Vitesse API

Generated by Doxygen 1.8.14

Contents

1 Ethernet Virtual Connections **1**

1.1 Port Configuration 1

1.2 Policer Configuration 1

1.3 EVCs 2

1.4 ECEs 2

1.5 EVC Statistics 3

1.6 ECE Statistics 3

2 Layer 2 **5**

2.1 MAC Address Table 5

2.2 Operational State 6

2.3 Spanning Tree 6

2.4 VLAN 6

2.5 VLAN Classification List 7

2.6 VLAN Translation 7

2.7 Port Isolation 8

2.8 Private VLAN 8

2.9 Asymmetric Private VLAN 8

2.10 Destination Port Groups 8

2.11 sFlow 9

2.12 Link Aggregation 9

2.13 Mirroring 9

2.14 Flooding Control 10

2.15 IPv4 Multicast 10

2.16 IPv6 Multicast 10

2.17 Ethernet Protection Switching 10

2.18 Ethernet Ring Protection Switching 11

3 MPLS API Overview **13**

3.1 API Usage 14

3.2 MPLS Statistics 15

4 Y.1731/IEEE802.1ag OAM **17**

4.1 - Vitesse OAM Processor 17

4.2 - Vitesse OAM Engine\n 17

5 Automatic Frame Injector. **19**

6 Security **21**

6.1 Port Authentication (802.1X) 21

6.2 Access Control List 21

6.2.1 Access Control Entry 21

6.2.2 Port Configuration 22

6.2.3 Policer Configuration 22

7 Data Structure Index **23**

7.1 Data Structures 23

8 File Index **35**

8.1 File List 35

9 Data Structure Documentation **37**

9.1 port_custom_conf_t Struct Reference 37

9.1.1 Detailed Description 37

9.1.2 Field Documentation 37

9.1.2.1 enable 38

9.1.2.2 autoneg 38

9.1.2.3 fdx 38

9.1.2.4 flow_control 38

9.1.2.5 pfc 38

9.1.2.6 speed 39

9.1.2.7 dual_media_fiber_speed 39

9.1.2.8 max_length . . . 39
9.1.2.9 power_mode . . . 39
9.1.2.10 exc_col_cont . . . 39
9.1.2.11 adv_dis . . . 40
9.1.2.12 max_tags . . . 40
9.1.2.13 oper_up . . . 40
9.1.2.14 frame_length_chk . . . 40
9.2 serdes_fields_t Struct Reference . . . 40
9.2.1 Detailed Description . . . 41
9.2.2 Field Documentation . . . 41
9.2.2.1 ob_post0 . . . 41
9.2.2.2 ob_sr . . . 41
9.3 tag_vtss_fdma_list Struct Reference . . . 41
9.3.1 Detailed Description . . . 42
9.3.2 Field Documentation . . . 42
9.3.2.1 frm_ptr . . . 42
9.3.2.2 act_len . . . 43
9.3.2.3 alloc_ptr . . . 43
9.3.2.4 rx_info . . . 43
9.3.2.5 sw_tstamp . . . 43
9.3.2.6 user . . . 44
9.3.2.7 next . . . 44
9.4 vtss_ace_frame_arp_t Struct Reference . . . 44
9.4.1 Detailed Description . . . 45
9.4.2 Field Documentation . . . 45
9.4.2.1 smac . . . 45
9.4.2.2 arp . . . 45
9.4.2.3 req . . . 45
9.4.2.4 unknown . . . 45
9.4.2.5 smac_match . . . 46

9.4.2.6 dmac_match . . . 46
9.4.2.7 length . . . 46
9.4.2.8 ip . . . 46
9.4.2.9 ethernet . . . 46
9.4.2.10 sip . . . 47
9.4.2.11 dip . . . 47
9.5 vtss_ace_frame_etype_t Struct Reference . . . 47
9.5.1 Detailed Description . . . 47
9.5.2 Field Documentation . . . 47
9.5.2.1 dmac . . . 48
9.5.2.2 smac . . . 48
9.5.2.3 etype . . . 48
9.5.2.4 data . . . 48
9.5.2.5 ptp . . . 48
9.6 vtss_ace_frame_ipv4_t Struct Reference . . . 49
9.6.1 Detailed Description . . . 49
9.6.2 Field Documentation . . . 49
9.6.2.1 ttl . . . 49
9.6.2.2 fragment . . . 50
9.6.2.3 options . . . 50
9.6.2.4 ds . . . 50
9.6.2.5 proto . . . 50
9.6.2.6 sip . . . 50
9.6.2.7 dip . . . 51
9.6.2.8 data . . . 51
9.6.2.9 sport . . . 51
9.6.2.10 dport . . . 51
9.6.2.11 tcp_fin . . . 51
9.6.2.12 tcp_syn . . . 52
9.6.2.13 tcp_rst . . . 52

9.6.2.14 tcp_psh . . . 52
9.6.2.15 tcp_ack . . . 52
9.6.2.16 tcp_urg . . . 52
9.6.2.17 sip_eq_dip . . . 53
9.6.2.18 sport_eq_dport . . . 53
9.6.2.19 seq_zero . . . 53
9.6.2.20 ptp . . . 53
9.6.2.21 sip_smac . . . 53
9.7 vtss_ace_frame_ipv6_t Struct Reference . . . 54
9.7.1 Detailed Description . . . 54
9.7.2 Field Documentation . . . 54
9.7.2.1 proto . . . 54
9.7.2.2 sip . . . 55
9.7.2.3 ttl . . . 55
9.7.2.4 ds . . . 55
9.7.2.5 data . . . 55
9.7.2.6 sport . . . 55
9.7.2.7 dport . . . 56
9.7.2.8 tcp_fin . . . 56
9.7.2.9 tcp_syn . . . 56
9.7.2.10 tcp_rst . . . 56
9.7.2.11 tcp_psh . . . 56
9.7.2.12 tcp_ack . . . 57
9.7.2.13 tcp_urg . . . 57
9.7.2.14 sip_eq_dip . . . 57
9.7.2.15 sport_eq_dport . . . 57
9.7.2.16 seq_zero . . . 57
9.7.2.17 ptp . . . 58
9.8 vtss_ace_frame_llc_t Struct Reference . . . 58
9.8.1 Detailed Description . . . 58

9.8.2 Field Documentation . . . 58
9.8.2.1 dmac . . . 58
9.8.2.2 smac . . . 59
9.8.2.3 llc . . . 59
9.9 vtss_ace_frame_snap_t Struct Reference . . . 59
9.9.1 Detailed Description . . . 59
9.9.2 Field Documentation . . . 59
9.9.2.1 dmac . . . 60
9.9.2.2 smac . . . 60
9.9.2.3 snap . . . 60
9.10 vtss_ace_ptp_t Struct Reference . . . 60
9.10.1 Detailed Description . . . 60
9.10.2 Field Documentation . . . 61
9.10.2.1 enable . . . 61
9.10.2.2 header . . . 61
9.11 vtss_ace_sip_smac_t Struct Reference . . . 61
9.11.1 Detailed Description . . . 61
9.11.2 Field Documentation . . . 62
9.11.2.1 enable . . . 62
9.11.2.2 sip . . . 62
9.11.2.3 smac . . . 62
9.12 vtss_ace_t Struct Reference . . . 62
9.12.1 Detailed Description . . . 63
9.12.2 Field Documentation . . . 63
9.12.2.1 id . . . 63
9.12.2.2 lookup . . . 63
9.12.2.3 isdx_enable . . . 64
9.12.2.4 isdx_disable . . . 64
9.12.2.5 port_list . . . 64
9.12.2.6 policy . . . 64

9.12.2.7 type . . . 64
9.12.2.8 action . . . 65
9.12.2.9 dmac_mc . . . 65
9.12.2.10 dmac_bc . . . 65
9.12.2.11 vlan . . . 65
9.12.2.12 etype . . . 65
9.12.2.13 llc . . . 66
9.12.2.14 snap . . . 66
9.12.2.15 arp . . . 66
9.12.2.16 ipv4 . . . 66
9.12.2.17 ipv6 . . . 66
9.12.2.18 frame . . . 67
9.13 vtss_ace_vlan_t Struct Reference . . . 67
9.13.1 Detailed Description . . . 67
9.13.2 Field Documentation . . . 67
9.13.2.1 vid . . . 67
9.13.2.2 usr_prio . . . 68
9.13.2.3 cfi . . . 68
9.13.2.4 tagged . . . 68
9.14 vtss_acl_action_t Struct Reference . . . 68
9.14.1 Detailed Description . . . 69
9.14.2 Field Documentation . . . 69
9.14.2.1 cpu . . . 69
9.14.2.2 cpu_once . . . 69
9.14.2.3 cpu_queue . . . 69
9.14.2.4 police . . . 69
9.14.2.5 policer_no . . . 70
9.14.2.6 evc_police . . . 70
9.14.2.7 evc_policer_id . . . 70
9.14.2.8 learn . . . 70

9.14.2.9 port_action . . . 70
9.14.2.10 port_list . . . 71
9.14.2.11 mirror . . . 71
9.14.2.12 ptp_action . . . 71
9.14.2.13 lm_cnt_disable . . . 71
9.14.2.14 mac_swap . . . 71
9.15 vtss_acl_policer_conf_t Struct Reference . . . 72
9.15.1 Detailed Description . . . 72
9.15.2 Field Documentation . . . 72
9.15.2.1 bit_rate_enable . . . 72
9.15.2.2 bit_rate . . . 72
9.15.2.3 rate . . . 73
9.16 vtss_acl_port_conf_t Struct Reference . . . 73
9.16.1 Detailed Description . . . 73
9.16.2 Field Documentation . . . 73
9.16.2.1 policy_no . . . 73
9.16.2.2 action . . . 74
9.17 vtss_afi_frm_dscr_t Struct Reference . . . 74
9.17.1 Detailed Description . . . 74
9.17.2 Field Documentation . . . 74
9.17.2.1 fps . . . 74
9.17.2.2 actual_fps . . . 75
9.18 vtss_aggr_mode_t Struct Reference . . . 75
9.18.1 Detailed Description . . . 75
9.18.2 Field Documentation . . . 75
9.18.2.1 smac_enable . . . 76
9.18.2.2 dmac_enable . . . 76
9.18.2.3 sip_dip_enable . . . 76
9.18.2.4 sport_dport_enable . . . 76
9.19 vtss_aneg_t Struct Reference . . . 76

9.19.1 Detailed Description . . . 77
9.19.2 Field Documentation . . . 77
9.19.2.1 obey_pause . . . 77
9.19.2.2 generate_pause . . . 77
9.20 vtss_api_lock_t Struct Reference . . . 77
9.20.1 Detailed Description . . . 78
9.20.2 Field Documentation . . . 78
9.20.2.1 inst . . . 78
9.20.2.2 function . . . 78
9.20.2.3 file . . . 78
9.20.2.4 line . . . 79
9.21 vtss_basic_counters_t Struct Reference . . . 79
9.21.1 Detailed Description . . . 79
9.21.2 Field Documentation . . . 79
9.21.2.1 rx_frames . . . 79
9.21.2.2 tx_frames . . . 80
9.22 vtss_chip_id_t Struct Reference . . . 80
9.22.1 Detailed Description . . . 80
9.22.2 Field Documentation . . . 80
9.22.2.1 part_number . . . 80
9.22.2.2 revision . . . 81
9.23 vtss_counter_pair_t Struct Reference . . . 81
9.23.1 Detailed Description . . . 81
9.23.2 Field Documentation . . . 81
9.23.2.1 frames . . . 81
9.23.2.2 bytes . . . 82
9.24 vtss_debug_info_t Struct Reference . . . 82
9.24.1 Detailed Description . . . 82
9.24.2 Field Documentation . . . 82
9.24.2.1 layer . . . 82

9.24.2.2 group . . . 83
9.24.2.3 chip_no . . . 83
9.24.2.4 port_list . . . 83
9.24.2.5 full . . . 83
9.24.2.6 clear . . . 83
9.24.2.7 vml_format . . . 84
9.25 vtss_debug_lock_t Struct Reference . . . 84
9.25.1 Detailed Description . . . 84
9.25.2 Field Documentation . . . 84
9.25.2.1 chip_no . . . 84
9.26 vtss_dgroup_port_conf_t Struct Reference . . . 85
9.26.1 Detailed Description . . . 85
9.26.2 Field Documentation . . . 85
9.26.2.1 dgroup_no . . . 85
9.27 vtss_dlb_policer_conf_t Struct Reference . . . 85
9.27.1 Detailed Description . . . 86
9.27.2 Field Documentation . . . 86
9.27.2.1 type . . . 86
9.27.2.2 enable . . . 86
9.27.2.3 cf . . . 86
9.27.2.4 line_rate . . . 86
9.27.2.5 cir . . . 87
9.27.2.6 cbs . . . 87
9.27.2.7 eir . . . 87
9.27.2.8 ebs . . . 87
9.28 vtss_ece_action_t Struct Reference . . . 87
9.28.1 Detailed Description . . . 88
9.28.2 Field Documentation . . . 88
9.28.2.1 dir . . . 88
9.28.2.2 rule . . . 88

9.28.2.3 tx_lookup . . . 89
9.28.2.4 pop_tag . . . 89
9.28.2.5 outer_tag . . . 89
9.28.2.6 inner_tag . . . 89
9.28.2.7 policer_id . . . 89
9.28.2.8 evc_id . . . 90
9.28.2.9 policy_no . . . 90
9.28.2.10 prio_enable . . . 90
9.28.2.11 prio . . . 90
9.28.2.12 dp_enable . . . 90
9.28.2.13 dp . . . 91
9.29 vtss_ece_frame_etype_t Struct Reference . . . 91
9.29.1 Detailed Description . . . 91
9.29.2 Field Documentation . . . 91
9.29.2.1 etype . . . 91
9.29.2.2 data . . . 92
9.30 vtss_ece_frame_ipv4_t Struct Reference . . . 92
9.30.1 Detailed Description . . . 92
9.30.2 Field Documentation . . . 92
9.30.2.1 dscp . . . 92
9.30.2.2 fragment . . . 93
9.30.2.3 proto . . . 93
9.30.2.4 sip . . . 93
9.30.2.5 dip . . . 93
9.30.2.6 sport . . . 93
9.30.2.7 dport . . . 94
9.31 vtss_ece_frame_ipv6_t Struct Reference . . . 94
9.31.1 Detailed Description . . . 94
9.31.2 Field Documentation . . . 94
9.31.2.1 dscp . . . 94

9.31.2.2 proto . . . 95
9.31.2.3 sip . . . 95
9.31.2.4 dip . . . 95
9.31.2.5 sport . . . 95
9.31.2.6 dport . . . 95
9.32 vtss_ece_frame_llc_t Struct Reference . . . 96
9.32.1 Detailed Description . . . 96
9.32.2 Field Documentation . . . 96
9.32.2.1 data . . . 96
9.33 vtss_ece_frame_snap_t Struct Reference . . . 96
9.33.1 Detailed Description . . . 97
9.33.2 Field Documentation . . . 97
9.33.2.1 data . . . 97
9.34 vtss_ece_inner_tag_t Struct Reference . . . 97
9.34.1 Detailed Description . . . 97
9.34.2 Field Documentation . . . 97
9.34.2.1 type . . . 98
9.34.2.2 vid . . . 98
9.34.2.3 pcp_mode . . . 98
9.34.2.4 pcp . . . 98
9.34.2.5 dei_mode . . . 98
9.34.2.6 dei . . . 99
9.35 vtss_ece_key_t Struct Reference . . . 99
9.35.1 Detailed Description . . . 99
9.35.2 Field Documentation . . . 99
9.35.2.1 port_list . . . 100
9.35.2.2 mac . . . 100
9.35.2.3 tag . . . 100
9.35.2.4 inner_tag . . . 100
9.35.2.5 type . . . 100

9.35.2.6 lookup . . . 101
9.35.2.7 etype . . . 101
9.35.2.8 llc . . . 101
9.35.2.9 snap . . . 101
9.35.2.10 ipv4 . . . 101
9.35.2.11 ipv6 . . . 102
9.35.2.12 frame . . . 102
9.36 vtss_ece_mac_t Struct Reference . . . 102
9.36.1 Detailed Description . . . 102
9.36.2 Field Documentation . . . 102
9.36.2.1 dmac . . . 103
9.36.2.2 dmac_mc . . . 103
9.36.2.3 dmac_bc . . . 103
9.36.2.4 smac . . . 103
9.37 vtss_ece_outer_tag_t Struct Reference . . . 103
9.37.1 Detailed Description . . . 104
9.37.2 Field Documentation . . . 104
9.37.2.1 enable . . . 104
9.37.2.2 vid . . . 104
9.37.2.3 pcp_mode . . . 104
9.37.2.4 pcp . . . 105
9.37.2.5 dei_mode . . . 105
9.37.2.6 dei . . . 105
9.38 vtss_ece_t Struct Reference . . . 105
9.38.1 Detailed Description . . . 105
9.38.2 Field Documentation . . . 106
9.38.2.1 id . . . 106
9.38.2.2 key . . . 106
9.38.2.3 action . . . 106
9.39 vtss_ece_tag_t Struct Reference . . . 106

9.39.1 Detailed Description . . . 107
9.39.2 Field Documentation . . . 107
9.39.2.1 vid . . . 107
9.39.2.2 pcp . . . 107
9.39.2.3 dei . . . 107
9.39.2.4 tagged . . . 107
9.39.2.5 s_tagged . . . 108
9.40 vtss_eee_port_conf_t Struct Reference . . . 108
9.40.1 Detailed Description . . . 108
9.40.2 Field Documentation . . . 108
9.40.2.1 eee_ena . . . 108
9.40.2.2 eee_fast_queues . . . 109
9.40.2.3 tx_tw . . . 109
9.40.2.4 lp_advertisement . . . 109
9.40.2.5 optimized_for_power . . . 109
9.41 vtss_eee_port_counter_t Struct Reference . . . 109
9.41.1 Detailed Description . . . 110
9.41.2 Field Documentation . . . 110
9.41.2.1 fill_level_get . . . 110
9.41.2.2 fill_level_thres . . . 110
9.41.2.3 fill_level . . . 110
9.41.2.4 tx_out_bytes_get . . . 111
9.41.2.5 tx_out_bytes . . . 111
9.42 vtss_eee_port_state_t Struct Reference . . . 111
9.42.1 Detailed Description . . . 111
9.42.2 Field Documentation . . . 111
9.42.2.1 select . . . 112
9.42.2.2 val . . . 112
9.43 vtss_eps_port_conf_t Struct Reference . . . 112
9.43.1 Detailed Description . . . 112

9.43.2 Field Documentation . . . 112
9.43.2.1 type . . . 113
9.43.2.2 port_no . . . 113
9.44 vtss_evc_conf_t Struct Reference . . . 113
9.44.1 Detailed Description . . . 113
9.44.2 Field Documentation . . . 113
9.44.2.1 policer_id . . . 114
9.44.2.2 learning . . . 114
9.44.2.3 pb . . . 114
9.44.2.4 mpls_tp . . . 114
9.44.2.5 network . . . 114
9.45 vtss_evc_counters_t Struct Reference . . . 115
9.45.1 Detailed Description . . . 115
9.45.2 Field Documentation . . . 115
9.45.2.1 rx_green . . . 115
9.45.2.2 rx_yellow . . . 115
9.45.2.3 rx_red . . . 116
9.45.2.4 rx_discard . . . 116
9.45.2.5 tx_discard . . . 116
9.45.2.6 tx_green . . . 116
9.45.2.7 tx_yellow . . . 116
9.46 vtss_evc_mpls_pw_info_t Struct Reference . . . 117
9.46.1 Detailed Description . . . 117
9.46.2 Field Documentation . . . 117
9.46.2.1 split_horizon . . . 117
9.46.2.2 ingress_xc . . . 117
9.46.2.3 egress_xc . . . 118
9.46.2.4 pw_num . . . 118
9.47 vtss_evc_mpls_tp_conf_t Struct Reference . . . 118
9.47.1 Detailed Description . . . 118

9.47.2 Field Documentation . . . 118
9.47.2.1 pw . . . 119
9.48 vtss_evc_oam_port_conf_t Struct Reference . . . 119
9.48.1 Detailed Description . . . 119
9.48.2 Field Documentation . . . 119
9.48.2.1 voe_idx . . . 119
9.49 vtss_evc_pb_conf_t Struct Reference . . . 120
9.49.1 Detailed Description . . . 120
9.49.2 Field Documentation . . . 120
9.49.2.1 nni . . . 120
9.49.2.2 ivid . . . 120
9.49.2.3 vid . . . 121
9.49.2.4 leaf_ivid . . . 121
9.49.2.5 leaf_vid . . . 121
9.49.2.6 leaf . . . 121
9.50 vtss_evc_port_conf_t Struct Reference . . . 121
9.50.1 Detailed Description . . . 122
9.50.2 Field Documentation . . . 122
9.50.2.1 dmac_dip . . . 122
9.50.2.2 key_type . . . 122
9.51 vtss_fan_conf_t Struct Reference . . . 122
9.51.1 Detailed Description . . . 123
9.51.2 Field Documentation . . . 123
9.51.2.1 fan_pwm_freq . . . 123
9.51.2.2 fan_low_pol . . . 123
9.51.2.3 fan_open_col . . . 123
9.51.2.4 type . . . 124
9.51.2.5 ppr . . . 124
9.52 vtss_fdma_cfg_t Struct Reference . . . 124
9.52.1 Detailed Description . . . 124

9.52.2 Field Documentation . . . 125
9.52.2.1 enable . . . 125
9.52.2.2 rx_mtu . . . 125
9.52.2.3 rx_buf_cnt . . . 125
9.52.2.4 rx_alloc_cb . . . 126
9.52.2.5 rx_dont_strip_vlan_tag . . . 127
9.52.2.6 rx_dont_reinsert_vlan_tag . . . 127
9.52.2.7 rx_allow_vlan_tag_mismatch . . . 127
9.52.2.8 rx_allow_multiple_dcbs . . . 127
9.52.2.9 rx_cb . . . 128
9.52.2.10 tx_buf_cnt . . . 128
9.52.2.11 tx_done_cb . . . 128
9.52.2.12 afi_buf_cnt . . . 129
9.52.2.13 afi_done_cb . . . 129
9.53 vtss_fdma_ch_cfg_t Struct Reference . . . 129
9.53.1 Detailed Description . . . 130
9.53.2 Field Documentation . . . 130
9.53.2.1 usage . . . 130
9.53.2.2 xtr_grp . . . 130
9.53.2.3 inj_grp_mask . . . 131
9.53.2.4 list . . . 131
9.53.2.5 xtr_cb . . . 132
9.53.2.6 prio . . . 132
9.53.2.7 chip_no . . . 133
9.54 vtss_fdma_throttle_cfg_t Struct Reference . . . 133
9.54.1 Detailed Description . . . 133
9.54.2 Field Documentation . . . 134
9.54.2.1 frm_limit_per_tick . . . 134
9.54.2.2 byte_limit_per_tick . . . 134
9.54.2.3 suspend_tick_cnt . . . 134

9.55 vtss_fdma_tx_info_t Struct Reference . . . 134
9.55.1 Detailed Description . . . 135
9.55.2 Field Documentation . . . 135
9.55.2.1 pre_cb_ctxt1 . . . 135
9.55.2.2 pre_cb_ctxt2 . . . 135
9.55.2.3 pre_cb . . . 136
9.55.2.4 afi_fps . . . 136
9.55.2.5 afi_type . . . 137
9.56 vtss_hqos_conf_t Struct Reference . . . 137
9.56.1 Detailed Description . . . 137
9.56.2 Field Documentation . . . 137
9.56.2.1 port_no . . . 137
9.56.2.2 shaper . . . 138
9.56.2.3 shaper_queue . . . 138
9.56.2.4 dwrr_cnt . . . 138
9.56.2.5 queue_pct . . . 138
9.56.2.6 min_rate . . . 138
9.57 vtss_hqos_port_conf_t Struct Reference . . . 139
9.57.1 Detailed Description . . . 139
9.57.2 Field Documentation . . . 139
9.57.2.1 sch_mode . . . 139
9.58 vtss_init_conf_t Struct Reference . . . 139
9.58.1 Detailed Description . . . 140
9.58.2 Field Documentation . . . 140
9.58.2.1 reg_read . . . 140
9.58.2.2 reg_write . . . 140
9.58.2.3 miim_read . . . 141
9.58.2.4 miim_write . . . 141
9.58.2.5 mmd_read . . . 141
9.58.2.6 mmd_read_inc . . . 141

9.58.2.7 mmd_write . . . 141
9.58.2.8 spi_read_write . . . 142
9.58.2.9 spi_32bit_read_write . . . 142
9.58.2.10 spi_64bit_read_write . . . 142
9.58.2.11 warm_start_enable . . . 142
9.58.2.12 restart_info_src . . . 142
9.58.2.13 restart_info_port . . . 143
9.58.2.14 pi . . . 143
9.58.2.15 serdes . . . 143
9.59 vtss_inst_create_t Struct Reference . . . 143
9.59.1 Detailed Description . . . 143
9.59.2 Field Documentation . . . 144
9.59.2.1 target . . . 144
9.60 vtss_intr_t Struct Reference . . . 144
9.60.1 Detailed Description . . . 144
9.60.2 Field Documentation . . . 144
9.60.2.1 link_change . . . 144
9.61 vtss_ip_addr_t Struct Reference . . . 145
9.61.1 Detailed Description . . . 145
9.61.2 Field Documentation . . . 145
9.61.2.1 type . . . 145
9.61.2.2 ipv4 . . . 145
9.61.2.3 ipv6 . . . 146
9.61.2.4 addr . . . 146
9.62 vtss_ip_network_t Struct Reference . . . 146
9.62.1 Detailed Description . . . 146
9.62.2 Field Documentation . . . 146
9.62.2.1 address . . . 146
9.62.2.2 prefix_size . . . 147
9.63 vtss_ipv4_network_t Struct Reference . . . 147

9.63.1 Detailed Description . . . 147
9.63.2 Field Documentation . . . 147
9.63.2.1 address . . . 147
9.63.2.2 prefix_size . . . 148
9.64 vtss_ipv4_uc_t Struct Reference . . . 148
9.64.1 Detailed Description . . . 148
9.64.2 Field Documentation . . . 148
9.64.2.1 network . . . 148
9.64.2.2 destination . . . 149
9.65 vtss_ipv6_network_t Struct Reference . . . 149
9.65.1 Detailed Description . . . 149
9.65.2 Field Documentation . . . 149
9.65.2.1 address . . . 149
9.65.2.2 prefix_size . . . 150
9.66 vtss_ipv6_t Struct Reference . . . 150
9.66.1 Detailed Description . . . 150
9.66.2 Field Documentation . . . 150
9.66.2.1 addr . . . 150
9.67 vtss_ipv6_uc_t Struct Reference . . . 151
9.67.1 Detailed Description . . . 151
9.67.2 Field Documentation . . . 151
9.67.2.1 network . . . 151
9.67.2.2 destination . . . 151
9.68 vtss_irq_conf_t Struct Reference . . . 151
9.68.1 Detailed Description . . . 152
9.68.2 Field Documentation . . . 152
9.68.2.1 external . . . 152
9.68.2.2 destination . . . 152
9.69 vtss_irq_status_t Struct Reference . . . 152
9.69.1 Detailed Description . . . 153

9.69.2 Field Documentation . . . 153
9.69.2.1 active . . . 153
9.69.2.2 raw_ident . . . 153
9.69.2.3 raw_status . . . 153
9.69.2.4 raw_mask . . . 154
9.70 vtss_l3_counters_t Struct Reference . . . 154
9.70.1 Detailed Description . . . 154
9.70.2 Field Documentation . . . 154
9.70.2.1 ipv4uc_received_octets . . . 154
9.70.2.2 ipv4uc_received_frames . . . 155
9.70.2.3 ipv6uc_received_octets . . . 155
9.70.2.4 ipv6uc_received_frames . . . 155
9.70.2.5 ipv4uc_transmitted_octets . . . 155
9.70.2.6 ipv4uc_transmitted_frames . . . 155
9.70.2.7 ipv6uc_transmitted_octets . . . 156
9.70.2.8 ipv6uc_transmitted_frames . . . 156
9.71 vtss_lcpll_status_t Struct Reference . . . 156
9.71.1 Detailed Description . . . 156
9.71.2 Field Documentation . . . 156
9.71.2.1 lock_status . . . 157
9.71.2.2 cal_done . . . 157
9.71.2.3 cal_error . . . 157
9.71.2.4 fsm_lock . . . 157
9.71.2.5 fsm_stat . . . 157
9.71.2.6 gain_stat . . . 158
9.72 vtss_learn_mode_t Struct Reference . . . 158
9.72.1 Detailed Description . . . 158
9.72.2 Field Documentation . . . 158
9.72.2.1 automatic . . . 158
9.72.2.2 cpu . . . 159

9.72.2.3 discard . . . 159
9.73 vtss_mac_t Struct Reference . . . 159
9.73.1 Detailed Description . . . 159
9.73.2 Field Documentation . . . 159
9.73.2.1 addr . . . 160
9.74 vtss_mac_table_entry_t Struct Reference . . . 160
9.74.1 Detailed Description . . . 160
9.74.2 Field Documentation . . . 160
9.74.2.1 vid_mac . . . 160
9.74.2.2 destination . . . 161
9.74.2.3 copy_to_cpu . . . 161
9.74.2.4 locked . . . 161
9.74.2.5 aged . . . 161
9.74.2.6 cpu_queue . . . 161
9.75 vtss_mac_table_status_t Struct Reference . . . 162
9.75.1 Detailed Description . . . 162
9.75.2 Field Documentation . . . 162
9.75.2.1 learned . . . 162
9.75.2.2 replaced . . . 162
9.75.2.3 moved . . . 163
9.75.2.4 aged . . . 163
9.76 vtss_mce_action_t Struct Reference . . . 163
9.76.1 Detailed Description . . . 164
9.76.2 Field Documentation . . . 164
9.76.2.1 port_list . . . 164
9.76.2.2 evc_counting . . . 164
9.76.2.3 evc_etree . . . 164
9.76.2.4 evc_id . . . 164
9.76.2.5 voe_idx . . . 165
9.76.2.6 outer_tag . . . 165

9.76.2.7 isdx . . . 165
9.76.2.8 rule . . . 165
9.76.2.9 tx_lookup . . . 165
9.76.2.10 oam_detect . . . 166
9.76.2.11 policer_id . . . 166
9.76.2.12 policy_no . . . 166
9.76.2.13 prio_enable . . . 166
9.76.2.14 prio . . . 166
9.76.2.15 vid . . . 167
9.76.2.16 pop_cnt . . . 167
9.77 vtss_mce_key_t Struct Reference . . . 167
9.77.1 Detailed Description . . . 167
9.77.2 Field Documentation . . . 168
9.77.2.1 port_list . . . 168
9.77.2.2 port_cpu . . . 168
9.77.2.3 tag . . . 168
9.77.2.4 inner_tag . . . 168
9.77.2.5 mel . . . 168
9.77.2.6 injected . . . 169
9.77.2.7 lookup . . . 169
9.77.2.8 dmac . . . 169
9.77.2.9 dmac_mc . . . 169
9.77.2.10 service_detect . . . 169
9.78 vtss_mce_outer_tag_t Struct Reference . . . 170
9.78.1 Detailed Description . . . 170
9.78.2 Field Documentation . . . 170
9.78.2.1 enable . . . 170
9.78.2.2 vid . . . 170
9.78.2.3 pcp_mode . . . 171
9.78.2.4 pcp . . . 171

9.78.2.5 dei_mode . . . 171
9.78.2.6 dei . . . 171
9.79 vtss_mce_port_info_t Struct Reference . . . 171
9.79.1 Detailed Description . . . 172
9.79.2 Field Documentation . . . 172
9.79.2.1 isdx . . . 172
9.80 vtss_mce_t Struct Reference . . . 172
9.80.1 Detailed Description . . . 172
9.80.2 Field Documentation . . . 173
9.80.2.1 tt_loop . . . 173
9.80.2.2 id . . . 173
9.80.2.3 key . . . 173
9.80.2.4 action . . . 173
9.81 vtss_mce_tag_t Struct Reference . . . 174
9.81.1 Detailed Description . . . 174
9.81.2 Field Documentation . . . 174
9.81.2.1 tagged . . . 174
9.81.2.2 s_tagged . . . 174
9.81.2.3 vid . . . 175
9.81.2.4 pcp . . . 175
9.81.2.5 dei . . . 175
9.82 vtss_mirror_conf_t Struct Reference . . . 175
9.82.1 Detailed Description . . . 175
9.82.2 Field Documentation . . . 176
9.82.2.1 port_no . . . 176
9.82.2.2 fwd_enable . . . 176
9.83 vtss_mpls_l2_t Struct Reference . . . 176
9.83.1 Detailed Description . . . 176
9.83.2 Field Documentation . . . 177
9.83.2.1 port . . . 177

9.83.2.2 peer_mac . . . 177
9.83.2.3 self_mac . . . 177
9.83.2.4 tag_type . . . 177
9.83.2.5 vid . . . 177
9.83.2.6 pcp . . . 178
9.83.2.7 dei . . . 178
9.83.2.8 section_oam . . . 178
9.84 vtss_mpls_label_t Struct Reference . . . 178
9.84.1 Detailed Description . . . 178
9.84.2 Field Documentation . . . 179
9.84.2.1 value . . . 179
9.84.2.2 tc . . . 179
9.84.2.3 ttl . . . 179
9.85 vtss_mpls_oam_conf_t Struct Reference . . . 179
9.85.1 Detailed Description . . . 180
9.85.2 Field Documentation . . . 180
9.85.2.1 type . . . 180
9.86 vtss_mpls_pw_conf_t Struct Reference . . . 180
9.86.1 Detailed Description . . . 180
9.86.2 Field Documentation . . . 180
9.86.2.1 is_pw . . . 181
9.86.2.2 process_cw . . . 181
9.86.2.3 cw . . . 181
9.87 vtss_mpls_qos_to_tc_map_entry_t Struct Reference . . . 181
9.87.1 Detailed Description . . . 181
9.87.2 Field Documentation . . . 182
9.87.2.1 dp0_tc . . . 182
9.87.2.2 dp1_tc . . . 182
9.88 vtss_mpls_segment_status_t Struct Reference . . . 182
9.88.1 Detailed Description . . . 182

9.88.2 Field Documentation . . . 182
9.88.2.1 state . . . 183
9.88.2.2 oam_active . . . 183
9.89 vtss_mpls_segment_t Struct Reference . . . 183
9.89.1 Detailed Description . . . 183
9.89.2 Field Documentation . . . 184
9.89.2.1 is_in . . . 184
9.89.2.2 e_lsp . . . 184
9.89.2.3 l_lsp_cos . . . 184
9.89.2.4 tc_qos_map_idx . . . 184
9.89.2.5 l2_idx . . . 185
9.89.2.6 label . . . 185
9.89.2.7 upstream . . . 185
9.89.2.8 hqos_id . . . 185
9.89.2.9 pw_conf . . . 185
9.89.2.10 oam_conf . . . 186
9.89.2.11 xc_idx . . . 186
9.89.2.12 server_idx . . . 186
9.90 vtss_mpls_tc_conf_t Struct Reference . . . 186
9.90.1 Detailed Description . . . 186
9.90.2 Field Documentation . . . 187
9.90.2.1 qos_to_tc_map . . . 187
9.90.2.2 tc_to_qos_map . . . 187
9.91 vtss_mpls_tc_to_qos_map_entry_t Struct Reference . . . 187
9.91.1 Detailed Description . . . 187
9.91.2 Field Documentation . . . 187
9.91.2.1 qos . . . 188
9.91.2.2 dp . . . 188
9.92 vtss_mpls_xc_counters_t Struct Reference . . . 188
9.92.1 Detailed Description . . . 188

9.92.2 Field Documentation . . . 188
9.92.2.1 rx_green . . . 189
9.92.2.2 rx_yellow . . . 189
9.92.2.3 rx_red . . . 189
9.92.2.4 rx_discard . . . 189
9.92.2.5 tx_discard . . . 189
9.92.2.6 tx_green . . . 190
9.92.2.7 tx_yellow . . . 190
9.93 vtss_mpls_xc_t Struct Reference . . . 190
9.93.1 Detailed Description . . . 190
9.93.2 Field Documentation . . . 191
9.93.2.1 type . . . 191
9.93.2.2 tc_tunnel_mode . . . 191
9.93.2.3 ttl_tunnel_mode . . . 191
9.93.2.4 in_seg_idx . . . 191
9.93.2.5 out_seg_idx . . . 191
9.93.2.6 flow_handle . . . 192
9.94 vtss_mtimer_t Struct Reference . . . 192
9.94.1 Detailed Description . . . 192
9.94.2 Field Documentation . . . 192
9.94.2.1 timeout . . . 192
9.94.2.2 now . . . 193
9.95 vtss_npi_conf_t Struct Reference . . . 193
9.95.1 Detailed Description . . . 193
9.95.2 Field Documentation . . . 193
9.95.2.1 enable . . . 193
9.95.2.2 port_no . . . 194
9.96 vtss_oam_ccm_counter_t Struct Reference . . . 194
9.96.1 Detailed Description . . . 194
9.96.2 Field Documentation . . . 194

9.96.2.1 rx_valid_count . . . 194
9.96.2.2 rx_invalid_count . . . 195
9.96.2.3 rx_invalid_seq_no . . . 195
9.97 vtss_oam_ccm_status_t Struct Reference . . . 195
9.97.1 Detailed Description . . . 195
9.97.2 Field Documentation . . . 195
9.97.2.1 period_err . . . 196
9.97.2.2 priority_err . . . 196
9.97.2.3 zero_period_err . . . 196
9.97.2.4 rx_rdi . . . 196
9.97.2.5 loc . . . 196
9.97.2.6 mep_id_err . . . 197
9.97.2.7 meg_id_err . . . 197
9.98 vtss_oam_event_mask_t Struct Reference . . . 197
9.98.1 Detailed Description . . . 197
9.98.2 Field Documentation . . . 197
9.98.2.1 voe_mask . . . 198
9.99 vtss_oam_ipt_conf_t Struct Reference . . . 198
9.99.1 Detailed Description . . . 198
9.99.2 Field Documentation . . . 198
9.99.2.1 enable . . . 198
9.99.2.2 service_voe_idx . . . 199
9.100 vtss_oam_lb_counter_t Struct Reference . . . 199
9.100.1 Detailed Description . . . 199
9.100.2 Field Documentation . . . 199
9.100.2.1 rx_lbr . . . 199
9.100.2.2 tx_lbm . . . 200
9.100.2.3 rx_lbr_trans_id_err . . . 200
9.101 vtss_oam_lm_counter_t Struct Reference . . . 200
9.101.1 Detailed Description . . . 201

9.101.2 Field Documentation . . . 201
9.101.2.1 tx_FCf . . . 201
9.101.2.2 rx_FCl . . . 201
9.101.2.3 down_mep . . . 201
9.101.2.4 rx_lmm_sample_seq_no . . . 201
9.101.2.5 rx_lmr_sample_seq_no . . . 202
9.101.2.6 rx_ccm_lm_sample_seq_no . . . 202
9.101.2.7 lmm . . . 202
9.101.2.8 lmr . . . 202
9.101.2.9 ccm_lm . . . 202
9.101.2.10 lmm_valid . . . 203
9.101.2.11 lmr_valid . . . 203
9.101.2.12 ccm_lm_valid . . . 203
9.101.2.13 lmm_sample_lost . . . 203
9.101.2.14 lmr_sample_lost . . . 203
9.101.2.15 ccm_lm_sample_lost . . . 204
9.101.2.16 up_mep . . . 204
9.101.2.17 lm_count . . . 204
9.102 vtss_oam_meg_id_t Struct Reference . . . 204
9.102.1 Detailed Description . . . 204
9.102.2 Field Documentation . . . 205
9.102.2.1 data . . . 205
9.103 vtss_oam_pdu_seen_status_t Struct Reference . . . 205
9.103.1 Detailed Description . . . 205
9.103.2 Field Documentation . . . 206
9.103.2.1 generic_seen . . . 206
9.103.2.2 unknown_seen . . . 206
9.103.2.3 ltm_seen . . . 206
9.103.2.4 ltr_seen . . . 206
9.103.2.5 lmm_seen . . . 206

9.103.2.6 lmr_seen . . . 207
9.103.2.7 lbm_seen . . . 207
9.103.2.8 lbr_seen . . . 207
9.103.2.9 dmm_seen . . . 207
9.103.2.10 dmr_seen . . . 207
9.103.2.11 one_dm_seen . . . 208
9.103.2.12 ccm_seen . . . 208
9.103.2.13 ccm_lm_seen . . . 208
9.104 vtss_oam_proc_conf_t Struct Reference . . . 208
9.104.1 Detailed Description . . . 209
9.104.2 Field Documentation . . . 209
9.104.2.1 meg_level . . . 209
9.104.2.2 dmac_check_type . . . 209
9.104.2.3 ccm_check_only . . . 209
9.104.2.4 copy_next_only . . . 209
9.104.2.5 copy_on_ccm_err . . . 210
9.104.2.6 copy_on_mel_too_low_err . . . 210
9.104.2.7 copy_on_ccm_more_than_one_tlv . . . 210
9.104.2.8 copy_on_dmac_err . . . 210
9.105 vtss_oam_proc_status_t Struct Reference . . . 210
9.105.1 Detailed Description . . . 211
9.105.2 Field Documentation . . . 211
9.105.2.1 rx_ccm_seq_no . . . 211
9.105.2.2 tx_next_ccm_seq_no . . . 211
9.105.2.3 rx_lbr_transaction_id . . . 212
9.105.2.4 rx_tst_seq_no . . . 212
9.105.2.5 tx_next_lbm_transaction_id . . . 212
9.105.2.6 rx_meg_level_err . . . 212
9.105.2.7 rx_meg_level_err_seen . . . 212
9.105.2.8 rx_dmac_err_seen . . . 213

9.105.2.9 tx_meg_level_err_seen . . . 213
9.106 vtss_oam_rx_tx_counter_t Struct Reference . . . 213
9.106.1 Detailed Description . . . 213
9.106.2 Field Documentation . . . 213
9.106.2.1 rx . . . 214
9.106.2.2 tx . . . 214
9.107 vtss_oam_sel_counter_t Struct Reference . . . 214
9.107.1 Detailed Description . . . 214
9.107.2 Field Documentation . . . 214
9.107.2.1 selected_frames . . . 215
9.107.2.2 nonselected_frames . . . 215
9.108 vtss_oam_ts_id_s Struct Reference . . . 215
9.108.1 Detailed Description . . . 215
9.108.2 Field Documentation . . . 215
9.108.2.1 voe_id . . . 216
9.108.2.2 voe_sq . . . 216
9.109 vtss_oam_ts_timestamp_s Struct Reference . . . 216
9.109.1 Detailed Description . . . 216
9.109.2 Field Documentation . . . 216
9.109.2.1 ts . . . 217
9.109.2.2 port_no . . . 217
9.109.2.3 ts_valid . . . 217
9.110 vtss_oam_tst_counter_t Struct Reference . . . 217
9.110.1 Detailed Description . . . 217
9.110.2 Field Documentation . . . 218
9.110.2.1 rx_tst . . . 218
9.110.2.2 tx_tst . . . 218
9.110.2.3 rx_tst_trans_id_err . . . 218
9.111 vtss_oam_voe_alloc_cfg_t Struct Reference . . . 218
9.111.1 Detailed Description . . . 219

9.111.2 Field Documentation . . . 219
9.111.2.1 phys_port . . . 219
9.112 vtss_oam_voe_ccm_conf_t Struct Reference . . . 219
9.112.1 Detailed Description . . . 219
9.112.2 Field Documentation . . . 220
9.112.2.1 enable . . . 220
9.112.2.2 copy_to_cpu . . . 220
9.112.2.3 forward . . . 220
9.112.2.4 count_as_selected . . . 220
9.112.2.5 count_as_data . . . 220
9.112.2.6 mepid . . . 221
9.112.2.7 megid . . . 221
9.112.2.8 tx_seq_no_auto_upd_op . . . 221
9.112.2.9 tx_seq_no . . . 221
9.112.2.10 rx_seq_no_check . . . 221
9.112.2.11 rx_seq_no . . . 222
9.112.2.12 rx_priority . . . 222
9.112.2.13 rx_period . . . 222
9.113 vtss_oam_voe_ccm_lm_conf_t Struct Reference . . . 222
9.113.1 Detailed Description . . . 223
9.113.2 Field Documentation . . . 223
9.113.2.1 enable . . . 223
9.113.2.2 copy_to_cpu . . . 223
9.113.2.3 forward . . . 223
9.113.2.4 count_as_selected . . . 223
9.113.2.5 count . . . 224
9.113.2.6 period . . . 224
9.114 vtss_oam_voe_conf_t Struct Reference . . . 224
9.114.1 Detailed Description . . . 225
9.114.2 Field Documentation . . . 225

9.114.2.1 voe_type . . . 225
9.114.2.2 unicast_mac . . . 225
9.114.2.3 mep_type . . . 225
9.114.2.4 svc_to_path . . . 225
9.114.2.5 svc_to_path_idx_w . . . 226
9.114.2.6 svc_to_path_idx_p . . . 226
9.114.2.7 loop_isdx_w . . . 226
9.114.2.8 loop_isdx_p . . . 226
9.114.2.9 loop_portidx_p . . . 226
9.114.2.10 sdlb_enable . . . 227
9.114.2.11 sdlb_idx . . . 227
9.114.2.12 proc . . . 227
9.114.2.13 generic . . . 227
9.114.2.14 unknown . . . 227
9.114.2.15 ccm . . . 228
9.114.2.16 ccm_lm . . . 228
9.114.2.17 single_ended_lm . . . 228
9.114.2.18 lb . . . 228
9.114.2.19 tst . . . 228
9.114.2.20 dm . . . 229
9.114.2.21 lt . . . 229
9.114.2.22 upmep . . . 229
9.115 vtss_oam_voe_counter_t Struct Reference . . . 229
9.115.1 Detailed Description . . . 230
9.115.2 Field Documentation . . . 230
9.115.2.1 ccm . . . 230
9.115.2.2 lm . . . 230
9.115.2.3 lb . . . 230
9.115.2.4 tst . . . 230
9.115.2.5 sel . . . 231

9.116 vtss_oam_voe_dm_conf_t Struct Reference . . . 231
9.116.1 Detailed Description . . . 231
9.116.2 Field Documentation . . . 231
9.116.2.1 enable_dmm . . . 231
9.116.2.2 enable_1dm . . . 232
9.116.2.3 copy_1dm_to_cpu . . . 232
9.116.2.4 copy_dmm_to_cpu . . . 232
9.116.2.5 copy_dmr_to_cpu . . . 232
9.116.2.6 forward_1dm . . . 232
9.116.2.7 forward_dmm . . . 233
9.116.2.8 forward_dmr . . . 233
9.116.2.9 count_as_selected . . . 233
9.116.2.10 count_as_data . . . 233
9.117 vtss_oam_voe_generic_conf_t Struct Reference . . . 233
9.117.1 Detailed Description . . . 234
9.117.2 Field Documentation . . . 234
9.117.2.1 enable . . . 234
9.117.2.2 copy_to_cpu . . . 234
9.117.2.3 forward . . . 234
9.117.2.4 count_as_selected . . . 235
9.117.2.5 count_as_data . . . 235
9.118 vtss_oam_voe_lb_conf_t Struct Reference . . . 235
9.118.1 Detailed Description . . . 235
9.118.2 Field Documentation . . . 236
9.118.2.1 enable . . . 236
9.118.2.2 copy_lbm_to_cpu . . . 236
9.118.2.3 copy_lbr_to_cpu . . . 236
9.118.2.4 forward_lbm . . . 236
9.118.2.5 forward_lbr . . . 236
9.118.2.6 count_as_selected . . . 237

9.118.2.7 count_as_data . . . 237
9.118.2.8 tx_update_transaction_id . . . 237
9.118.2.9 tx_transaction_id . . . 237
9.118.2.10 rx_transaction_id . . . 237
9.119 vtss_oam_voe_lm_counter_conf_t Struct Reference . . . 238
9.119.1 Detailed Description . . . 238
9.119.2 Field Documentation . . . 238
9.119.2.1 priority_mask . . . 238
9.119.2.2 yellow . . . 238
9.120 vtss_oam_voe_lt_conf_t Struct Reference . . . 238
9.120.1 Detailed Description . . . 239
9.120.2 Field Documentation . . . 239
9.120.2.1 enable . . . 239
9.120.2.2 copy_ltm_to_cpu . . . 239
9.120.2.3 copy_ltr_to_cpu . . . 239
9.120.2.4 forward_ltm . . . 240
9.120.2.5 forward_ltr . . . 240
9.120.2.6 count_as_selected . . . 240
9.120.2.7 count_as_data . . . 240
9.121 vtss_oam_voe_se_lm_conf_t Struct Reference . . . 240
9.121.1 Detailed Description . . . 241
9.121.2 Field Documentation . . . 241
9.121.2.1 enable . . . 241
9.121.2.2 copy_lmm_to_cpu . . . 241
9.121.2.3 copy_lmr_to_cpu . . . 241
9.121.2.4 forward_lmm . . . 242
9.121.2.5 forward_lmr . . . 242
9.121.2.6 count_as_selected . . . 242
9.121.2.7 count_as_data . . . 242
9.121.2.8 count . . . 242

9.122 vtss_oam_voe_tst_conf_t Struct Reference . . . 243
9.122.1 Detailed Description . . . 243
9.122.2 Field Documentation . . . 243
9.122.2.1 enable . . . 243
9.122.2.2 copy_to_cpu . . . 243
9.122.2.3 forward . . . 244
9.122.2.4 count_as_selected . . . 244
9.122.2.5 count_as_data . . . 244
9.122.2.6 tx_seq_no_auto_update . . . 244
9.122.2.7 tx_seq_no . . . 244
9.122.2.8 rx_seq_no . . . 245
9.123 vtss_oam_voe_unknown_conf_t Struct Reference . . . 245
9.123.1 Detailed Description . . . 245
9.123.2 Field Documentation . . . 245
9.123.2.1 enable . . . 245
9.123.2.2 copy_to_cpu . . . 246
9.123.2.3 count_as_selected . . . 246
9.123.2.4 count_as_data . . . 246
9.124 vtss_oam_voe_up_mep_conf_t Struct Reference . . . 246
9.124.1 Detailed Description . . . 246
9.124.2 Field Documentation . . . 247
9.124.2.1 discard_rx . . . 247
9.124.2.2 loopback . . . 247
9.124.2.3 port . . . 247
9.125 vtss_oam_vop_conf_t Struct Reference . . . 247
9.125.1 Detailed Description . . . 248
9.125.2 Field Documentation . . . 248
9.125.2.1 enable_all_voe . . . 248
9.125.2.2 ccm_lm_enable_rx_fcf_in_reserved_field . . . 248
9.125.2.3 down_mep_lmr_proprietary_fcf_use . . . 248

9.125.2.4 common_multicast_dmac . . . 248
9.125.2.5 external_cpu_portmask . . . 249
9.125.2.6 sdlb_cpy_copy_idx . . . 249
9.125.2.7 pdu_type . . . 249
9.126 vtss_oam_vop_extract_conf_t Struct Reference . . . 249
9.126.1 Detailed Description . . . 249
9.126.2 Field Documentation . . . 250
9.126.2.1 to_front . . . 250
9.126.2.2 rx_queue . . . 250
9.127 vtss_oam_vop_generic_opcode_conf_t Struct Reference . . . 250
9.127.1 Detailed Description . . . 250
9.127.2 Field Documentation . . . 251
9.127.2.1 opcode . . . 251
9.127.2.2 check_dmac . . . 251
9.127.2.3 extract . . . 251
9.128 vtss_oam_vop_pdu_type_conf_t Struct Reference . . . 251
9.128.1 Detailed Description . . . 252
9.128.2 Field Documentation . . . 252
9.128.2.1 generic . . . 252
9.128.2.2 ccm . . . 252
9.128.2.3 ccm_lm . . . 253
9.128.2.4 lt . . . 253
9.128.2.5 dmm . . . 253
9.128.2.6 dmr . . . 253
9.128.2.7 lmm . . . 253
9.128.2.8 lmr . . . 254
9.128.2.9 lbm . . . 254
9.128.2.10 lbr . . . 254
9.128.2.11 err . . . 254
9.128.2.12 other . . . 254

9.129 vtss_os_timestamp_t Struct Reference . . . 255
9.129.1 Detailed Description . . . 255
9.129.2 Field Documentation . . . 255
9.129.2.1 hw_cnt . . . 255
9.130 vtss_packet_dma_conf_t Struct Reference . . . 255
9.130.1 Detailed Description . . . 255
9.130.2 Field Documentation . . . 256
9.130.2.1 dma_enable . . . 256
9.131 vtss_packet_frame_info_t Struct Reference . . . 256
9.131.1 Detailed Description . . . 256
9.131.2 Field Documentation . . . 256
9.131.2.1 port_no . . . 256
9.131.2.2 vid . . . 257
9.131.2.3 port_tx . . . 257
9.131.2.4 aggr_rx_disable . . . 257
9.131.2.5 aggr_tx_disable . . . 257
9.132 vtss_packet_port_filter_t Struct Reference . . . 257
9.132.1 Detailed Description . . . 258
9.132.2 Field Documentation . . . 258
9.132.2.1 filter . . . 258
9.132.2.2 tpid . . . 258
9.133 vtss_packet_port_info_t Struct Reference . . . 258
9.133.1 Detailed Description . . . 259
9.133.2 Field Documentation . . . 259
9.133.2.1 port_no . . . 259
9.133.2.2 vid . . . 259
9.133.2.3 aggr_rx_disable . . . 259
9.133.2.4 aggr_tx_disable . . . 260
9.134 vtss_packet_rx_conf_t Struct Reference . . . 260
9.134.1 Detailed Description . . . 260

9.134.2 Field Documentation . . . 260
9.134.2.1 queue . . . 260
9.134.2.2 reg . . . 261
9.134.2.3 map . . . 261
9.134.2.4 grp_map . . . 261
9.134.2.5 shaper_rate . . . 261
9.135 vtss_packet_rx_header_t Struct Reference . . . 261
9.135.1 Detailed Description . . . 262
9.135.2 Field Documentation . . . 262
9.135.2.1 length . . . 262
9.135.2.2 port_no . . . 262
9.135.2.3 queue_mask . . . 262
9.135.2.4 learn . . . 263
9.135.2.5 arrived_tagged . . . 263
9.135.2.6 tag . . . 263
9.136 vtss_packet_rx_info_t Struct Reference . . . 263
9.136.1 Detailed Description . . . 264
9.136.2 Field Documentation . . . 264
9.136.2.1 hints . . . 265
9.136.2.2 length . . . 265
9.136.2.3 port_no . . . 265
9.136.2.4 glag_no . . . 266
9.136.2.5 tag_type . . . 266
9.136.2.6 tag . . . 267
9.136.2.7 stripped_tag . . . 267
9.136.2.8 xtr_qu_mask . . . 268
9.136.2.9 cos . . . 268
9.136.2.10 acl_hit . . . 268
9.136.2.11 acl_idx . . . 269
9.136.2.12 sw_tstamp . . . 269

9.136.2.13 tstamp_id . . . 269
9.136.2.14 tstamp_id_decoded . . . 270
9.136.2.15 hw_tstamp . . . 270
9.136.2.16 hw_tstamp_decoded . . . 270
9.136.2.17 sflow_type . . . 271
9.136.2.18 sflow_port_no . . . 271
9.136.2.19 oam_info . . . 272
9.136.2.20 oam_info_decoded . . . 272
9.136.2.21 isdx . . . 272
9.137 vtss_packet_rx_meta_t Struct Reference . . . 273
9.137.1 Detailed Description . . . 273
9.137.2 Field Documentation . . . 273
9.137.2.1 no_wait . . . 273
9.137.2.2 chip_no . . . 274
9.137.2.3 xtr_qu . . . 274
9.137.2.4 etype . . . 275
9.137.2.5 fcs . . . 275
9.137.2.6 sw_tstamp . . . 276
9.137.2.7 length . . . 276
9.138 vtss_packet_rx_port_conf_t Struct Reference . . . 277
9.138.1 Detailed Description . . . 277
9.138.2 Field Documentation . . . 277
9.138.2.1 ipmc_ctrl_reg . . . 277
9.138.2.2 igmp_reg . . . 278
9.138.2.3 mld_reg . . . 278
9.138.2.4 bpdu_reg . . . 278
9.138.2.5 garp_reg . . . 278
9.139 vtss_packet_rx_queue_conf_t Struct Reference . . . 278
9.139.1 Detailed Description . . . 279
9.139.2 Field Documentation . . . 279

9.139.2.1 size . . . 279
9.139.2.2 npi . . . 279
9.140 vtss_packet_rx_queue_map_t Struct Reference . . . 279
9.140.1 Detailed Description . . . 280
9.140.2 Field Documentation . . . 280
9.140.2.1 bpdu_queue . . . 280
9.140.2.2 garp_queue . . . 280
9.140.2.3 learn_queue . . . 280
9.140.2.4 igmp_queue . . . 281
9.140.2.5 ipmc_ctrl_queue . . . 281
9.140.2.6 mac_vid_queue . . . 281
9.140.2.7 stack_queue . . . 281
9.140.2.8 sflow_queue . . . 281
9.140.2.9 lrn_all_queue . . . 282
9.141 vtss_packet_rx_queue_npi_conf_t Struct Reference . . . 282
9.141.1 Detailed Description . . . 282
9.141.2 Field Documentation . . . 282
9.141.2.1 enable . . . 282
9.142 vtss_packet_rx_reg_t Struct Reference . . . 283
9.142.1 Detailed Description . . . 283
9.142.2 Field Documentation . . . 283
9.142.2.1 bpdu_cpu_only . . . 283
9.142.2.2 garp_cpu_only . . . 283
9.142.2.3 ipmc_ctrl_cpu_copy . . . 284
9.142.2.4 igmp_cpu_only . . . 284
9.142.2.5 mld_cpu_only . . . 284
9.143 vtss_packet_tx_ifh_t Struct Reference . . . 284
9.143.1 Detailed Description . . . 284
9.143.2 Field Documentation . . . 285
9.143.2.1 length . . . 285

9.143.2.2 ifh . . . 285
9.144 vtss_packet_tx_info_t Struct Reference . . . 285
9.144.1 Detailed Description . . . 286
9.144.2 Field Documentation . . . 286
9.144.2.1 switch_frm . . . 286
9.144.2.2 dst_port_mask . . . 287
9.144.2.3 frm_len . . . 287
9.144.2.4 tag . . . 288
9.144.2.5 aggr_code . . . 288
9.144.2.6 cos . . . 289
9.144.2.7 ptp_action . . . 289
9.144.2.8 ptp_id . . . 290
9.144.2.9 ptp_timestamp . . . 290
9.144.2.10 latch_timestamp . . . 291
9.144.2.11 oam_type . . . 291
9.144.2.12 isdx . . . 292
9.144.2.13 isdx_dont_use . . . 292
9.144.2.14 dp . . . 293
9.144.2.15 masquerade_port . . . 293
9.144.2.16 pdu_offset . . . 294
9.144.2.17 afi_id . . . 294
9.145 vtss_phy_10g_fifo_sync_t Struct Reference . . . 295
9.145.1 Detailed Description . . . 295
9.145.2 Field Documentation . . . 295
9.145.2.1 bypass_in_api . . . 295
9.145.2.2 skip_rev_check . . . 296
9.145.2.3 pr . . . 296
9.146 vtss_phy_aneg_t Struct Reference . . . 296
9.146.1 Detailed Description . . . 296
9.146.2 Field Documentation . . . 297

9.146.2.1 speed_10m_hdx . . . 297
9.146.2.2 speed_10m_fdx . . . 297
9.146.2.3 speed_100m_hdx . . . 297
9.146.2.4 speed_100m_fdx . . . 297
9.146.2.5 speed_1g_fdx . . . 297
9.146.2.6 speed_1g_hdx . . . 298
9.146.2.7 symmetric_pause . . . 298
9.146.2.8 asymmetric_pause . . . 298
9.146.2.9 tx_remote_fault . . . 298
9.147 vtss_phy_clock_conf_t Struct Reference . . . 298
9.147.1 Detailed Description . . . 299
9.147.2 Field Documentation . . . 299
9.147.2.1 src . . . 299
9.147.2.2 freq . . . 299
9.147.2.3 squelch . . . 299
9.148 vtss_phy_conf_1g_t Struct Reference . . . 300
9.148.1 Detailed Description . . . 300
9.148.2 Field Documentation . . . 300
9.148.2.1 cfg . . . 300
9.148.2.2 val . . . 300
9.148.2.3 master . . . 301
9.149 vtss_phy_conf_t Struct Reference . . . 301
9.149.1 Detailed Description . . . 301
9.149.2 Field Documentation . . . 301
9.149.2.1 mode . . . 301
9.149.2.2 forced . . . 302
9.149.2.3 aneg . . . 302
9.149.2.4 mdi . . . 302
9.149.2.5 flf . . . 302
9.149.2.6 sigdet . . . 302

9.149.2.7 unidir . . . 303
9.149.2.8 mac_if_pcs . . . 303
9.149.2.9 media_if_pcs . . . 303
9.149.2.10 force_ams_sel . . . 303
9.149.2.11 skip_coma . . . 303
9.150 vtss_phy_daisy_chain_conf_t Struct Reference . . . 304
9.150.1 Detailed Description . . . 304
9.150.2 Field Documentation . . . 304
9.150.2.1 spi_daisy_input . . . 304
9.150.2.2 spi_daisy_output . . . 304
9.151 vtss_phy_eee_conf_t Struct Reference . . . 304
9.151.1 Detailed Description . . . 305
9.151.2 Field Documentation . . . 305
9.151.2.1 eee_mode . . . 305
9.151.2.2 eee_ena_phy . . . 305
9.152 vtss_phy_enhanced_led_control_t Struct Reference . . . 305
9.152.1 Detailed Description . . . 306
9.152.2 Field Documentation . . . 306
9.152.2.1 ser_led_output_1 . . . 306
9.152.2.2 ser_led_output_2 . . . 306
9.152.2.3 ser_led_frame_rate . . . 306
9.152.2.4 ser_led_select . . . 307
9.153 vtss_phy_forced_t Struct Reference . . . 307
9.153.1 Detailed Description . . . 307
9.153.2 Field Documentation . . . 307
9.153.2.1 speed . . . 307
9.153.2.2 fdx . . . 308
9.154 vtss_phy_led_mode_select_t Struct Reference . . . 308
9.154.1 Detailed Description . . . 308
9.154.2 Field Documentation . . . 308

9.154.2.1 mode . . . 308
9.154.2.2 number . . . 309
9.155 vtss_phy_loopback_t Struct Reference . . . 309
9.155.1 Detailed Description . . . 309
9.155.2 Field Documentation . . . 309
9.155.2.1 far_end_enable . . . 309
9.155.2.2 near_end_enable . . . 310
9.155.2.3 connector_enable . . . 310
9.155.2.4 mac_serdes_input_enable . . . 310
9.155.2.5 mac_serdes_facility_enable . . . 310
9.155.2.6 mac_serdes_equipment_enable . . . 310
9.155.2.7 media_serdes_input_enable . . . 311
9.155.2.8 media_serdes_facility_enable . . . 311
9.155.2.9 media_serdes_equipment_enable . . . 311
9.156 vtss_phy_ltc_freq_synth_s Struct Reference . . . 311
9.156.1 Detailed Description . . . 311
9.156.2 Field Documentation . . . 312
9.156.2.1 enable . . . 312
9.156.2.2 high_duty_cycle . . . 312
9.156.2.3 low_duty_cycle . . . 312
9.157 vtss_phy_mac_serd_pcs_cntl_t Struct Reference . . . 312
9.157.1 Detailed Description . . . 313
9.157.2 Field Documentation . . . 313
9.157.2.1 disable . . . 313
9.157.2.2 restart . . . 313
9.157.2.3 pd_enable . . . 314
9.157.2.4 aneg_restart . . . 314
9.157.2.5 force_adv_ability . . . 314
9.157.2.6 sgmii_in_pre . . . 314
9.157.2.7 sgmii_out_pre . . . 314

9.157.2.8 serdes_aneg_ena . . . 315
9.157.2.9 serdes_pol_inv_in . . . 315
9.157.2.10 serdes_pol_inv_out . . . 315
9.157.2.11 fast_link_stat_ena . . . 315
9.157.2.12 inhibit_odd_start . . . 315
9.158 vtss_phy_media_serd_pcs_cntl_t Struct Reference . . . 316
9.158.1 Detailed Description . . . 316
9.158.2 Field Documentation . . . 316
9.158.2.1 remote_fault . . . 316
9.158.2.2 aneg_pd_detect . . . 316
9.158.2.3 force_adv_ability . . . 317
9.158.2.4 serdes_pol_inv_in . . . 317
9.158.2.5 serdes_pol_inv_out . . . 317
9.158.2.6 inhibit_odd_start . . . 317
9.158.2.7 force_hls . . . 317
9.158.2.8 force_fefi . . . 318
9.158.2.9 force_fefi_value . . . 318
9.159 vtss_phy_power_conf_t Struct Reference . . . 318
9.159.1 Detailed Description . . . 318
9.159.2 Field Documentation . . . 318
9.159.2.1 mode . . . 319
9.160 vtss_phy_power_status_t Struct Reference . . . 319
9.160.1 Detailed Description . . . 319
9.160.2 Field Documentation . . . 319
9.160.2.1 level . . . 319
9.161 vtss_phy_reset_conf_t Struct Reference . . . 320
9.161.1 Detailed Description . . . 320
9.161.2 Field Documentation . . . 320
9.161.2.1 mac_if . . . 320
9.161.2.2 media_if . . . 320

9.161.2.3 rgmii . . . 321
9.161.2.4 tbi . . . 321
9.161.2.5 force . . . 321
9.161.2.6 pkt_mode . . . 321
9.161.2.7 i_cpu_en . . . 321
9.162 vtss_phy_rgmii_conf_t Struct Reference . . . 322
9.162.1 Detailed Description . . . 322
9.162.2 Field Documentation . . . 322
9.162.2.1 rx_clk_skew_ps . . . 322
9.162.2.2 tx_clk_skew_ps . . . 322
9.163 vtss_phy_statistic_t Struct Reference . . . 322
9.163.1 Detailed Description . . . 323
9.163.2 Field Documentation . . . 323
9.163.2.1 cu_good . . . 323
9.163.2.2 cu_bad . . . 323
9.163.2.3 serdes_tx_good . . . 323
9.163.2.4 serdes_tx_bad . . . 324
9.163.2.5 rx_err_cnt_base_tx . . . 324
9.163.2.6 media_mac_serdes_good . . . 324
9.163.2.7 media_mac_serdes_crc . . . 324
9.164 vtss_phy_status_1g_t Struct Reference . . . 325
9.164.1 Detailed Description . . . 325
9.164.2 Field Documentation . . . 325
9.164.2.1 master_cfg_fault . . . 325
9.164.2.2 master . . . 325
9.165 vtss_phy_tbi_conf_t Struct Reference . . . 325
9.165.1 Detailed Description . . . 326
9.165.2 Field Documentation . . . 326
9.165.2.1 aneg_enable . . . 326
9.166 vtss_phy_timestamp_t Struct Reference . . . 326

9.166.1 Detailed Description . . . 326
9.166.2 Field Documentation . . . 327
9.166.2.1 high . . . 327
9.166.2.2 low . . . 327
9.166.2.3 seconds . . . 327
9.166.2.4 nanoseconds . . . 327
9.167 vtss_phy_ts_ach_conf_t Struct Reference . . . 327
9.167.1 Detailed Description . . . 328
9.167.2 Field Documentation . . . 328
9.167.2.1 value [1/2] . . . 328
9.167.2.2 mask [1/2] . . . 328
9.167.2.3 version . . . 329
9.167.2.4 value [2/2] . . . 329
9.167.2.5 mask [2/2] . . . 329
9.167.2.6 channel_type . . . 329
9.167.2.7 proto_id . . . 329
9.167.2.8 comm_opt . . . 329
9.168 vtss_phy_ts_alt_clock_mode_s Struct Reference . . . 330
9.168.1 Detailed Description . . . 330
9.168.2 Field Documentation . . . 330
9.168.2.1 pps_ls_lpbk . . . 330
9.168.2.2 ls_lpbk . . . 330
9.168.2.3 ls_pps_lpbk . . . 331
9.169 vtss_phy_ts_eng_init_conf_t Struct Reference . . . 331
9.169.1 Detailed Description . . . 331
9.169.2 Field Documentation . . . 331
9.169.2.1 eng_used . . . 331
9.169.2.2 encap_type . . . 332
9.169.2.3 flow_match_mode . . . 332
9.169.2.4 flow_st_index . . . 332

9.169.2.5 flow_end_index . . . 332
9.170 vtss_phy_ts_engine_action_t Struct Reference . . . 332
9.170.1 Detailed Description . . . 333
9.170.2 Field Documentation . . . 333
9.170.2.1 action_ptp . . . 333
9.170.2.2 action_gen . . . 333
9.170.2.3 ptp_conf . . . 334
9.170.2.4 oam_conf . . . 334
9.170.2.5 gen_conf . . . 334
9.170.2.6 action . . . 334
9.171 vtss_phy_ts_engine_flow_conf_t Struct Reference . . . 334
9.171.1 Detailed Description . . . 335
9.171.2 Field Documentation . . . 335
9.171.2.1 eng_mode . . . 335
9.171.2.2 channel_map . . . 335
9.171.2.3 ptp . . . 336
9.171.2.4 oam . . . 336
9.171.2.5 gen . . . 336
9.171.2.6 flow_conf . . . 336
9.172 vtss_phy_ts_eth_conf_t Struct Reference . . . 336
9.172.1 Detailed Description . . . 337
9.172.2 Field Documentation . . . 338
9.172.2.1 pbb_en . . . 338
9.172.2.2 etype . . . 338
9.172.2.3 tpid . . . 338
9.172.2.4 comm_opt . . . 338
9.172.2.5 flow_en . . . 338
9.172.2.6 addr_match_mode . . . 339
9.172.2.7 addr_match_select . . . 339
9.172.2.8 mac_addr . . . 339

9.172.2.9 vlan_check . . . 339
9.172.2.10 num_tag . . . 339
9.172.2.11 outer_tag_type . . . 340
9.172.2.12 inner_tag_type . . . 340
9.172.2.13 tag_range_mode . . . 340
9.172.2.14 upper . . . 340
9.172.2.15 lower . . . 340
9.172.2.16 range [1/2] . . . 341
9.172.2.17 val [1/2] . . . 341
9.172.2.18 mask [1/2] . . . 341
9.172.2.19 value [1/2] . . . 341
9.172.2.20 outer_tag . . . 341
9.172.2.21 range [2/2] . . . 341
9.172.2.22 value [2/2] . . . 341
9.172.2.23 val [2/2] . . . 342
9.172.2.24 mask [2/2] . . . 342
9.172.2.25 _tag . . . 342
9.172.2.26 inner_tag . . . 342
9.172.2.27 flow_opt . . . 342
9.173 vtss_phy_ts_fifo_conf_t Struct Reference . . . 342
9.173.1 Detailed Description . . . 343
9.173.2 Field Documentation . . . 343
9.173.2.1 detect_only . . . 343
9.173.2.2 eng_recov . . . 343
9.173.2.3 eng_minE . . . 343
9.173.2.4 skip_rev_check . . . 344
9.174 vtss_phy_ts_fifo_sig_t Struct Reference . . . 344
9.174.1 Detailed Description . . . 344
9.174.2 Field Documentation . . . 344
9.174.2.1 sig_mask . . . 344

9.174.2.2 msg_type . . . 345
9.174.2.3 domain_num . . . 345
9.174.2.4 src_port_identity . . . 345
9.174.2.5 sequence_id . . . 345
9.174.2.6 dest_ip . . . 345
9.174.2.7 src_ip . . . 346
9.174.2.8 dest_mac . . . 346
9.175 vtss_phy_ts_gen_conf_t Struct Reference . . . 346
9.175.1 Detailed Description . . . 346
9.175.2 Field Documentation . . . 347
9.175.2.1 flow_offset . . . 347
9.175.2.2 next_prot_offset . . . 347
9.175.2.3 comm_opt . . . 347
9.175.2.4 flow_en . . . 347
9.175.2.5 data . . . 347
9.175.2.6 mask . . . 348
9.175.2.7 flow_opt . . . 348
9.176 vtss_phy_ts_generic_action_t Struct Reference . . . 348
9.176.1 Detailed Description . . . 348
9.176.2 Field Documentation . . . 348
9.176.2.1 enable . . . 349
9.176.2.2 channel_map . . . 349
9.176.2.3 flow_id . . . 349
9.176.2.4 data . . . 349
9.176.2.5 mask . . . 349
9.176.2.6 ts_type . . . 350
9.176.2.7 ts_offset . . . 350
9.177 vtss_phy_ts_generic_flow_conf_t Struct Reference . . . 350
9.177.1 Detailed Description . . . 350
9.177.2 Field Documentation . . . 350

9.177.2.1 eth1_opt . . . 351
9.177.2.2 gen_opt . . . 351
9.178 vtss_phy_ts_ietf_mpls_ach_oam_conf_t Struct Reference . . . 351
9.178.1 Detailed Description . . . 351
9.178.2 Field Documentation . . . 351
9.178.2.1 delaym_type . . . 352
9.178.2.2 ts_format . . . 352
9.178.2.3 ds . . . 352
9.179 vtss_phy_ts_init_conf_t Struct Reference . . . 352
9.179.1 Detailed Description . . . 353
9.179.2 Field Documentation . . . 353
9.179.2.1 clk_freq . . . 353
9.179.2.2 clk_src . . . 353
9.179.2.3 rx_ts_pos . . . 353
9.179.2.4 rx_ts_len . . . 353
9.179.2.5 tx_fifo_mode . . . 354
9.179.2.6 tx_ts_len . . . 354
9.179.2.7 tx_fifo_spi_conf . . . 354
9.179.2.8 tx_fifo_hi_clk_cycs . . . 354
9.179.2.9 tx_fifo_lo_clk_cycs . . . 354
9.179.2.10 tc_op_mode . . . 355
9.179.2.11 auto_clear_ls . . . 355
9.179.2.12 macsec_ena . . . 355
9.179.2.13 chk_ing_modified . . . 355
9.179.2.14 one_step_txfifo . . . 355
9.180 vtss_phy_ts_ip_conf_t Struct Reference . . . 356
9.180.1 Detailed Description . . . 356
9.180.2 Field Documentation . . . 356
9.180.2.1 ip_mode . . . 356
9.180.2.2 sport_val . . . 357

9.180.2.3 sport_mask . . . 357
9.180.2.4 dport_val . . . 357
9.180.2.5 dport_mask . . . 357
9.180.2.6 comm_opt . . . 357
9.180.2.7 flow_en . . . 358
9.180.2.8 match_mode . . . 358
9.180.2.9 addr . . . 358
9.180.2.10 mask . . . 358
9.180.2.11 ipv4 . . . 358
9.180.2.12 ipv6 . . . 359
9.180.2.13 ip_addr . . . 359
9.180.2.14 flow_opt . . . 359
9.181 vtss_phy_ts_mpls_conf_t Struct Reference . . . 359
9.181.1 Detailed Description . . . 360
9.181.2 Field Documentation . . . 360
9.181.2.1 cw_en . . . 360
9.181.2.2 comm_opt . . . 360
9.181.2.3 flow_en . . . 360
9.181.2.4 stack_depth . . . 360
9.181.2.5 stack_ref_point . . . 361
9.181.2.6 top . . . 361
9.181.2.7 frst_lvl_after_top . . . 361
9.181.2.8 snd_lvl_after_top . . . 361
9.181.2.9 thrd_lvl_after_top . . . 361
9.181.2.10 top_down . . . 362
9.181.2.11 end . . . 362
9.181.2.12 frst_lvl_before_end . . . 362
9.181.2.13 snd_lvl_before_end . . . 362
9.181.2.14 thrd_lvl_before_end . . . 362
9.181.2.15 bottom_up . . . 362

9.181.2.16 stack_level . . . 363
9.181.2.17 flow_opt . . . 363
9.182 vtss_phy_ts_mpls_lvl_rng_t Struct Reference . . . 363
9.182.1 Detailed Description . . . 363
9.182.2 Field Documentation . . . 363
9.182.2.1 lower . . . 363
9.182.2.2 upper . . . 364
9.183 vtss_phy_ts_nphase_status_t Struct Reference . . . 364
9.183.1 Detailed Description . . . 364
9.183.2 Field Documentation . . . 364
9.183.2.1 enable . . . 364
9.183.2.2 CALIB_ERR . . . 365
9.183.2.3 CALIB_DONE . . . 365
9.184 vtss_phy_ts_oam_engine_action_t Struct Reference . . . 365
9.184.1 Detailed Description . . . 365
9.184.2 Field Documentation . . . 366
9.184.2.1 enable . . . 366
9.184.2.2 y1731_en . . . 366
9.184.2.3 channel_map . . . 366
9.184.2.4 version . . . 366
9.184.2.5 y1731_oam_conf . . . 366
9.184.2.6 ietf_oam_conf . . . 367
9.184.2.7 oam_conf . . . 367
9.185 vtss_phy_ts_oam_engine_flow_conf_t Struct Reference . . . 367
9.185.1 Detailed Description . . . 367
9.185.2 Field Documentation . . . 367
9.185.2.1 eth1_opt . . . 368
9.185.2.2 eth2_opt . . . 368
9.185.2.3 mpls_opt . . . 368
9.185.2.4 ach_opt . . . 368

9.186 vtss_phy_ts_pps_config_s Struct Reference . . . 368
9.186.1 Detailed Description . . . 369
9.186.2 Field Documentation . . . 369
9.186.2.1 pps_width_adj . . . 369
9.186.2.2 pps_offset . . . 369
9.186.2.3 pps_output_enable . . . 369
9.187 vtss_phy_ts_ptp_conf_t Struct Reference . . . 370
9.187.1 Detailed Description . . . 370
9.187.2 Field Documentation . . . 370
9.187.2.1 range_en . . . 370
9.187.2.2 val . . . 370
9.187.2.3 mask . . . 371
9.187.2.4 value . . . 371
9.187.2.5 upper . . . 371
9.187.2.6 lower . . . 371
9.187.2.7 range . . . 371
9.187.2.8 domain . . . 371
9.188 vtss_phy_ts_ptp_engine_action_t Struct Reference . . . 372
9.188.1 Detailed Description . . . 372
9.188.2 Field Documentation . . . 372
9.188.2.1 enable . . . 372
9.188.2.2 channel_map . . . 372
9.188.2.3 ptp_conf . . . 373
9.188.2.4 clk_mode . . . 373
9.188.2.5 delaym_type . . . 373
9.188.2.6 cf_update . . . 373
9.189 vtss_phy_ts_ptp_engine_flow_conf_t Struct Reference . . . 373
9.189.1 Detailed Description . . . 374
9.189.2 Field Documentation . . . 374
9.189.2.1 eth1_opt . . . 374

9.189.2.2 eth2_opt . . . 374
9.189.2.3 ip1_opt . . . 374
9.189.2.4 ip2_opt . . . 375
9.189.2.5 mpls_opt . . . 375
9.189.2.6 ach_opt . . . 375
9.190 vtss_phy_ts_sertod_conf_t Struct Reference . . . 375
9.190.1 Detailed Description . . . 375
9.190.2 Field Documentation . . . 376
9.190.2.1 ip_enable . . . 376
9.190.2.2 op_enable . . . 376
9.190.2.3 ls_inv . . . 376
9.191 vtss_phy_ts_stats_t Struct Reference . . . 376
9.191.1 Detailed Description . . . 377
9.191.2 Field Documentation . . . 377
9.191.2.1 ingr_pream_shrink_err . . . 377
9.191.2.2 egr_pream_shrink_err . . . 377
9.191.2.3 ingr_fcs_err . . . 377
9.191.2.4 egr_fcs_err . . . 378
9.191.2.5 ingr_frm_mod_cnt . . . 378
9.191.2.6 egr_frm_mod_cnt . . . 378
9.191.2.7 ts_fifo_tx_cnt . . . 378
9.191.2.8 ts_fifo_drop_cnt . . . 378
9.192 vtss_phy_ts_y1731_oam_conf_t Struct Reference . . . 379
9.192.1 Detailed Description . . . 379
9.192.2 Field Documentation . . . 379
9.192.2.1 range_en . . . 379
9.192.2.2 delaym_type . . . 379
9.192.2.3 val . . . 380
9.192.2.4 mask . . . 380
9.192.2.5 value . . . 380

9.192.2.6 upper . . . 380
9.192.2.7 lower . . . 380
9.192.2.8 range . . . 380
9.192.2.9 meg_level . . . 381
9.193 vtss_phy_type_t Struct Reference . . . 381
9.193.1 Detailed Description . . . 381
9.193.2 Field Documentation . . . 381
9.193.2.1 part_number . . . 381
9.193.2.2 revision . . . 382
9.193.2.3 port_cnt . . . 382
9.193.2.4 channel_id . . . 382
9.193.2.5 base_port_no . . . 382
9.193.2.6 phy_api_base_no . . . 382
9.194 vtss_phy_veriphy_result_t Struct Reference . . . 383
9.194.1 Detailed Description . . . 383
9.194.2 Field Documentation . . . 383
9.194.2.1 link . . . 383
9.194.2.2 status . . . 383
9.194.2.3 length . . . 384
9.195 vtss_phy_wol_conf_t Struct Reference . . . 384
9.195.1 Detailed Description . . . 384
9.195.2 Field Documentation . . . 384
9.195.2.1 secure_on_enable . . . 384
9.195.2.2 wol_mac . . . 385
9.195.2.3 wol_pass . . . 385
9.195.2.4 wol_passwd_len . . . 385
9.195.2.5 magic_pkt_cnt . . . 385
9.196 vtss_pi_conf_t Struct Reference . . . 385
9.196.1 Detailed Description . . . 386
9.196.2 Field Documentation . . . 386

9.196.2.1 cs_wait_ns . . . 386
9.197 vtss_policer_ext_t Struct Reference . . . 386
9.197.1 Detailed Description . . . 386
9.197.2 Field Documentation . . . 387
9.197.2.1 frame_rate . . . 387
9.197.2.2 flow_control . . . 387
9.198 vtss_policer_t Struct Reference . . . 387
9.198.1 Detailed Description . . . 387
9.198.2 Field Documentation . . . 387
9.198.2.1 level . . . 388
9.198.2.2 rate . . . 388
9.199 vtss_port_bridge_counters_t Struct Reference . . . 388
9.199.1 Detailed Description . . . 388
9.199.2 Field Documentation . . . 388
9.199.2.1 dot1dTpPortInDiscards . . . 389
9.200 vtss_port_clause_37_adv_t Struct Reference . . . 389
9.200.1 Detailed Description . . . 389
9.200.2 Field Documentation . . . 389
9.200.2.1 fdx . . . 389
9.200.2.2 hdx . . . 390
9.200.2.3 symmetric_pause . . . 390
9.200.2.4 asymmetric_pause . . . 390
9.200.2.5 remote_fault . . . 390
9.200.2.6 acknowledge . . . 390
9.200.2.7 next_page . . . 391
9.201 vtss_port_clause_37_control_t Struct Reference . . . 391
9.201.1 Detailed Description . . . 391
9.201.2 Field Documentation . . . 391
9.201.2.1 enable . . . 391
9.201.2.2 advertisement . . . 392

9.202 vtss_port_conf_t Struct Reference . . . 392
9.202.1 Detailed Description . . . 392
9.202.2 Field Documentation . . . 392
9.202.2.1 if_type . . . 393
9.202.2.2 sd_enable . . . 393
9.202.2.3 sd_active_high . . . 393
9.202.2.4 sd_internal . . . 393
9.202.2.5 frame_gaps . . . 393
9.202.2.6 power_down . . . 394
9.202.2.7 speed . . . 394
9.202.2.8 fdx . . . 394
9.202.2.9 flow_control . . . 394
9.202.2.10 max_frame_length . . . 394
9.202.2.11 frame_length_chk . . . 395
9.202.2.12 max_tags . . . 395
9.202.2.13 exc_col_cont . . . 395
9.202.2.14 xaui_rx_lane_flip . . . 395
9.202.2.15 xaui_tx_lane_flip . . . 395
9.202.2.16 loop . . . 396
9.202.2.17 serdes . . . 396
9.203 vtss_port_counters_t Struct Reference . . . 396
9.203.1 Detailed Description . . . 396
9.203.2 Field Documentation . . . 396
9.203.2.1 rmon . . . 397
9.203.2.2 if_group . . . 397
9.203.2.3 ethernet_like . . . 397
9.203.2.4 bridge . . . 397
9.203.2.5 prop . . . 397
9.204 vtss_port_ethernet_like_counters_t Struct Reference . . . 398
9.204.1 Detailed Description . . . 398

9.204.2 Field Documentation . . . 398
9.204.2.1 dot3InPauseFrames . . . 398
9.204.2.2 dot3OutPauseFrames . . . 398
9.205 vtss_port_flow_control_conf_t Struct Reference . . . 398
9.205.1 Detailed Description . . . 399
9.205.2 Field Documentation . . . 399
9.205.2.1 obey . . . 399
9.205.2.2 generate . . . 399
9.205.2.3 smac . . . 399
9.205.2.4 pfc . . . 400
9.206 vtss_port_frame_gaps_t Struct Reference . . . 400
9.206.1 Detailed Description . . . 400
9.206.2 Field Documentation . . . 400
9.206.2.1 hdx_gap_1 . . . 400
9.206.2.2 hdx_gap_2 . . . 401
9.206.2.3 fdx_gap . . . 401
9.207 vtss_port_if_group_counters_t Struct Reference . . . 401
9.207.1 Detailed Description . . . 401
9.207.2 Field Documentation . . . 402
9.207.2.1 ifInOctets . . . 402
9.207.2.2 ifInUcastPkts . . . 402
9.207.2.3 ifInMulticastPkts . . . 402
9.207.2.4 ifInBroadcastPkts . . . 402
9.207.2.5 ifInNUcastPkts . . . 402
9.207.2.6 ifInDiscards . . . 403
9.207.2.7 ifInErrors . . . 403
9.207.2.8 ifOutOctets . . . 403
9.207.2.9 ifOutUcastPkts . . . 403
9.207.2.10 ifOutMulticastPkts . . . 403
9.207.2.11 ifOutBroadcastPkts . . . 404

9.207.2.12 ifOutNUcastPkts . . . 404
9.207.2.13 ifOutDiscards . . . 404
9.207.2.14 ifOutErrors . . . 404
9.208 vtss_port_ifh_t Struct Reference . . . 404
9.208.1 Detailed Description . . . 405
9.208.2 Field Documentation . . . 405
9.208.2.1 ena_inj_header . . . 405
9.208.2.2 ena_xtr_header . . . 405
9.209 vtss_port_map_t Struct Reference . . . 405
9.209.1 Detailed Description . . . 406
9.209.2 Field Documentation . . . 406
9.209.2.1 chip_port . . . 406
9.209.2.2 chip_no . . . 406
9.209.2.3 miim_controller . . . 406
9.209.2.4 miim_addr . . . 407
9.209.2.5 miim_chip_no . . . 407
9.210 vtss_port_proprietary_counters_t Struct Reference . . . 407
9.210.1 Detailed Description . . . 407
9.210.2 Field Documentation . . . 407
9.210.2.1 rx_prio . . . 408
9.210.2.2 tx_prio . . . 408
9.211 vtss_port_rmon_counters_t Struct Reference . . . 408
9.211.1 Detailed Description . . . 409
9.211.2 Field Documentation . . . 409
9.211.2.1 rx_etherStatsDropEvents . . . 409
9.211.2.2 rx_etherStatsOctets . . . 409
9.211.2.3 rx_etherStatsPkts . . . 409
9.211.2.4 rx_etherStatsBroadcastPkts . . . 409
9.211.2.5 rx_etherStatsMulticastPkts . . . 410
9.211.2.6 rx_etherStatsCRCAlignErrors . . . 410

9.211.2.7 rx_etherStatsUndersizePkts . . . 410
9.211.2.8 rx_etherStatsOversizePkts . . . 410
9.211.2.9 rx_etherStatsFragments . . . 410
9.211.2.10 rx_etherStatsJabbers . . . 411
9.211.2.11 rx_etherStatsPkts64Octets . . . 411
9.211.2.12 rx_etherStatsPkts65to127Octets . . . 411
9.211.2.13 rx_etherStatsPkts128to255Octets . . . 411
9.211.2.14 rx_etherStatsPkts256to511Octets . . . 411
9.211.2.15 rx_etherStatsPkts512to1023Octets . . . 412
9.211.2.16 rx_etherStatsPkts1024to1518Octets . . . 412
9.211.2.17 rx_etherStatsPkts1519toMaxOctets . . . 412
9.211.2.18 tx_etherStatsDropEvents . . . 412
9.211.2.19 tx_etherStatsOctets . . . 412
9.211.2.20 tx_etherStatsPkts . . . 413
9.211.2.21 tx_etherStatsBroadcastPkts . . . 413
9.211.2.22 tx_etherStatsMulticastPkts . . . 413
9.211.2.23 tx_etherStatsCollisions . . . 413
9.211.2.24 tx_etherStatsPkts64Octets . . . 413
9.211.2.25 tx_etherStatsPkts65to127Octets . . . 414
9.211.2.26 tx_etherStatsPkts128to255Octets . . . 414
9.211.2.27 tx_etherStatsPkts256to511Octets . . . 414
9.211.2.28 tx_etherStatsPkts512to1023Octets . . . 414
9.211.2.29 tx_etherStatsPkts1024to1518Octets . . . 414
9.211.2.30 tx_etherStatsPkts1519toMaxOctets . . . 415
9.212 vtss_port_serdes_conf_t Struct Reference . . . 415
9.212.1 Detailed Description . . . 415
9.212.2 Field Documentation . . . 415
9.212.2.1 sfp_dac . . . 415
9.213 vtss_port_sgmii_aneg_t Struct Reference . . . 416
9.213.1 Detailed Description . . . 416

9.213.2 Field Documentation . . . 416
9.213.2.1 link . . . 416
9.213.2.2 fdx . . . 416
9.213.2.3 hdx . . . 417
9.213.2.4 speed_10M . . . 417
9.213.2.5 speed_100M . . . 417
9.213.2.6 speed_1G . . . 417
9.213.2.7 aneg_complete . . . 417
9.214 vtss_port_status_t Struct Reference . . . 418
9.214.1 Detailed Description . . . 418
9.214.2 Field Documentation . . . 418
9.214.2.1 link_down . . . 418
9.214.2.2 link . . . 418
9.214.2.3 speed . . . 419
9.214.2.4 fdx . . . 419
9.214.2.5 remote_fault . . . 419
9.214.2.6 aneg_complete . . . 419
9.214.2.7 unidirectional_ability . . . 419
9.214.2.8 aneg . . . 420
9.214.2.9 mdi_cross . . . 420
9.214.2.10 fiber . . . 420
9.214.2.11 copper . . . 420
9.215 vtss_qce_action_t Struct Reference . . . 420
9.215.1 Detailed Description . . . 421
9.215.2 Field Documentation . . . 421
9.215.2.1 prio_enable . . . 421
9.215.2.2 prio . . . 421
9.215.2.3 dp_enable . . . 421
9.215.2.4 dp . . . 422
9.215.2.5 dscp_enable . . . 422

9.215.2.6 dscp . . . 422
9.215.2.7 pcp_dei_enable . . . 422
9.215.2.8 pcp . . . 422
9.215.2.9 dei . . . 423
9.215.2.10 policy_no_enable . . . 423
9.215.2.11 policy_no . . . 423
9.216 vtss_qce_frame_etype_t Struct Reference . . . 423
9.216.1 Detailed Description . . . 423
9.216.2 Field Documentation . . . 424
9.216.2.1 etype . . . 424
9.216.2.2 data . . . 424
9.217 vtss_qce_frame_ipv4_t Struct Reference . . . 424
9.217.1 Detailed Description . . . 424
9.217.2 Field Documentation . . . 425
9.217.2.1 fragment . . . 425
9.217.2.2 dscp . . . 425
9.217.2.3 proto . . . 425
9.217.2.4 sip . . . 425
9.217.2.5 dip . . . 425
9.217.2.6 sport . . . 426
9.217.2.7 dport . . . 426
9.218 vtss_qce_frame_ipv6_t Struct Reference . . . 426
9.218.1 Detailed Description . . . 426
9.218.2 Field Documentation . . . 426
9.218.2.1 dscp . . . 427
9.218.2.2 proto . . . 427
9.218.2.3 sip . . . 427
9.218.2.4 dip . . . 427
9.218.2.5 sport . . . 427
9.218.2.6 dport . . . 428

9.219 vtss_qce_frame_llc_t Struct Reference . . . 428
9.219.1 Detailed Description . . . 428
9.219.2 Field Documentation . . . 428
9.219.2.1 data . . . 428
9.220 vtss_qce_frame_snap_t Struct Reference . . . 429
9.220.1 Detailed Description . . . 429
9.220.2 Field Documentation . . . 429
9.220.2.1 data . . . 429
9.221 vtss_qce_key_t Struct Reference . . . 429
9.221.1 Detailed Description . . . 430
9.221.2 Field Documentation . . . 430
9.221.2.1 port_list . . . 430
9.221.2.2 mac . . . 430
9.221.2.3 tag . . . 431
9.221.2.4 inner_tag . . . 431
9.221.2.5 type . . . 431
9.221.2.6 etype . . . 431
9.221.2.7 llc . . . 431
9.221.2.8 snap . . . 432
9.221.2.9 ipv4 . . . 432
9.221.2.10 ipv6 . . . 432
9.221.2.11 frame . . . 432
9.222 vtss_qce_mac_t Struct Reference . . . 432
9.222.1 Detailed Description . . . 433
9.222.2 Field Documentation . . . 433
9.222.2.1 dmac_mc . . . 433
9.222.2.2 dmac_bc . . . 433
9.222.2.3 dmac . . . 433
9.222.2.4 smac . . . 434
9.223 vtss_qce_t Struct Reference . . . 434

9.223.1 Detailed Description . . . 434
9.223.2 Field Documentation . . . 434
9.223.2.1 id . . . 434
9.223.2.2 key . . . 435
9.223.2.3 action . . . 435
9.224 vtss_qce_tag_t Struct Reference . . . 435
9.224.1 Detailed Description . . . 435
9.224.2 Field Documentation . . . 435
9.224.2.1 vid . . . 436
9.224.2.2 pcp . . . 436
9.224.2.3 dei . . . 436
9.224.2.4 tagged . . . 436
9.224.2.5 s_tag . . . 436
9.225 vtss_qos_conf_t Struct Reference . . . 437
9.225.1 Detailed Description . . . 437
9.225.2 Field Documentation . . . 437
9.225.2.1 prios . . . 437
9.225.2.2 dscp_trust . . . 438
9.225.2.3 dscp_qos_class_map . . . 438
9.225.2.4 dscp_dp_level_map . . . 438
9.225.2.5 dscp_qos_map . . . 438
9.225.2.6 dscp_qos_map_dp1 . . . 438
9.225.2.7 dscp_remark . . . 439
9.225.2.8 dscp_translate_map . . . 439
9.225.2.9 dscp_remap . . . 439
9.225.2.10 dscp_remap_dp1 . . . 439
9.225.2.11 policer_uc . . . 439
9.225.2.12 policer_uc_frame_rate . . . 440
9.225.2.13 policer_uc_mode . . . 440
9.225.2.14 policer_mc . . . 440

9.225.2.15 policer_mc_frame_rate . . . 440
9.225.2.16 policer_mc_mode . . . 440
9.225.2.17 policer_bc . . . 441
9.225.2.18 policer_bc_frame_rate . . . 441
9.225.2.19 policer_bc_mode . . . 441
9.225.2.20 red_v2 . . . 441
9.226 vtss_qos_port_conf_t Struct Reference . . . 441
9.226.1 Detailed Description . . . 442
9.226.2 Field Documentation . . . 442
9.226.2.1 policer_port . . . 442
9.226.2.2 policer_ext_port . . . 443
9.226.2.3 policer_queue . . . 443
9.226.2.4 shaper_port . . . 443
9.226.2.5 shaper_queue . . . 443
9.226.2.6 excess_enable . . . 443
9.226.2.7 default_prio . . . 444
9.226.2.8 usr_prio . . . 444
9.226.2.9 default_dpl . . . 444
9.226.2.10 default_dei . . . 444
9.226.2.11 tag_class_enable . . . 444
9.226.2.12 qos_class_map . . . 445
9.226.2.13 dp_level_map . . . 445
9.226.2.14 dscp_class_enable . . . 445
9.226.2.15 dscp_mode . . . 445
9.226.2.16 dscp_emode . . . 445
9.226.2.17 dscp_translate . . . 446
9.226.2.18 tag_remark_mode . . . 446
9.226.2.19 tag_default_pcp . . . 446
9.226.2.20 tag_default_dei . . . 446
9.226.2.21 tag_pcp_map . . . 446

9.226.2.22 tag_dei_map . . . 447
9.226.2.23 dwrr_enable . . . 447
9.226.2.24 dwrr_cnt . . . 447
9.226.2.25 queue_pct . . . 447
9.226.2.26 dmac_dip . . . 447
9.226.2.27 key_type . . . 448
9.227 vtss_rcpll_status_t Struct Reference . . . 448
9.227.1 Detailed Description . . . 448
9.227.2 Field Documentation . . . 448
9.227.2.1 out_of_range . . . 448
9.227.2.2 cal_error . . . 449
9.227.2.3 cal_not_done . . . 449
9.228 vtss_red_v2_t Struct Reference . . . 449
9.228.1 Detailed Description . . . 449
9.228.2 Field Documentation . . . 449
9.228.2.1 enable . . . 450
9.228.2.2 min_fl . . . 450
9.228.2.3 max . . . 450
9.228.2.4 max_unit . . . 450
9.229 vtss_restart_status_t Struct Reference . . . 450
9.229.1 Detailed Description . . . 451
9.229.2 Field Documentation . . . 451
9.229.2.1 restart . . . 451
9.229.2.2 prev_version . . . 451
9.229.2.3 cur_version . . . 451
9.230 vtss_routing_entry_t Struct Reference . . . 452
9.230.1 Detailed Description . . . 452
9.230.2 Field Documentation . . . 452
9.230.2.1 type . . . 452
9.230.2.2 ipv4_uc . . . 452

9.230.2.3 ipv6_uc . . . 453
9.230.2.4 route . . . 453
9.230.2.5 vlan . . . 453
9.231 vtss_secure_on_passwd_t Struct Reference . . . 453
9.231.1 Detailed Description . . . 453
9.231.2 Field Documentation . . . 454
9.231.2.1 passwd . . . 454
9.232 vtss_serdes_macro_conf_t Struct Reference . . . 454
9.232.1 Detailed Description . . . 454
9.232.2 Field Documentation . . . 454
9.232.2.1 serdes1g_vdd . . . 454
9.232.2.2 serdes6g_vdd . . . 455
9.232.2.3 ib_cterm_ena . . . 455
9.232.2.4 qsgmii . . . 455
9.233 vtss_sflow_port_conf_t Struct Reference . . . 455
9.233.1 Detailed Description . . . 456
9.233.2 Field Documentation . . . 456
9.233.2.1 type . . . 456
9.233.2.2 sampling_rate . . . 456
9.234 vtss_sgpio_conf_t Struct Reference . . . 456
9.234.1 Detailed Description . . . 457
9.234.2 Field Documentation . . . 457
9.234.2.1 bmode . . . 457
9.234.2.2 bit_count . . . 457
9.234.2.3 port_conf . . . 457
9.235 vtss_sgpio_port_conf_t Struct Reference . . . 457
9.235.1 Detailed Description . . . 458
9.235.2 Field Documentation . . . 458
9.235.2.1 enabled . . . 458
9.235.2.2 mode . . . 458

9.235.2.3 int_pol_high . . . 458
9.236 vtss_sgpio_port_data_t Struct Reference . . . 459
9.236.1 Detailed Description . . . 459
9.236.2 Field Documentation . . . 459
9.236.2.1 value . . . 459
9.237 vtss_shaper_t Struct Reference . . . 459
9.237.1 Detailed Description . . . 460
9.237.2 Field Documentation . . . 460
9.237.2.1 level . . . 460
9.237.2.2 rate . . . 460
9.237.2.3 ebs . . . 460
9.237.2.4 eir . . . 460
9.237.2.5 mode . . . 461
9.238 vtss_synce_clock_in_t Struct Reference . . . 461
9.238.1 Detailed Description . . . 461
9.238.2 Field Documentation . . . 461
9.238.2.1 port_no . . . 461
9.238.2.2 squelsh . . . 462
9.238.2.3 enable . . . 462
9.239 vtss_synce_clock_out_t Struct Reference . . . 462
9.239.1 Detailed Description . . . 462
9.239.2 Field Documentation . . . 462
9.239.2.1 divider . . . 463
9.239.2.2 enable . . . 463
9.240 vtss_tci_t Struct Reference . . . 463
9.240.1 Detailed Description . . . 463
9.240.2 Field Documentation . . . 463
9.240.2.1 vid . . . 464
9.240.2.2 cfi . . . 464
9.240.2.3 tagprio . . . 464

9.241 vtss_timeofday_t Struct Reference . . . 464
9.241.1 Detailed Description . . . 464
9.241.2 Field Documentation . . . 465
9.241.2.1 sec [1/2] . . . 465
9.241.2.2 sec [2/2] . . . 465
9.242 vtss_timestamp_t Struct Reference . . . 465
9.242.1 Detailed Description . . . 465
9.242.2 Field Documentation . . . 466
9.242.2.1 sec_msb . . . 466
9.242.2.2 seconds . . . 466
9.242.2.3 nanoseconds . . . 466
9.243 vtss_trace_conf_t Struct Reference . . . 466
9.243.1 Detailed Description . . . 467
9.243.2 Field Documentation . . . 467
9.243.2.1 level . . . 467
9.244 vtss_ts_alt_clock_mode_t Struct Reference . . . 467
9.244.1 Detailed Description . . . 467
9.244.2 Field Documentation . . . 467
9.244.2.1 one_pps_out . . . 468
9.244.2.2 one_pps_in . . . 468
9.244.2.3 save . . . 468
9.244.2.4 load . . . 468
9.245 vtss_ts_ext_clock_mode_t Struct Reference . . . 468
9.245.1 Detailed Description . . . 469
9.245.2 Field Documentation . . . 469
9.245.2.1 one_pps_mode . . . 469
9.245.2.2 enable . . . 469
9.245.2.3 freq . . . 469
9.246 vtss_ts_id_t Struct Reference . . . 470
9.246.1 Detailed Description . . . 470

9.246.2 Field Documentation . . . 470
9.246.2.1 ts_id . . . 470
9.247 vtss_ts_internal_mode_t Struct Reference . . . 470
9.247.1 Detailed Description . . . 471
9.247.2 Field Documentation . . . 471
9.247.2.1 int_fmt . . . 471
9.248 vtss_ts_operation_mode_t Struct Reference . . . 471
9.248.1 Detailed Description . . . 471
9.248.2 Field Documentation . . . 471
9.248.2.1 mode . . . 472
9.249 vtss_ts_timestamp_alloc_t Struct Reference . . . 472
9.249.1 Detailed Description . . . 472
9.249.2 Field Documentation . . . 472
9.249.2.1 port_mask . . . 472
9.249.2.2 context . . . 473
9.249.2.3 cb . . . 473
9.250 vtss_ts_timestamp_t Struct Reference . . . 473
9.250.1 Detailed Description . . . 473
9.250.2 Field Documentation . . . 473
9.250.2.1 ts . . . 474
9.250.2.2 id . . . 474
9.250.2.3 context . . . 474
9.250.2.4 ts_valid . . . 474
9.251 vtss_vcap_ip_t Struct Reference . . . 474
9.251.1 Detailed Description . . . 475
9.251.2 Field Documentation . . . 475
9.251.2.1 value . . . 475
9.251.2.2 mask . . . 475
9.252 vtss_vcap_port_conf_t Struct Reference . . . 475
9.252.1 Detailed Description . . . 476

9.252.2 Field Documentation . . . 476
9.252.2.1 key_type_is1_1 . . . 476
9.252.2.2 dmac_dip_1 . . . 476
9.253 vtss_vcap_u128_t Struct Reference . . . 476
9.253.1 Detailed Description . . . 477
9.253.2 Field Documentation . . . 477
9.253.2.1 value . . . 477
9.253.2.2 mask . . . 477
9.254 vtss_vcap_u16_t Struct Reference . . . 477
9.254.1 Detailed Description . . . 478
9.254.2 Field Documentation . . . 478
9.254.2.1 value . . . 478
9.254.2.2 mask . . . 478
9.255 vtss_vcap_u24_t Struct Reference . . . 478
9.255.1 Detailed Description . . . 479
9.255.2 Field Documentation . . . 479
9.255.2.1 value . . . 479
9.255.2.2 mask . . . 479
9.256 vtss_vcap_u32_t Struct Reference . . . 479
9.256.1 Detailed Description . . . 480
9.256.2 Field Documentation . . . 480
9.256.2.1 value . . . 480
9.256.2.2 mask . . . 480
9.257 vtss_vcap_u40_t Struct Reference . . . 480
9.257.1 Detailed Description . . . 481
9.257.2 Field Documentation . . . 481
9.257.2.1 value . . . 481
9.257.2.2 mask . . . 481
9.258 vtss_vcap_u48_t Struct Reference . . . 481
9.258.1 Detailed Description . . . 482

9.258.2 Field Documentation . . . 482
9.258.2.1 value . . . 482
9.258.2.2 mask . . . 482
9.259 vtss_vcap_u8_t Struct Reference . . . 482
9.259.1 Detailed Description . . . 483
9.259.2 Field Documentation . . . 483
9.259.2.1 value . . . 483
9.259.2.2 mask . . . 483
9.260 vtss_vcap_udp_tcp_t Struct Reference . . . 483
9.260.1 Detailed Description . . . 484
9.260.2 Field Documentation . . . 484
9.260.2.1 in_range . . . 484
9.260.2.2 low . . . 484
9.260.2.3 high . . . 484
9.261 vtss_vcap_vid_t Struct Reference . . . 484
9.261.1 Detailed Description . . . 485
9.261.2 Field Documentation . . . 485
9.261.2.1 value . . . 485
9.261.2.2 mask . . . 485
9.262 vtss_vcap_vr_t Struct Reference . . . 485
9.262.1 Detailed Description . . . 486
9.262.2 Field Documentation . . . 486
9.262.2.1 type . . . 486
9.262.2.2 value . . . 486
9.262.2.3 mask . . . 487
9.262.2.4 v . . . 487
9.262.2.5 low . . . 487
9.262.2.6 high . . . 487
9.262.2.7 r . . . 487
9.262.2.8 vr . . . 487

9.263 vtss_vce_action_t Struct Reference . . . 488
9.263.1 Detailed Description . . . 488
9.263.2 Field Documentation . . . 488
9.263.2.1 vid . . . 488
9.263.2.2 policy_no . . . 488
9.264 vtss_vce_frame_etype_t Struct Reference . . . 488
9.264.1 Detailed Description . . . 489
9.264.2 Field Documentation . . . 489
9.264.2.1 etype . . . 489
9.264.2.2 data . . . 489
9.265 vtss_vce_frame_ipv4_t Struct Reference . . . 489
9.265.1 Detailed Description . . . 490
9.265.2 Field Documentation . . . 490
9.265.2.1 fragment . . . 490
9.265.2.2 options . . . 490
9.265.2.3 dscp . . . 490
9.265.2.4 proto . . . 491
9.265.2.5 sip . . . 491
9.265.2.6 dport . . . 491
9.266 vtss_vce_frame_ipv6_t Struct Reference . . . 491
9.266.1 Detailed Description . . . 492
9.266.2 Field Documentation . . . 492
9.266.2.1 dscp . . . 492
9.266.2.2 proto . . . 492
9.266.2.3 sip . . . 492
9.266.2.4 dport . . . 492
9.267 vtss_vce_frame_llc_t Struct Reference . . . 493
9.267.1 Detailed Description . . . 493
9.267.2 Field Documentation . . . 493
9.267.2.1 data . . . 493

9.268 vtss_vce_frame_snap_t Struct Reference . . . 493
9.268.1 Detailed Description . . . 494
9.268.2 Field Documentation . . . 494
9.268.2.1 data . . . 494
9.269 vtss_vce_key_t Struct Reference . . . 494
9.269.1 Detailed Description . . . 494
9.269.2 Field Documentation . . . 495
9.269.2.1 port_list . . . 495
9.269.2.2 mac . . . 495
9.269.2.3 tag . . . 495
9.269.2.4 type . . . 495
9.269.2.5 etype . . . 495
9.269.2.6 llc . . . 496
9.269.2.7 snap . . . 496
9.269.2.8 ipv4 . . . 496
9.269.2.9 ipv6 . . . 496
9.269.2.10 frame . . . 496
9.270 vtss_vce_mac_t Struct Reference . . . 497
9.270.1 Detailed Description . . . 497
9.270.2 Field Documentation . . . 497
9.270.2.1 dmac_mc . . . 497
9.270.2.2 dmac_bc . . . 497
9.270.2.3 smac . . . 498
9.271 vtss_vce_t Struct Reference . . . 498
9.271.1 Detailed Description . . . 498
9.271.2 Field Documentation . . . 498
9.271.2.1 id . . . 498
9.271.2.2 key . . . 499
9.271.2.3 action . . . 499
9.272 vtss_vce_tag_t Struct Reference . . . 499

9.272.1 Detailed Description . . . 499
9.272.2 Field Documentation . . . 499
9.272.2.1 vid . . . 500
9.272.2.2 pcp . . . 500
9.272.2.3 dei . . . 500
9.272.2.4 tagged . . . 500
9.272.2.5 s_tag . . . 500
9.273 vtss_vcl_port_conf_t Struct Reference . . . 501
9.273.1 Detailed Description . . . 501
9.273.2 Field Documentation . . . 501
9.273.2.1 dmac_dip . . . 501
9.274 vtss_vid_mac_t Struct Reference . . . 501
9.274.1 Detailed Description . . . 502
9.274.2 Field Documentation . . . 502
9.274.2.1 vid . . . 502
9.274.2.2 mac . . . 502
9.275 vtss_vlan_conf_t Struct Reference . . . 502
9.275.1 Detailed Description . . . 502
9.275.2 Field Documentation . . . 503
9.275.2.1 s_etype . . . 503
9.276 vtss_vlan_port_conf_t Struct Reference . . . 503
9.276.1 Detailed Description . . . 503
9.276.2 Field Documentation . . . 503
9.276.2.1 port_type . . . 504
9.276.2.2 pvid . . . 504
9.276.2.3 untagged_vid . . . 504
9.276.2.4 frame_type . . . 504
9.276.2.5 ingress_filter . . . 504
9.276.2.6 s_etype . . . 505
9.277 vtss_vlan_tag_t Struct Reference . . . 505

9.277.1 Detailed Description 505
9.277.2 Field Documentation 505
9.277.2.1 tpid 505
9.277.2.2 pcp 506
9.277.2.3 dei 506
9.277.2.4 vid 506
9.278 vtss_vlan_trans_grp2vlan_conf_t Struct Reference 506
9.278.1 Detailed Description 506
9.278.2 Field Documentation 507
9.278.2.1 group_id 507
9.278.2.2 vid 507
9.278.2.3 trans_vid 507
9.279 vtss_vlan_trans_port2grp_conf_t Struct Reference 507
9.279.1 Detailed Description 508
9.279.2 Field Documentation 508
9.279.2.1 group_id 508
9.279.2.2 ports 508
9.280 vtss_vlan_vid_conf_t Struct Reference 508
9.280.1 Detailed Description 509
9.280.2 Field Documentation 509
9.280.2.1 learning 509
9.280.2.2 mirror 509
9.280.2.3 fid 509
9.281 vtss_wol_mac_addr_t Struct Reference 509
9.281.1 Detailed Description 510
9.281.2 Field Documentation 510
9.281.2.1 addr 510

10 File Documentation 511

10.1 vtss_api/include/vtss/api/l2_types.h File Reference . . . 511

10.1.1 Detailed Description . . . 511

10.1.2 Enumeration Type Documentation . . . 511

10.1.2.1 vtss_sflow_type_t . . . 511

10.2 vtss_api/include/vtss/api/options.h File Reference . . . 512

10.2.1 Detailed Description . . . 514

10.2.2 Macro Definition Documentation . . . 514

10.2.2.1 VTSS_ARCH_SERVAL . . . 514

10.2.2.2 VTSS_ARCH_SERVAL_CE . . . 514

10.2.2.3 VTSS_ARCH_SERVAL_CPU . . . 515

10.2.2.4 VTSS_FEATURE_WARM_START [1/2] . . . 515

10.2.2.5 VTSS_FEATURE_MISC . . . 515

10.2.2.6 VTSS_FEATURE_PORT_CONTROL . . . 515

10.2.2.7 VTSS_FEATURE_PORT_IFH . . . 515

10.2.2.8 VTSS_FEATURE_CLAUSE_37 . . . 516

10.2.2.9 VTSS_FEATURE_EXC_COL_CONT . . . 516

10.2.2.10 VTSS_FEATURE_PORT_CNT_BRIDGE . . . 516

10.2.2.11 VTSS_FEATURE_PFC . . . 516

10.2.2.12 VTSS_FEATURE_QOS . . . 516

10.2.2.13 VTSS_FEATURE_QCL . . . 517

10.2.2.14 VTSS_FEATURE_QCL_V2 . . . 517

10.2.2.15 VTSS_FEATURE_QCL_DMAC_DIP . . . 517

10.2.2.16 VTSS_FEATURE_QCL_KEY_TYPE . . . 517

10.2.2.17 VTSS_FEATURE_QCL_KEY_S_TAG . . . 517

10.2.2.18 VTSS_FEATURE_QCL_KEY_INNER_TAG . . . 518

10.2.2.19 VTSS_FEATURE_QCL_KEY_DMAC . . . 518

10.2.2.20 VTSS_FEATURE_QCL_KEY_DIP . . . 518

10.2.2.21 VTSS_FEATURE_QCL_PCP_DEI_ACTION . . . 518

10.2.2.22 VTSS_FEATURE_QCL_POLICY_ACTION . . . 518

10.2.2.23 VTSS_FEATURE_QOS_POLICER_UC_SWITCH 519
10.2.2.24 VTSS_FEATURE_QOS_POLICER_MC_SWITCH 519
10.2.2.25 VTSS_FEATURE_QOS_POLICER_BC_SWITCH 519
10.2.2.26 VTSS_FEATURE_QOS_PORT_POLICER_EXT 519
10.2.2.27 VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS 519
10.2.2.28 VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC 520
10.2.2.29 VTSS_FEATURE_QOS_QUEUE_TX 520
10.2.2.30 VTSS_FEATURE_QOS_QUEUE_POLICER 520
10.2.2.31 VTSS_FEATURE_QOS_SCHEDULER_V2 520
10.2.2.32 VTSS_FEATURE_QOS_SCHEDULER_DWRR_CNT 520
10.2.2.33 VTSS_FEATURE_QOS_TAG_REMARK_V2 521
10.2.2.34 VTSS_FEATURE_QOS_CLASSIFICATION_V2 521
10.2.2.35 VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS 521
10.2.2.36 VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB 521
10.2.2.37 VTSS_FEATURE_QOS_EGRESS_SHAPERS_DLB 521
10.2.2.38 VTSS_FEATURE_QOS_EGRESS_SHAPERS_RT 522
10.2.2.39 VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE 522
10.2.2.40 VTSS_FEATURE_QOS_DSCP_REMARK 522
10.2.2.41 VTSS_FEATURE_QOS_DSCP_REMARK_V2 522
10.2.2.42 VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWARE 522
10.2.2.43 VTSS_FEATURE_QOS_WRED_V2 523
10.2.2.44 VTSS_FEATURE_QOS_POLICER_DLB 523
10.2.2.45 VTSS_FEATURE_QOS_CPU_PORT_SHAPER 523
10.2.2.46 VTSS_FEATURE_PACKET 523
10.2.2.47 VTSS_FEATURE_PACKET_TX 523
10.2.2.48 VTSS_FEATURE_PACKET_RX 524
10.2.2.49 VTSS_FEATURE_PACKET_GROUPING 524
10.2.2.50 VTSS_FEATURE_PACKET_PORT_REG 524
10.2.2.51 VTSS_FEATURE_LAYER2 524
10.2.2.52 VTSS_FEATURE_PVLAN 524

10.2.2.53 VTSS_FEATURE_VLAN_PORT_V2 525
10.2.2.54 VTSS_FEATURE_VLAN_TX_TAG 525
10.2.2.55 VTSS_FEATURE_VLAN_SVL 525
10.2.2.56 VTSS_FEATURE_MAC_AGE_AUTO 525
10.2.2.57 VTSS_FEATURE_MAC_CPU_QUEUE 525
10.2.2.58 VTSS_FEATURE_EEE 526
10.2.2.59 VTSS_FEATURE_VCAP [1/2] 526
10.2.2.60 VTSS_FEATURE_ACL 526
10.2.2.61 VTSS_FEATURE_ACL_V2 526
10.2.2.62 VTSS_FEATURE_VCL 526
10.2.2.63 VTSS_FEATURE_NPI 527
10.2.2.64 VTSS_FEATURE_LED_POW_REDUC 527
10.2.2.65 VTSS_FEATURE_VLAN_TRANSLATION 527
10.2.2.66 VTSS_FEATURE_SFLOW 527
10.2.2.67 VTSS_FEATURE_MIRROR_CPU 527
10.2.2.68 VTSS_FEATURE_IRQ_CONTROL 528
10.2.2.69 VTSS_FEATURE_SERDES_MACRO_SETTINGS [1/2] 528
10.2.2.70 TESLA_ING_TS_ERRFIX 528
10.2.2.71 VIPER_B_FIFO_RESET 528
10.2.2.72 VTSS_TS_FIFO_SYNC_LOOPBACK 528
10.2.2.73 VTSS_TS_FIFO_MEDIA_SWAP_SYNC 529
10.2.2.74 VTSS_PHY_TS_SPI_CLK_THRU_PPS0 529
10.2.2.75 VTSS_FEATURE_INTERRUPTS 529
10.2.2.76 VTSS_FEATURE_SERDES_MACRO_SETTINGS [2/2] 529
10.2.2.77 VTSS_FEATURE_SYNCE 529
10.2.2.78 VTSS_FEATURE_SERIAL_GPIO 530
10.2.2.79 VTSS_FEATURE_FAN 530
10.2.2.80 VTSS_FEATURE_PHY_TIMESTAMP 530
10.2.2.81 VTSS_FEATURE_TIMESTAMP 530
10.2.2.82 VTSS_FEATURE_TIMESTAMP_ONE_STEP 530

10.2.2.83 VTSS_FEATURE_TIMESTAMP_LATENCY_COMP 531
10.2.2.84 VTSS_FEATURE_TIMESTAMP_ORG_TIME 531
10.2.2.85 VTSS_FEATURE_TIMESTAMP_P2P_DELAY_COMP 531
10.2.2.86 VTSS_FEATURE_TIMESTAMP_ASYMMETRY_COMP 531
10.2.2.87 VTSS_FEATURE_PTP_RS422 531
10.2.2.88 VTSS_OPT_VCORE_III 532
10.2.2.89 VTSS_FEATURE_FDMA 532
10.2.2.90 VTSS_FEATURE_EVC 532
10.2.2.91 VTSS_FEATURE_E_TREE 532
10.2.2.92 VTSS_FEATURE_OAM 532
10.2.2.93 VTSS_FEATURE_AFI_SWC 533
10.2.2.94 VTSS_AFI_V1 533
10.2.2.95 VTSS_FEATURE_MPLS 533
10.2.2.96 VTSS_FEATURE_HQOS 533
10.2.2.97 VTSS_CHIP_CU_PHY 533
10.2.2.98 VTSS_OPT_TRACE 534
10.2.2.99 VTSS_OPT_FDMA_IRQ_CONTEXT 534
10.2.2.100 VTSS_OPT_FDMA_DEBUG 534
10.2.2.101 VTSS_OPT_VAUI_EQ_CTRL 534
10.2.2.102 VTSS_OPT_PORT_COUNT 534
10.2.2.103 VTSS_PHY_OPT_VERIPHY 535
10.2.2.104 VTSS_FEATURE_WARM_START [2/2] 535
10.2.2.105 VTSS_FEATURE_VCAP [2/2] 535
10.3 vtss_api/include/vtss/api/phy.h File Reference 535
10.3.1 Detailed Description 536
10.3.2 Macro Definition Documentation 536
10.3.2.1 VTSS_PHY_POWER_ACTIPHY_BIT 536
10.3.2.2 VTSS_PHY_POWER_DYNAMIC_BIT 536
10.3.3 Enumeration Type Documentation 536
10.3.3.1 vtss_phy_power_mode_t 536

10.3.3.2 vtss_phy_veriphy_status_t . . . 537

10.4 vtss_api/include/vtss/api/port.h File Reference . . . 537

10.4.1 Detailed Description . . . 539

10.4.2 Macro Definition Documentation . . . 539

10.4.2.1 PORT_CAP_NONE . . . 539

10.4.2.2 PORT_CAP_AUTONEG . . . 540

10.4.2.3 PORT_CAP_10M_HDX . . . 540

10.4.2.4 PORT_CAP_10M_FDX . . . 540

10.4.2.5 PORT_CAP_100M_HDX . . . 540

10.4.2.6 PORT_CAP_100M_FDX . . . 540

10.4.2.7 PORT_CAP_1G_FDX . . . 541

10.4.2.8 PORT_CAP_2_5G_FDX . . . 541

10.4.2.9 PORT_CAP_5G_FDX . . . 541

10.4.2.10 PORT_CAP_10G_FDX . . . 541

10.4.2.11 PORT_CAP_FLOW_CTRL . . . 541

10.4.2.12 PORT_CAP_COPPER . . . 542

10.4.2.13 PORT_CAP_FIBER . . . 542

10.4.2.14 PORT_CAP_DUAL_COPPER . . . 542

10.4.2.15 PORT_CAP_DUAL_FIBER . . . 542

10.4.2.16 PORT_CAP_SD_ENABLE . . . 542

10.4.2.17 PORT_CAP_SD_HIGH . . . 543

10.4.2.18 PORT_CAP_SD_INTERNAL . . . 543

10.4.2.19 PORT_CAP_DUAL_FIBER_100FX . . . 543

10.4.2.20 PORT_CAP_XAUI_LANE_FLIP . . . 543

10.4.2.21 PORT_CAP_VTSS_10G_PHY . . . 543

10.4.2.22 PORT_CAP_SFP_DETECT . . . 544

10.4.2.23 PORT_CAP_STACKING . . . 544

10.4.2.24 PORT_CAP_DUAL_SFP_DETECT . . . 544

10.4.2.25 PORT_CAP_SFP_ONLY . . . 544

10.4.2.26 PORT_CAP_DUAL_COPPER_100FX . . . 544

10.4.2.27 PORT_CAP_HDX . . . 545
10.4.2.28 PORT_CAP_TRI_SPEED_FDX . . . 545
10.4.2.29 PORT_CAP_TRI_SPEED . . . 545
10.4.2.30 PORT_CAP_1G_PHY . . . 545
10.4.2.31 PORT_CAP_TRI_SPEED_COPPER . . . 545
10.4.2.32 PORT_CAP_TRI_SPEED_FIBER . . . 546
10.4.2.33 PORT_CAP_TRI_SPEED_DUAL_COPPER . . . 546
10.4.2.34 PORT_CAP_TRI_SPEED_DUAL_FIBER . . . 546
10.4.2.35 PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX . . . 546
10.4.2.36 PORT_CAP_ANY_FIBER . . . 546
10.4.2.37 PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED . . . 547
10.4.2.38 PORT_CAP_SPEED_DUAL_ANY_FIBER . . . 547
10.4.2.39 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER . . . 547
10.4.2.40 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED . . . 547
10.4.2.41 PORT_CAP_DUAL_FIBER_1000X . . . 547
10.4.2.42 PORT_CAP_SFP_1G . . . 548
10.4.2.43 PORT_CAP_SFP_2_5G . . . 548
10.4.2.44 PORT_CAP_SFP_SD_HIGH . . . 548
10.4.2.45 PORT_CAP_2_5G_TRI_SPEED_FDX . . . 548
10.4.2.46 PORT_CAP_2_5G_TRI_SPEED . . . 548
10.4.2.47 PORT_CAP_2_5G_TRI_SPEED_COPPER . . . 549
10.4.3 Typedef Documentation . . . 549
10.4.3.1 port_cap_t . . . 549
10.4.4 Enumeration Type Documentation . . . 549
10.4.4.1 vtss_port_speed_t . . . 549
10.4.4.2 vtss_fiber_port_speed_t . . . 550
10.5 vtss_api/include/vtss/api/types.h File Reference . . . 550
10.5.1 Detailed Description . . . 557
10.5.2 Macro Definition Documentation . . . 557
10.5.2.1 PRIu64 . . . 558

10.5.2.2 PRIi64 558
10.5.2.3 PRIx64 558
10.5.2.4 VTSS_BIT64 558
10.5.2.5 VTSS_BITMASK64 558
10.5.2.6 VTSS_EXTRACT_BITFIELD64 559
10.5.2.7 VTSS_ENCODE_BITFIELD64 559
10.5.2.8 VTSS_ENCODE_BITMASK64 559
10.5.2.9 TRUE 559
10.5.2.10 FALSE 560
10.5.2.11 VTSS_PACKET_RATE_DISABLED 560
10.5.2.12 VTSS_PORT_COUNT [1/2] 560
10.5.2.13 VTSS_PORT_COUNT [2/2] 560
10.5.2.14 VTSS_PORTS 560
10.5.2.15 VTSS_PORT_NO_NONE 561
10.5.2.16 VTSS_PORT_NO_CPU 561
10.5.2.17 VTSS_PORT_NO_START 561
10.5.2.18 VTSS_PORT_NO_END 561
10.5.2.19 VTSS_PORT_ARRAY_SIZE 561
10.5.2.20 VTSS_PORT_IS_PORT 562
10.5.2.21 VTSS_PRIOS 562
10.5.2.22 VTSS_PRIO_NO_NONE 562
10.5.2.23 VTSS_PRIO_START 562
10.5.2.24 VTSS_PRIO_END 562
10.5.2.25 VTSS_PRIO_ARRAY_SIZE 563
10.5.2.26 VTSS_QUEUES 563
10.5.2.27 VTSS_QUEUE_START 563
10.5.2.28 VTSS_QUEUE_END 563
10.5.2.29 VTSS_QUEUE_ARRAY_SIZE 563
10.5.2.30 VTSS_PCPS 564
10.5.2.31 VTSS_PCP_START 564

10.5.2.32 VTSS_PCP_END 564
10.5.2.33 VTSS_PCP_ARRAY_SIZE 564
10.5.2.34 VTSS_DEIS 564
10.5.2.35 VTSS_DEI_START 565
10.5.2.36 VTSS_DEI_END 565
10.5.2.37 VTSS_DEI_ARRAY_SIZE 565
10.5.2.38 VTSS_DPLS 565
10.5.2.39 VTSS_DPL_START 565
10.5.2.40 VTSS_DPL_END 566
10.5.2.41 VTSS_DPL_ARRAY_SIZE 566
10.5.2.42 VTSS_BITRATE_DISABLED 566
10.5.2.43 VTSS_VID_NULL 566
10.5.2.44 VTSS_VID_DEFAULT 566
10.5.2.45 VTSS_VID_RESERVED 567
10.5.2.46 VTSS_VIDS 567
10.5.2.47 VTSS_VID_ALL 567
10.5.2.48 VTSS_ETYPE_VTSS 567
10.5.2.49 VTSS_MAC_ADDR_SZ_BYTES 567
10.5.2.50 MAC_ADDR_BROADCAST 568
10.5.2.51 VTSS_EVCS 568
10.5.2.52 VTSS_ISDX_NONE 568
10.5.2.53 VTSS_AGGRS 568
10.5.2.54 VTSS_AGGR_NO_NONE 568
10.5.2.55 VTSS_AGGR_NO_START 569
10.5.2.56 VTSS_AGGR_NO_END 569
10.5.2.57 VTSS_GLAGS 569
10.5.2.58 VTSS_GLAG_NO_NONE 569
10.5.2.59 VTSS_GLAG_NO_START 569
10.5.2.60 VTSS_GLAG_NO_END 570
10.5.2.61 VTSS_GLAG_PORTS 570

10.5.2.62 VTSS_GLAG_PORT_START 570
10.5.2.63 VTSS_GLAG_PORT_END 570
10.5.2.64 VTSS_GLAG_PORT_ARRAY_SIZE 570
10.5.2.65 VTSS_PACKET_RX_QUEUE_CNT 571
10.5.2.66 VTSS_PACKET_RX_GRP_CNT 571
10.5.2.67 VTSS_PACKET_TX_GRP_CNT 571
10.5.2.68 VTSS_PACKET_RX_QUEUE_NONE 571
10.5.2.69 VTSS_PACKET_RX_QUEUE_START 571
10.5.2.70 VTSS_PACKET_RX_QUEUE_END 572
10.5.2.71 VTSS_ACL_POLICERS 572
10.5.2.72 VTSS_ACL_POLICER_NO_START 572
10.5.2.73 VTSS_ACL_POLICER_NO_END 572
10.5.2.74 VTSS_ACL_POLICY_NO_NONE 572
10.5.2.75 VTSS_ACL_POLICY_NO_MIN 573
10.5.2.76 VTSS_ACL_POLICY_NO_MAX 573
10.5.2.77 VTSS_ACL_POLICIES 573
10.5.2.78 VTSS_ACL_POLICY_NO_START 573
10.5.2.79 VTSS_ACL_POLICY_NO_END 573
10.5.2.80 VTSS_HQOS_COUNT 574
10.5.2.81 VTSS_HQOS_ID_NONE 574
10.5.2.82 VTSS_ONE_MIA 574
10.5.2.83 VTSS_ONE_MILL 574
10.5.2.84 VTSS_MAX_TIMEINTERVAL 574
10.5.2.85 VTSS_INTERVAL_SEC 575
10.5.2.86 VTSS_INTERVAL_MS 575
10.5.2.87 VTSS_INTERVAL_US 575
10.5.2.88 VTSS_INTERVAL_NS 575
10.5.2.89 VTSS_INTERVAL_PS 575
10.5.2.90 VTSS_SEC_NS_INTERVAL 576
10.5.2.91 VTSS_CLOCK_IDENTITY_LENGTH 576

10.5.2.92 VTSS_SYNCE_CLK_PORT_ARRAY_SIZE . . . 576
10.5.3 Typedef Documentation . . . 576
10.5.3.1 i8 . . . 576
10.5.3.2 i16 . . . 577
10.5.3.3 i32 . . . 577
10.5.3.4 i64 . . . 577
10.5.3.5 u8 . . . 577
10.5.3.6 u16 . . . 577
10.5.3.7 u32 . . . 578
10.5.3.8 u64 . . . 578
10.5.3.9 BOOL . . . 578
10.5.3.10 uintptr_t . . . 578
10.5.3.11 vtss_mac_addr_t . . . 578
10.5.3.12 vtss_isdx_t . . . 579
10.5.3.13 vtss_packet_rx_grp_t . . . 579
10.5.3.14 vtss_packet_tx_grp_t . . . 579
10.5.4 Enumeration Type Documentation . . . 579
10.5.4.1 anonymous enum . . . 580
10.5.4.2 vtss_mem_flags_t . . . 582
10.5.4.3 vtss_port_interface_t . . . 583
10.5.4.4 vtss_serdes_mode_t . . . 583
10.5.4.5 vtss_storm_policer_mode_t . . . 584
10.5.4.6 vtss_policer_type_t . . . 584
10.5.4.7 vtss_vlan_frame_t . . . 584
10.5.4.8 vtss_packet_reg_type_t . . . 586
10.5.4.9 vtss_vdd_t . . . 586
10.5.4.10 vtss_ip_type_t . . . 586
10.5.4.11 vtss_vcap_bit_t . . . 587
10.5.4.12 vtss_vcap_vr_type_t . . . 587
10.5.4.13 vtss_vcap_key_type_t . . . 587

10.5.4.14 vtss_ece_dir_t . . . 588
10.5.4.15 vtss_ece_pop_tag_t . . . 588
10.5.4.16 vtss_ece_rule_t . . . 588
10.5.4.17 vtss_ece_tx_lookup_t . . . 589
10.5.4.18 vtss_ece_pcp_mode_t . . . 589
10.5.4.19 vtss_ece_dei_mode_t . . . 589
10.5.4.20 vtss_ece_inner_tag_type_t . . . 590
10.5.4.21 vtss_hqos_sch_mode_t . . . 590
10.6 vtss_api/include/vtss_ae_api.h File Reference . . . 590
10.6.1 Detailed Description . . . 591
10.7 vtss_api/include/vtss_afi_api.h File Reference . . . 591
10.7.1 Detailed Description . . . 591
10.8 vtss_api/include/vtss_aneg_api.h File Reference . . . 591
10.8.1 Detailed Description . . . 591
10.9 vtss_api/include/vtss_api.h File Reference . . . 592
10.9.1 Detailed Description . . . 592
10.10 vtss_api/include/vtss_evc_api.h File Reference . . . 592
10.10.1 Detailed Description . . . 595
10.10.2 Macro Definition Documentation . . . 596
10.10.2.1 VTSS_EVC_POLICERS . . . 596
10.10.2.2 VTSS_EVC_POLICER_ID_DISCARD . . . 596
10.10.2.3 VTSS_EVC_POLICER_ID_NONE . . . 596
10.10.2.4 VTSS_EVC_POLICER_ID_EVC . . . 596
10.10.2.5 VTSS_EVC_ID_NONE . . . 596
10.10.2.6 VTSS_EVC_MPLS_PW_CNT . . . 597
10.10.2.7 VTSS_ECE_ID_LAST . . . 597
10.10.2.8 VTSS_MCE_ID_LAST . . . 597
10.10.2.9 VTSS_MCE_ISDX_NONE . . . 597
10.10.2.10 VTSS_MCE_ISDX_NEW . . . 597
10.10.2.11 VTSS_ISDX_CPU_TX . . . 598

10.10.2.12 VTSS_MCE_POP_NONE . . . 598
10.10.3 Enumeration Type Documentation . . . 598
10.10.3.1 vtss_ece_type_t . . . 598
10.10.3.2 vtss_ece_port_t . . . 598
10.10.3.3 vtss_mce_tx_lookup_t . . . 599
10.10.3.4 vtss_mce_oam_detect_t . . . 599
10.10.3.5 vtss_mce_rule_t . . . 599
10.10.4 Function Documentation . . . 600
10.10.4.1 vtss_evc_port_conf_get() . . . 600
10.10.4.2 vtss_evc_port_conf_set() . . . 600
10.10.4.3 vtss_evc_policer_conf_get() . . . 601
10.10.4.4 vtss_evc_policer_conf_set() . . . 601
10.10.4.5 vtss_evc_add() . . . 602
10.10.4.6 vtss_evc_del() . . . 602
10.10.4.7 vtss_evc_get() . . . 602
10.10.4.8 vtss_evc_hqos_map_get() . . . 603
10.10.4.9 vtss_evc_hqos_map_set() . . . 603
10.10.4.10 vtss_ece_init() . . . 604
10.10.4.11 vtss_ece_add() . . . 604
10.10.4.12 vtss_ece_del() . . . 605
10.10.4.13 vtss_evc_oam_port_conf_get() . . . 605
10.10.4.14 vtss_evc_oam_port_conf_set() . . . 606
10.10.4.15 vtss_mce_init() . . . 606
10.10.4.16 vtss_mce_add() . . . 606
10.10.4.17 vtss_mce_del() . . . 607
10.10.4.18 vtss_mce_port_info_get() . . . 607
10.10.4.19 vtss_evc_counters_get() . . . 608
10.10.4.20 vtss_evc_counters_clear() . . . 608
10.10.4.21 vtss_ece_counters_get() . . . 609
10.10.4.22 vtss_ece_counters_clear() . . . 609

10.10.4.23 vtss_mce_counters_get() . . . 609
10.10.4.24 vtss_mce_counters_clear() . . . 610
10.11 vtss_api/include/vtss_fdma_api.h File Reference . . . 610
10.11.1 Detailed Description . . . 612
10.11.2 Macro Definition Documentation . . . 612
10.11.2.1 VTSS_FDMA_CH_CNT . . . 612
10.11.2.2 VTSS_PHYS_PORT_CNT . . . 613
10.11.2.3 VTSS_FDMA_DCB_SIZE_BYTES . . . 613
10.11.2.4 VTSS_FDMA_HDR_SIZE_BYTES . . . 613
10.11.2.5 VTSS_FDMA_MAX_DATA_PER_DCB_BYTES . . . 613
10.11.2.6 VTSS_FDMA_MIN_FRAME_SIZE_BYTES . . . 613
10.11.2.7 VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BYTES . . . 614
10.11.2.8 VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_BYTES . . . 614
10.11.2.9 VTSS_FDMA_MIN_DATA_PER_NON_SOF_DCB_BYTES . . . 614
10.11.2.10 VTSS_FDMA_MAX_FRAME_SIZE_BYTES . . . 614
10.11.2.11 VTSS_OS_DCACHE_LINE_SIZE_BYTES . . . 614
10.11.2.12 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED . . . 615
10.11.2.13 VTSS_AFI_FPS_MAX . . . 615
10.11.3 Typedef Documentation . . . 615
10.11.3.1 vtss_fdma_ch_t . . . 615
10.11.3.2 vtss_fdma_list_t . . . 617
10.11.4 Enumeration Type Documentation . . . 617
10.11.4.1 vtss_fdma_afi_type_t . . . 617
10.11.4.2 vtss_fdma_ch_usage_t . . . 618
10.11.4.3 vtss_fdma_dcb_type_t . . . 618
10.11.5 Function Documentation . . . 618
10.11.5.1 vtss_fdma_uninit() . . . 619
10.11.5.2 vtss_fdma_cfg() . . . 619
10.11.5.3 vtss_fdma_dcb_release() . . . 620
10.11.5.4 vtss_fdma_tx() . . . 620

10.11.5.5 vtss_fdma_tx_info_init() . . . 621
10.11.5.6 vtss_fdma_afi_cancel() . . . 622
10.11.5.7 vtss_fdma_dcb_get() . . . 622
10.11.5.8 vtss_fdma_throttle_cfg_get() . . . 623
10.11.5.9 vtss_fdma_throttle_cfg_set() . . . 624
10.11.5.10 vtss_fdma_throttle_tick() . . . 624
10.11.5.11 vtss_fdma_stats_clr() . . . 625
10.11.5.12 vtss_fdma_irq_handler() . . . 625
10.12 vtss_api/include/vtss_gfp_api.h File Reference . . . 626
10.12.1 Detailed Description . . . 627
10.13 vtss_api/include/vtss_hqos_api.h File Reference . . . 627
10.13.1 Detailed Description . . . 628
10.13.2 Function Documentation . . . 628
10.13.2.1 vtss_hqos_port_conf_get() . . . 628
10.13.2.2 vtss_hqos_port_conf_set() . . . 628
10.13.2.3 vtss_hqos_add() . . . 629
10.13.2.4 vtss_hqos_del() . . . 629
10.13.2.5 vtss_hqos_get() . . . 630
10.13.2.6 vtss_hqos_min_rate_calc() . . . 630
10.13.2.7 vtss_hqos_port_min_rate_calc() . . . 630
10.14 vtss_api/include/vtss_i2c_api.h File Reference . . . 631
10.14.1 Detailed Description . . . 631
10.15 vtss_api/include/vtss_init_api.h File Reference . . . 631
10.15.1 Detailed Description . . . 633
10.15.2 Macro Definition Documentation . . . 633
10.15.2.1 VTSS_I2C_NO_MULTIPLEXER . . . 634
10.15.3 Typedef Documentation . . . 634
10.15.3.1 vtss_reg_read_t . . . 634
10.15.3.2 vtss_reg_write_t . . . 634
10.15.3.3 vtss_i2c_read_t . . . 635

10.15.3.4 vtss_i2c_write_t . . . 635
10.15.3.5 vtss_spi_read_write_t . . . 636
10.15.3.6 vtss_spi_32bit_read_write_t . . . 636
10.15.3.7 vtss_spi_64bit_read_write_t . . . 637
10.15.3.8 vtss_miim_read_t . . . 637
10.15.3.9 vtss_miim_write_t . . . 638
10.15.3.10 vtss_mmd_read_t . . . 638
10.15.3.11 vtss_mmd_read_inc_t . . . 639
10.15.3.12 vtss_mmd_write_t . . . 639
10.15.4 Enumeration Type Documentation . . . 640
10.15.4.1 vtss_target_type_t . . . 640
10.15.4.2 vtss_restart_t . . . 641
10.15.5 Function Documentation . . . 641
10.15.5.1 vtss_inst_get() . . . 641
10.15.5.2 vtss_inst_create() . . . 641
10.15.5.3 vtss_inst_destroy() . . . 642
10.15.5.4 vtss_init_conf_get() . . . 642
10.15.5.5 vtss_init_conf_set() . . . 642
10.15.5.6 vtss_restart_conf_end() . . . 643
10.15.5.7 vtss_restart_status_get() . . . 643
10.15.5.8 vtss_restart_conf_get() . . . 644
10.15.5.9 vtss_restart_conf_set() . . . 644
10.16 vtss_api/include/vtss_l2_api.h File Reference . . . 644
10.16.1 Detailed Description . . . 651
10.16.2 Macro Definition Documentation . . . 652
10.16.2.1 VTSS_MAC_ADDRS . . . 652
10.16.2.2 VTSS_MSTIS . . . 652
10.16.2.3 VTSS_MSTI_START . . . 652
10.16.2.4 VTSS_MSTI_END . . . 652
10.16.2.5 VTSS_MSTI_ARRAY_SIZE . . . 652

10.16.2.6 VTSS_VCL_IDS . . . 653
10.16.2.7 VTSS_VCL_ID_START . . . 653
10.16.2.8 VTSS_VCL_ID_END . . . 653
10.16.2.9 VTSS_VCL_ARRAY_SIZE . . . 653
10.16.2.10 VTSS_VCE_ID_LAST . . . 653
10.16.2.11 VTSS_VLAN_TRANS_GROUP_MAX_CNT . . . 654
10.16.2.12 VTSS_VLAN_TRANS_MAX_CNT . . . 654
10.16.2.13 VTSS_VLAN_TRANS_NULL_GROUP_ID . . . 654
10.16.2.14 VTSS_VLAN_TRANS_FIRST_GROUP_ID . . . 654
10.16.2.15 VTSS_VLAN_TRANS_VID_START . . . 654
10.16.2.16 VTSS_VLAN_TRANS_MAX_VLAN_ID . . . 655
10.16.2.17 VTSS_VLAN_TRANS_LAST_GROUP_ID . . . 655
10.16.2.18 VTSS_VLAN_TRANS_VALID_GROUP_CHECK . . . 655
10.16.2.19 VTSS_VLAN_TRANS_VALID_VLAN_CHECK . . . 655
10.16.2.20 VTSS_VLAN_TRANS_NULL_CHECK . . . 656
10.16.2.21 VTSS_VLAN_TRANS_PORT_BF_SIZE . . . 656
10.16.2.22 VTSS_PVLANS . . . 656
10.16.2.23 VTSS_PVLAN_NO_START . . . 656
10.16.2.24 VTSS_PVLAN_NO_END . . . 656
10.16.2.25 VTSS_PVLAN_ARRAY_SIZE . . . 657
10.16.2.26 VTSS_PVLAN_NO_DEFAULT . . . 657
10.16.2.27 VTSS_ERPIS . . . 657
10.16.2.28 VTSS_ERPI_START . . . 657
10.16.2.29 VTSS_ERPI_END . . . 657
10.16.2.30 VTSS_ERPI_ARRAY_SIZE . . . 658
10.16.3 Typedef Documentation . . . 658
10.16.3.1 vtss_vt_id_t . . . 658
10.16.4 Enumeration Type Documentation . . . 658
10.16.4.1 vtss_stp_state_t . . . 658
10.16.4.2 vtss_vlan_port_type_t . . . 658

10.16.4.3 vtss_vlan_tx_tag_t . . . 659
10.16.4.4 vtss_vce_type_t . . . 659
10.16.4.5 vtss_eps_port_type_t . . . 660
10.16.4.6 vtss_eps_selector_t . . . 660
10.16.4.7 vtss_erps_state_t . . . 660
10.16.5 Function Documentation . . . 660
10.16.5.1 vtss_mac_table_add() . . . 661
10.16.5.2 vtss_mac_table_del() . . . 661
10.16.5.3 vtss_mac_table_get() . . . 661
10.16.5.4 vtss_mac_table_get_next() . . . 662
10.16.5.5 vtss_mac_table_age_time_get() . . . 662
10.16.5.6 vtss_mac_table_age_time_set() . . . 663
10.16.5.7 vtss_mac_table_age() . . . 663
10.16.5.8 vtss_mac_table_vlan_age() . . . 663
10.16.5.9 vtss_mac_table_flush() . . . 664
10.16.5.10 vtss_mac_table_port_flush() . . . 664
10.16.5.11 vtss_mac_table_vlan_flush() . . . 665
10.16.5.12 vtss_mac_table_vlan_port_flush() . . . 665
10.16.5.13 vtss_mac_table_status_get() . . . 665
10.16.5.14 vtss_learn_port_mode_get() . . . 666
10.16.5.15 vtss_learn_port_mode_set() . . . 666
10.16.5.16 vtss_port_state_get() . . . 667
10.16.5.17 vtss_port_state_set() . . . 667
10.16.5.18 vtss_stp_port_state_get() . . . 668
10.16.5.19 vtss_stp_port_state_set() . . . 668
10.16.5.20 vtss_mstp_vlan_msti_get() . . . 668
10.16.5.21 vtss_mstp_vlan_msti_set() . . . 669
10.16.5.22 vtss_mstp_port_msti_state_get() . . . 669
10.16.5.23 vtss_mstp_port_msti_state_set() . . . 670
10.16.5.24 vtss_vlan_conf_get() . . . 670

10.16.5.25 vtss_vlan_conf_set() . . . 671
10.16.5.26 vtss_vlan_port_conf_get() . . . 671
10.16.5.27 vtss_vlan_port_conf_set() . . . 671
10.16.5.28 vtss_vlan_port_members_get() . . . 672
10.16.5.29 vtss_vlan_port_members_set() . . . 672
10.16.5.30 vtss_vlan_vid_conf_get() . . . 673
10.16.5.31 vtss_vlan_vid_conf_set() . . . 673
10.16.5.32 vtss_vlan_tx_tag_get() . . . 674
10.16.5.33 vtss_vlan_tx_tag_set() . . . 674
10.16.5.34 vtss_vcl_port_conf_get() . . . 674
10.16.5.35 vtss_vcl_port_conf_set() . . . 675
10.16.5.36 vtss_vce_init() . . . 675
10.16.5.37 vtss_vce_add() . . . 676
10.16.5.38 vtss_vce_del() . . . 676
10.16.5.39 vtss_vlan_trans_group_add() . . . 677
10.16.5.40 vtss_vlan_trans_group_del() . . . 677
10.16.5.41 vtss_vlan_trans_group_get() . . . 677
10.16.5.42 vtss_vlan_trans_group_to_port_set() . . . 678
10.16.5.43 vtss_vlan_trans_group_to_port_get() . . . 678
10.16.5.44 vtss_vcap_port_conf_get() . . . 679
10.16.5.45 vtss_vcap_port_conf_set() . . . 679
10.16.5.46 vtss_isolated_vlan_get() . . . 680
10.16.5.47 vtss_isolated_vlan_set() . . . 680
10.16.5.48 vtss_isolated_port_members_get() . . . 680
10.16.5.49 vtss_isolated_port_members_set() . . . 681
10.16.5.50 vtss_pvlan_port_members_get() . . . 681
10.16.5.51 vtss_pvlan_port_members_set() . . . 682
10.16.5.52 vtss_apvlan_port_members_get() . . . 682
10.16.5.53 vtss_apvlan_port_members_set() . . . 682
10.16.5.54 vtss_dgroup_port_conf_get() . . . 683

10.16.5.55 vtss_dgroup_port_conf_set() 683
10.16.5.56 vtss_sflow_port_conf_get() 684
10.16.5.57 vtss_sflow_port_conf_set() 684
10.16.5.58 vtss_sflow_sampling_rate_convert() 685
10.16.5.59 vtss_aggr_port_members_get() 685
10.16.5.60 vtss_aggr_port_members_set() 686
10.16.5.61 vtss_aggr_mode_get() 686
10.16.5.62 vtss_aggr_mode_set() 686
10.16.5.63 vtss_mirror_conf_get() 687
10.16.5.64 vtss_mirror_conf_set() 687
10.16.5.65 vtss_mirror_monitor_port_get() 688
10.16.5.66 vtss_mirror_monitor_port_set() 688
10.16.5.67 vtss_mirror_ingress_ports_get() 688
10.16.5.68 vtss_mirror_ingress_ports_set() 689
10.16.5.69 vtss_mirror_egress_ports_get() 689
10.16.5.70 vtss_mirror_egress_ports_set() 690
10.16.5.71 vtss_mirror_cpu_ingress_get() 690
10.16.5.72 vtss_mirror_cpu_ingress_set() 690
10.16.5.73 vtss_mirror_cpu_egress_get() 691
10.16.5.74 vtss_mirror_cpu_egress_set() 691
10.16.5.75 vtss_uc_flood_members_get() 692
10.16.5.76 vtss_uc_flood_members_set() 692
10.16.5.77 vtss_mc_flood_members_get() 692
10.16.5.78 vtss_mc_flood_members_set() 693
10.16.5.79 vtss_ipv4_mc_flood_members_get() 693
10.16.5.80 vtss_ipv4_mc_flood_members_set() 694
10.16.5.81 vtss_ipv6_mc_flood_members_get() 694
10.16.5.82 vtss_ipv6_mc_flood_members_set() 694
10.16.5.83 vtss_ipv6_mc_ctrl_flood_get() 695
10.16.5.84 vtss_ipv6_mc_ctrl_flood_set() 695

10.16.5.85 vtss_eps_port_conf_get() . . . 695
10.16.5.86 vtss_eps_port_conf_set() . . . 696
10.16.5.87 vtss_eps_port_selector_get() . . . 696
10.16.5.88 vtss_eps_port_selector_set() . . . 697
10.16.5.89 vtss_erps_vlan_member_get() . . . 697
10.16.5.90 vtss_erps_vlan_member_set() . . . 698
10.16.5.91 vtss_erps_port_state_get() . . . 698
10.16.5.92 vtss_erps_port_state_set() . . . 699
10.17 vtss_api/include/vtss_l3_api.h File Reference . . . 699
10.17.1 Detailed Description . . . 699
10.18 vtss_api/include/vtss_mac10g_api.h File Reference . . . 699
10.18.1 Detailed Description . . . 699
10.19 vtss_api/include/vtss_misc_api.h File Reference . . . 700
10.19.1 Detailed Description . . . 705
10.19.2 Macro Definition Documentation . . . 705
10.19.2.1 VTSS_OS_TIMESTAMP_TYPE . . . 706
10.19.2.2 VTSS_OS_TIMESTAMP . . . 706
10.19.3 Typedef Documentation . . . 706
10.19.3.1 tod_get_ns_cnt_cb_t . . . 706
10.19.4 Enumeration Type Documentation . . . 706
10.19.4.1 vtss_trace_layer_t . . . 706
10.19.4.2 vtss_trace_group_t . . . 707
10.19.4.3 vtss_trace_level_t . . . 707
10.19.4.4 vtss_debug_layer_t . . . 708
10.19.4.5 vtss_debug_group_t . . . 708
10.19.4.6 vtss_gpio_mode_t . . . 709
10.19.4.7 vtss_sgpio_mode_t . . . 710
10.19.4.8 vtss_sgpio_bmode_t . . . 710
10.19.4.9 vtss_irq_t . . . 710
10.19.4.10 vtss_eee_state_select_t . . . 711

10.19.5 Function Documentation . . . 711
10.19.5.1 vtss_trace_conf_get() . . . 711
10.19.5.2 vtss_trace_conf_set() . . . 712
10.19.5.3 vtss_callout_trace_printf() . . . 712
10.19.5.4 vtss_callout_trace_hex_dump() . . . 713
10.19.5.5 vtss_debug_info_get() . . . 713
10.19.5.6 vtss_debug_info_print() . . . 714
10.19.5.7 vtss_callout_lock() . . . 714
10.19.5.8 vtss_callout_unlock() . . . 714
10.19.5.9 vtss_debug_lock() . . . 715
10.19.5.10 vtss_debug_unlock() . . . 715
10.19.5.11 vtss_reg_read() . . . 715
10.19.5.12 vtss_reg_write() . . . 716
10.19.5.13 vtss_reg_write_masked() . . . 716
10.19.5.14 vtss_intr_sticky_clear() . . . 717
10.19.5.15 vtss_chip_id_get() . . . 717
10.19.5.16 vtss_poll_1sec() . . . 718
10.19.5.17 vtss_ptp_event_poll() . . . 718
10.19.5.18 vtss_ptp_event_enable() . . . 719
10.19.5.19 vtss_dev_all_event_poll() . . . 719
10.19.5.20 vtss_dev_all_event_enable() . . . 720
10.19.5.21 vtss_gpio_mode_set() . . . 720
10.19.5.22 vtss_gpio_direction_set() . . . 720
10.19.5.23 vtss_gpio_read() . . . 721
10.19.5.24 vtss_gpio_write() . . . 721
10.19.5.25 vtss_gpio_event_poll() . . . 722
10.19.5.26 vtss_gpio_event_enable() . . . 722
10.19.5.27 vtss_sgpio_conf_get() . . . 723
10.19.5.28 vtss_sgpio_conf_set() . . . 723
10.19.5.29 vtss_sgpio_read() . . . 724

10.19.5.30 vtss_sgpio_event_poll() . . . 724
10.19.5.31 vtss_sgpio_event_enable() . . . 725
10.19.5.32 vtss_intr_cfg() . . . 725
10.19.5.33 vtss_intr_mask_set() . . . 726
10.19.5.34 vtss_intr_status_get() . . . 726
10.19.5.35 vtss_intr_pol_negation() . . . 726
10.19.5.36 vtss_irq_conf_get() . . . 727
10.19.5.37 vtss_irq_conf_set() . . . 727
10.19.5.38 vtss_irq_status_get_and_mask() . . . 728
10.19.5.39 vtss_irq_enable() . . . 728
10.19.5.40 vtss_tod_get_ns_cnt() . . . 728
10.19.5.41 vtss_tod_set_ns_cnt_cb() . . . 729
10.19.5.42 vtss_temp_sensor_init() . . . 729
10.19.5.43 vtss_temp_sensor_get() . . . 729
10.19.5.44 vtss_fan_rotation_get() . . . 730
10.19.5.45 vtss_fan_cool_lvl_set() . . . 730
10.19.5.46 vtss_fan_controller_init() . . . 730
10.19.5.47 vtss_fan_cool_lvl_get() . . . 731
10.19.5.48 vtss_eee_port_conf_set() . . . 731
10.19.5.49 vtss_eee_port_state_set() . . . 732
10.19.5.50 vtss_eee_port_counter_get() . . . 732
10.19.5.51 vtss_debug_reg_check_set() . . . 733
10.20 vtss_api/include/vtss_mpls_api.h File Reference . . . 733
10.20.1 Detailed Description . . . 736
10.20.2 Macro Definition Documentation . . . 736
10.20.2.1 VTSS_MPLS_LABEL_VALUE_DONTCARE . . . 736
10.20.2.2 VTSS_MPLS_TC_VALUE_DONTCARE . . . 736
10.20.2.3 VTSS_MPLS_IDX_UNDEFINED . . . 736
10.20.2.4 VTSS_MPLS_IDX_IS_UNDEF . . . 736
10.20.2.5 VTSS_MPLS_IDX_IS_DEF . . . 737

10.20.2.6 VTSS_MPLS_IDX_UNDEF . . . 737
10.20.2.7 VTSS_MPLS_QOS_TO_TC_MAP_CNT . . . 737
10.20.2.8 VTSS_MPLS_QOS_TO_TC_ENTRY_CNT . . . 737
10.20.2.9 VTSS_MPLS_TC_TO_QOS_MAP_CNT . . . 737
10.20.2.10 VTSS_MPLS_TC_TO_QOS_ENTRY_CNT . . . 738
10.20.3 Typedef Documentation . . . 738
10.20.3.1 vtss_mpls_label_value_t . . . 738
10.20.3.2 vtss_mpls_tc_t . . . 738
10.20.3.3 vtss_mpls_cos_t . . . 738
10.20.3.4 vtss_mpls_qos_t . . . 738
10.20.3.5 vtss_mpls_segment_idx_t . . . 739
10.20.3.6 vtss_mpls_xc_idx_t . . . 739
10.20.3.7 vtss_mpls_l2_idx_t . . . 739
10.20.4 Enumeration Type Documentation . . . 739
10.20.4.1 vtss_mll_tagtype_t . . . 739
10.20.4.2 vtss_mpls_segment_state_t . . . 740
10.20.4.3 vtss_mpls_xc_type_t . . . 740
10.20.4.4 vtss_mpls_tunnel_mode_t . . . 740
10.20.4.5 vtss_mpls_oam_type_t . . . 741
10.20.5 Function Documentation . . . 741
10.20.5.1 vtss_mpls_l2_alloc() . . . 741
10.20.5.2 vtss_mpls_l2_free() . . . 742
10.20.5.3 vtss_mpls_l2_get() . . . 742
10.20.5.4 vtss_mpls_l2_set() . . . 743
10.20.5.5 vtss_mpls_l2_segment_attach() . . . 743
10.20.5.6 vtss_mpls_l2_segment_detach() . . . 744
10.20.5.7 vtss_mpls_segment_alloc() . . . 744
10.20.5.8 vtss_mpls_segment_free() . . . 745
10.20.5.9 vtss_mpls_segment_get() . . . 745
10.20.5.10 vtss_mpls_segment_set() . . . 745

10.20.5.11 vtss_mpls_segment_status_get() . . . 746
10.20.5.12 vtss_mpls_segment_server_attach() . . . 747
10.20.5.13 vtss_mpls_segment_server_detach() . . . 747
10.20.5.14 vtss_mpls_xc_alloc() . . . 748
10.20.5.15 vtss_mpls_xc_free() . . . 748
10.20.5.16 vtss_mpls_xc_get() . . . 748
10.20.5.17 vtss_mpls_xc_set() . . . 750
10.20.5.18 vtss_mpls_xc_segment_attach() . . . 750
10.20.5.19 vtss_mpls_xc_segment_detach() . . . 751
10.20.5.20 vtss_mpls_tc_conf_get() . . . 751
10.20.5.21 vtss_mpls_tc_conf_set() . . . 752
10.20.5.22 vtss_mpls_xc_counters_get() . . . 752
10.20.5.23 vtss_mpls_xc_counters_clear() . . . 753
10.21 vtss_api/include/vtss_oam_api.h File Reference . . . 753
10.21.1 Detailed Description . . . 757
10.21.2 Macro Definition Documentation . . . 757
10.21.2.1 VTSS_OAM_PATH_SERVICE_VOE_CNT . . . 757
10.21.2.2 VTSS_OAM_PORT_VOE_CNT . . . 757
10.21.2.3 VTSS_OAM_PORT_VOE_BASE_IDX . . . 758
10.21.2.4 VTSS_OAM_VOE_CNT . . . 758
10.21.2.5 VTSS_OAM_GENERIC_OPCODE_CFG_CNT . . . 758
10.21.2.6 VTSS_OAM_VOE_IDX_NONE . . . 758
10.21.2.7 VTSS_OAM_EVENT_MASK_ARRAY_SIZE . . . 758
10.21.2.8 VTSS_OAM_VOE_EVENT_CCM_PERIOD . . . 759
10.21.2.9 VTSS_OAM_VOE_EVENT_CCM_PRIORITY . . . 759
10.21.2.10 VTSS_OAM_VOE_EVENT_CCM_ZERO_PERIOD . . . 759
10.21.2.11 VTSS_OAM_VOE_EVENT_CCM_RX_RDI . . . 759
10.21.2.12 VTSS_OAM_VOE_EVENT_CCM_LOC . . . 759
10.21.2.13 VTSS_OAM_VOE_EVENT_CCM_MEP_ID . . . 760
10.21.2.14 VTSS_OAM_VOE_EVENT_CCM_MEG_ID . . . 760

10.21.2.15 VTSS_OAM_VOE_EVENT_MEG_LEVEL . . . 760
10.21.2.16 VTSS_OAM_VOE_EVENT_MASK . . . 760
10.21.2.17 VTSS_OAM_CNT_CCM . . . 760
10.21.2.18 VTSS_OAM_CNT_LM . . . 761
10.21.2.19 VTSS_OAM_CNT_LB . . . 761
10.21.2.20 VTSS_OAM_CNT_TST . . . 761
10.21.2.21 VTSS_OAM_CNT_SEL . . . 761
10.21.2.22 VTSS_OAM_CNT_ALL . . . 761
10.21.3 Function Documentation . . . 761
10.21.3.1 vtss_oam_vop_conf_get() . . . 761
10.21.3.2 vtss_oam_vop_conf_set() . . . 762
10.21.3.3 vtss_oam_event_poll() . . . 762
10.21.3.4 vtss_oam_voe_alloc() . . . 763
10.21.3.5 vtss_oam_voe_free() . . . 763
10.21.3.6 vtss_oam_voe_conf_get() . . . 763
10.21.3.7 vtss_oam_voe_conf_set() . . . 764
10.21.3.8 vtss_oam_voe_event_enable() . . . 764
10.21.3.9 vtss_oam_voe_event_poll() . . . 765
10.21.3.10 vtss_oam_voe_ccm_arm_hitme() . . . 765
10.21.3.11 vtss_oam_voe_ccm_set_rdi_flag() . . . 766
10.21.3.12 vtss_oam_ccm_status_get() . . . 766
10.21.3.13 vtss_oam_pdu_seen_status_get() . . . 766
10.21.3.14 vtss_oam_proc_status_get() . . . 767
10.21.3.15 vtss_oam_voe_counter_update() . . . 767
10.21.3.16 vtss_oam_voe_counter_update_serval() . . . 768
10.21.3.17 vtss_oam_voe_counter_get() . . . 768
10.21.3.18 vtss_oam_voe_counter_clear() . . . 769
10.21.3.19 vtss_oam_ipt_conf_get() . . . 769
10.21.3.20 vtss_oam_ipt_conf_set() . . . 769
10.22 vtss_api/include/vtss_oha_api.h File Reference . . . 770

10.22.1 Detailed Description . . . 770
10.23 vtss_api/include/vtss_os.h File Reference . . . 770
10.23.1 Detailed Description . . . 770
10.24 vtss_api/include/vtss_os_custom.h File Reference . . . 770
10.24.1 Detailed Description . . . 771
10.24.2 Macro Definition Documentation . . . 771
10.24.2.1 uint . . . 771
10.24.2.2 ulong . . . 771
10.24.2.3 VTSS_MSLEEP . . . 772
10.24.2.4 VTSS_MTIMER_START . . . 772
10.24.2.5 VTSS_MTIMER_TIMEOUT . . . 772
10.24.2.6 VTSS_MTIMER_CANCEL . . . 772
10.24.2.7 VTSS_DIV64 . . . 772
10.24.2.8 VTSS_MOD64 . . . 773
10.24.2.9 VTSS_LABS . . . 773
10.24.2.10 VTSS_LLABS . . . 773
10.24.2.11 VTSS_OS_CTZ . . . 773
10.24.2.12 VTSS_OS_CTZ64 . . . 774
10.24.2.13 VTSS_OS_MALLOC . . . 774
10.24.2.14 VTSS_OS_FREE . . . 774
10.24.2.15 VTSS_OS_RAND . . . 774
10.24.3 Typedef Documentation . . . 775
10.24.3.1 vtss_mtimer_t . . . 775
10.25 vtss_api/include/vtss_os_ecos.h File Reference . . . 775
10.25.1 Detailed Description . . . 776
10.25.2 Macro Definition Documentation . . . 776
10.25.2.1 VTSS_MSLEEP . . . 776
10.25.2.2 VTSS_NSLEEP . . . 777
10.25.2.3 VTSS_MTIMER_START . . . 777
10.25.2.4 VTSS_MTIMER_TIMEOUT . . . 777

10.25.2.5 VTSS_MTIMER_CANCEL . . . 777
10.25.2.6 VTSS_TIME_OF_DAY . . . 777
10.25.2.7 VTSS_DIV64 . . . 778
10.25.2.8 VTSS_MOD64 . . . 778
10.25.2.9 VTSS_LABS . . . 778
10.25.2.10 VTSS_LLABS . . . 778
10.25.2.11 VTSS_OS_CTZ . . . 779
10.25.2.12 VTSS_OS_CTZ64 . . . 779
10.25.2.13 VTSS_OS_MALLOC . . . 779
10.25.2.14 VTSS_OS_FREE . . . 780
10.25.2.15 VTSS_OS_RAND . . . 780
10.25.2.16 VTSS_OS_REORDER_BARRIER . . . 780
10.25.2.17 VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED . . . 780
10.25.2.18 VTSS_OS_DCACHE_LINE_SIZE_BYTES . . . 781
10.25.2.19 VTSS_OS_DCACHE_INVALIDATE . . . 781
10.25.2.20 VTSS_OS_DCACHE_FLUSH . . . 781
10.25.2.21 VTSS_OS_VIRT_TO_PHYS . . . 781
10.25.2.22 VTSS_OS_BIG_ENDIAN . . . 782
10.25.2.23 VTSS_OS_NTOHL . . . 782
10.25.2.24 VTSS_OS_SCHEDULER_FLAGS . . . 782
10.25.2.25 VTSS_OS_SCHEDULER_LOCK . . . 782
10.25.2.26 VTSS_OS_SCHEDULER_UNLOCK . . . 783
10.25.2.27 VTSS_OS_INTERRUPT_FLAGS . . . 783
10.25.2.28 VTSS_OS_INTERRUPT_DISABLE . . . 783
10.25.2.29 VTSS_OS_INTERRUPT_RESTORE . . . 783
10.25.3 Typedef Documentation . . . 783
10.25.3.1 vtss_mtimer_t . . . 784
10.25.4 Function Documentation . . . 784
10.25.4.1 llabs() . . . 784
10.25.4.2 vtss_callout_malloc() . . . 784

10.25.4.3 vtss_callout_free() . . . 785
10.26 vtss_api/include/vtss_os_linux.h File Reference . . . 785
10.26.1 Detailed Description . . . 786
10.26.2 Macro Definition Documentation . . . 786
10.26.2.1 VTSS_OS_BIG_ENDIAN . . . 786
10.26.2.2 VTSS_OS_NTOHL . . . 786
10.26.2.3 VTSS_NSLEEP . . . 786
10.26.2.4 VTSS_MSLEEP . . . 787
10.26.2.5 VTSS_MTIMER_START . . . 787
10.26.2.6 VTSS_MTIMER_TIMEOUT . . . 788
10.26.2.7 VTSS_MTIMER_CANCEL . . . 788
10.26.2.8 VTSS_TIME_OF_DAY . . . 788
10.26.2.9 VTSS_OS_SCHEDULER_FLAGS . . . 788
10.26.2.10 VTSS_OS_SCHEDULER_LOCK . . . 789
10.26.2.11 VTSS_OS_SCHEDULER_UNLOCK . . . 789
10.26.2.12 VTSS_DIV64 . . . 789
10.26.2.13 VTSS_MOD64 . . . 789
10.26.2.14 VTSS_LABS . . . 790
10.26.2.15 VTSS_LLABS . . . 790
10.26.2.16 VTSS_OS_CTZ . . . 790
10.26.2.17 VTSS_OS_CTZ64 . . . 791
10.26.2.18 VTSS_OS_MALLOC . . . 791
10.26.2.19 VTSS_OS_FREE . . . 791
10.26.2.20 VTSS_OS_RAND . . . 792
10.27 vtss_api/include/vtss_otn_api.h File Reference . . . 792
10.27.1 Detailed Description . . . 792
10.28 vtss_api/include/vtss_packet_api.h File Reference . . . 792
10.28.1 Detailed Description . . . 795
10.28.2 Macro Definition Documentation . . . 795
10.28.2.1 VTSS_PRIO_SUPER . . . 796

10.28.2.2 VTSS_AFI_SLOT_CNT . . . 796
10.28.2.3 VTSS_AFI_ID_NONE . . . 796
10.28.2.4 VTSS_AFI_FPS_MAX . . . 796
10.28.2.5 VTSS_JR1_PACKET_HDR_SIZE_BYTES . . . 796
10.28.2.6 VTSS_JR2_PACKET_HDR_SIZE_BYTES . . . 797
10.28.2.7 VTSS_SVL_PACKET_HDR_SIZE_BYTES . . . 797
10.28.2.8 VTSS_L26_PACKET_HDR_SIZE_BYTES . . . 797
10.28.2.9 VTSS_PACKET_HDR_SIZE_BYTES . . . 797
10.28.2.10 VTSS_SVL_RX_IFH_SIZE . . . 797
10.28.2.11 VTSS_JR1_RX_IFH_SIZE . . . 798
10.28.2.12 VTSS_L26_RX_IFH_SIZE . . . 798
10.28.2.13 VTSS_JR2_RX_IFH_SIZE . . . 798
10.28.2.14 VTSS_PACKET_TX_IFH_MAX . . . 798
10.28.3 Enumeration Type Documentation . . . 798
10.28.3.1 vtss_packet_filter_t . . . 798
10.28.3.2 vtss_packet_oam_type_t . . . 799
10.28.3.3 vtss_packet_ptp_action_t . . . 799
10.28.3.4 vtss_tag_type_t . . . 800
10.28.3.5 vtss_packet_rx_hints_t . . . 800
10.28.3.6 vtss_packet_tx_vstax_t . . . 801
10.28.4 Function Documentation . . . 801
10.28.4.1 vtss_npi_conf_get() . . . 802
10.28.4.2 vtss_npi_conf_set() . . . 802
10.28.4.3 vtss_packet_rx_conf_get() . . . 802
10.28.4.4 vtss_packet_rx_conf_set() . . . 803
10.28.4.5 vtss_packet_rx_port_conf_get() . . . 803
10.28.4.6 vtss_packet_rx_port_conf_set() . . . 803
10.28.4.7 vtss_packet_rx_frame_get() . . . 804
10.28.4.8 vtss_packet_rx_frame_get_raw() . . . 805
10.28.4.9 vtss_packet_rx_frame_discard() . . . 805

10.28.4.10 vtss_packet_tx_frame_port() . . . 806
10.28.4.11 vtss_packet_tx_frame_port_vlan() . . . 806
10.28.4.12 vtss_packet_tx_frame_vlan() . . . 807
10.28.4.13 vtss_packet_frame_info_init() . . . 807
10.28.4.14 vtss_packet_frame_filter() . . . 807
10.28.4.15 vtss_packet_port_info_init() . . . 808
10.28.4.16 vtss_packet_port_filter_get() . . . 808
10.28.4.17 vtss_afi_alloc() . . . 809
10.28.4.18 vtss_afi_free() . . . 809
10.28.4.19 vtss_packet_rx_hdr_decode() . . . 810
10.28.4.20 vtss_packet_tx_hdr_encode() . . . 810
10.28.4.21 vtss_packet_tx_hdr_compile() . . . 811
10.28.4.22 vtss_packet_tx_frame() . . . 811
10.28.4.23 vtss_packet_tx_info_init() . . . 812
10.28.4.24 vtss_packet_dma_conf_get() . . . 812
10.28.4.25 vtss_packet_dma_conf_set() . . . 813
10.28.4.26 vtss_packet_dma_offset() . . . 813
10.29 vtss_api/include/vtss_pcs_10gbase_r_api.h File Reference . . . 814
10.29.1 Detailed Description . . . 814
10.30 vtss_api/include/vtss_phy_10g_api.h File Reference . . . 814
10.30.1 Detailed Description . . . 814
10.31 vtss_api/include/vtss_phy_api.h File Reference . . . 814
10.31.1 Detailed Description . . . 824
10.31.2 Macro Definition Documentation . . . 824
10.31.2.1 MAX_CFG_BUF_SIZE . . . 824
10.31.2.2 MAX_STAT_BUF_SIZE . . . 824
10.31.2.3 VTSS_PHY_POWER_ACTIPHY_BIT . . . 825
10.31.2.4 VTSS_PHY_POWER_DYNAMIC_BIT . . . 825
10.31.2.5 VTSS_PHY_ACTIPHY_PWR . . . 825
10.31.2.6 VTSS_PHY_LINK_DOWN_PWR . . . 825

10.31.2.7 VTSS_PHY_LINK_UP_FULL_PWR 825
10.31.2.8 VTSS_PHY_RECOV_CLK1 826
10.31.2.9 VTSS_PHY_RECOV_CLK2 826
10.31.2.10 VTSS_PHY_RECOV_CLK_NUM 826
10.31.2.11 VTSS_PHY_PAGE_STANDARD 826
10.31.2.12 VTSS_PHY_PAGE_EXTENDED 826
10.31.2.13 VTSS_PHY_PAGE_EXTENDED_2 827
10.31.2.14 VTSS_PHY_PAGE_EXTENDED_3 827
10.31.2.15 VTSS_PHY_PAGE_EXTENDED_4 827
10.31.2.16 VTSS_PHY_PAGE_GPIO 827
10.31.2.17 VTSS_PHY_PAGE_1588 827
10.31.2.18 VTSS_PHY_PAGE_MACSEC 828
10.31.2.19 VTSS_PHY_PAGE_TEST 828
10.31.2.20 VTSS_PHY_PAGE_TR 828
10.31.2.21 VTSS_PHY_PAGE_0x2DAF 828
10.31.2.22 VTSS_PHY_REG_STANDARD 828
10.31.2.23 VTSS_PHY_REG_EXTENDED 829
10.31.2.24 VTSS_PHY_REG_GPIO 829
10.31.2.25 VTSS_PHY_REG_TEST 829
10.31.2.26 VTSS_PHY_REG_TR 829
10.31.2.27 VTSS_PHY_LINK_LOS_EV 829
10.31.2.28 VTSS_PHY_LINK_FFAIL_EV 830
10.31.2.29 VTSS_PHY_LINK_AMS_EV 830
10.31.2.30 VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV 830
10.31.2.31 VTSS_PHY_LINK_FDX_STATE_CHANGE_EV 830
10.31.2.32 VTSS_PHY_LINK_AUTO_NEG_ERROR_EV 830
10.31.2.33 VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV 831
10.31.2.34 VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV 831
10.31.2.35 VTSS_PHY_LINK_SYMBOL_ERR_INT_EV 831
10.31.2.36 VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV 831

10.31.2.37 VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV 831
10.31.2.38 VTSS_PHY_LINK_FALSE_CARRIER_INT_EV 832
10.31.2.39 VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV 832
10.31.2.40 VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV 832
10.31.2.41 VTSS_PHY_LINK_RX_ER_INT_EV 832
10.31.2.42 VTSS_PHY_LINK_EXTENDED_REG_INT_EV 832
10.31.2.43 VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV 833
10.31.2.44 VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV 833
10.31.2.45 VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV 833
10.31.2.46 VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV 833
10.31.2.47 VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV 833
10.31.2.48 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV 834
10.31.2.49 VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV 834
10.31.2.50 VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV 834
10.31.2.51 VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV 834
10.31.2.52 VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV 834
10.31.2.53 VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV 835
10.31.2.54 VTSS_PHY_LINK_EXT_MEM_INT_RING_EV 835
10.31.2.55 MAX_WOL_MAC_ADDR_SIZE 835
10.31.2.56 MAX_WOL_PASSWD_SIZE 835
10.31.2.57 MIN_WOL_PASSWD_SIZE 835
10.31.3 Typedef Documentation 836
10.31.3.1 vtss_phy_recov_clk_t 836
10.31.3.2 vtss_phy_led_intensity 836
10.31.4 Enumeration Type Documentation 836
10.31.4.1 vtss_phy_led_mode_t 836
10.31.4.2 vtss_phy_media_interface_t 837
10.31.4.3 vtss_phy_mdi_t 838
10.31.4.4 rgmii_skew_delay_psec_t 838
10.31.4.5 vtss_phy_forced_reset_t 839

10.31.4.6 vtss_phy_pkt_mode_t . . . 839
10.31.4.7 vtss_phy_mode_t . . . 839
10.31.4.8 vtss_phy_fast_link_fail_t . . . 840
10.31.4.9 vtss_phy_sigdet_polarity_t . . . 840
10.31.4.10 vtss_phy_unidirectional_t . . . 840
10.31.4.11 vtss_phy_mac_serd_pcs_sgmii_pre . . . 840
10.31.4.12 vtss_phy_media_rem_fault_t . . . 842
10.31.4.13 vtss_phy_media_force_ams_sel_t . . . 842
10.31.4.14 vtss_phy_clk_source_t . . . 843
10.31.4.15 vtss_phy_freq_t . . . 843
10.31.4.16 vtss_phy_clk_squelch . . . 843
10.31.4.17 vtss_phy_gpio_mode_t . . . 844
10.31.4.18 lb_type . . . 844
10.31.4.19 vtss_wol_passwd_len_type_t . . . 845
10.31.4.20 vtss_fefi_mode_t . . . 845
10.31.5 Function Documentation . . . 845
10.31.5.1 vtss_phy_pre_reset() . . . 845
10.31.5.2 vtss_phy_post_reset() . . . 846
10.31.5.3 vtss_phy_pre_system_reset() . . . 846
10.31.5.4 vtss_phy_reset() . . . 847
10.31.5.5 vtss_phy_reset_get() . . . 847
10.31.5.6 vtss_phy_chip_temp_get() . . . 847
10.31.5.7 vtss_phy_chip_temp_init() . . . 848
10.31.5.8 vtss_phy_conf_get() . . . 848
10.31.5.9 vtss_phy_conf_set() . . . 849
10.31.5.10 vtss_phy_status_get() . . . 849
10.31.5.11 vtss_phy_cl37_lp_abil_get() . . . 850
10.31.5.12 vtss_phy_id_get() . . . 850
10.31.5.13 vtss_phy_conf_1g_get() . . . 850
10.31.5.14 vtss_phy_conf_1g_set() . . . 851

10.31.5.15 vtss_phy_status_1g_get() . . . 851
10.31.5.16 vtss_phy_power_conf_get() . . . 852
10.31.5.17 vtss_phy_power_conf_set() . . . 852
10.31.5.18 vtss_phy_power_status_get() . . . 853
10.31.5.19 vtss_phy_clock_conf_set() . . . 853
10.31.5.20 vtss_phy_clock_conf_get() . . . 853
10.31.5.21 vtss_phy_i2c_read() . . . 854
10.31.5.22 vtss_phy_i2c_write() . . . 855
10.31.5.23 vtss_phy_read() . . . 855
10.31.5.24 vtss_phy_read_page() . . . 856
10.31.5.25 vtss_phy_mmd_read() . . . 856
10.31.5.26 vtss_phy_mmd_write() . . . 857
10.31.5.27 vtss_phy_write() . . . 857
10.31.5.28 vtss_phy_write_masked() . . . 858
10.31.5.29 vtss_phy_write_masked_page() . . . 858
10.31.5.30 vtss_phy_gpio_mode() . . . 859
10.31.5.31 vtss_phy_gpio_get() . . . 859
10.31.5.32 vtss_phy_gpio_set() . . . 860
10.31.5.33 vtss_phy_veriphy_start() . . . 860
10.31.5.34 vtss_phy_veriphy_get() . . . 861
10.31.5.35 vtss_phy_led_mode_set() . . . 861
10.31.5.36 vtss_phy_led_intensity_set() . . . 862
10.31.5.37 vtss_phy_led_intensity_get() . . . 862
10.31.5.38 vtss_phy_enhanced_led_control_init() . . . 862
10.31.5.39 vtss_phy_enhanced_led_control_init_get() . . . 863
10.31.5.40 vtss_phy_coma_mode_disable() . . . 863
10.31.5.41 vtss_phy_coma_mode_enable() . . . 864
10.31.5.42 vga_adc_debug() . . . 864
10.31.5.43 vtss_phy_port_eee_capable() . . . 864
10.31.5.44 vtss_phy_eee_ena() . . . 865

10.31.5.45 vtss_phy_eee_conf_get() 865
10.31.5.46 vtss_phy_eee_conf_set() 866
10.31.5.47 vtss_phy_eee_power_save_state_get() 866
10.31.5.48 vtss_phy_eee_link_partner_advertisements_get() 867
10.31.5.49 vtss_phy_event_enable_set() 867
10.31.5.50 vtss_phy_event_enable_get() 868
10.31.5.51 vtss_phy_event_poll() 868
10.31.5.52 vtss_squelch_workaround() 868
10.31.5.53 vtss_phy_csr_wr() 869
10.31.5.54 vtss_phy_csr_rd() 869
10.31.5.55 vtss_phy_statistic_get() 871
10.31.5.56 vtss_phy_do_page_chk_set() 871
10.31.5.57 vtss_phy_do_page_chk_get() 872
10.31.5.58 vtss_phy_loopback_set() 872
10.31.5.59 vtss_phy_loopback_get() 872
10.31.5.60 vtss_phy_is_8051_crc_ok() 873
10.31.5.61 vtss_phy_cfg_ob_post0() 873
10.31.5.62 vtss_phy_cfg_ib_cterm() 874
10.31.5.63 vtss_phy_atom12_patch_setttings_get() 874
10.31.5.64 vtss_phy_reg_decode_status() 875
10.31.5.65 vtss_phy_flowcontrol_decode_status() 875
10.31.5.66 vtss_phy_debug_stat_print() 876
10.31.5.67 vtss_phy_warm_start_failed_get() 876
10.31.5.68 vtss_phy_debug_phyinfo_print() 877
10.31.5.69 vtss_phy_debug_register_dump() 877
10.31.5.70 vtss_phy_detect_base_ports() 878
10.31.5.71 vtss_phy_ext_connector_loopback() 878
10.31.5.72 vtss_phy_serdes_sgmii_loopback() 878
10.31.5.73 vtss_phy_serdes_fmedia_loopback() 879
10.31.5.74 vtss_phy_debug_regdump_print() 879

10.31.5.75 vtss_phy_wol_enable() . . . 880
10.31.5.76 vtss_phy_wol_conf_get() . . . 880
10.31.5.77 vtss_phy_wol_conf_set() . . . 881
10.31.5.78 vtss_phy_patch_settings_get() . . . 881
10.31.5.79 vtss_phy_serdes6g_rcpll_status_get() . . . 882
10.31.5.80 vtss_phy_serdes1g_rcpll_status_get() . . . 882
10.31.5.81 vtss_phy_lcpll_status_get() . . . 883
10.31.5.82 vtss_phy_reset_lcpll() . . . 883
10.31.5.83 vtss_phy_sd6g_ob_post_rd() . . . 884
10.31.5.84 vtss_phy_sd6g_ob_post_wr() . . . 884
10.31.5.85 vtss_phy_sd6g_ob_lev_rd() . . . 885
10.31.5.86 vtss_phy_sd6g_ob_lev_wr() . . . 885
10.31.5.87 vtss_phy_mac_media_inhibit_odd_start() . . . 886
10.31.5.88 vtss_phy_fefi_get() . . . 886
10.31.5.89 vtss_phy_fefi_set() . . . 886
10.31.5.90 vtss_phy_fefi_detect() . . . 887
10.31.5.91 vtss_phy_mse_100m_get() . . . 887
10.31.5.92 vtss_phy_mse_1000m_get() . . . 888
10.31.5.93 vtss_phy_read_tr_addr() . . . 888
10.31.5.94 vtss_phy_is_viper_revB() . . . 889
10.31.5.95 vtss_phy_ext_event_poll() . . . 889
10.31.5.96 vtss_phy_status_inst_poll() . . . 890
10.31.5.97 vtss_phy_macsec_csr_sd6g_rd() . . . 890
10.31.5.98 vtss_phy_macsec_csr_sd6g_wr() . . . 891
10.31.5.99 vtss_phy_sd6g_mac_serdes_conf() . . . 891
10.32 vtss_api/include/vtss_phy_ts_api.h File Reference . . . 891
10.32.1 Detailed Description . . . 900
10.32.2 Macro Definition Documentation . . . 900
10.32.2.1 VTSS_PHY_TS_FIFO_SIG_SRC_IP . . . 901
10.32.2.2 VTSS_PHY_TS_FIFO_SIG_DEST_IP . . . 901

10.32.2.3 VTSS_PHY_TS_FIFO_SIG_MSG_TYPE 901
10.32.2.4 VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM 901
10.32.2.5 VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID 901
10.32.2.6 VTSS_PHY_TS_FIFO_SIG_SEQ_ID 902
10.32.2.7 VTSS_PHY_TS_FIFO_SIG_DEST_MAC 902
10.32.2.8 VTSS_PHY_TS_FIFO_SIG_SH_SPORT_ID 902
10.32.2.9 VTSS_PHY_TS_SIG_LEN 902
10.32.2.10 VTSS_PHY_TS_SIG_TIME_STAMP_LEN 902
10.32.2.11 VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN 903
10.32.2.12 VTSS_PHY_TS_SIG_SEQ_ID_LEN 903
10.32.2.13 VTSS_PHY_TS_SIG_MSG_TYPE_LEN 903
10.32.2.14 VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN 903
10.32.2.15 VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN 903
10.32.2.16 VTSS_PHY_TS_SIG_DEST_IP_LEN 904
10.32.2.17 VTSS_PHY_TS_SIG_SRC_IP_LEN 904
10.32.2.18 VTSS_PHY_TS_SIG_DEST_MAC_LEN 904
10.32.2.19 VTSS_PHY_TS_SIG_SH_SPORT_ID_LEN 904
10.32.2.20 VTSS_PTP_IP_1588_VERSION_2_1 904
10.32.2.21 VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT 905
10.32.2.22 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST 905
10.32.2.23 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST 905
10.32.2.24 VTSS_PHY_TS_ETH_MATCH_DEST_ADDR 905
10.32.2.25 VTSS_PHY_TS_ETH_MATCH_SRC_ADDR 905
10.32.2.26 VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST 906
10.32.2.27 VTSS_PHY_TS_TAG_TYPE_C 906
10.32.2.28 VTSS_PHY_TS_TAG_TYPE_S 906
10.32.2.29 VTSS_PHY_TS_TAG_TYPE_I 906
10.32.2.30 VTSS_PHY_TS_TAG_TYPE_B 906
10.32.2.31 VTSS_PHY_TS_TAG_RANGE_NONE 907
10.32.2.32 VTSS_PHY_TS_TAG_RANGE_OUTER 907

10.32.2.33 VTSS_PHY_TS_TAG_RANGE_INNER 907
10.32.2.34 VTSS_PHY_TS_IP_VER_4 907
10.32.2.35 VTSS_PHY_TS_IP_VER_6 907
10.32.2.36 VTSS_PHY_TS_IP_MATCH_SRC 908
10.32.2.37 VTSS_PHY_TS_IP_MATCH_DEST 908
10.32.2.38 VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST 908
10.32.2.39 VTSS_PHY_TS_MPLS_STACK_DEPTH_1 908
10.32.2.40 VTSS_PHY_TS_MPLS_STACK_DEPTH_2 908
10.32.2.41 VTSS_PHY_TS_MPLS_STACK_DEPTH_3 909
10.32.2.42 VTSS_PHY_TS_MPLS_STACK_DEPTH_4 909
10.32.2.43 VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP 909
10.32.2.44 VTSS_PHY_TS_MPLS_STACK_REF_POINT_END 909
10.32.2.45 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0 909
10.32.2.46 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1 910
10.32.2.47 VTSS_PHY_TS_INGR_ENGINE_ERR 910
10.32.2.48 VTSS_PHY_TS_INGR_RW_PREAM_ERR 910
10.32.2.49 VTSS_PHY_TS_INGR_RW_FCS_ERR 910
10.32.2.50 VTSS_PHY_TS_EGR_ENGINE_ERR 910
10.32.2.51 VTSS_PHY_TS_EGR_RW_FCS_ERR 911
10.32.2.52 VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED 911
10.32.2.53 VTSS_PHY_TS_EGR_FIFO_OVERFLOW 911
10.32.2.54 VTSS_PHY_TS_DATA_IN_RSRVD_FIELD 911
10.32.2.55 VTSS_PHY_TS_LTC_NEW_PPS_INTRPT 911
10.32.2.56 VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD 912
10.32.2.57 VTSS_PHY_TS_8487_XAUI_SEL_0 912
10.32.2.58 VTSS_PHY_TS_8487_XAUI_SEL_1 912
10.32.2.59 VTSS_PHY_TS_INGR_DATAPATH_RESET 912
10.32.2.60 VTSS_PHY_TS_EGR_DATAPATH_RESET 912
10.32.2.61 VTSS_PHY_TS_INGR_LTC1_RESET 913
10.32.2.62 VTSS_PHY_TS_EGR_LTC2_RESET 913

10.32.2.63 VTSS_PHY_TS_EGR_FIFO_RESET . . . 913
10.32.3 Typedef Documentation . . . 913
10.32.3.1 vtss_phy_ts_alt_clock_mode_t . . . 913
10.32.3.2 vtss_phy_ts_scaled_ppb_t . . . 913
10.32.3.3 vtss_phy_ts_fifo_sig_mask_t . . . 914
10.32.3.4 vtss_phy_ts_fifo_read . . . 914
10.32.3.5 vtss_phy_ts_engine_channel_map_t . . . 914
10.32.3.6 vtss_phy_ts_event_t . . . 914
10.32.3.7 vtss_phy_ts_8487_xaui_sel_t . . . 914
10.32.3.8 vtss_phy_ts_soft_reset_t . . . 915
10.32.4 Enumeration Type Documentation . . . 915
10.32.4.1 vtss_phy_ts_todadj_status_t . . . 915
10.32.4.2 vtss_phy_ts_fifo_status_t . . . 915
10.32.4.3 vtss_phy_ts_encap_t . . . 916
10.32.4.4 vtss_phy_ts_engine_t . . . 916
10.32.4.5 vtss_phy_ts_engine_flow_match_t . . . 916
10.32.4.6 vtss_phy_ts_ptp_clock_mode_t . . . 917
10.32.4.7 vtss_phy_ts_ptp_delaym_type_t . . . 917
10.32.4.8 vtss_phy_ts_y1731_oam_delaym_type_t . . . 918
10.32.4.9 vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t . . . 918
10.32.4.10 vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t . . . 919
10.32.4.11 vtss_phy_ts_action_format . . . 919
10.32.4.12 vtss_phy_ts_clockfreq_t . . . 919
10.32.4.13 vtss_phy_ts_clock_src_t . . . 920
10.32.4.14 vtss_phy_ts_rxtimestamp_pos_t . . . 920
10.32.4.15 vtss_phy_ts_rxtimestamp_len_t . . . 921
10.32.4.16 vtss_phy_ts_fifo_mode_t . . . 921
10.32.4.17 vtss_phy_ts_fifo_timestamp_len_t . . . 921
10.32.4.18 vtss_phy_ts_tc_op_mode_t . . . 922
10.32.4.19 vtss_phy_ts_nphase_sampler_t . . . 922

10.32.4.20 vtss_phy_ts_ptp_message_type_t . . . 923
10.32.5 Function Documentation . . . 923
10.32.5.1 vtss_phy_ts_alt_clock_saved_get() . . . 923
10.32.5.2 vtss_phy_ts_alt_clock_mode_get() . . . 923
10.32.5.3 vtss_phy_ts_alt_clock_mode_set() . . . 924
10.32.5.4 vtss_phy_ts_pps_conf_set() . . . 924
10.32.5.5 vtss_phy_ts_pps_conf_get() . . . 925
10.32.5.6 vtss_phy_ts_ingress_latency_set() . . . 925
10.32.5.7 vtss_phy_ts_ingress_latency_get() . . . 926
10.32.5.8 vtss_phy_ts_egress_latency_set() . . . 926
10.32.5.9 vtss_phy_ts_egress_latency_get() . . . 927
10.32.5.10 vtss_phy_ts_path_delay_set() . . . 927
10.32.5.11 vtss_phy_ts_path_delay_get() . . . 928
10.32.5.12 vtss_phy_ts_delay_asymmetry_set() . . . 928
10.32.5.13 vtss_phy_ts_delay_asymmetry_get() . . . 929
10.32.5.14 vtss_phy_ts_ptptime_set() . . . 929
10.32.5.15 vtss_phy_ts_ptptime_set_done() . . . 930
10.32.5.16 vtss_phy_ts_ptptime_arm() . . . 930
10.32.5.17 vtss_phy_ts_ptptime_get() . . . 931
10.32.5.18 vtss_phy_ts_load_ptptime_get() . . . 931
10.32.5.19 vtss_phy_ts_sertod_set() . . . 931
10.32.5.20 vtss_phy_ts_sertod_get() . . . 932
10.32.5.21 vtss_phy_ts_loadpulse_delay_set() . . . 932
10.32.5.22 vtss_phy_ts_loadpulse_delay_get() . . . 933
10.32.5.23 vtss_phy_ts_clock_rateadj_set() . . . 933
10.32.5.24 vtss_phy_ts_clock_rateadj_get() . . . 934
10.32.5.25 vtss_phy_ts_clock_rateadj_ppm_set() . . . 934
10.32.5.26 vtss_phy_ts_clock_rateadj_ppm_get() . . . 934
10.32.5.27 vtss_phy_ts_ptptime_adj1ns() . . . 935
10.32.5.28 vtss_phy_ts_timeofday_offset_set() . . . 935

10.32.5.29 vtss_phy_ts_ongoing_adjustment() 936
10.32.5.30 vtss_phy_ts_ltc_freq_synth_pulse_set() 936
10.32.5.31 vtss_phy_ts_ltc_freq_synth_pulse_get() 937
10.32.5.32 vtss_phy_daisy_conf_set() 937
10.32.5.33 vtss_phy_daisy_conf_get() 937
10.32.5.34 vtss_phy_ts_fifo_sig_set() 938
10.32.5.35 vtss_phy_ts_fifo_sig_get() 938
10.32.5.36 vtss_phy_ts_fifo_empty() 939
10.32.5.37 vtss_phy_ts_fifo_read_install() 939
10.32.5.38 vtss_phy_ts_fifo_read_cb_get() 940
10.32.5.39 vtss_phy_ts_ingress_engine_init() 940
10.32.5.40 vtss_phy_ts_ingress_engine_init_conf_get() 941
10.32.5.41 vtss_phy_ts_ingress_engine_clear() 941
10.32.5.42 vtss_phy_ts_egress_engine_init() 942
10.32.5.43 vtss_phy_ts_egress_engine_init_conf_get() 943
10.32.5.44 vtss_phy_ts_egress_engine_clear() 943
10.32.5.45 vtss_phy_ts_ingress_engine_conf_set() 944
10.32.5.46 vtss_phy_ts_ingress_engine_conf_get() 944
10.32.5.47 vtss_phy_ts_egress_engine_conf_set() 945
10.32.5.48 vtss_phy_ts_egress_engine_conf_get() 945
10.32.5.49 vtss_phy_ts_ingress_engine_action_set() 946
10.32.5.50 vtss_phy_ts_ingress_engine_action_get() 946
10.32.5.51 vtss_phy_ts_egress_engine_action_set() 948
10.32.5.52 vtss_phy_ts_egress_engine_action_get() 948
10.32.5.53 vtss_phy_ts_event_enable_set() 950
10.32.5.54 vtss_phy_ts_event_enable_get() 950
10.32.5.55 vtss_phy_ts_event_poll() 951
10.32.5.56 vtss_phy_ts_stats_get() 951
10.32.5.57 vtss_phy_ts_correction_overflow_get() 952
10.32.5.58 vtss_phy_ts_mode_set() 952

10.32.5.59 vtss_phy_ts_mode_get() . . . 953
10.32.5.60 vtss_phy_ts_init() . . . 953
10.32.5.61 vtss_phy_ts_init_conf_get() . . . 953
10.32.5.62 vtss_phy_ts_nphase_status_get() . . . 954
10.32.5.63 vtss_phy_ts_hiacc_set() . . . 954
10.32.5.64 vtss_phy_ts_hiacc_get() . . . 955
10.32.5.65 vtss_phy_ts_block_soft_reset() . . . 955
10.32.5.66 vtss_phy_ts_new_spi_mode_set() . . . 956
10.32.5.67 vtss_phy_ts_new_spi_mode_get() . . . 956
10.32.5.68 vtss_phy_1588_csr_reg_write() . . . 957
10.32.5.69 vtss_phy_1588_csr_reg_read() . . . 957
10.32.5.70 vtss_phy_ts_status_check() . . . 958
10.32.5.71 vtss_phy_ts_10g_extended_fifo_sync() . . . 958
10.32.5.72 vtss_phy_ts_viper_fifo_reset() . . . 959
10.32.5.73 vtss_phy_ts_tesla_tsp_fifo_sync() . . . 959
10.32.5.74 vtss_phy_1g_ts_fifo_sync() . . . 960
10.32.5.75 vtss_phy_1588_debug_reg_read() . . . 960
10.32.5.76 vtss_phy_ts_flow_clear_cf_set() . . . 961
10.33 vtss_api/include/vtss_port_api.h File Reference . . . 961
10.33.1 Detailed Description . . . 963
10.33.2 Macro Definition Documentation . . . 963
10.33.2.1 CHIP_PORT_UNUSED . . . 963
10.33.2.2 VTSS_FRAME_GAP_DEFAULT . . . 964
10.33.2.3 VTSS_MAX_FRAME_LENGTH_STANDARD . . . 964
10.33.2.4 VTSS_MAX_FRAME_LENGTH_MAX [1/2] . . . 964
10.33.2.5 VTSS_MAX_FRAME_LENGTH_MAX [2/2] . . . 964
10.33.3 Enumeration Type Documentation . . . 964
10.33.3.1 vtss_miim_controller_t . . . 964
10.33.3.2 vtss_internal_bw_t . . . 965
10.33.3.3 vtss_port_clause_37_remote_fault_t . . . 965

10.33.3.4 vtss_port_max_tags_t . . . 966
10.33.3.5 vtss_port_loop_t . . . 966
10.33.3.6 vtss_port_forward_t . . . 966
10.33.4 Function Documentation . . . 967
10.33.4.1 vtss_port_map_set() . . . 967
10.33.4.2 vtss_port_map_get() . . . 967
10.33.4.3 vtss_port_clause_37_control_get() . . . 967
10.33.4.4 vtss_port_clause_37_control_set() . . . 968
10.33.4.5 vtss_port_conf_set() . . . 968
10.33.4.6 vtss_port_conf_get() . . . 969
10.33.4.7 vtss_port_status_get() . . . 969
10.33.4.8 vtss_port_counters_update() . . . 970
10.33.4.9 vtss_port_counters_clear() . . . 970
10.33.4.10 vtss_port_counters_get() . . . 970
10.33.4.11 vtss_port_basic_counters_get() . . . 971
10.33.4.12 vtss_port_forward_state_get() . . . 971
10.33.4.13 vtss_port_forward_state_set() . . . 972
10.33.4.14 vtss_port_ifh_conf_set() . . . 972
10.33.4.15 vtss_port_ifh_conf_get() . . . 973
10.33.4.16 vtss_miim_read() . . . 973
10.33.4.17 vtss_miim_write() . . . 974
10.34 vtss_api/include/vtss_qos_api.h File Reference . . . 974
10.34.1 Detailed Description . . . 976
10.34.2 Macro Definition Documentation . . . 976
10.34.2.1 VTSS_PORT_POLICERS . . . 976
10.34.2.2 VTSS_PORT_POLICER_CPU_QUEUES . . . 976
10.34.2.3 VTSS_QCL_IDS . . . 977
10.34.2.4 VTSS_QCL_ID_START . . . 977
10.34.2.5 VTSS_QCL_ID_END . . . 977
10.34.2.6 VTSS_QCL_ARRAY_SIZE . . . 977

10.34.2.7 VTSS_QCE_ID_LAST . . . 977
10.34.3 Enumeration Type Documentation . . . 977
10.34.3.1 vtss_wred_v2_max_t . . . 977
10.34.3.2 vtss_shaper_mode_t . . . 978
10.34.3.3 vtss_tag_remark_mode_t . . . 978
10.34.3.4 vtss_dscp_mode_t . . . 978
10.34.3.5 vtss_dscp_emode_t . . . 979
10.34.3.6 vtss_qce_type_t . . . 979
10.34.4 Function Documentation . . . 980
10.34.4.1 vtss_qos_conf_get() . . . 980
10.34.4.2 vtss_qos_conf_set() . . . 980
10.34.4.3 vtss_qos_port_conf_get() . . . 980
10.34.4.4 vtss_qos_port_conf_set() . . . 981
10.34.4.5 vtss_qce_init() . . . 981
10.34.4.6 vtss_qce_add() . . . 982
10.34.4.7 vtss_qce_del() . . . 982
10.34.4.8 vtss_qos_shaper_calibrate() . . . 983
10.35 vtss_api/include/vtss_rab_api.h File Reference . . . 983
10.35.1 Detailed Description . . . 983
10.36 vtss_api/include/vtss_security_api.h File Reference . . . 983
10.36.1 Detailed Description . . . 986
10.36.2 Macro Definition Documentation . . . 986
10.36.2.1 VTSS_ACE_ID_LAST . . . 986
10.36.3 Enumeration Type Documentation . . . 986
10.36.3.1 vtss_auth_state_t . . . 986
10.36.3.2 vtss_acl_port_action_t . . . 987
10.36.3.3 vtss_acl_ptp_action_t . . . 987
10.36.3.4 vtss_ace_type_t . . . 988
10.36.3.5 vtss_ace_bit_t . . . 988
10.36.4 Function Documentation . . . 988

10.36.4.1 vtss_auth_port_state_get() . . . 988
10.36.4.2 vtss_auth_port_state_set() . . . 989
10.36.4.3 vtss_acl_policer_conf_get() . . . 989
10.36.4.4 vtss_acl_policer_conf_set() . . . 990
10.36.4.5 vtss_acl_port_conf_get() . . . 990
10.36.4.6 vtss_acl_port_conf_set() . . . 991
10.36.4.7 vtss_acl_port_counter_get() . . . 991
10.36.4.8 vtss_acl_port_counter_clear() . . . 991
10.36.4.9 vtss_ace_init() . . . 992
10.36.4.10 vtss_ace_add() . . . 992
10.36.4.11 vtss_ace_del() . . . 993
10.36.4.12 vtss_ace_counter_get() . . . 993
10.36.4.13 vtss_ace_counter_clear() . . . 994
10.37 vtss_api/include/vtss_sfi4_api.h File Reference . . . 994
10.37.1 Detailed Description . . . 994
10.38 vtss_api/include/vtss_sync_api.h File Reference . . . 994
10.38.1 Detailed Description . . . 995
10.38.2 Macro Definition Documentation . . . 995
10.38.2.1 VTSS_SYNCE_CLK_A . . . 995
10.38.2.2 VTSS_SYNCE_CLK_B . . . 996
10.38.2.3 VTSS_SYNCE_CLK_MAX . . . 996
10.38.3 Enumeration Type Documentation . . . 996
10.38.3.1 vtss_synce_divider_t . . . 996
10.38.4 Function Documentation . . . 996
10.38.4.1 vtss_synce_clock_out_set() . . . 996
10.38.4.2 vtss_synce_clock_out_get() . . . 997
10.38.4.3 vtss_synce_clock_in_set() . . . 997
10.38.4.4 vtss_synce_clock_in_get() . . . 998
10.39 vtss_api/include/vtss_tfi5_api.h File Reference . . . 998
10.39.1 Detailed Description . . . 998

10.40 vtss_api/include/vtss_ts_api.h File Reference . . . 998
10.40.1 Detailed Description . . . 1002
10.40.2 Typedef Documentation . . . 1002
10.40.2.1 vtss_ts_alt_clock_mode_t . . . 1002
10.40.3 Function Documentation . . . 1002
10.40.3.1 vtss_ts_timeofday_set() . . . 1002
10.40.3.2 vtss_ts_timeofday_set_delta() . . . 1003
10.40.3.3 vtss_ts_timeofday_offset_set() . . . 1003
10.40.3.4 vtss_ts_adjtimer_one_sec() . . . 1003
10.40.3.5 vtss_ts_ongoing_adjustment() . . . 1004
10.40.3.6 vtss_ts_timeofday_get() . . . 1004
10.40.3.7 vtss_ts_timeofday_next_pps_get() . . . 1005
10.40.3.8 vtss_ts_adjtimer_set() . . . 1005
10.40.3.9 vtss_ts_adjtimer_get() . . . 1005
10.40.3.10 vtss_ts_freq_offset_get() . . . 1006
10.40.3.11 vtss_ts_alt_clock_saved_get() . . . 1006
10.40.3.12 vtss_ts_alt_clock_mode_get() . . . 1007
10.40.3.13 vtss_ts_alt_clock_mode_set() . . . 1007
10.40.3.14 vtss_ts_timeofday_next_pps_set() . . . 1007
10.40.3.15 vtss_ts_external_clock_mode_get() . . . 1008
10.40.3.16 vtss_ts_external_clock_mode_set() . . . 1008
10.40.3.17 vtss_ts_external_clock_saved_get() . . . 1008
10.40.3.18 vtss_ts_ingress_latency_set() . . . 1009
10.40.3.19 vtss_ts_ingress_latency_get() . . . 1009
10.40.3.20 vtss_ts_p2p_delay_set() . . . 1010
10.40.3.21 vtss_ts_p2p_delay_get() . . . 1010
10.40.3.22 vtss_ts_egress_latency_set() . . . 1011
10.40.3.23 vtss_ts_egress_latency_get() . . . 1011
10.40.3.24 vtss_ts_delay_asymmetry_set() . . . 1011
10.40.3.25 vtss_ts_delay_asymmetry_get() . . . 1012

10.40.3.26 vtss_ts_operation_mode_set() . . . 1012
10.40.3.27 vtss_ts_operation_mode_get() . . . 1013
10.40.3.28 vtss_ts_internal_mode_set() . . . 1013
10.40.3.29 vtss_ts_internal_mode_get() . . . 1013
10.40.3.30 vtss_tx_timestamp_update() . . . 1014
10.40.3.31 vtss_rx_timestamp_get() . . . 1014
10.40.3.32 vtss_rx_timestamp_id_release() . . . 1015
10.40.3.33 vtss_rx_master_timestamp_get() . . . 1015
10.40.3.34 vtss_oam_timestamp_get() . . . 1016
10.40.3.35 vtss_tx_timestamp_idx_alloc() . . . 1016
10.40.3.36 vtss_timestamp_age() . . . 1016
10.40.3.37 vtss_ts_status_change() . . . 1017
10.41 vtss_api/include/vtss_upi_api.h File Reference . . . 1017
10.41.1 Detailed Description . . . 1017
10.42 vtss_api/include/vtss_wis_api.h File Reference . . . 1017
10.42.1 Detailed Description . . . 1018
10.43 vtss_api/include/vtss_xaui_api.h File Reference . . . 1018
10.43.1 Detailed Description . . . 1018
10.44 vtss_api/include/vtss_xfi_api.h File Reference . . . 1018
10.44.1 Detailed Description . . . 1018

Index **1019**

Chapter 1

Ethernet Virtual Connections

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

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

1.1 Port Configuration

The following EVC functions are available per port.

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

1.2 Policer Configuration

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

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

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

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

By default, all EVC policers are disabled.

1.3 EVCs

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

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

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

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

By default, no EVCs are setup.

1.4 ECEs

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

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

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

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

By default, no ECEs are setup.

1.5 EVC Statistics

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

- vtss_evc_counters_get() is used to get the counters for an EVC and port.
- vtss_evc_counters_clear() is used to clear the counters for an EVC and port.

1.6 ECE Statistics

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

- vtss_ece_counters_get() is used to get the counters for an EVC and port.
- vtss_ece_counters_clear() is used to clear the counters for an EVC and port.

Chapter 2

Layer 2

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

2.1 MAC Address Table

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

- vtss_mac_table_add() is used to add a static entry.
- vtss_mac_table_del() is used to delete a static entry.
- vtss_mac_table_get() is used to lookup a specific entry.
- vtss_mac_table_get_next() is used to get the next entry for table traversal.
- vtss_mac_table_age_time_get() is used to get the age time.
- vtss_mac_table_age_time_set() is used to set the age time.
- vtss_mac_table_age() is used for manual age scan.
- vtss_mac_table_vlan_age() is used for manual age scan per VLAN.
- vtss_mac_table_flush() is used to flush all dynamic entries.
- vtss_mac_table_port_flush() is used to flush dynamic entries per port.
- vtss_mac_table_vlan_flush() is used to flush dynamic entries per VLAN.
- vtss_mac_table_vlan_port_flush() is used to flush dynamic entries per VLAN and port.
- vtss_mac_table_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 idendified 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() convers 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.

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.

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

MPLS API Overview

The MPLS API is made up of the following main elements:

- Segments
- Cross-connects
- Layer-2 configuration

Together, these elements allow for the creation of MPLS label-swap LSPs, PWE3 pseudo-wire head and tail ends and LSP tunnel head and tail ends.

The available facilities are oriented towards the transportation of MPLS- encapsulated Ethernet frames, including Ethernet OAM, and MPLS-encapsulated MPLS OAM frames. IP is not supported by current hardware capabilities, and is therefore absent from the present API.

Segment & Layer-2

The Segment is the fundamental MPLS entity, designating either an incoming unidirectional LSP (in-segment) or an outgoing unidirectional LSP (out- segment).

A segment thus holds an MPLS label, TC information as well as other details necessary to configure the LSP.

Segments may be nested: One segment may be Server to other segments, Clients. Each Client segment may itself be Server to other Clients, and so on; only limited by the HW capabilities.

In order to move MPLS packets in and out of a port, a Layer-2 configuration must be created; it holds the relevant L2 info (port, peer MAC, tag type and VID, own MAC) enabling RX and TX.

A segment must therefore be attached to a L2 entry in order to be able to process MPLS. Obviously, only the ultimate server segment ("outermost segment") can have L2 attached.

Finally, each segment must be attached to exactly one Cross-connect; see below.

Cross-connect (XC)

Every segment must be attached to a Cross Connect (XC). The XC serves these purposes:

- Connect an in- and out-segment, enabling a Label Swap operation (LSR)
- For initiating/terminating LSPs/PWs, identify the type.

Ethernet Processing

Ethernet payload processing is handled by EVCs configured to use MPLS pseudo-wires as server layer. Thus, a typical setup for the Ethernet service ingress direction:

- An EVC, configured with one or more ECEs to accept certain traffic and add a service-delimiting S-tag to the payload Ethernet frame
- A pseudo-wire out-segment attached to an XC. This adds an inner MPLS label and an optional PW control word
- An LSP tunnel out-segment attach to an XC. This adds an outer MPLS label on top of the PW label. Then the PW out-segment is attached to the LSP tunnel out-segment; it has become server to the PW
- The LSP out-segment is attached to an L2 entry which provides the link- layer information required to form a valid Ethernet frame: Destination MAC, source MAC, optional VLAN tag, MPLS Ethertype, customer Ethernet frame (which includes the S-tag). FCS for the complete frame is automatically calculated and appended to the outgoing frame.

Ethernet service egress processing is largely symmetric to ingress. Instead of the EVC and out-segments adding an S-tag and labels, the tag and labels are removed by ECE rules and MPLS in-segments before the resulting payload frame is output.

3.1 API Usage

HW allocation

A segment will attempt to acquire HW resources when it is sufficiently configured. What constitutes "sufficient" depends on many different factors, detailed later. This HW allocation may fail if the underlying resources are unavailable/consumed; in that case the segment will stay in this state and there will be no allocation retrying except when explicitly initiated by a call to vtss_mpls_segment_set() or one of the attachment functions (attach to L2, segment, XC).

Segment states

A segment can either be UNCONFigured, CONFigured or UP.

UNCONF means that the segment lacks configuration in order to operate.

CONF means that the segment has sufficient configuration to operate, but that:

- either the underlying server segment isn't UP,
- or the segment hasn't been able to allocate hardware resources

UP indicates that:

- the configuration is sufficient
- the underlying server (if any) is UP
- all necessary HW resources (if any) have been allocated

Please note that UP does not indicate whether the ultimately underlying L2 port is up or not. The MPLS API considers it a resource, not a server layer.

3.2 MPLS Statistics

MPLS statistics include green/yellow/red/discarded frames and bytes for the following types of setups and traffic:

- LSR LSPs. The traffic is counted on the in- and out-segments attached to an XC

The Serval-1 hardware does not support MPLS OAM injection/extraction, nor are frames handled directly by the MPLS API but instead other parts of the chip API. Therefore, OAM counting must be handled by the application code running on the CPU.

Ethernet payload traffic is counted via the endpoint EVCs/ECEs available in the EVC API. Endpoint PWs only carry Ethernet frames and MPLS PW OAM, and tunnel LSPs only carry MPLS OAM, so the application must manually accumulate HW-based EVC counters and software-based MPLS OAM counters.

Chapter 4

Y.1731/IEEE802.1ag OAM

The Vitesse Serval architecture supports a large number of OAM features in hardware, either completely or with CPU assist.

The OAM functionality is logically divided into two main areas:

1) The Vitesse OAM Processor, the VOP. It is responsible for handling certain global settings across all hardware OAM instances 2) The Vitesse OAM Engines, VOEs. Each VOE implements functionality for supporting an OAM MIP or MEP.

4.1 - Vitesse OAM Processor

The VOP configures various global parameters that influence overall OAM PDU processing.

4.2 - Vitesse OAM Engine\n

A VOE is the hardware entity responsible for per-MEP/MIP processing.

Before a VOE can be used, it must be allocated. When it is no longer needed, it must be released.

Upon allocation a VOE index value is returned; that value must then be used for all subsequent calls to VOE-related functions.

Thus, the lifetime for a VOE will, in pseudo code, resemble this:

vtss_oam_voe_idx_t voe_idx; vtss_oam_voe_alloc_cfg_t data;

vtss_oam_voe_alloc(..., VTSS_OAM_VOE_..., &data, &voe_idx) vtss_oam_voe_conf_get(..., voe_idx, ...) vtss_oam_voe_conf_set() // (operational life of VOE: cfg changes, poll, counter get/clear, etc.) voe_oam_↩ voe_free(voe_idx)

Chapter 5

Automatic Frame Injector.

The automatic frame injector allows for periodic transmission of frames from within the switch core.

The flow for setting up a frame to be periodically injected is as follows for external CPUs:
1) Allocate resources for the frame using vtss_afi_alloc(). This function takes a structure in which the application must specify properties, like transmissions per seconds.
2) The vtss_afi_alloc() function returns a unique ID, which must be transferred in vtss_packet_tx_info_t::afi_id of the structure that is passed to vtss_packet_tx_hdr_encode().
3) Transmit the frame with the injection header that came out of vtss_packet_tx_hdr_encode().
4) In order to cancel the frame transmission, call vtss_afi_free() with the ID returned originally by vtss_afi_alloc().
5) If the port onto which the frame is transmitted gets a link-down event, the Vitesse API will autonomously pause frame transmission. When the link comes back up, the external CPU will have to re-transmit the frame with the ID retrieved in the original call to vtss_afi_alloc(), because internally in the API, the frame is seen as paused, not cancelled.

If using the FDMA driver to inject AFI-based frames, the FDMA driver will take care of allocating, pausing, resuming, and freeing frames. So in this event, the application should not call any of the vtss_afi_XXX() functions, but use the FDMA driver's interface (vtss_fdma_tx_info_t).

Chapter 6

Security

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

6.1 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.

6.2 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.

6.2.1 Access Control Entry

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

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

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

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

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

6.2.2 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.

6.2.3 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 7

Data Structure Index

7.1 Data Structures

Here are the data structures with brief descriptions:

port_custom_conf_t
Port configuration 37
serdes_fields_t
Serdes fields 40
tag_vtss_fdma_list
Software DCB structure 41
vtss_ace_frame_arp_t
Frame data for VTSS_ACE_TYPE_ARP 44
vtss_ace_frame_etype_t
Frame data for VTSS_ACE_TYPE_ETYPE 47
vtss_ace_frame_ipv4_t
Frame data for VTSS_ACE_TYPE_IPV4 49
vtss_ace_frame_ipv6_t
Frame data for VTSS_ACE_TYPE_IPV6 54
vtss_ace_frame_llc_t
Frame data for VTSS_ACE_TYPE_LLC 58
vtss_ace_frame_snap_t
Frame data for VTSS_ACE_TYPE_SNAP 59
vtss_ace_ptp_t
PTP header filtering 60
vtss_ace_sip_smac_t
SIP/SMAC filtering 61
vtss_ace_t
Access Control Entry 62
vtss_ace_vlan_t
ACE VLAN information 67
vtss_acl_action_t
ACL Action 68
vtss_acl_policer_conf_t
ACL policer configuration 72
vtss_acl_port_conf_t
ACL port configuration 73
vtss_afi_frm_dscr_t
AFI Frame description structure 74
vtss_aggr_mode_t
Aggregation traffic distribution mode 75

vtss_aneg_t
Auto negotiation struct . . . 76
vtss_api_lock_t
API lock structure . . . 77
vtss_basic_counters_t
Basic counters structure . . . 79
vtss_chip_id_t
Chip ID . . . 80
vtss_counter_pair_t
Counter pair . . . 81
vtss_debug_info_t
Debug information structure . . . 82
vtss_debug_lock_t
API debug lock structure . . . 84
vtss_dgroup_port_conf_t
Destination group port configuration . . . 85
vtss_dlb_policer_conf_t
Dual leaky buckets policer configuration . . . 85
vtss_ece_action_t
ECE action . . . 87
vtss_ece_frame_etype_t
ECE Ethernet Type information . . . 91
vtss_ece_frame_ipv4_t
ECE IPv4 information . . . 92
vtss_ece_frame_ipv6_t
ECE IPv6 information . . . 94
vtss_ece_frame_llc_t
ECE LLC information . . . 96
vtss_ece_frame_snap_t
ECE SNAP information . . . 96
vtss_ece_inner_tag_t
ECE inner tag . . . 97
vtss_ece_key_t
ECE key . . . 99
vtss_ece_mac_t
ECE MAC information . . . 102
vtss_ece_outer_tag_t
ECE outer tag . . . 103
vtss_ece_t
EVC Control Entry . . . 105
vtss_ece_tag_t
ECE tag information . . . 106
vtss_eee_port_conf_t
EEE port configuration . . . 108
vtss_eee_port_counter_t
EEE port counters (JR only) . . . 109
vtss_eee_port_state_t
EEE port state (JR only) . . . 111
vtss_eps_port_conf_t
Port protection configuration . . . 112
vtss_evc_conf_t
EVC configuration (excluding UNIs) . . . 113
vtss_evc_counters_t
EVC/ECE counters . . . 115
vtss_evc_mpls_pw_info_t
MPLS Pseudo Wire configuration, used when attaching a PW to an EVC . . . 117
vtss_evc_mpls_tp_conf_t
MPLS specific EVC configuration . . . 118

vtss_evc_oam_port_conf_t
EVC OAM port configuration . . . 119
vtss_evc_pb_conf_t
PB specific EVC configuration . . . 120
vtss_evc_port_conf_t
EVC port configuration . . . 121
vtss_fan_conf_t
Fan specifications . . . 122
vtss_fdma_cfg_t
FDMA configuration structure . . . 124
vtss_fdma_ch_cfg_t
Channel configuration structure . . . 129
vtss_fdma_throttle_cfg_t . . . 133
vtss_fdma_tx_info_t
FDMA Injection Properties . . . 134
vtss_hqos_conf_t
Egress QoS setup per HQoS ID . . . 137
vtss_hqos_port_conf_t
HQoS port configuration . . . 139
vtss_init_conf_t
Initialization configuration . . . 139
vtss_inst_create_t
Create structure . . . 143
vtss_intr_t
Interrupt source structure . . . 144
vtss_ip_addr_t
Either an IPv4 or IPv6 address . . . 145
vtss_ip_network_t
IPv6 network . . . 146
vtss_ipv4_network_t
IPv4 network . . . 147
vtss_ipv4_uc_t
IPv4 unicast routing entry . . . 148
vtss_ipv6_network_t
IPv6 network . . . 149
vtss_ipv6_t
IPv6 address/mask . . . 150
vtss_ipv6_uc_t
IPv6 routing entry . . . 151
vtss_irq_conf_t
Interrupt configuration options . . . 151
vtss_irq_status_t
Interrupt status structure . . . 152
vtss_l3_counters_t
Routing interface statics counter . . . 154
vtss_lcpll_status_t
Structure for Get PHY LC-PLL status . . . 156
vtss_learn_mode_t
Learning mode . . . 158
vtss_mac_t
MAC Address . . . 159
vtss_mac_table_entry_t
MAC address entry . . . 160
vtss_mac_table_status_t
MAC address table status . . . 162
vtss_mce_action_t
MCE action . . . 163

vtss_mce_key_t
MCE key . . . 167
vtss_mce_outer_tag_t
ECE outer tag . . . 170
vtss_mce_port_info_t
MEP Control Entry port information . . . 171
vtss_mce_t
MEP Control Entry . . . 172
vtss_mce_tag_t
MCE tag information . . . 174
vtss_mirror_conf_t
Mirror configuration . . . 175
vtss_mpls_l2_t
Layer-2 configuration . . . 176
vtss_mpls_label_t
MPLS label. The TTL value is only relevant for LER out-segments . . . 178
vtss_mpls_oam_conf_t
OAM configuration . . . 179
vtss_mpls_pw_conf_t
Pseudo-wire configuration . . . 180
vtss_mpls_qos_to_tc_map_entry_t
QoS-to-TC map entry . . . 181
vtss_mpls_segment_status_t
Segment status. Contains various kinds of status/state information for a segment . . . 182
vtss_mpls_segment_t
Segment configuration . . . 183
vtss_mpls_tc_conf_t
TC/QoS map configuration . . . 186
vtss_mpls_tc_to_qos_map_entry_t
TC-to-(QoS,DP) map entry . . . 187
vtss_mpls_xc_counters_t
MPLS cross-connect counters . . . 188
vtss_mpls_xc_t
Cross Connect function for unidirectional LSP/PW . . . 190
vtss_mtimer_t
Timer structure . . . 192
vtss_npi_conf_t
NPI configuration . . . 193
vtss_oam_ccm_counter_t
CCM counters . . . 194
vtss_oam_ccm_status_t
Status for most recently processed RX CCM PDU . . . 195
vtss_oam_event_mask_t
Event polling across all VOEs . . . 197
vtss_oam_ipt_conf_t
OAM-related configuration for the IPT table . . . 198
vtss_oam_lb_counter_t
LB counter . . . 199
vtss_oam_lm_counter_t
Up-/Down-MEP LM counters and per-priority RX/TX LM counters . . . 200
vtss_oam_meg_id_t
MEG ID type . . . 204
vtss_oam_pdu_seen_status_t
Indicate whether one or more PDUs of various types has been seen since last check. A PDU must pass MEG Level and DMAC checks before being indicated here. The values are cleared after reading . . . 205
vtss_oam_proc_conf_t
PDU processing checks . . . 208

vtss_oam_proc_status_t
Processing status: change indications and status values for most recently processed PDU(s). ..._err: Status for the most recently processed PDU. ..._seen: This has happened at least once since last poll. Cleared after reading . . . 210
vtss_oam_rx_tx_counter_t
Simple RX/TX counter structure . . . 213
vtss_oam_sel_counter_t
RX/TX OAM PDU counters. By default, all OAM PDU types are counted as 'non-selected', but by setting the appropriate count_as_selected field in the configuration structures, a PDU type can be "moved" to the 'selected' counter . . . 214
vtss_oam_ts_id_s
Parameter for requesting an oam timestamp . . . 215
vtss_oam_ts_timestamp_s
Parameter for returning an oam timestamp . . . 216
vtss_oam_tst_counter_t
TST counter . . . 217
vtss_oam_voe_alloc_cfg_t
Extra data used by vtss_oam_voe_alloc(). Only relevant for port VOEs . . . 218
vtss_oam_voe_ccm_conf_t
CCM configuration . . . 219
vtss_oam_voe_ccm_lm_conf_t
CCM-LM (dual-ended LM) configuration. NOTE that this configuration extends and augments the CCM configuration in struct vtss_oam_voe_ccm_conf_t. Thus, in order to configure CCM-LM, both structs need to be filled in . . . 222
vtss_oam_voe_conf_t
Main VOE configuration structure . . . 224
vtss_oam_voe_counter_t
Per-VOE counters . . . 229
vtss_oam_voe_dm_conf_t
One- and Two-way DM (1DM / DMM) configuration . . . 231
vtss_oam_voe_generic_conf_t
Generic OAM opcode configuration . . . 233
vtss_oam_voe_lb_conf_t
LB configuration. Note that the tx_... members are ignored if LB is already active. If they are to be changed, LB must be disabled and re-enabled with the new values. In other words, the values are only applied at enable-time . . . 235
vtss_oam_voe_lm_counter_conf_t
LM counter configuration . . . 238
vtss_oam_voe_lt_conf_t
LT configuration . . . 238
vtss_oam_voe_se_lm_conf_t
Single-ended LM configuration . . . 240
vtss_oam_voe_tst_conf_t
TST configuration. Note that the tx_... members are ignored if TST is already active. If they are to be changed, TST must be disabled and re-enabled with the new values. In other words, the values are only applied at enable-time . . . 243
vtss_oam_voe_unknown_conf_t
Unknown opcode configuration. An unknown opcode is one that isn't a) explicitly supported (e.g. CCM or LTM) or b) configured as a generic opcode . . . 245
vtss_oam_voe_up_mep_conf_t
Up-MEP configuration . . . 246
vtss_oam_vop_conf_t
VOP configuration. Once the VOP is configured, VOEs can be configured . . . 247
vtss_oam_vop_extract_conf_t
Configuration for CPU/frontport extraction. Configure if a particular OAM PDU type is extracted to the CPU, or to a front port . . . 249

vtss_oam_vop_generic_opcode_conf_t
Configuration for generic opcodes. Similar to vtss_oam_vop_extract_conf_t, this structure configures extraction for a generic opcode, and allows for DMAC checking for incoming OAM PDUs with that opcode value . . . 250
vtss_oam_vop_pdu_type_conf_t
Global per-PDU type configuration . . . 251
vtss_os_timestamp_t . . . 255
vtss_packet_dma_conf_t . . . 255
vtss_packet_frame_info_t
Information about frame . . . 256
vtss_packet_port_filter_t
Packet information for each port . . . 257
vtss_packet_port_info_t
Port info structure . . . 258
vtss_packet_rx_conf_t
CPU Rx configuration . . . 260
vtss_packet_rx_header_t
System frame header describing received frame . . . 261
vtss_packet_rx_info_t
Decoded extraction header properties . . . 263
vtss_packet_rx_meta_t
Input structure to vtss_packet_rx_hdr_decode() . . . 273
vtss_packet_rx_port_conf_t
Packet registration per port . . . 277
vtss_packet_rx_queue_conf_t
CPU Rx queue configuration . . . 278
vtss_packet_rx_queue_map_t
CPU Rx queue map . . . 279
vtss_packet_rx_queue_npi_conf_t
CPU Rx queue NPI configuration . . . 282
vtss_packet_rx_reg_t
CPU Rx packet registration . . . 283
vtss_packet_tx_ifh_t
Compiled Tx Frame Header . . . 284
vtss_packet_tx_info_t
Injection Properties . . . 285
vtss_phy_10g_fifo_sync_t
10G OOS workaround options . . . 295
vtss_phy_aneg_t
PHY auto negotiation advertisement . . . 296
vtss_phy_clock_conf_t
PHY clock configuration . . . 298
vtss_phy_conf_1g_t
PHY 1G configuration . . . 300
vtss_phy_conf_t
PHY configuration . . . 301
vtss_phy_daisy_chain_conf_t
SPI daisy chain configuration . . . 304
vtss_phy_eee_conf_t
EEE configuration . . . 304
vtss_phy_enhanced_led_control_t
Enhanced LED control . . . 305
vtss_phy_forced_t
PHY forced mode configuration . . . 307
vtss_phy_led_mode_select_t
LED model selection . . . 308
vtss_phy_loopback_t
1G Phy loopbacks . . . 309

vtss_phy_ltc_freq_synth_s
Frequency systhesis pulse configuration . . . 311
vtss_phy_mac_serd_pcs_cntl_t
PHY MAC SerDes PCS Control, Reg16E3 . . . 312
vtss_phy_media_serd_pcs_cntl_t
PHY MEDIA SerDes PCS Control, Reg23E3 . . . 316
vtss_phy_power_conf_t
PHY power configuration . . . 318
vtss_phy_power_status_t
PHY power status . . . 319
vtss_phy_reset_conf_t
PHY reset structure . . . 320
vtss_phy_rgmii_conf_t
PHY RGMII configuration . . . 322
vtss_phy_statistic_t
Phy statistic information . . . 322
vtss_phy_status_1g_t
PHY 1G status . . . 325
vtss_phy_tbi_conf_t
PHY TBI configuration . . . 325
vtss_phy_timestamp_t
PHY timestamp in seconds and nanoseconds (10 bytes Timestamp) . . . 326
vtss_phy_ts_ach_conf_t
Analyzer ACH comparator configuration options . . . 327
vtss_phy_ts_alt_clock_mode_s
Parameter for setting the alternative clock mode . . . 330
vtss_phy_ts_eng_init_conf_t
Defines the basic engine parameters passed to the engine_init_conf_get() function . . . 331
vtss_phy_ts_engine_action_t
Engine Action configuration options . . . 332
vtss_phy_ts_engine_flow_conf_t
Analyzer flow configuration options . . . 334
vtss_phy_ts_eth_conf_t
Analyzer Ethernet comparator configuration options . . . 336
vtss_phy_ts_fifo_conf_t
Defines the params for FIFO SYNC function . . . 342
vtss_phy_ts_fifo_sig_t
Tx TSFIFO entry signature . . . 344
vtss_phy_ts_gen_conf_t
Analyzer Generic data configuration options using IP comparator . . . 346
vtss_phy_ts_generic_action_t
Generic Action configuration option . . . 348
vtss_phy_ts_generic_flow_conf_t
Generic engine flow configuration options . . . 350
vtss_phy_ts_ietf_mpls_ach_oam_conf_t
Analyzer OAM comparator, IETF MPLS ACH OAM Packet format configuration options . . . 351
vtss_phy_ts_init_conf_t
Defines the initial parameters to be passed to init function . . . 352
vtss_phy_ts_ip_conf_t
Analyzer IP comparator configuration options . . . 356
vtss_phy_ts_mpls_conf_t
Analyzer MPLS comparator configuration options . . . 359
vtss_phy_ts_mpls_lvl_rng_t
MPLS level range . . . 363
vtss_phy_ts_nphase_status_t
N-phase status . . . 364
vtss_phy_ts_oam_engine_action_t
OAM Action configuration options . . . 365

vtss_phy_ts_oam_engine_flow_conf_t
OAM engine flow configuration options . . . 367
vtss_phy_ts_pps_config_s
PPS Configuration . . . 368
vtss_phy_ts_ptp_conf_t
Analyzer PTP comparator configuration options . . . 370
vtss_phy_ts_ptp_engine_action_t
Analyzer PTP action configuration options . . . 372
vtss_phy_ts_ptp_engine_flow_conf_t
PTP engine flow configuration options . . . 373
vtss_phy_ts_sertod_conf_t
PHY timestamp in seconds and nanoseconds (10 bytes Timestamp) . . . 375
vtss_phy_ts_stats_t
Timestamping Statistics . . . 376
vtss_phy_ts_y1731_oam_conf_t
Analyzer OAM comparator, Y.1731 OAM Packet format configuration options . . . 379
vtss_phy_type_t
Phy type information . . . 381
vtss_phy_veriphy_result_t
VeriPHY result . . . 383
vtss_phy_wol_conf_t
Structure for Get/Set Wake-On-LAN configuration . . . 384
vtss_pi_conf_t
PI configuration . . . 385
vtss_policer_ext_t
Policer Extensions . . . 386
vtss_policer_t
Policer . . . 387
vtss_port_bridge_counters_t
Port bridge counter structure (RFC 4188) . . . 388
vtss_port_clause_37_adv_t
Advertisement control data for Clause 37 aneg . . . 389
vtss_port_clause_37_control_t
Auto-negotiation control parameter struct . . . 391
vtss_port_conf_t
Port configuration structure . . . 392
vtss_port_counters_t
Port counter structure . . . 396
vtss_port_ethernet_like_counters_t
Ethernet-like Interface counter structure (RFC 3635) . . . 398
vtss_port_flow_control_conf_t
Flow control setup . . . 398
vtss_port_frame_gaps_t
Inter frame gap structure . . . 400
vtss_port_if_group_counters_t
Interfaces Group counter structure (RFC 2863) . . . 401
vtss_port_ifh_t
Port Internal Frame Header structure . . . 404
vtss_port_map_t
Port map structure . . . 405
vtss_port_proprietary_counters_t
Port proprietary counter structure . . . 407
vtss_port_rmon_counters_t
RMON counter structure (RFC 2819) . . . 408
vtss_port_serdes_conf_t
SFI Serdes configuration . . . 415
vtss_port_sgmii_aneg_t
Advertisement control data for SGMII aneg . . . 416

vtss_port_status_t
Port status parameter struct . . . 418
vtss_qce_action_t
QCE action . . . 420
vtss_qce_frame_etype_t
Frame data for VTSS_QCE_TYPE_ETYPE . . . 423
vtss_qce_frame_ipv4_t
Frame data for VTSS_QCE_TYPE_IPV4 . . . 424
vtss_qce_frame_ipv6_t
Frame data for VTSS_QCE_TYPE_IPV6 . . . 426
vtss_qce_frame_llc_t
Frame data for VTSS_QCE_TYPE_LLC . . . 428
vtss_qce_frame_snap_t
Frame data for VTSS_QCE_TYPE_SNAP . . . 429
vtss_qce_key_t
QCE key . . . 429
vtss_qce_mac_t
QCE MAC information . . . 432
vtss_qce_t
QoS Control Entry . . . 434
vtss_qce_tag_t
QCE tag information . . . 435
vtss_qos_conf_t
All parameters below are defined per chip . . . 437
vtss_qos_port_conf_t
QoS setup per port . . . 441
vtss_rcpll_status_t
Structure for Get PHY RC-PLL status . . . 448
vtss_red_v2_t
Random Early Detection configuration struct version 2 (per queue, per dpl - switch global) . . . 449
vtss_restart_status_t
Restart status . . . 450
vtss_routing_entry_t
Routing entry . . . 452
vtss_secure_on_passwd_t
Structure for Wake-On-LAN Secure-On Password . . . 453
vtss_serdes_macro_conf_t
Serdes macro configuration . . . 454
vtss_sflow_port_conf_t
SFlow configuration structure . . . 455
vtss_sgpio_conf_t
SGPIO configuration for a group . . . 456
vtss_sgpio_port_conf_t
SGPIO port configuration . . . 457
vtss_sgpio_port_data_t
SGPIO read data for a port . . . 459
vtss_shaper_t
Shaper . . . 459
vtss_synce_clock_in_t
Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port . . . 461
vtss_synce_clock_out_t
Struct containing configuration for a recovered clock output port . . . 462
vtss_tci_t
Tag Control Information (according to IEEE 802.1Q) . . . 463
vtss_timeofday_t
Time of day structure . . . 464

vtss_timestamp_t
Time stamp in seconds and nanoseconds . . . 465
vtss_trace_conf_t
Trace group configuration . . . 466
vtss_ts_alt_clock_mode_t
Parameter for setting the alternative clock mode . . . 467
vtss_ts_ext_clock_mode_t
External clock output configuration . . . 468
vtss_ts_id_t
Timestamp identifier . . . 470
vtss_ts_internal_mode_t
Hardware timestamping format mode for internal ports . . . 470
vtss_ts_operation_mode_t
Timestamp operation . . . 471
vtss_ts_timestamp_alloc_t
Timestamp allocation . . . 472
vtss_ts_timestamp_t
Timestamp structure . . . 473
vtss_vcap_ip_t
VCAP IPv4 address value and mask . . . 474
vtss_vcap_port_conf_t
VCAP port configuration . . . 475
vtss_vcap_u128_t
VCAP 128 bit value and mask . . . 476
vtss_vcap_u16_t
VCAP 16 bit value and mask . . . 477
vtss_vcap_u24_t
VCAP 24 bit value and mask . . . 478
vtss_vcap_u32_t
VCAP 32 bit value and mask . . . 479
vtss_vcap_u40_t
VCAP 40 bit value and mask . . . 480
vtss_vcap_u48_t
VCAP 48 bit value and mask . . . 481
vtss_vcap_u8_t
VCAP 8 bit value and mask . . . 482
vtss_vcap_udp_tcp_t
VCAP UDP/TCP port range . . . 483
vtss_vcap_vid_t
VCAP VLAN ID value and mask . . . 484
vtss_vcap_vr_t
VCAP universal value or range . . . 485
vtss_vce_action_t
VCE Action . . . 488
vtss_vce_frame_etype_t
Frame data for VTSS_VCE_TYPE_ETYPE . . . 488
vtss_vce_frame_ipv4_t
Frame data for VTSS_VCE_TYPE_IPV4 . . . 489
vtss_vce_frame_ipv6_t
Frame data for VTSS_VCE_TYPE_IPV6 . . . 491
vtss_vce_frame_llc_t
Frame data for VTSS_VCE_TYPE_LLC . . . 493
vtss_vce_frame_snap_t
Frame data for VTSS_VCE_TYPE_SNAP . . . 493
vtss_vce_key_t
VCE Key . . . 494
vtss_vce_mac_t
VCE MAC header information . . . 497

vtss_vce_t
VLAN Control Entry 498
vtss_vce_tag_t
VCE tag information 499
vtss_vcl_port_conf_t
VCL port configuration 501
vtss_vid_mac_t
MAC Address in specific VLAN 501
vtss_vlan_conf_t
VLAN configuration 502
vtss_vlan_port_conf_t
VLAN port configuration 503
vtss_vlan_tag_t 505
vtss_vlan_trans_grp2vlan_conf_t
VLAN translation group-to-VLAN configuration 506
vtss_vlan_trans_port2grp_conf_t
VLAN translation port-to-group configuration 507
vtss_vlan_vid_conf_t
VLAN ID configuration 508
vtss_wol_mac_addr_t
Structure for Wake-On-LAN MAC Address 509

Chapter 8

File Index

8.1 File List

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

vtss_api/include/vtss_ae_api.h
Ae API 590
vtss_api/include/vtss_afi_api.h
AFI API 591
vtss_api/include/vtss_aneg_api.h
ANEG API 591
vtss_api/include/vtss_api.h
Vitesse API main header file 592
vtss_api/include/vtss_evc_api.h
EVC API 592
vtss_api/include/vtss_fdma_api.h
Frame DMA API 610
vtss_api/include/vtss_gfp_api.h
GFP API 626
vtss_api/include/vtss_hqos_api.h
HQoS API 627
vtss_api/include/vtss_i2c_api.h
I2C API 631
vtss_api/include/vtss_init_api.h
Initialization API 631
vtss_api/include/vtss_l2_api.h
Layer 2 API 644
vtss_api/include/vtss_l3_api.h
L3 routing API 699
vtss_api/include/vtss_mac10g_api.h
MAC10G API 699
vtss_api/include/**vtss_macsec_api.h** . . . . . **??**
vtss_api/include/vtss_misc_api.h
Miscellaneous API 700
vtss_api/include/vtss_mpls_api.h
MPLS API 733
vtss_api/include/vtss_oam_api.h
OAM API 753
vtss_api/include/vtss_oha_api.h
OHA API 770

vtss_api/include/vtss_os.h
OS Layer API . . . 770
vtss_api/include/vtss_os_custom.h
OS custom header file . . . 770
vtss_api/include/vtss_os_ecos.h
ECos OS API . . . 775
vtss_api/include/vtss_os_linux.h
Linux OS API . . . 785
vtss_api/include/vtss_otn_api.h
OTN API . . . 792
vtss_api/include/vtss_packet_api.h
Packet API . . . 792
vtss_api/include/vtss_pcs_10gbase_r_api.h
PCS_10BASE_R API . . . 814
vtss_api/include/vtss_phy_10g_api.h
10G PHY API . . . 814
vtss_api/include/vtss_phy_api.h
PHY API . . . 814
vtss_api/include/vtss_phy_ts_api.h
PHY TimeStamping API . . . 891
vtss_api/include/vtss_port_api.h
Port API . . . 961
vtss_api/include/vtss_qos_api.h
QoS API . . . 974
vtss_api/include/vtss_rab_api.h
RAB API . . . 983
vtss_api/include/vtss_security_api.h
Security API . . . 983
vtss_api/include/vtss_sfi4_api.h
SFI4 API . . . 994
vtss_api/include/vtss_sync_api.h
Synchronization API . . . 994
vtss_api/include/vtss_tfi5_api.h
TFI5 API . . . 998
vtss_api/include/vtss_ts_api.h
TimeStamping API . . . 998
vtss_api/include/vtss_upi_api.h
Define UPI API interface . . . 1017
vtss_api/include/vtss_wis_api.h
EWIS layer API . . . 1017
vtss_api/include/vtss_xaui_api.h
XAUI API . . . 1018
vtss_api/include/vtss_xfi_api.h
XFI API . . . 1018
vtss_api/include/vtss/api/l2_types.h
Layer 2 Public API Header for l2 . . . 511
vtss_api/include/vtss/api/options.h
Features and options . . . 512
vtss_api/include/vtss/api/phy.h
PHY Public API Header . . . 535
vtss_api/include/vtss/api/port.h
Port Public API Header . . . 537
vtss_api/include/vtss/api/types.h
Generic types API . . . 550

Chapter 9

Data Structure Documentation

9.1 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

9.1.1 Detailed Description

Port configuration.

Definition at line 269 of file port.h.

9.1.2 Field Documentation

9.1.2.1 enable

```
BOOL port_custom_conf_t::enable
```

Admin enable/disable

Definition at line 270 of file port.h.

9.1.2.2 autoneg

```
BOOL port_custom_conf_t::autoneg
```

Auto negotiation

Definition at line 271 of file port.h.

9.1.2.3 fdx

```
BOOL port_custom_conf_t::fdx
```

Forced duplex mode

Definition at line 272 of file port.h.

9.1.2.4 flow_control

```
BOOL port_custom_conf_t::flow_control
```

Flow control (Standard 802.3x)

Definition at line 273 of file port.h.

9.1.2.5 pfc

```
BOOL port_custom_conf_t::pfc[VTSS_PRIOS]
```

Priority Flow control (802.1Qbb)

Definition at line 275 of file port.h.

9.1.2.6 speed

```
vtss_port_speed_t port_custom_conf_t::speed
```

Forced port speed

Definition at line 277 of file port.h.

9.1.2.7 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.

9.1.2.8 max_length

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

Max frame length

Definition at line 279 of file port.h.

9.1.2.9 power_mode

```
vtss_phy_power_mode_t port_custom_conf_t::power_mode
```

PHY power mode

Definition at line 281 of file port.h.

9.1.2.10 exc_col_cont

```
BOOL port_custom_conf_t::exc_col_cont
```

Excessive collision continuation

Definition at line 283 of file port.h.

9.1.2.11 adv_dis

```
u8 port_custom_conf_t::adv_dis
```

Auto neg advertisement disable

Definition at line 284 of file port.h.

9.1.2.12 max_tags

```
u8 port_custom_conf_t::max_tags
```

Maximum number of tags

Definition at line 285 of file port.h.

9.1.2.13 oper_up

```
BOOL port_custom_conf_t::oper_up
```

Force operational state up

Definition at line 286 of file port.h.

9.1.2.14 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

9.2 serdes_fields_t Struct Reference

Serdes fields.

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

Data Fields

- u32 ob_post0
- u32 ob_sr

9.2.1 Detailed Description

Serdes fields.

Definition at line 357 of file vtss_init_api.h.

9.2.2 Field Documentation

9.2.2.1 ob_post0

```
u32 serdes_fields_t::ob_post0
```

Trace length

Definition at line 358 of file vtss_init_api.h.

9.2.2.2 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

9.3 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
- struct tag_vtss_fdma_list * next

9.3.1 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.

9.3.2 Field Documentation

9.3.2.1 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.

9.3.2.2 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.

9.3.2.3 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.

9.3.2.4 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.

9.3.2.5 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.

9.3.2.6 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.

9.3.2.7 next

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

XTR:
Points to the next entry in the list or NULL if it's the last. Set by user on initialization of list. Continuously updated by vtss_fdma.c afterwards.
INJ:
Points to the next fragment of the frame and set by user on a per-frame basis. Last fragment of a frame must set ->next to NULL. Once handed to vtss_fdma.c, the driver code takes over. **AFI:**
Must be NULL (AFI frames must be contained in one fragment (due to injection from multiple GPDMA channels into the same injection group)). Internally the FDMA driver uses it to link multiple user AFI frames onto the same AFI channel.

Definition at line 716 of file vtss_fdma_api.h.

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

- vtss_api/include/vtss_fdma_api.h

9.4 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

9.4.1 Detailed Description

Frame data for VTSS_ACE_TYPE_ARP.

Definition at line 450 of file vtss_security_api.h.

9.4.2 Field Documentation

9.4.2.1 smac

`vtss_ace_u48_t vtss_ace_frame_arp_t::smac`

SMAC

Definition at line 452 of file vtss_security_api.h.

9.4.2.2 arp

`vtss_ace_bit_t vtss_ace_frame_arp_t::arp`

Opcode ARP/RARP

Definition at line 453 of file vtss_security_api.h.

9.4.2.3 req

`vtss_ace_bit_t vtss_ace_frame_arp_t::req`

Opcode request/reply

Definition at line 454 of file vtss_security_api.h.

9.4.2.4 unknown

`vtss_ace_bit_t vtss_ace_frame_arp_t::unknown`

Opcode unknown

Definition at line 455 of file vtss_security_api.h.

9.4.2.5 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.

9.4.2.6 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.

9.4.2.7 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.

9.4.2.8 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.

9.4.2.9 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.

9.4.2.10 sip

```
vtss_ace_ip_t vtss_ace_frame_arp_t::sip
```

Sender IP address

Definition at line 461 of file vtss_security_api.h.

9.4.2.11 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

9.5 vtss_ace_frame_etype_t Struct Reference

Frame data for VTSS_ACE_TYPE_ETYPE.

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

Data Fields

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

9.5.1 Detailed Description

Frame data for VTSS_ACE_TYPE_ETYPE.

Definition at line 422 of file vtss_security_api.h.

9.5.2 Field Documentation

9.5.2.1 dmac

`vtss_ace_u48_t vtss_ace_frame_etype_t::dmac`

DMAC

Definition at line 424 of file vtss_security_api.h.

9.5.2.2 smac

`vtss_ace_u48_t vtss_ace_frame_etype_t::smac`

SMAC

Definition at line 425 of file vtss_security_api.h.

9.5.2.3 etype

`vtss_ace_u16_t vtss_ace_frame_etype_t::etype`

Ethernet Type value

Definition at line 426 of file vtss_security_api.h.

9.5.2.4 data

`vtss_ace_u16_t vtss_ace_frame_etype_t::data`

MAC data

Definition at line 427 of file vtss_security_api.h.

9.5.2.5 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

9.6 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

9.6.1 Detailed Description

Frame data for VTSS_ACE_TYPE_IPV4.

Definition at line 466 of file vtss_security_api.h.

9.6.2 Field Documentation

9.6.2.1 ttl

```
vtss_ace_bit_t vtss_ace_frame_ipv4_t::ttl
```

TTL zero

Definition at line 468 of file vtss_security_api.h.

9.6.2.2 fragment

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::fragment`

Fragment

Definition at line 469 of file vtss_security_api.h.

9.6.2.3 options

`vtss_ace_bit_t vtss_ace_frame_ipv4_t::options`

Header options

Definition at line 470 of file vtss_security_api.h.

9.6.2.4 ds

`vtss_ace_u8_t vtss_ace_frame_ipv4_t::ds`

DS field

Definition at line 471 of file vtss_security_api.h.

9.6.2.5 proto

`vtss_ace_u8_t vtss_ace_frame_ipv4_t::proto`

Protocol

Definition at line 472 of file vtss_security_api.h.

9.6.2.6 sip

`vtss_ace_ip_t vtss_ace_frame_ipv4_t::sip`

Source IP address

Definition at line 473 of file vtss_security_api.h.

9.6.2.7 dip

```
vtss_ace_ip_t vtss_ace_frame_ipv4_t::dip
```

Destination IP address

Definition at line 474 of file vtss_security_api.h.

9.6.2.8 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.

9.6.2.9 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.

9.6.2.10 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.

9.6.2.11 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.

9.6.2.12 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.

9.6.2.13 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.

9.6.2.14 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.

9.6.2.15 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.

9.6.2.16 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.

9.6.2.17 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.

9.6.2.18 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.

9.6.2.19 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.

9.6.2.20 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.

9.6.2.21 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

9.7 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

9.7.1 Detailed Description

Frame data for VTSS_ACE_TYPE_IPV6.

Definition at line 494 of file vtss_security_api.h.

9.7.2 Field Documentation

9.7.2.1 proto

```
vtss_ace_u8_t vtss_ace_frame_ipv6_t::proto
```

IPv6 protocol

Definition at line 496 of file vtss_security_api.h.

9.7.2.2 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.

9.7.2.3 ttl

`vtss_ace_bit_t vtss_ace_frame_ipv6_t::ttl`

TTL zero

Definition at line 498 of file vtss_security_api.h.

9.7.2.4 ds

`vtss_ace_u8_t vtss_ace_frame_ipv6_t::ds`

DS field

Definition at line 499 of file vtss_security_api.h.

9.7.2.5 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.

9.7.2.6 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.

9.7.2.7 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.

9.7.2.8 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.

9.7.2.9 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.

9.7.2.10 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.

9.7.2.11 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.

9.7.2.12 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.

9.7.2.13 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.

9.7.2.14 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.

9.7.2.15 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.

9.7.2.16 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.

9.7.2.17 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

9.8 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

9.8.1 Detailed Description

Frame data for VTSS_ACE_TYPE_LLC.

Definition at line 434 of file vtss_security_api.h.

9.8.2 Field Documentation

9.8.2.1 dmac

```
vtss_ace_u48_t vtss_ace_frame_llc_t::dmac
```

DMAC

Definition at line 436 of file vtss_security_api.h.

9.8.2.2 smac

```
vtss_ace_u48_t vtss_ace_frame_llc_t::smac
```

SMAC

Definition at line 437 of file vtss_security_api.h.

9.8.2.3 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

9.9 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

9.9.1 Detailed Description

Frame data for VTSS_ACE_TYPE_SNAP.

Definition at line 442 of file vtss_security_api.h.

9.9.2 Field Documentation

9.9.2.1 dmac

```
vtss_ace_u48_t vtss_ace_frame_snap_t::dmac
```

DMAC

Definition at line 444 of file vtss_security_api.h.

9.9.2.2 smac

```
vtss_ace_u48_t vtss_ace_frame_snap_t::smac
```

SMAC

Definition at line 445 of file vtss_security_api.h.

9.9.2.3 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

9.10 vtss_ace_ptp_t Struct Reference

PTP header filtering.

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

Data Fields

- BOOL enable
- vtss_ace_u32_t header

9.10.1 Detailed Description

PTP header filtering.

Definition at line 391 of file vtss_security_api.h.

9.10.2 Field Documentation

9.10.2.1 enable

```
BOOL vtss_ace_ptp_t::enable
```

Enable PTP header filtering

Definition at line 393 of file vtss_security_api.h.

9.10.2.2 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

9.11 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

9.11.1 Detailed Description

SIP/SMAC filtering.

Definition at line 398 of file vtss_security_api.h.

9.11.2 Field Documentation

9.11.2.1 enable

```
BOOL vtss_ace_sip_smac_t::enable
```

Enable SIP/SMAC filtering

Definition at line 400 of file vtss_security_api.h.

9.11.2.2 sip

```
vtss_ip_t vtss_ace_sip_smac_t::sip
```

SIP

Definition at line 401 of file vtss_security_api.h.

9.11.2.3 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

9.12 vtss_ace_t Struct Reference

Access Control Entry.

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

Data Fields

- vtss_ace_id_t id
- u8 lookup
- BOOL isdx_enable
- BOOL isdx_disable
- BOOL port_list [VTSS_PORT_ARRAY_SIZE]
- vtss_ace_u8_t policy
- vtss_ace_type_t type
- vtss_acl_action_t action
- vtss_ace_bit_t dmac_mc
- vtss_ace_bit_t dmac_bc
- vtss_ace_vlan_t vlan
- union {
  vtss_ace_frame_etype_t etype
  vtss_ace_frame_llc_t llc
  vtss_ace_frame_snap_t snap
  vtss_ace_frame_arp_t arp
  vtss_ace_frame_ipv4_t ipv4
  vtss_ace_frame_ipv6_t ipv6
  } frame

9.12.1 Detailed Description

Access Control Entry.

Definition at line 518 of file vtss_security_api.h.

9.12.2 Field Documentation

9.12.2.1 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.

9.12.2.2 lookup

```
u8 vtss_ace_t::lookup
```

Lookup, any non-zero value means second lookup

Definition at line 522 of file vtss_security_api.h.

9.12.2.3 isdx_enable

```
BOOL vtss_ace_t::isdx_enable
```

Use VID value for ISDX value

Definition at line 523 of file vtss_security_api.h.

9.12.2.4 isdx_disable

```
BOOL vtss_ace_t::isdx_disable
```

Match only frames with ISDX zero

Definition at line 524 of file vtss_security_api.h.

9.12.2.5 port_list

```
BOOL vtss_ace_t::port_list[VTSS_PORT_ARRAY_SIZE]
```

Port list

Definition at line 530 of file vtss_security_api.h.

9.12.2.6 policy

```
vtss_ace_u8_t vtss_ace_t::policy
```

Policy number

Definition at line 532 of file vtss_security_api.h.

9.12.2.7 type

```
vtss_ace_type_t vtss_ace_t::type
```

ACE frame type

Definition at line 533 of file vtss_security_api.h.

9.12.2.8 action

`vtss_acl_action_t vtss_ace_t::action`

ACE action

Definition at line 534 of file vtss_security_api.h.

9.12.2.9 dmac_mc

`vtss_ace_bit_t vtss_ace_t::dmac_mc`

Multicast DMAC

Definition at line 536 of file vtss_security_api.h.

9.12.2.10 dmac_bc

`vtss_ace_bit_t vtss_ace_t::dmac_bc`

Broadcast DMAC

Definition at line 537 of file vtss_security_api.h.

9.12.2.11 vlan

`vtss_ace_vlan_t vtss_ace_t::vlan`

VLAN Tag

Definition at line 539 of file vtss_security_api.h.

9.12.2.12 etype

`vtss_ace_frame_etype_t vtss_ace_t::etype`

VTSS_ACE_TYPE_ETYPE

Definition at line 544 of file vtss_security_api.h.

9.12.2.13 llc

`vtss_ace_frame_llc_t vtss_ace_t::llc`

VTSS_ACE_TYPE_LLC

Definition at line 545 of file vtss_security_api.h.

9.12.2.14 snap

`vtss_ace_frame_snap_t vtss_ace_t::snap`

VTSS_ACE_TYPE_SNAP

Definition at line 546 of file vtss_security_api.h.

9.12.2.15 arp

`vtss_ace_frame_arp_t vtss_ace_t::arp`

VTSS_ACE_TYPE_ARP

Definition at line 547 of file vtss_security_api.h.

9.12.2.16 ipv4

`vtss_ace_frame_ipv4_t vtss_ace_t::ipv4`

VTSS_ACE_TYPE_IPV4

Definition at line 548 of file vtss_security_api.h.

9.12.2.17 ipv6

`vtss_ace_frame_ipv6_t vtss_ace_t::ipv6`

VTSS_ACE_TYPE_IPV6

Definition at line 549 of file vtss_security_api.h.

9.12.2.18 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

9.13 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

9.13.1 Detailed Description

ACE VLAN information.

Definition at line 411 of file vtss_security_api.h.

9.13.2 Field Documentation

9.13.2.1 vid

```
vtss_ace_vid_t vtss_ace_vlan_t::vid
```

VLAN ID (12 bit)

Definition at line 413 of file vtss_security_api.h.

9.13.2.2 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.

9.13.2.3 cfi

```
vtss_ace_bit_t vtss_ace_vlan_t::cfi
```

CFI/DEI

Definition at line 415 of file vtss_security_api.h.

9.13.2.4 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

9.14 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
- BOOL lm_cnt_disable
- BOOL mac_swap

9.14.1 Detailed Description

ACL Action.

Definition at line 238 of file vtss_security_api.h.

9.14.2 Field Documentation

9.14.2.1 cpu

```
BOOL vtss_acl_action_t::cpu
```

Forward to CPU

Definition at line 240 of file vtss_security_api.h.

9.14.2.2 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.

9.14.2.3 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.

9.14.2.4 police

```
BOOL vtss_acl_action_t::police
```

Enable policer

Definition at line 243 of file vtss_security_api.h.

9.14.2.5 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.

9.14.2.6 evc_police

```
BOOL vtss_acl_action_t::evc_police
```

Enable EVC policer

Definition at line 247 of file vtss_security_api.h.

9.14.2.7 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.

9.14.2.8 learn

```
BOOL vtss_acl_action_t::learn
```

Allow learning

Definition at line 251 of file vtss_security_api.h.

9.14.2.9 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.

9.14.2.10 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.

9.14.2.11 mirror

```
BOOL vtss_acl_action_t::mirror
```

Enable mirroring

Definition at line 260 of file vtss_security_api.h.

9.14.2.12 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.

9.14.2.13 lm_cnt_disable

```
BOOL vtss_acl_action_t::lm_cnt_disable
```

Disable OAM LM Tx counting

Definition at line 270 of file vtss_security_api.h.

9.14.2.14 mac_swap

```
BOOL vtss_acl_action_t::mac_swap
```

Swap SMAC and DMAC

Definition at line 271 of file vtss_security_api.h.

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

- vtss_api/include/vtss_security_api.h

9.15 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

9.15.1 Detailed Description

ACL policer configuration.

Definition at line 155 of file vtss_security_api.h.

9.15.2 Field Documentation

9.15.2.1 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.

9.15.2.2 bit_rate

```
vtss_bitrate_t vtss_acl_policer_conf_t::bit_rate
```

Bit rate

Definition at line 158 of file vtss_security_api.h.

9.15.2.3 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

9.16 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

9.16.1 Detailed Description

ACL port configuration.

Definition at line 276 of file vtss_security_api.h.

9.16.2 Field Documentation

9.16.2.1 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.

9.16.2.2 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

9.17 vtss_afi_frm_dscr_t Struct Reference

AFI Frame description structure.

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

Data Fields

- u32 fps
- u32 actual_fps

9.17.1 Detailed Description

AFI Frame description structure.

Contains properties that describe how a frame should be injected periodically. Upon exit from vtss_afi_alloc() it contains additional information about the injection. For future compatibility, memset() this structure to 0 before filling it in.

Definition at line 608 of file vtss_packet_api.h.

9.17.2 Field Documentation

9.17.2.1 fps

```
u32 vtss_afi_frm_dscr_t::fps
```

[IN] Frame rate (in frames per second) at which a frame should be transmitted.

Definition at line 614 of file vtss_packet_api.h.

9.17.2.2 actual_fps

```
u32 vtss_afi_frm_dscr_t::actual_fps
```

[OUT] On some platforms, it may not be possible to achieve the requested frame rate in one single go. A work-around for this is to inject the same frame with different frame rates. Typically, a particular desired frame rate can be obtained by transmitting the same frame a number of times. vtss_afi_alloc() fills in actual_fps with the number of frames that the AFI will actually send. It is guaranteed that actual_fps <= fps, so that vtss_afi_alloc() can be called repeatedly with the missing frame count.

The API supports two different allocation methods. Which one is currently active depends on the value of the VTSS_OPT_AFI_OPTIMIZE_FOR_TIMERS compile- time flag.

If VTSS_OPT_AFI_OPTIMIZE_FOR_TIMERS is 0 (default/legacy), the AFI code attempts to use as few frame-slots as possible to achieve the desired rate. This is at the expense of timers. Setting VTSS_OPT_AFI_CAL_TYPE to 1 causes the AFI code to optimize for timers rather than frame-slots. This might be a better solution, because there are much more frame-slots than timers. When optimizing for timers, each timer is pre-configured to provide a fixed number of frames per second. In this mode the application may have to call into the vtss_afi_alloc() function more times than had it been in the "optimize-for- frame-slots" mode.

Definition at line 649 of file vtss_packet_api.h.

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

- vtss_api/include/vtss_packet_api.h

9.18 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

9.18.1 Detailed Description

Aggregation traffic distribution mode.

Definition at line 39 of file l2_types.h.

9.18.2 Field Documentation

9.18.2.1 smac_enable

```
BOOL vtss_aggr_mode_t::smac_enable
```

Source MAC address

Definition at line 41 of file l2_types.h.

9.18.2.2 dmac_enable

```
BOOL vtss_aggr_mode_t::dmac_enable
```

Destination MAC address

Definition at line 42 of file l2_types.h.

9.18.2.3 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.

9.18.2.4 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

9.19 vtss_aneg_t Struct Reference

Auto negotiation struct.

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

Data Fields

- BOOL obey_pause
- BOOL generate_pause

9.19.1 Detailed Description

Auto negotiation struct.

Definition at line 482 of file types.h.

9.19.2 Field Documentation

9.19.2.1 obey_pause

```
BOOL vtss_aneg_t::obey_pause
```

This port should obey PAUSE frames

Definition at line 484 of file types.h.

9.19.2.2 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

9.20 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

9.20.1 Detailed Description

API lock structure.

Definition at line 285 of file vtss_misc_api.h.

9.20.2 Field Documentation

9.20.2.1 inst

```
vtss_inst_t vtss_api_lock_t::inst
```

Target instance reference

Definition at line 286 of file vtss_misc_api.h.

9.20.2.2 function

```
const char* vtss_api_lock_t::function
```

Function name

Definition at line 287 of file vtss_misc_api.h.

9.20.2.3 file

```
const char* vtss_api_lock_t::file
```

File name

Definition at line 288 of file vtss_misc_api.h.

9.20.2.4 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

9.21 vtss_basic_counters_t Struct Reference

Basic counters structure.

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

Data Fields

- u32 rx_frames
- u32 tx_frames

9.21.1 Detailed Description

Basic counters structure.

Definition at line 377 of file vtss_port_api.h.

9.21.2 Field Documentation

9.21.2.1 rx_frames

```
u32 vtss_basic_counters_t::rx_frames
```

Rx frames

Definition at line 379 of file vtss_port_api.h.

9.21.2.2 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

9.22 vtss_chip_id_t Struct Reference

Chip ID.

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

Data Fields

- u16 part_number
- u16 revision

9.22.1 Detailed Description

Chip ID.

Definition at line 401 of file vtss_misc_api.h.

9.22.2 Field Documentation

9.22.2.1 part_number

```
u16 vtss_chip_id_t::part_number
```

BCD encoded part number

Definition at line 403 of file vtss_misc_api.h.

9.22.2.2 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

9.23 vtss_counter_pair_t Struct Reference

Counter pair.

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

Data Fields

- vtss_counter_t frames
- vtss_counter_t bytes

9.23.1 Detailed Description

Counter pair.

Definition at line 1111 of file types.h.

9.23.2 Field Documentation

9.23.2.1 frames

```
vtss_counter_t vtss_counter_pair_t::frames
```

Number of frames

Definition at line 1112 of file types.h.

9.23.2.2 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

9.24 vtss_debug_info_t Struct Reference

Debug information structure.

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

Data Fields

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

9.24.1 Detailed Description

Debug information structure.

Definition at line 243 of file vtss_misc_api.h.

9.24.2 Field Documentation

9.24.2.1 layer

```
vtss_debug_layer_t vtss_debug_info_t::layer
```

Layer

Definition at line 244 of file vtss_misc_api.h.

9.24.2.2 group

```
vtss_debug_group_t vtss_debug_info_t::group
```

Function group

Definition at line 245 of file vtss_misc_api.h.

9.24.2.3 chip_no

```
vtss_chip_no_t vtss_debug_info_t::chip_no
```

Chip number, multi-chip targets

Definition at line 246 of file vtss_misc_api.h.

9.24.2.4 port_list

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

Port list

Definition at line 247 of file vtss_misc_api.h.

9.24.2.5 full

```
BOOL vtss_debug_info_t::full
```

Full information dump

Definition at line 248 of file vtss_misc_api.h.

9.24.2.6 clear

```
BOOL vtss_debug_info_t::clear
```

Clear counters

Definition at line 249 of file vtss_misc_api.h.

9.24.2.7 vml_format

```
BOOL vtss_debug_info_t::vml_format
```

VML format register dump

Definition at line 250 of file vtss_misc_api.h.

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

- vtss_api/include/vtss_misc_api.h

9.25 vtss_debug_lock_t Struct Reference

API debug lock structure.

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

Data Fields

- vtss_chip_no_t chip_no

9.25.1 Detailed Description

API debug lock structure.

Definition at line 307 of file vtss_misc_api.h.

9.25.2 Field Documentation

9.25.2.1 chip_no

```
vtss_chip_no_t vtss_debug_lock_t::chip_no
```

Chip number (if multi-chip instance).

Definition at line 308 of file vtss_misc_api.h.

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

- vtss_api/include/vtss_misc_api.h

9.26 vtss_dgroup_port_conf_t Struct Reference

Destination group port configuration.

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

Data Fields

- vtss_dgroup_no_t dgroup_no

9.26.1 Detailed Description

Destination group port configuration.

Definition at line 1394 of file vtss_l2_api.h.

9.26.2 Field Documentation

9.26.2.1 dgroup_no

```
vtss_dgroup_no_t vtss_dgroup_port_conf_t::dgroup_no
```

Destination port group

Definition at line 1395 of file vtss_l2_api.h.

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

- vtss_api/include/vtss_l2_api.h

9.27 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

9.27.1 Detailed Description

Dual leaky buckets policer configuration.

Definition at line 237 of file vtss_qos_api.h.

9.27.2 Field Documentation

9.27.2.1 type

```
vtss_policer_type_t vtss_dlb_policer_conf_t::type
```

Policer type

Definition at line 238 of file vtss_qos_api.h.

9.27.2.2 enable

```
BOOL vtss_dlb_policer_conf_t::enable
```

Enable/disable policer

Definition at line 239 of file vtss_qos_api.h.

9.27.2.3 cf

```
BOOL vtss_dlb_policer_conf_t::cf
```

Coupling Flag

Definition at line 243 of file vtss_qos_api.h.

9.27.2.4 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.

9.27.2.5 cir

```
vtss_bitrate_t vtss_dlb_policer_conf_t::cir
```

Committed Information Rate

Definition at line 245 of file vtss_qos_api.h.

9.27.2.6 cbs

```
vtss_burst_level_t vtss_dlb_policer_conf_t::cbs
```

Committed Burst Size

Definition at line 246 of file vtss_qos_api.h.

9.27.2.7 eir

```
vtss_bitrate_t vtss_dlb_policer_conf_t::eir
```

Excess Information Rate

Definition at line 247 of file vtss_qos_api.h.

9.27.2.8 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

9.28 vtss_ece_action_t Struct Reference

ECE action.

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

Data Fields

- vtss_ece_dir_t dir
- vtss_ece_rule_t rule
- vtss_ece_tx_lookup_t tx_lookup
- vtss_ece_pop_tag_t pop_tag
- vtss_ece_outer_tag_t outer_tag
- vtss_ece_inner_tag_t inner_tag
- vtss_evc_policer_id_t policer_id
- vtss_evc_id_t evc_id
- vtss_acl_policy_no_t policy_no
- BOOL prio_enable
- vtss_prio_t prio
- BOOL dp_enable
- vtss_dp_level_t dp

9.28.1 Detailed Description

ECE action.

Definition at line 557 of file vtss_evc_api.h.

9.28.2 Field Documentation

9.28.2.1 dir

```
vtss_ece_dir_t vtss_ece_action_t::dir
```

Traffic direction

Definition at line 558 of file vtss_evc_api.h.

9.28.2.2 rule

```
vtss_ece_rule_t vtss_ece_action_t::rule
```

Rule type

Definition at line 560 of file vtss_evc_api.h.

9.28.2.3 tx_lookup

`vtss_ece_tx_lookup_t vtss_ece_action_t::tx_lookup`

Egress lookup type

Definition at line 561 of file vtss_evc_api.h.

9.28.2.4 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.

9.28.2.5 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.

9.28.2.6 inner_tag

`vtss_ece_inner_tag_t vtss_ece_action_t::inner_tag`

Egress inner VLAN tag (optional)

Definition at line 566 of file vtss_evc_api.h.

9.28.2.7 policer_id

`vtss_evc_policer_id_t vtss_ece_action_t::policer_id`

Policer ID

Definition at line 567 of file vtss_evc_api.h.

9.28.2.8 evc_id

```
vtss_evc_id_t vtss_ece_action_t::evc_id
```

EVC ID

Definition at line 569 of file vtss_evc_api.h.

9.28.2.9 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.

9.28.2.10 prio_enable

```
BOOL vtss_ece_action_t::prio_enable
```

Enable priority classification

Definition at line 572 of file vtss_evc_api.h.

9.28.2.11 prio

```
vtss_prio_t vtss_ece_action_t::prio
```

Priority (QoS class)

Definition at line 573 of file vtss_evc_api.h.

9.28.2.12 dp_enable

```
BOOL vtss_ece_action_t::dp_enable
```

Enable DP classification

Definition at line 576 of file vtss_evc_api.h.

9.28.2.13 dp

```
vtss_dp_level_t vtss_ece_action_t::dp
```

Drop precedence

Definition at line 577 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.29 vtss_ece_frame_etype_t Struct Reference

ECE Ethernet Type information.

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

Data Fields

- vtss_vcap_u16_t etype
- vtss_vcap_u32_t data

9.29.1 Detailed Description

ECE Ethernet Type information.

Definition at line 444 of file vtss_evc_api.h.

9.29.2 Field Documentation

9.29.2.1 etype

```
vtss_vcap_u16_t vtss_ece_frame_etype_t::etype
```

Ethernet Type value

Definition at line 445 of file vtss_evc_api.h.

9.29.2.2 data

```
vtss_vcap_u32_t vtss_ece_frame_etype_t::data
```

MAC data

Definition at line 446 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.30 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_ip_t dip
- vtss_vcap_vr_t sport
- vtss_vcap_vr_t dport

9.30.1 Detailed Description

ECE IPv4 information.

Definition at line 461 of file vtss_evc_api.h.

9.30.2 Field Documentation

9.30.2.1 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.

9.30.2.2 fragment

```
vtss_vcap_bit_t vtss_ece_frame_ipv4_t::fragment
```

Fragment

Definition at line 464 of file vtss_evc_api.h.

9.30.2.3 proto

```
vtss_vcap_u8_t vtss_ece_frame_ipv4_t::proto
```

Protocol

Definition at line 465 of file vtss_evc_api.h.

9.30.2.4 sip

```
vtss_vcap_ip_t vtss_ece_frame_ipv4_t::sip
```

Source IP address

Definition at line 466 of file vtss_evc_api.h.

9.30.2.5 dip

```
vtss_vcap_ip_t vtss_ece_frame_ipv4_t::dip
```

Destination IP address

Definition at line 468 of file vtss_evc_api.h.

9.30.2.6 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.

9.30.2.7 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

9.31 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_u128_t dip
- vtss_vcap_vr_t sport
- vtss_vcap_vr_t dport

9.31.1 Detailed Description

ECE IPv6 information.

Definition at line 476 of file vtss_evc_api.h.

9.31.2 Field Documentation

9.31.2.1 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.

9.31.2.2 proto

```
vtss_vcap_u8_t vtss_ece_frame_ipv6_t::proto
```

Protocol

Definition at line 479 of file vtss_evc_api.h.

9.31.2.3 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.

9.31.2.4 dip

```
vtss_vcap_u128_t vtss_ece_frame_ipv6_t::dip
```

Destination IP address (32 LSB)

Definition at line 482 of file vtss_evc_api.h.

9.31.2.5 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.

9.31.2.6 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

9.32 vtss_ece_frame_llc_t Struct Reference

ECE LLC information.

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

Data Fields

- vtss_vcap_u48_t data

9.32.1 Detailed Description

ECE LLC information.

Definition at line 450 of file vtss_evc_api.h.

9.32.2 Field Documentation

9.32.2.1 data

```
vtss_vcap_u48_t vtss_ece_frame_llc_t::data
```

Data

Definition at line 451 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.33 vtss_ece_frame_snap_t Struct Reference

ECE SNAP information.

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

Data Fields

- vtss_vcap_u48_t data

9.33.1 Detailed Description

ECE SNAP information.

Definition at line 455 of file vtss_evc_api.h.

9.33.2 Field Documentation

9.33.2.1 data

```
vtss_vcap_u48_t vtss_ece_frame_snap_t::data
```

Data

Definition at line 456 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.34 vtss_ece_inner_tag_t Struct Reference

ECE inner tag.

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

Data Fields

- vtss_ece_inner_tag_type_t type
- vtss_vid_t vid
- vtss_ece_pcp_mode_t pcp_mode
- vtss_tagprio_t pcp
- vtss_ece_dei_mode_t dei_mode
- vtss_dei_t dei

9.34.1 Detailed Description

ECE inner tag.

Definition at line 538 of file vtss_evc_api.h.

9.34.2 Field Documentation

9.34.2.1 type

`vtss_ece_inner_tag_type_t vtss_ece_inner_tag_t::type`

Tag type

Definition at line 540 of file vtss_evc_api.h.

9.34.2.2 vid

`vtss_vid_t vtss_ece_inner_tag_t::vid`

VLAN ID

Definition at line 541 of file vtss_evc_api.h.

9.34.2.3 pcp_mode

`vtss_ece_pcp_mode_t vtss_ece_inner_tag_t::pcp_mode`

PCP mode

Definition at line 543 of file vtss_evc_api.h.

9.34.2.4 pcp

`vtss_tagprio_t vtss_ece_inner_tag_t::pcp`

PCP value

Definition at line 548 of file vtss_evc_api.h.

9.34.2.5 dei_mode

`vtss_ece_dei_mode_t vtss_ece_inner_tag_t::dei_mode`

DEI mode

Definition at line 550 of file vtss_evc_api.h.

9.34.2.6 dei

```
vtss_dei_t vtss_ece_inner_tag_t::dei
```

DEI value

Definition at line 552 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.35 vtss_ece_key_t Struct Reference

ECE key.

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

Data Fields

- vtss_ece_port_t port_list [VTSS_PORT_ARRAY_SIZE]
- vtss_ece_mac_t mac
- vtss_ece_tag_t tag
- vtss_ece_tag_t inner_tag
- vtss_ece_type_t type
- u8 lookup
- union {
  - vtss_ece_frame_etype_t etype
  - vtss_ece_frame_llc_t llc
  - vtss_ece_frame_snap_t snap
  - vtss_ece_frame_ipv4_t ipv4
  - vtss_ece_frame_ipv6_t ipv6

  } frame

9.35.1 Detailed Description

ECE key.

Definition at line 490 of file vtss_evc_api.h.

9.35.2 Field Documentation

9.35.2.1 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.

9.35.2.2 mac

```
vtss_ece_mac_t vtss_ece_key_t::mac
```

MAC header

Definition at line 493 of file vtss_evc_api.h.

9.35.2.3 tag

```
vtss_ece_tag_t vtss_ece_key_t::tag
```

Tag

Definition at line 494 of file vtss_evc_api.h.

9.35.2.4 inner_tag

```
vtss_ece_tag_t vtss_ece_key_t::inner_tag
```

Inner tag

Definition at line 496 of file vtss_evc_api.h.

9.35.2.5 type

```
vtss_ece_type_t vtss_ece_key_t::type
```

Frame type

Definition at line 498 of file vtss_evc_api.h.

9.35.2.6 lookup

```
u8 vtss_ece_key_t::lookup
```

Lookup, any non-zero value means second lookup

Definition at line 500 of file vtss_evc_api.h.

9.35.2.7 etype

```
vtss_ece_frame_etype_t vtss_ece_key_t::etype
```

VTSS_ECE_TYPE_ETYPE

Definition at line 507 of file vtss_evc_api.h.

9.35.2.8 llc

```
vtss_ece_frame_llc_t vtss_ece_key_t::llc
```

VTSS_ECE_TYPE_LLC

Definition at line 508 of file vtss_evc_api.h.

9.35.2.9 snap

```
vtss_ece_frame_snap_t vtss_ece_key_t::snap
```

VTSS_ECE_TYPE_SNAP

Definition at line 509 of file vtss_evc_api.h.

9.35.2.10 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.

9.35.2.11 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.

9.35.2.12 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

9.36 vtss_ece_mac_t Struct Reference

ECE MAC information.

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

Data Fields

- vtss_vcap_u48_t dmac
- vtss_vcap_bit_t dmac_mc
- vtss_vcap_bit_t dmac_bc
- vtss_vcap_u48_t smac

9.36.1 Detailed Description

ECE MAC information.

Definition at line 420 of file vtss_evc_api.h.

9.36.2 Field Documentation

9.36.2.1 dmac

```
vtss_vcap_u48_t vtss_ece_mac_t::dmac
```

DMAC

Definition at line 423 of file vtss_evc_api.h.

9.36.2.2 dmac_mc

```
vtss_vcap_bit_t vtss_ece_mac_t::dmac_mc
```

Multicast DMAC

Definition at line 426 of file vtss_evc_api.h.

9.36.2.3 dmac_bc

```
vtss_vcap_bit_t vtss_ece_mac_t::dmac_bc
```

Broadcast DMAC

Definition at line 427 of file vtss_evc_api.h.

9.36.2.4 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

9.37 vtss_ece_outer_tag_t Struct Reference

ECE outer tag.

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

Data Fields

- BOOL enable
- vtss_vid_t vid
- vtss_ece_pcp_mode_t pcp_mode
- vtss_tagprio_t pcp
- vtss_ece_dei_mode_t dei_mode
- vtss_dei_t dei

9.37.1 Detailed Description

ECE outer tag.

Definition at line 517 of file vtss_evc_api.h.

9.37.2 Field Documentation

9.37.2.1 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.

9.37.2.2 vid

```
vtss_vid_t vtss_ece_outer_tag_t::vid
```

VLAN ID (VTSS_ECE_DIR_NNI_TO_UNI only)

Definition at line 521 of file vtss_evc_api.h.

9.37.2.3 pcp_mode

```
vtss_ece_pcp_mode_t vtss_ece_outer_tag_t::pcp_mode
```

PCP mode

Definition at line 524 of file vtss_evc_api.h.

9.37.2.4 pcp

```
vtss_tagprio_t vtss_ece_outer_tag_t::pcp
```

PCP value

Definition at line 529 of file vtss_evc_api.h.

9.37.2.5 dei_mode

```
vtss_ece_dei_mode_t vtss_ece_outer_tag_t::dei_mode
```

DEI mode

Definition at line 531 of file vtss_evc_api.h.

9.37.2.6 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

9.38 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

9.38.1 Detailed Description

EVC Control Entry.

Definition at line 582 of file vtss_evc_api.h.

9.38.2 Field Documentation

9.38.2.1 id

`vtss_ece_id_t vtss_ece_t::id`

Entry ID

Definition at line 583 of file vtss_evc_api.h.

9.38.2.2 key

`vtss_ece_key_t vtss_ece_t::key`

ECE key

Definition at line 584 of file vtss_evc_api.h.

9.38.2.3 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

9.39 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

9.39.1 Detailed Description

ECE tag information.

Definition at line 433 of file vtss_evc_api.h.

9.39.2 Field Documentation

9.39.2.1 vid

`vtss_vcap_vr_t vtss_ece_tag_t::vid`

VLAN ID (12 bit)

Definition at line 435 of file vtss_evc_api.h.

9.39.2.2 pcp

`vtss_vcap_u8_t vtss_ece_tag_t::pcp`

PCP (3 bit)

Definition at line 436 of file vtss_evc_api.h.

9.39.2.3 dei

`vtss_vcap_bit_t vtss_ece_tag_t::dei`

DEI

Definition at line 437 of file vtss_evc_api.h.

9.39.2.4 tagged

`vtss_vcap_bit_t vtss_ece_tag_t::tagged`

Tagged/untagged frame

Definition at line 438 of file vtss_evc_api.h.

9.39.2.5 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

9.40 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

9.40.1 Detailed Description

EEE port configuration.

Definition at line 1219 of file vtss_misc_api.h.

9.40.2 Field Documentation

9.40.2.1 eee_ena

```
BOOL vtss_eee_port_conf_t::eee_ena
```

Enable EEE

Definition at line 1221 of file vtss_misc_api.h.

9.40.2.2 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.

9.40.2.3 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.

9.40.2.4 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.

9.40.2.5 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

9.41 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

9.41.1 Detailed Description

EEE port counters (JR only)

Definition at line 1246 of file vtss_misc_api.h.

9.41.2 Field Documentation

9.41.2.1 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.

9.41.2.2 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.

9.41.2.3 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.

9.41.2.4 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.

9.41.2.5 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

9.42 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

9.42.1 Detailed Description

EEE port state (JR only)

Definition at line 1239 of file vtss_misc_api.h.

9.42.2 Field Documentation

9.42.2.1 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.

9.42.2.2 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

9.43 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

9.43.1 Detailed Description

Port protection configuration.

Definition at line 2141 of file vtss_l2_api.h.

9.43.2 Field Documentation

9.43.2.1 type

```
vtss_eps_port_type_t vtss_eps_port_conf_t::type
```

Protection type

Definition at line 2143 of file vtss_l2_api.h.

9.43.2.2 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

9.44 vtss_evc_conf_t Struct Reference

EVC configuration (excluding UNIs)

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

Data Fields

- vtss_evc_policer_id_t policer_id
- BOOL learning
- struct {
  - vtss_evc_pb_conf_t pb
  - vtss_evc_mpls_tp_conf_t mpls_tp

  } network

9.44.1 Detailed Description

EVC configuration (excluding UNIs)

Definition at line 309 of file vtss_evc_api.h.

9.44.2 Field Documentation

9.44.2.1 policer_id

```
vtss_evc_policer_id_t vtss_evc_conf_t::policer_id
```

Policer ID

Definition at line 311 of file vtss_evc_api.h.

9.44.2.2 learning

```
BOOL vtss_evc_conf_t::learning
```

Enable/disable learning

Definition at line 313 of file vtss_evc_api.h.

9.44.2.3 pb

```
vtss_evc_pb_conf_t vtss_evc_conf_t::pb
```

PB specific configuration

Definition at line 316 of file vtss_evc_api.h.

9.44.2.4 mpls_tp

```
vtss_evc_mpls_tp_conf_t vtss_evc_conf_t::mpls_tp
```

MPLS-TP specific configuration

Definition at line 318 of file vtss_evc_api.h.

9.44.2.5 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

9.45 vtss_evc_counters_t Struct Reference

EVC/ECE counters.

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

Data Fields

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

9.45.1 Detailed Description

EVC/ECE counters.

Definition at line 1127 of file types.h.

9.45.2 Field Documentation

9.45.2.1 rx_green

```
vtss_counter_pair_t vtss_evc_counters_t::rx_green
```

Rx green frames/bytes

Definition at line 1128 of file types.h.

9.45.2.2 rx_yellow

```
vtss_counter_pair_t vtss_evc_counters_t::rx_yellow
```

Rx yellow frames/bytes

Definition at line 1129 of file types.h.

9.45.2.3 rx_red

```
vtss_counter_pair_t vtss_evc_counters_t::rx_red
```

Rx red frames/bytes

Definition at line 1130 of file types.h.

9.45.2.4 rx_discard

```
vtss_counter_pair_t vtss_evc_counters_t::rx_discard
```

Rx discarded frames/bytes

Definition at line 1131 of file types.h.

9.45.2.5 tx_discard

```
vtss_counter_pair_t vtss_evc_counters_t::tx_discard
```

Tx discarded frames/bytes

Definition at line 1132 of file types.h.

9.45.2.6 tx_green

```
vtss_counter_pair_t vtss_evc_counters_t::tx_green
```

Tx green frames/bytes

Definition at line 1133 of file types.h.

9.45.2.7 tx_yellow

```
vtss_counter_pair_t vtss_evc_counters_t::tx_yellow
```

Tx yellow frames/bytes

Definition at line 1134 of file types.h.

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

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

9.46 vtss_evc_mpls_pw_info_t Struct Reference

MPLS Pseudo Wire configuration, used when attaching a PW to an EVC.

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

Data Fields

- BOOL split_horizon
- vtss_mpls_xc_idx_t ingress_xc
- vtss_mpls_xc_idx_t egress_xc
- u32 pw_num

9.46.1 Detailed Description

MPLS Pseudo Wire configuration, used when attaching a PW to an EVC.

Definition at line 295 of file vtss_evc_api.h.

9.46.2 Field Documentation

9.46.2.1 split_horizon

```
BOOL vtss_evc_mpls_pw_info_t::split_horizon
```

TRUE == this PW obeys split horizon; FALSE == it does not

Definition at line 296 of file vtss_evc_api.h.

9.46.2.2 ingress_xc

```
vtss_mpls_xc_idx_t vtss_evc_mpls_pw_info_t::ingress_xc
```

XC for ingress unidirectional pseudo-wire. VTSS_MPLS_IDX_UNDEFINED == unused

Definition at line 297 of file vtss_evc_api.h.

9.46.2.3 egress_xc

```
vtss_mpls_xc_idx_t vtss_evc_mpls_pw_info_t::egress_xc
```

XC for egress unidirectional pseudo-wire. VTSS_MPLS_IDX_UNDEFINED == unused

Definition at line 298 of file vtss_evc_api.h.

9.46.2.4 pw_num

```
u32 vtss_evc_mpls_pw_info_t::pw_num
```

Pseudo-wire number. Zero == unused entry

Definition at line 299 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.47 vtss_evc_mpls_tp_conf_t Struct Reference

MPLS specific EVC configuration.

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

Data Fields

- vtss_evc_mpls_pw_info_t pw [VTSS_EVC_MPLS_PW_CNT]

9.47.1 Detailed Description

MPLS specific EVC configuration.

Definition at line 303 of file vtss_evc_api.h.

9.47.2 Field Documentation

9.47.2.1 pw

```
vtss_evc_mpls_pw_info_t vtss_evc_mpls_tp_conf_t::pw[VTSS_EVC_MPLS_PW_CNT]
```

MPLS pseudo-wire configuration. NOTE: All unused entries must be at the end of the array; no "holes" are allowed

Definition at line 304 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.48 vtss_evc_oam_port_conf_t Struct Reference

EVC OAM port configuration.

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

Data Fields

- vtss_oam_voe_idx_t voe_idx

9.48.1 Detailed Description

EVC OAM port configuration.

Definition at line 628 of file vtss_evc_api.h.

9.48.2 Field Documentation

9.48.2.1 voe_idx

```
vtss_oam_voe_idx_t vtss_evc_oam_port_conf_t::voe_idx
```

VOE index or VTSS_OAM_VOE_IDX_NONE

Definition at line 629 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.49 vtss_evc_pb_conf_t Struct Reference

PB specific EVC configuration.

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

Data Fields

- BOOL nni [VTSS_PORT_ARRAY_SIZE]
- vtss_vid_t ivid
- vtss_vid_t vid
- vtss_vid_t leaf_ivid
- vtss_vid_t leaf_vid
- BOOL leaf [VTSS_PORT_ARRAY_SIZE]

9.49.1 Detailed Description

PB specific EVC configuration.

Definition at line 275 of file vtss_evc_api.h.

9.49.2 Field Documentation

9.49.2.1 nni

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

NNI configuration

Definition at line 276 of file vtss_evc_api.h.

9.49.2.2 ivid

```
vtss_vid_t vtss_evc_pb_conf_t::ivid
```

Internal VID

Definition at line 277 of file vtss_evc_api.h.

9.49.2.3 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.

9.49.2.4 leaf_ivid

```
vtss_vid_t vtss_evc_pb_conf_t::leaf_ivid
```

Leaf internal VID

Definition at line 280 of file vtss_evc_api.h.

9.49.2.5 leaf_vid

```
vtss_vid_t vtss_evc_pb_conf_t::leaf_vid
```

Leaf NNI VID of outer tag

Definition at line 281 of file vtss_evc_api.h.

9.49.2.6 leaf

```
BOOL vtss_evc_pb_conf_t::leaf[VTSS_PORT_ARRAY_SIZE]
```

UNI leaf

Definition at line 282 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.50 vtss_evc_port_conf_t Struct Reference

EVC port configuration.

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

Data Fields

- BOOL dmac_dip
- vtss_vcap_key_type_t key_type

9.50.1 Detailed Description

EVC port configuration.

Definition at line 150 of file vtss_evc_api.h.

9.50.2 Field Documentation

9.50.2.1 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.

9.50.2.2 key_type

```
vtss_vcap_key_type_t vtss_evc_port_conf_t::key_type
```

Key type for received frames (default NORMAL)

Definition at line 161 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.51 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

9.51.1 Detailed Description

Fan specifications.

Definition at line 1148 of file vtss_misc_api.h.

9.51.2 Field Documentation

9.51.2.1 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.

9.51.2.2 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.

9.51.2.3 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.

9.51.2.4 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.

9.51.2.5 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

9.52 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 ∗cntxt, vtss_fdma_list_t ∗list)
- u32 tx_buf_cnt
- void(∗ tx_done_cb)(void ∗cntxt, struct tag_vtss_fdma_list ∗list, vtss_rc rc)
- u32 afi_buf_cnt
- void(∗ afi_done_cb)(void ∗cntxt, struct tag_vtss_fdma_list ∗list, vtss_rc rc)

9.52.1 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.

9.52.2 Field Documentation

9.52.2.1 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.

9.52.2.2 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 ∗ 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.

9.52.2.3 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.

9.52.2.4 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 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 of 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

sz	[IN] Number of bytes to allocate.
list	[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.
mem_flags	[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.

9.52.2.5 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.

9.52.2.6 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.

9.52.2.7 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.

9.52.2.8 rx_allow_multiple_dcbs

```
BOOL vtss_fdma_cfg_t::rx_allow_multiple_dcbs
```

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

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

Definition at line 1964 of file vtss_fdma_api.h.

9.52.2.9 rx_cb

```
vtss_fdma_list_t*(* vtss_fdma_cfg_t::rx_cb) (void *cntxt, 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. cntxt: The cntxt 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 deatils): (@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.

9.52.2.10 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.

9.52.2.11 tx_done_cb

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

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

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

The parameters to the callback function are as follows:

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

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

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

Definition at line 2015 of file vtss_fdma_api.h.

9.52.2.12 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.

9.52.2.13 afi_done_cb

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

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

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

The parameters to the callback function are as follows:

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

9.53 vtss_fdma_ch_cfg_t Struct Reference

Channel configuration structure.

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

Data Fields

- vtss_fdma_ch_usage_t usage
- vtss_packet_rx_grp_t xtr_grp
- u32 inj_grp_mask
- vtss_fdma_list_t * list
- vtss_fdma_list_t *(* xtr_cb)(void *cntxt, vtss_fdma_list_t *list, vtss_packet_rx_queue_t qu)
- int prio
- vtss_chip_no_t chip_no

9.53.1 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.

9.53.2 Field Documentation

9.53.2.1 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_U↩SAGE_UNUSED.

Definition at line 1603 of file vtss_fdma_api.h.

9.53.2.2 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.

9.53.2.3 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.

9.53.2.4 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.

9.53.2.5 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. cntxt: 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 deatils): (@dcb), @∗data, @act_len, (@next). Once delivered to the Packet Module, the FDMA will not use the list again. The FDMA can, however, be handed back a (new) list of software DCBs by returning the list when the xtr_cb() returns. Alternatively, use the vtss_fdma_xtr_add_dcbs() function.
  3. qu: The extraction queue that this frame was extracted from. This may be useful if all extraction callback functions map to the same function.

  – Expected return value from xtr_cb():
    Non-NULL if the Packet Module wishes to give the FDMA new DCBs to extract to. In the case where the Packet Module handles the frame directly, it could as well pass back the list that xtr_cb() was called with right away. In the case where the frame is passed up to higher levels, like an IP stack, the Packet Module will probably return NULL in the call to xtr_cb() and provide a replacement list at a later point in time with a call to vtss_fdma_xtr_add_dcbs().

INJ/AFI:
Unused.

Definition at line 1709 of file vtss_fdma_api.h.

9.53.2.6 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.

9.53.2.7 chip_no

```
vtss_chip_no_t vtss_fdma_ch_cfg_t::chip_no
```

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

INJ/AFI:
Unused.

Definition at line 1745 of file vtss_fdma_api.h.

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

- vtss_api/include/vtss_fdma_api.h

9.54 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]

9.54.1 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.

9.54.2 Field Documentation

9.54.2.1 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.

9.54.2.2 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.

9.54.2.3 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

9.55 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

9.55.1 Detailed Description

FDMA Injection Properties.

Definition at line 742 of file vtss_fdma_api.h.

9.55.2 Field Documentation

9.55.2.1 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.

9.55.2.2 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.

9.55.2.3 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.

9.55.2.4 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.

9.55.2.5 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.

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

- vtss_api/include/vtss_fdma_api.h

9.56 vtss_hqos_conf_t Struct Reference

Egress QoS setup per HQoS ID.

```
#include <vtss_hqos_api.h>
```

Data Fields

- vtss_port_no_t port_no
- vtss_shaper_t shaper
- vtss_shaper_t shaper_queue [VTSS_QUEUE_ARRAY_SIZE]
- u8 dwrr_cnt
- vtss_pct_t queue_pct [VTSS_QUEUE_ARRAY_SIZE]
- vtss_bitrate_t min_rate

9.56.1 Detailed Description

Egress QoS setup per HQoS ID.

Definition at line 84 of file vtss_hqos_api.h.

9.56.2 Field Documentation

9.56.2.1 port_no

```
vtss_port_no_t vtss_hqos_conf_t::port_no
```

Port number

Definition at line 85 of file vtss_hqos_api.h.

9.56.2.2 shaper

```
vtss_shaper_t vtss_hqos_conf_t::shaper
```

Egress shaper

Definition at line 86 of file vtss_hqos_api.h.

9.56.2.3 shaper_queue

```
vtss_shaper_t vtss_hqos_conf_t::shaper_queue[VTSS_QUEUE_ARRAY_SIZE]
```

Egress queue shapers

Definition at line 87 of file vtss_hqos_api.h.

9.56.2.4 dwrr_cnt

```
u8 vtss_hqos_conf_t::dwrr_cnt
```

Number of queues running Deficit Weighted Round Robin scheduling

Definition at line 88 of file vtss_hqos_api.h.

9.56.2.5 queue_pct

```
vtss_pct_t vtss_hqos_conf_t::queue_pct[VTSS_QUEUE_ARRAY_SIZE]
```

Queue percentages

Definition at line 89 of file vtss_hqos_api.h.

9.56.2.6 min_rate

```
vtss_bitrate_t vtss_hqos_conf_t::min_rate
```

Configured minimum bandwidth. Unit: kbps.

Definition at line 90 of file vtss_hqos_api.h.

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

- vtss_api/include/vtss_hqos_api.h

9.57 vtss_hqos_port_conf_t Struct Reference

HQoS port configuration.

```
#include <vtss_hqos_api.h>
```

Data Fields

- vtss_hqos_sch_mode_t sch_mode

9.57.1 Detailed Description

HQoS port configuration.

Definition at line 53 of file vtss_hqos_api.h.

9.57.2 Field Documentation

9.57.2.1 sch_mode

```
vtss_hqos_sch_mode_t vtss_hqos_port_conf_t::sch_mode
```

Scheduling mode

Definition at line 54 of file vtss_hqos_api.h.

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

- vtss_api/include/vtss_hqos_api.h

9.58 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_pi_conf_t pi
- vtss_serdes_macro_conf_t serdes

9.58.1 Detailed Description

Initialization configuration.

Definition at line 515 of file vtss_init_api.h.

9.58.2 Field Documentation

9.58.2.1 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.

9.58.2.2 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.

9.58.2.3 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.

9.58.2.4 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.

9.58.2.5 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.

9.58.2.6 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.

9.58.2.7 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.

9.58.2.8 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.

9.58.2.9 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.

9.58.2.10 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.

9.58.2.11 warm_start_enable

```
BOOL vtss_init_conf_t::warm_start_enable
```

Allow warm start

Definition at line 535 of file vtss_init_api.h.

9.58.2.12 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.

9.58.2.13 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.

9.58.2.14 pi

```
vtss_pi_conf_t vtss_init_conf_t::pi
```

Parallel Interface configuration

Definition at line 547 of file vtss_init_api.h.

9.58.2.15 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

9.59 vtss_inst_create_t Struct Reference

Create structure.

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

Data Fields

- vtss_target_type_t target

9.59.1 Detailed Description

Create structure.

Definition at line 78 of file vtss_init_api.h.

9.59.2 Field Documentation

9.59.2.1 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

9.60 vtss_intr_t Struct Reference

Interrupt source structure.

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

Data Fields

- BOOL link_change [VTSS_PORT_ARRAY_SIZE]

9.60.1 Detailed Description

Interrupt source structure.

Definition at line 913 of file vtss_misc_api.h.

9.60.2 Field Documentation

9.60.2.1 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

9.61 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

9.61.1 Detailed Description

Either an IPv4 or IPv6 address.

Definition at line 813 of file types.h.

9.61.2 Field Documentation

9.61.2.1 type

```
vtss_ip_type_t vtss_ip_addr_t::type
```

Union type

Definition at line 814 of file types.h.

9.61.2.2 ipv4

```
vtss_ipv4_t vtss_ip_addr_t::ipv4
```

IPv4 address

Definition at line 816 of file types.h.

9.61.2.3 ipv6

```
vtss_ipv6_t vtss_ip_addr_t::ipv6
```

IPv6 address

Definition at line 817 of file types.h.

9.61.2.4 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

9.62 vtss_ip_network_t Struct Reference

IPv6 network.

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

Data Fields

- vtss_ip_addr_t address
- vtss_prefix_size_t prefix_size

9.62.1 Detailed Description

IPv6 network.

Definition at line 836 of file types.h.

9.62.2 Field Documentation

9.62.2.1 address

```
vtss_ip_addr_t vtss_ip_network_t::address
```

Network address

Definition at line 838 of file types.h.

9.62.2.2 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

9.63 vtss_ipv4_network_t Struct Reference

IPv4 network.

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

Data Fields

- vtss_ipv4_t address
- vtss_prefix_size_t prefix_size

9.63.1 Detailed Description

IPv4 network.

Definition at line 822 of file types.h.

9.63.2 Field Documentation

9.63.2.1 address

```
vtss_ipv4_t vtss_ipv4_network_t::address
```

Network address

Definition at line 824 of file types.h.

9.63.2.2 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

9.64 vtss_ipv4_uc_t Struct Reference

IPv4 unicast routing entry.

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

Data Fields

- vtss_ipv4_network_t network
- vtss_ipv4_t destination

9.64.1 Detailed Description

IPv4 unicast routing entry.

Definition at line 852 of file types.h.

9.64.2 Field Documentation

9.64.2.1 network

```
vtss_ipv4_network_t vtss_ipv4_uc_t::network
```

Network to route

Definition at line 854 of file types.h.

9.64.2.2 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

9.65 vtss_ipv6_network_t Struct Reference

IPv6 network.

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

Data Fields

- vtss_ipv6_t address
- vtss_prefix_size_t prefix_size

9.65.1 Detailed Description

IPv6 network.

Definition at line 829 of file types.h.

9.65.2 Field Documentation

9.65.2.1 address

```
vtss_ipv6_t vtss_ipv6_network_t::address
```

Network address

Definition at line 831 of file types.h.

9.65.2.2 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

9.66 vtss_ipv6_t Struct Reference

IPv6 address/mask.

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

Data Fields

- u8 addr [16]

9.66.1 Detailed Description

IPv6 address/mask.

Definition at line 797 of file types.h.

9.66.2 Field Documentation

9.66.2.1 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

9.67 vtss_ipv6_uc_t Struct Reference

IPv6 routing entry.

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

Data Fields

- vtss_ipv6_network_t network
- vtss_ipv6_t destination

9.67.1 Detailed Description

IPv6 routing entry.

Definition at line 860 of file types.h.

9.67.2 Field Documentation

9.67.2.1 network

```
vtss_ipv6_network_t vtss_ipv6_uc_t::network
```

Network to route

Definition at line 862 of file types.h.

9.67.2.2 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

9.68 vtss_irq_conf_t Struct Reference

Interrupt configuration options.

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

Data Fields

- BOOL external
- u8 destination

9.68.1 Detailed Description

Interrupt configuration options.

Definition at line 972 of file vtss_misc_api.h.

9.68.2 Field Documentation

9.68.2.1 external

```
BOOL vtss_irq_conf_t::external
```

Redirect to external IRQ

Definition at line 973 of file vtss_misc_api.h.

9.68.2.2 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

9.69 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

9.69.1 Detailed Description

Interrupt status structure.

Definition at line 980 of file vtss_misc_api.h.

9.69.2 Field Documentation

9.69.2.1 active

```
u32 vtss_irq_status_t::active
```

Bitmap for pending IRQs (VTSS_IRQ_xxx)

Definition at line 981 of file vtss_misc_api.h.

9.69.2.2 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.

9.69.2.3 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.

9.69.2.4 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

9.70 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

9.70.1 Detailed Description

Routing interface statics counter.

Definition at line 886 of file types.h.

9.70.2 Field Documentation

9.70.2.1 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.

9.70.2.2 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.

9.70.2.3 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.

9.70.2.4 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.

9.70.2.5 ipv4uc_transmitted_octets

```
u64 vtss_l3_counters_t::ipv4uc_transmitted_octets
```

IPv4UC octets transmitted

Definition at line 892 of file types.h.

9.70.2.6 ipv4uc_transmitted_frames

```
u64 vtss_l3_counters_t::ipv4uc_transmitted_frames
```

IPv4UC frames transmitted

Definition at line 893 of file types.h.

9.70.2.7 ipv6uc_transmitted_octets

```
u64 vtss_l3_counters_t::ipv6uc_transmitted_octets
```

IPv6UC octets transmitted

Definition at line 894 of file types.h.

9.70.2.8 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

9.71 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

9.71.1 Detailed Description

Structure for Get PHY LC-PLL status.

Definition at line 1777 of file vtss_phy_api.h.

9.71.2 Field Documentation

9.71.2.1 lock_status

```
u8 vtss_lcpll_status_t::lock_status
```

PLL lock status

Definition at line 1778 of file vtss_phy_api.h.

9.71.2.2 cal_done

```
u8 vtss_lcpll_status_t::cal_done
```

Calibration status

Definition at line 1779 of file vtss_phy_api.h.

9.71.2.3 cal_error

```
u8 vtss_lcpll_status_t::cal_error
```

Calibration Error indication

Definition at line 1780 of file vtss_phy_api.h.

9.71.2.4 fsm_lock

```
u8 vtss_lcpll_status_t::fsm_lock
```

FSM lock status

Definition at line 1781 of file vtss_phy_api.h.

9.71.2.5 fsm_stat

```
u8 vtss_lcpll_status_t::fsm_stat
```

FSM internal status

Definition at line 1782 of file vtss_phy_api.h.

9.71.2.6 gain_stat

```
u8 vtss_lcpll_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

9.72 vtss_learn_mode_t Struct Reference

Learning mode.

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

Data Fields

- BOOL automatic
- BOOL cpu
- BOOL discard

9.72.1 Detailed Description

Learning mode.

Definition at line 379 of file vtss_l2_api.h.

9.72.2 Field Documentation

9.72.2.1 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.

9.72.2.2 cpu

```
BOOL vtss_learn_mode_t::cpu
```

Learn frames copied to CPU (default disabled)

Definition at line 382 of file vtss_l2_api.h.

9.72.2.3 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

9.73 vtss_mac_t Struct Reference

MAC Address.

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

Data Fields

- u8 addr [6]

9.73.1 Detailed Description

MAC Address.

Definition at line 643 of file types.h.

9.73.2 Field Documentation

9.73.2.1 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

9.74 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

9.74.1 Detailed Description

MAC address entry.

Definition at line 119 of file vtss_l2_api.h.

9.74.2 Field Documentation

9.74.2.1 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.

9.74.2.2 destination

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

Dest. ports

Definition at line 122 of file vtss_l2_api.h.

9.74.2.3 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.

9.74.2.4 locked

```
BOOL vtss_mac_table_entry_t::locked
```

Locked/static flag

Definition at line 124 of file vtss_l2_api.h.

9.74.2.5 aged

```
BOOL vtss_mac_table_entry_t::aged
```

Age flag

Definition at line 125 of file vtss_l2_api.h.

9.74.2.6 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

9.75 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

9.75.1 Detailed Description

MAC address table status.

Definition at line 358 of file vtss_l2_api.h.

9.75.2 Field Documentation

9.75.2.1 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.

9.75.2.2 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.

9.75.2.3 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.

9.75.2.4 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

9.76 vtss_mce_action_t Struct Reference

MCE action.

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

Data Fields

- BOOL port_list [VTSS_PORT_ARRAY_SIZE]
- BOOL evc_counting
- BOOL evc_etree
- vtss_evc_id_t evc_id
- vtss_oam_voe_idx_t voe_idx
- vtss_mce_outer_tag_t outer_tag
- vtss_isdx_t isdx
- vtss_mce_rule_t rule
- vtss_mce_tx_lookup_t tx_lookup
- vtss_mce_oam_detect_t oam_detect
- vtss_evc_policer_id_t policer_id
- vtss_acl_policy_no_t policy_no
- BOOL prio_enable
- vtss_prio_t prio
- vtss_vid_t vid
- u8 pop_cnt

9.76.1 Detailed Description

MCE action.

Definition at line 759 of file vtss_evc_api.h.

9.76.2 Field Documentation

9.76.2.1 port_list

```
BOOL vtss_mce_action_t::port_list[VTSS_PORT_ARRAY_SIZE]
```

Egress port list

Definition at line 762 of file vtss_evc_api.h.

9.76.2.2 evc_counting

```
BOOL vtss_mce_action_t::evc_counting
```

OAM frames hitting this rule must be counting as EVC frames

Definition at line 763 of file vtss_evc_api.h.

9.76.2.3 evc_etree

```
BOOL vtss_mce_action_t::evc_etree
```

OAM frames hitting this rule must be forwarded as in an E-TREE. ISDX mask must be used restricting forwarding and no ES0 created

Definition at line 764 of file vtss_evc_api.h.

9.76.2.4 evc_id

```
vtss_evc_id_t vtss_mce_action_t::evc_id
```

EVC ID

Definition at line 765 of file vtss_evc_api.h.

9.76.2.5 voe_idx

`vtss_oam_voe_idx_t vtss_mce_action_t::voe_idx`

VOE index or VTSS_OAM_VOE_IDX_NONE

Definition at line 766 of file vtss_evc_api.h.

9.76.2.6 outer_tag

`vtss_mce_outer_tag_t vtss_mce_action_t::outer_tag`

Egress outer tag

Definition at line 767 of file vtss_evc_api.h.

9.76.2.7 isdx

`vtss_isdx_t vtss_mce_action_t::isdx`

ISDX or VTSS_MCE_ISDX_NONE or VTSS_MCE_ISDX_NEW

Definition at line 768 of file vtss_evc_api.h.

9.76.2.8 rule

`vtss_mce_rule_t vtss_mce_action_t::rule`

Rule type

Definition at line 769 of file vtss_evc_api.h.

9.76.2.9 tx_lookup

`vtss_mce_tx_lookup_t vtss_mce_action_t::tx_lookup`

Egress lookup type

Definition at line 770 of file vtss_evc_api.h.

9.76.2.10 oam_detect

`vtss_mce_oam_detect_t vtss_mce_action_t::oam_detect`

OAM detection

Definition at line 771 of file vtss_evc_api.h.

9.76.2.11 policer_id

`vtss_evc_policer_id_t vtss_mce_action_t::policer_id`

Policer ID

Definition at line 774 of file vtss_evc_api.h.

9.76.2.12 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.

9.76.2.13 prio_enable

`BOOL vtss_mce_action_t::prio_enable`

Enable priority control

Definition at line 777 of file vtss_evc_api.h.

9.76.2.14 prio

`vtss_prio_t vtss_mce_action_t::prio`

Selected priority

Definition at line 778 of file vtss_evc_api.h.

9.76.2.15 vid

```
vtss_vid_t vtss_mce_action_t::vid
```

Replace VID

Definition at line 779 of file vtss_evc_api.h.

9.76.2.16 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

9.77 vtss_mce_key_t Struct Reference

MCE key.

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

Data Fields

- BOOL port_list [VTSS_PORT_ARRAY_SIZE]
- BOOL port_cpu
- vtss_mce_tag_t tag
- vtss_mce_tag_t inner_tag
- vtss_vcap_u8_t mel
- vtss_vcap_bit_t injected
- u8 lookup
- vtss_vcap_u48_t dmac
- vtss_vcap_bit_t dmac_mc
- BOOL service_detect

9.77.1 Detailed Description

MCE key.

Definition at line 682 of file vtss_evc_api.h.

9.77.2 Field Documentation

9.77.2.1 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.

9.77.2.2 port_cpu

```
BOOL vtss_mce_key_t::port_cpu
```

CPU is the only ingress port

Definition at line 686 of file vtss_evc_api.h.

9.77.2.3 tag

```
vtss_mce_tag_t vtss_mce_key_t::tag
```

Outer tag

Definition at line 687 of file vtss_evc_api.h.

9.77.2.4 inner_tag

```
vtss_mce_tag_t vtss_mce_key_t::inner_tag
```

Inner tag

Definition at line 688 of file vtss_evc_api.h.

9.77.2.5 mel

```
vtss_vcap_u8_t vtss_mce_key_t::mel
```

MEG level (7 bit)

Definition at line 689 of file vtss_evc_api.h.

9.77.2.6 injected

```
vtss_vcap_bit_t vtss_mce_key_t::injected
```

CPU injected

Definition at line 690 of file vtss_evc_api.h.

9.77.2.7 lookup

```
u8 vtss_mce_key_t::lookup
```

Lookup, any non-zero value means second ingress lookup

Definition at line 691 of file vtss_evc_api.h.

9.77.2.8 dmac

```
vtss_vcap_u48_t vtss_mce_key_t::dmac
```

DMAC

Definition at line 692 of file vtss_evc_api.h.

9.77.2.9 dmac_mc

```
vtss_vcap_bit_t vtss_mce_key_t::dmac_mc
```

Multicast DMAC

Definition at line 693 of file vtss_evc_api.h.

9.77.2.10 service_detect

```
BOOL vtss_mce_key_t::service_detect
```

Detection of OAM or Service frames

Definition at line 694 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.78 vtss_mce_outer_tag_t Struct Reference

ECE outer tag.

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

Data Fields

- BOOL enable
- vtss_vid_t vid
- vtss_mce_pcp_mode_t pcp_mode
- vtss_tagprio_t pcp
- vtss_mce_dei_mode_t dei_mode
- vtss_dei_t dei

9.78.1 Detailed Description

ECE outer tag.

Definition at line 718 of file vtss_evc_api.h.

9.78.2 Field Documentation

9.78.2.1 enable

```
BOOL vtss_mce_outer_tag_t::enable
```

Enable tag

Definition at line 720 of file vtss_evc_api.h.

9.78.2.2 vid

```
vtss_vid_t vtss_mce_outer_tag_t::vid
```

VLAN ID

Definition at line 721 of file vtss_evc_api.h.

9.78.2.3 pcp_mode

```
vtss_mce_pcp_mode_t vtss_mce_outer_tag_t::pcp_mode
```

PCP mode

Definition at line 722 of file vtss_evc_api.h.

9.78.2.4 pcp

```
vtss_tagprio_t vtss_mce_outer_tag_t::pcp
```

PCP value

Definition at line 723 of file vtss_evc_api.h.

9.78.2.5 dei_mode

```
vtss_mce_dei_mode_t vtss_mce_outer_tag_t::dei_mode
```

DEI mode

Definition at line 724 of file vtss_evc_api.h.

9.78.2.6 dei

```
vtss_dei_t vtss_mce_outer_tag_t::dei
```

DEI value

Definition at line 725 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.79 vtss_mce_port_info_t Struct Reference

MEP Control Entry port information.

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

Data Fields

- vtss_isdx_t isdx

9.79.1 Detailed Description

MEP Control Entry port information.

Definition at line 830 of file vtss_evc_api.h.

9.79.2 Field Documentation

9.79.2.1 isdx

```
vtss_isdx_t vtss_mce_port_info_t::isdx
```

Ingress service index

Definition at line 832 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.80 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

9.80.1 Detailed Description

MEP Control Entry.

Definition at line 784 of file vtss_evc_api.h.

9.80.2 Field Documentation

9.80.2.1 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.

9.80.2.2 id

```
vtss_mce_id_t vtss_mce_t::id
```

Entry ID

Definition at line 787 of file vtss_evc_api.h.

9.80.2.3 key

```
vtss_mce_key_t vtss_mce_t::key
```

MCE key

Definition at line 788 of file vtss_evc_api.h.

9.80.2.4 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

9.81 vtss_mce_tag_t Struct Reference

MCE tag information.

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

Data Fields

- vtss_vcap_bit_t tagged
- vtss_vcap_bit_t s_tagged
- vtss_vcap_vid_t vid
- vtss_vcap_u8_t pcp
- vtss_vcap_bit_t dei

9.81.1 Detailed Description

MCE tag information.

Definition at line 671 of file vtss_evc_api.h.

9.81.2 Field Documentation

9.81.2.1 tagged

```
vtss_vcap_bit_t vtss_mce_tag_t::tagged
```

Tagged/untagged frame

Definition at line 673 of file vtss_evc_api.h.

9.81.2.2 s_tagged

```
vtss_vcap_bit_t vtss_mce_tag_t::s_tagged
```

S-tagged/C-tagged frame

Definition at line 674 of file vtss_evc_api.h.

9.81.2.3 vid

```
vtss_vcap_vid_t vtss_mce_tag_t::vid
```

VLAN ID (12 bit)

Definition at line 675 of file vtss_evc_api.h.

9.81.2.4 pcp

```
vtss_vcap_u8_t vtss_mce_tag_t::pcp
```

PCP (3 bit)

Definition at line 676 of file vtss_evc_api.h.

9.81.2.5 dei

```
vtss_vcap_bit_t vtss_mce_tag_t::dei
```

DEI

Definition at line 677 of file vtss_evc_api.h.

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

- vtss_api/include/vtss_evc_api.h

9.82 vtss_mirror_conf_t Struct Reference

Mirror configuration.

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

Data Fields

- vtss_port_no_t port_no
- BOOL fwd_enable

9.82.1 Detailed Description

Mirror configuration.

Definition at line 1681 of file vtss_l2_api.h.

9.82.2 Field Documentation

9.82.2.1 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.

9.82.2.2 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

9.83 vtss_mpls_l2_t Struct Reference

Layer-2 configuration.

```
#include <vtss_mpls_api.h>
```

Data Fields

- vtss_port_no_t port
- vtss_mac_t peer_mac
- vtss_mac_t self_mac
- vtss_mll_tagtype_t tag_type
- vtss_vid_t vid
- vtss_tagprio_t pcp
- vtss_dei_t dei
- BOOL section_oam

9.83.1 Detailed Description

Layer-2 configuration.

Definition at line 307 of file vtss_mpls_api.h.

9.83.2 Field Documentation

9.83.2.1 port

`vtss_port_no_t vtss_mpls_l2_t::port`

Port number

Definition at line 308 of file vtss_mpls_api.h.

9.83.2.2 peer_mac

`vtss_mac_t vtss_mpls_l2_t::peer_mac`

MAC address of peer MPLS entity

Definition at line 309 of file vtss_mpls_api.h.

9.83.2.3 self_mac

`vtss_mac_t vtss_mpls_l2_t::self_mac`

MAC address of this MPLS entity

Definition at line 310 of file vtss_mpls_api.h.

9.83.2.4 tag_type

`vtss_mll_tagtype_t vtss_mpls_l2_t::tag_type`

Tag type

Definition at line 311 of file vtss_mpls_api.h.

9.83.2.5 vid

`vtss_vid_t vtss_mpls_l2_t::vid`

VLAN ID, if tag_type isn't untagged

Definition at line 312 of file vtss_mpls_api.h.

9.83.2.6 pcp

```
vtss_tagprio_t vtss_mpls_l2_t::pcp
```

VLAN PCP; only relevant for tagged egress

Definition at line 313 of file vtss_mpls_api.h.

9.83.2.7 dei

```
vtss_dei_t vtss_mpls_l2_t::dei
```

VLAN DEI; only relevant for tagged egress

Definition at line 314 of file vtss_mpls_api.h.

9.83.2.8 section_oam

```
BOOL vtss_mpls_l2_t::section_oam
```

TRUE => accept incoming segment OAM. Note: Only supports ethertype for downsteam-assigned labels

Definition at line 315 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.84 vtss_mpls_label_t Struct Reference

MPLS label. The TTL value is only relevant for LER out-segments.

```
#include <vtss_mpls_api.h>
```

Data Fields

- vtss_mpls_label_value_t value
- vtss_mpls_tc_t tc
- u8 ttl

9.84.1 Detailed Description

MPLS label. The TTL value is only relevant for LER out-segments.

Definition at line 265 of file vtss_mpls_api.h.

9.84.2 Field Documentation

9.84.2.1 value

```
vtss_mpls_label_value_t vtss_mpls_label_t::value
```

Label value

Definition at line 266 of file vtss_mpls_api.h.

9.84.2.2 tc

```
vtss_mpls_tc_t vtss_mpls_label_t::tc
```

Used unless TC mapping is installed

Definition at line 267 of file vtss_mpls_api.h.

9.84.2.3 ttl

```
u8 vtss_mpls_label_t::ttl
```

Used when pushing label (not for swap)

Definition at line 268 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.85 vtss_mpls_oam_conf_t Struct Reference

OAM configuration.

```
#include <vtss_mpls_api.h>
```

Data Fields

- vtss_mpls_oam_type_t type

9.85.1 Detailed Description

OAM configuration.

Definition at line 300 of file vtss_mpls_api.h.

9.85.2 Field Documentation

9.85.2.1 type

```
vtss_mpls_oam_type_t vtss_mpls_oam_conf_t::type
```

OAM type

Definition at line 301 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.86 vtss_mpls_pw_conf_t Struct Reference

Pseudo-wire configuration.

```
#include <vtss_mpls_api.h>
```

Data Fields

- BOOL is_pw
- BOOL process_cw
- u32 cw

9.86.1 Detailed Description

Pseudo-wire configuration.

Definition at line 321 of file vtss_mpls_api.h.

9.86.2 Field Documentation

9.86.2.1 is_pw

```
BOOL vtss_mpls_pw_conf_t::is_pw
```

TRUE == this segment is used for a PW

Definition at line 322 of file vtss_mpls_api.h.

9.86.2.2 process_cw

```
BOOL vtss_mpls_pw_conf_t::process_cw
```

TRUE for in-segment: Check and pop CW. TRUE for out-segment: Add CW in cw

Definition at line 323 of file vtss_mpls_api.h.

9.86.2.3 cw

```
u32 vtss_mpls_pw_conf_t::cw
```

Control Word to use for out-segments

Definition at line 324 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.87 vtss_mpls_qos_to_tc_map_entry_t Struct Reference

QoS-to-TC map entry.

```
#include <vtss_mpls_api.h>
```

Data Fields

- vtss_mpls_tc_t dp0_tc
- vtss_mpls_tc_t dp1_tc

9.87.1 Detailed Description

QoS-to-TC map entry.

Definition at line 280 of file vtss_mpls_api.h.

9.87.2 Field Documentation

9.87.2.1 dp0_tc

```
vtss_mpls_tc_t vtss_mpls_qos_to_tc_map_entry_t::dp0_tc
```

TC value for DP == 0. Valid range is [0..7]

Definition at line 281 of file vtss_mpls_api.h.

9.87.2.2 dp1_tc

```
vtss_mpls_tc_t vtss_mpls_qos_to_tc_map_entry_t::dp1_tc
```

TC value for DP == 1. Valid range is [0..7]

Definition at line 282 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.88 vtss_mpls_segment_status_t Struct Reference

Segment status. Contains various kinds of status/state information for a segment.

```
#include <vtss_mpls_api.h>
```

Data Fields

- vtss_mpls_segment_state_t state
- BOOL oam_active

9.88.1 Detailed Description

Segment status. Contains various kinds of status/state information for a segment.

Definition at line 225 of file vtss_mpls_api.h.

9.88.2 Field Documentation

9.88.2.1 state

```
vtss_mpls_segment_state_t vtss_mpls_segment_status_t::state
```

Segement state

Definition at line 226 of file vtss_mpls_api.h.

9.88.2.2 oam_active

```
BOOL vtss_mpls_segment_status_t::oam_active
```

TRUE if OAM is configured AND valid given the current label stack AND any HW has been successfully allocated

Definition at line 227 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.89 vtss_mpls_segment_t Struct Reference

Segment configuration.

```
#include <vtss_mpls_api.h>
```

Data Fields

- BOOL is_in
- BOOL e_lsp
- vtss_mpls_cos_t l_lsp_cos
- i8 tc_qos_map_idx
- vtss_mpls_l2_idx_t l2_idx
- vtss_mpls_label_t label
- BOOL upstream
- vtss_hqos_id_t hqos_id
- vtss_mpls_pw_conf_t pw_conf
- vtss_mpls_oam_conf_t oam_conf
- vtss_mpls_xc_idx_t xc_idx
- vtss_mpls_segment_idx_t server_idx

9.89.1 Detailed Description

Segment configuration.

All indices are VTSS_MPLS_IDX_UNDEFINED if unused/uninitialized.

Definition at line 333 of file vtss_mpls_api.h.

9.89.2 Field Documentation

9.89.2.1 is_in

```
BOOL vtss_mpls_segment_t::is_in
```

[RO] TRUE == in-segment; FALSE == out-segment

Definition at line 334 of file vtss_mpls_api.h.

9.89.2.2 e_lsp

```
BOOL vtss_mpls_segment_t::e_lsp
```

[RW] TRUE == E-LSP; FALSE == L-LSP

Definition at line 335 of file vtss_mpls_api.h.

9.89.2.3 l_lsp_cos

```
vtss_mpls_cos_t vtss_mpls_segment_t::l_lsp_cos
```

[RW] Fixed COS: Ingress TC is mapped to this value for L-LSPs

Definition at line 336 of file vtss_mpls_api.h.

9.89.2.4 tc_qos_map_idx

```
i8 vtss_mpls_segment_t::tc_qos_map_idx
```

[RW] In-segment: Ingress TC-to-(QoS,DP) map: L-LSP: Only DP is mapped; E-LSP: Both QoS and DP are mapped. VTSS_MPLS_IDX_UNDEF = no mapping. Out-segment: Egress (QoS,DP)-to-TC map. VTSS_MPLS_IDX_UNDEF = no mapping

Definition at line 337 of file vtss_mpls_api.h.

9.89.2.5 l2_idx

```
vtss_mpls_l2_idx_t vtss_mpls_segment_t::l2_idx
```

[RO] L2 index

Definition at line 339 of file vtss_mpls_api.h.

9.89.2.6 label

```
vtss_mpls_label_t vtss_mpls_segment_t::label
```

[RW] Label value, TC, TTL

Definition at line 340 of file vtss_mpls_api.h.

9.89.2.7 upstream

```
BOOL vtss_mpls_segment_t::upstream
```

[RW] TRUE == Label is upstream-assigned

Definition at line 341 of file vtss_mpls_api.h.

9.89.2.8 hqos_id

```
vtss_hqos_id_t vtss_mpls_segment_t::hqos_id
```

[RW] Hierarchical QoS ID or VTSS_HQOS_ID_NONE for no HQoS. Only valid for LSR in-segments

Definition at line 343 of file vtss_mpls_api.h.

9.89.2.9 pw_conf

```
vtss_mpls_pw_conf_t vtss_mpls_segment_t::pw_conf
```

[RW] Pseudo-wire configuration

Definition at line 345 of file vtss_mpls_api.h.

9.89.2.10 oam_conf

```
vtss_mpls_oam_conf_t vtss_mpls_segment_t::oam_conf
```

[RW] OAM configuration

Definition at line 346 of file vtss_mpls_api.h.

9.89.2.11 xc_idx

```
vtss_mpls_xc_idx_t vtss_mpls_segment_t::xc_idx
```

[RO] Index of XC using this segment

Definition at line 347 of file vtss_mpls_api.h.

9.89.2.12 server_idx

```
vtss_mpls_segment_idx_t vtss_mpls_segment_t::server_idx
```

[RO] Server (outer) segment

Definition at line 348 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.90 vtss_mpls_tc_conf_t Struct Reference

TC/QoS map configuration.

```
#include <vtss_mpls_api.h>
```

Data Fields

- vtss_mpls_qos_to_tc_map_entry_t qos_to_tc_map [VTSS_MPLS_QOS_TO_TC_MAP_CNT][VTSS_MPLS_QOS_TO_TC_EN
- vtss_mpls_tc_to_qos_map_entry_t tc_to_qos_map [VTSS_MPLS_TC_TO_QOS_MAP_CNT][VTSS_MPLS_TC_TO_QOS_EN

9.90.1 Detailed Description

TC/QoS map configuration.

Definition at line 292 of file vtss_mpls_api.h.

9.90.2 Field Documentation

9.90.2.1 qos_to_tc_map

```
vtss_mpls_qos_to_tc_map_entry_t vtss_mpls_tc_conf_t::qos_to_tc_map[VTSS_MPLS_QOS_TO_TC_MAP_CNT][VTSS_MPLS_QOS_
```

(QoS, DP) => TC tables. Access with [map][qos].dp0_tc (or .dp1_tc)

Definition at line 293 of file vtss_mpls_api.h.

9.90.2.2 tc_to_qos_map

```
vtss_mpls_tc_to_qos_map_entry_t vtss_mpls_tc_conf_t::tc_to_qos_map[VTSS_MPLS_TC_TO_QOS_MAP_CNT][VTSS_MPLS_TC_T
```

TC => (QoS, DP) tables.

Definition at line 294 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.91 vtss_mpls_tc_to_qos_map_entry_t Struct Reference

TC-to-(QoS,DP) map entry.

```
#include <vtss_mpls_api.h>
```

Data Fields

- vtss_mpls_qos_t qos
- u8 dp

9.91.1 Detailed Description

TC-to-(QoS,DP) map entry.

Definition at line 286 of file vtss_mpls_api.h.

9.91.2 Field Documentation

9.91.2.1 qos

```
vtss_mpls_qos_t vtss_mpls_tc_to_qos_map_entry_t::qos
```

QoS value, valid range [0..7]

Definition at line 287 of file vtss_mpls_api.h.

9.91.2.2 dp

```
u8 vtss_mpls_tc_to_qos_map_entry_t::dp
```

DP value, valid range [0..1]

Definition at line 288 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.92 vtss_mpls_xc_counters_t Struct Reference

MPLS cross-connect counters.

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

Data Fields

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

9.92.1 Detailed Description

MPLS cross-connect counters.

Definition at line 1147 of file types.h.

9.92.2 Field Documentation

9.92.2.1 rx_green

```
vtss_counter_pair_t vtss_mpls_xc_counters_t::rx_green
```

Rx green frames/bytes

Definition at line 1148 of file types.h.

9.92.2.2 rx_yellow

```
vtss_counter_pair_t vtss_mpls_xc_counters_t::rx_yellow
```

Rx yellow frames/bytes

Definition at line 1149 of file types.h.

9.92.2.3 rx_red

```
vtss_counter_pair_t vtss_mpls_xc_counters_t::rx_red
```

Rx red frames/bytes

Definition at line 1150 of file types.h.

9.92.2.4 rx_discard

```
vtss_counter_pair_t vtss_mpls_xc_counters_t::rx_discard
```

Rx discarded frames/bytes

Definition at line 1151 of file types.h.

9.92.2.5 tx_discard

```
vtss_counter_pair_t vtss_mpls_xc_counters_t::tx_discard
```

Tx discarded frames/bytes

Definition at line 1152 of file types.h.

9.92.2.6 tx_green

```
vtss_counter_pair_t vtss_mpls_xc_counters_t::tx_green
```

Tx green frames/bytes

Definition at line 1153 of file types.h.

9.92.2.7 tx_yellow

```
vtss_counter_pair_t vtss_mpls_xc_counters_t::tx_yellow
```

Tx yellow frames/bytes

Definition at line 1154 of file types.h.

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

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

9.93 vtss_mpls_xc_t Struct Reference

Cross Connect function for unidirectional LSP/PW.

```
#include <vtss_mpls_api.h>
```

Data Fields

- vtss_mpls_xc_type_t type
- vtss_mpls_tunnel_mode_t tc_tunnel_mode
- vtss_mpls_tunnel_mode_t ttl_tunnel_mode
- vtss_mpls_segment_idx_t in_seg_idx
- vtss_mpls_segment_idx_t out_seg_idx
- vtss_isdx_t flow_handle

9.93.1 Detailed Description

Cross Connect function for unidirectional LSP/PW.

A transit (label-swapped) LSP has both an in- and out-segment A terminating LSP/PW only has an in-segment. An initiating LSP/PW only has an out-segment.

A segment which isn't attached to an XC cannot transport traffic.

Most fields should not be set directly; treat them as R/O and use the appropriate vtss_mpls_...() functions for manipulation.

All indices are VTSS_MPLS_IDX_UNDEFINED if unused/uninitialized.

Definition at line 366 of file vtss_mpls_api.h.

9.93.2 Field Documentation

9.93.2.1 type

`vtss_mpls_xc_type_t vtss_mpls_xc_t::type`

[RO] XC type

Definition at line 367 of file vtss_mpls_api.h.

9.93.2.2 tc_tunnel_mode

`vtss_mpls_tunnel_mode_t vtss_mpls_xc_t::tc_tunnel_mode`

[RW] DiffServ tunneling mode for TC. Default: VTSS_MPLS_TUNNEL_MODE_PIPE

Definition at line 368 of file vtss_mpls_api.h.

9.93.2.3 ttl_tunnel_mode

`vtss_mpls_tunnel_mode_t vtss_mpls_xc_t::ttl_tunnel_mode`

[RW] DiffServ tunneling mode for TTL. Default: VTSS_MPLS_TUNNEL_MODE_PIPE

Definition at line 369 of file vtss_mpls_api.h.

9.93.2.4 in_seg_idx

`vtss_mpls_segment_idx_t vtss_mpls_xc_t::in_seg_idx`

[RO] Index of in-segment

Definition at line 370 of file vtss_mpls_api.h.

9.93.2.5 out_seg_idx

`vtss_mpls_segment_idx_t vtss_mpls_xc_t::out_seg_idx`

[RO] Index of out-segment

Definition at line 371 of file vtss_mpls_api.h.

9.93.2.6 flow_handle

```
vtss_isdx_t vtss_mpls_xc_t::flow_handle
```

[RO] Opaque and unique flow handle for MPLS frames extracted to CPU; only valid when in-seg is attached and state UP. Value is ignored when writing to the XC config. NOTE: Value may change if in-seg goes out of state UP and back.

Definition at line 372 of file vtss_mpls_api.h.

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

- vtss_api/include/vtss_mpls_api.h

9.94 vtss_mtimer_t Struct Reference

Timer structure.

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

Data Fields

- struct timeval timeout
- struct timeval now

9.94.1 Detailed Description

Timer structure.

Definition at line 88 of file vtss_os_linux.h.

9.94.2 Field Documentation

9.94.2.1 timeout

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

Timeout

Definition at line 89 of file vtss_os_linux.h.

9.94.2.2 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

9.95 vtss_npi_conf_t Struct Reference

NPI configuration.

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

Data Fields

- BOOL enable
- vtss_port_no_t port_no

9.95.1 Detailed Description

NPI configuration.

Definition at line 48 of file vtss_packet_api.h.

9.95.2 Field Documentation

9.95.2.1 enable

```
BOOL vtss_npi_conf_t::enable
```

Enable NPI port

Definition at line 49 of file vtss_packet_api.h.

9.95.2.2 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

9.96 vtss_oam_ccm_counter_t Struct Reference

CCM counters.

```
#include <vtss_oam_api.h>
```

Data Fields

- u64 rx_valid_count
- u64 rx_invalid_count
- u64 rx_invalid_seq_no

9.96.1 Detailed Description

CCM counters.

Definition at line 818 of file vtss_oam_api.h.

9.96.2 Field Documentation

9.96.2.1 rx_valid_count

```
u64 vtss_oam_ccm_counter_t::rx_valid_count
```

Count of valid RX CCM PDUs

Definition at line 819 of file vtss_oam_api.h.

9.96.2.2 rx_invalid_count

```
u64 vtss_oam_ccm_counter_t::rx_invalid_count
```

Count of invalid RX CCM PDUs

Definition at line 820 of file vtss_oam_api.h.

9.96.2.3 rx_invalid_seq_no

```
u64 vtss_oam_ccm_counter_t::rx_invalid_seq_no
```

Count of RX CCM PDUs with invalid sequence number. NOTE: Inexact for Serval; do not rely on precise value.

Definition at line 829 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.97 vtss_oam_ccm_status_t Struct Reference

Status for most recently processed RX CCM PDU.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL period_err
- BOOL priority_err
- BOOL zero_period_err
- BOOL rx_rdi
- BOOL loc
- BOOL mep_id_err
- BOOL meg_id_err

9.97.1 Detailed Description

Status for most recently processed RX CCM PDU.

Definition at line 666 of file vtss_oam_api.h.

9.97.2 Field Documentation

9.97.2.1 period_err

```
BOOL vtss_oam_ccm_status_t::period_err
```

Status for last RX frame

Definition at line 667 of file vtss_oam_api.h.

9.97.2.2 priority_err

```
BOOL vtss_oam_ccm_status_t::priority_err
```

Status for last RX frame

Definition at line 668 of file vtss_oam_api.h.

9.97.2.3 zero_period_err

```
BOOL vtss_oam_ccm_status_t::zero_period_err
```

Status for last RX frame

Definition at line 669 of file vtss_oam_api.h.

9.97.2.4 rx_rdi

```
BOOL vtss_oam_ccm_status_t::rx_rdi
```

Status for last RX frame

Definition at line 670 of file vtss_oam_api.h.

9.97.2.5 loc

```
BOOL vtss_oam_ccm_status_t::loc
```

Status for last RX frame

Definition at line 671 of file vtss_oam_api.h.

9.97.2.6 mep_id_err

```
BOOL vtss_oam_ccm_status_t::mep_id_err
```

Status for last RX frame

Definition at line 672 of file vtss_oam_api.h.

9.97.2.7 meg_id_err

```
BOOL vtss_oam_ccm_status_t::meg_id_err
```

Status for last RX frame

Definition at line 673 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.98 vtss_oam_event_mask_t Struct Reference

Event polling across all VOEs.

```
#include <vtss_oam_api.h>
```

Data Fields

- u32 voe_mask [VTSS_OAM_EVENT_MASK_ARRAY_SIZE]

9.98.1 Detailed Description

Event polling across all VOEs.

Definition at line 241 of file vtss_oam_api.h.

9.98.2 Field Documentation

9.98.2.1 voe_mask

```
u32 vtss_oam_event_mask_t::voe_mask[VTSS_OAM_EVENT_MASK_ARRAY_SIZE]
```

Bit mask for VOEs with pending interrupts. LSB of [0] is VOE index 0.

Definition at line 242 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.99 vtss_oam_ipt_conf_t Struct Reference

OAM-related configuration for the IPT table.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- vtss_oam_voe_idx_t service_voe_idx

9.99.1 Detailed Description

OAM-related configuration for the IPT table.

Definition at line 941 of file vtss_oam_api.h.

9.99.2 Field Documentation

9.99.2.1 enable

```
BOOL vtss_oam_ipt_conf_t::enable
```

TRUE => Use this MEP index

Definition at line 942 of file vtss_oam_api.h.

9.99.2.2 service_voe_idx

```
vtss_oam_voe_idx_t vtss_oam_ipt_conf_t::service_voe_idx
```

Associated (Service) VOE

Definition at line 943 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.100 vtss_oam_lb_counter_t Struct Reference

LB counter.

```
#include <vtss_oam_api.h>
```

Data Fields

- u64 rx_lbr
- u64 tx_lbm
- u64 rx_lbr_trans_id_err

9.100.1 Detailed Description

LB counter.

Definition at line 846 of file vtss_oam_api.h.

9.100.2 Field Documentation

9.100.2.1 rx_lbr

```
u64 vtss_oam_lb_counter_t::rx_lbr
```

Count of RX LBR PDUs

Definition at line 847 of file vtss_oam_api.h.

9.100.2.2 tx_lbm

```
u64 vtss_oam_lb_counter_t::tx_lbm
```

Count of TX LBM PDUs

Definition at line 848 of file vtss_oam_api.h.

9.100.2.3 rx_lbr_trans_id_err

```
u64 vtss_oam_lb_counter_t::rx_lbr_trans_id_err
```

Count of RX'd LBR PDUs with invalid transaction ID. NOTE: Inexact for Serval; do not rely on precise value.

Definition at line 857 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.101 vtss_oam_lm_counter_t Struct Reference

Up-/Down-MEP LM counters and per-priority RX/TX LM counters.

```
#include <vtss_oam_api.h>
```

Data Fields

- struct {
  - u32 tx_FCf
  - u32 rx_FCl

  } down_mep

- struct {
  - u32 rx_lmm_sample_seq_no
  - u32 rx_lmr_sample_seq_no
  - u32 rx_ccm_lm_sample_seq_no
  - u32 lmm
  - u32 lmr
  - u32 ccm_lm
  - BOOL lmm_valid
  - BOOL lmr_valid
  - BOOL ccm_lm_valid
  - BOOL lmm_sample_lost
  - BOOL lmr_sample_lost
  - BOOL ccm_lm_sample_lost

  } up_mep

- vtss_oam_rx_tx_counter_t lm_count [VTSS_PRIO_ARRAY_SIZE]

9.101.1 Detailed Description

Up-/Down-MEP LM counters and per-priority RX/TX LM counters.

Definition at line 781 of file vtss_oam_api.h.

9.101.2 Field Documentation

9.101.2.1 tx_FCf

```
u32 vtss_oam_lm_counter_t::tx_FCf
```

CCM-LM: Latest received FCf. Cannot be cleared.

Definition at line 783 of file vtss_oam_api.h.

9.101.2.2 rx_FCl

```
u32 vtss_oam_lm_counter_t::rx_FCl
```

CCM-LM: Latest sampled CCM-LM RX frame counter. Cannot be cleared.

Definition at line 784 of file vtss_oam_api.h.

9.101.2.3 down_mep

```
struct { ... } vtss_oam_lm_counter_t::down_mep
```

Down-MEP specific

9.101.2.4 rx_lmm_sample_seq_no

```
u32 vtss_oam_lm_counter_t::rx_lmm_sample_seq_no
```

When LM values are sampled by Serval they cannot be provided as updates to the received PDU. Instead the most recent values are stored in the VOE and a sample sequence number is updated and sent to the CPU as PDU sideband data. It is then up to the application code to match an extracted LM PDU with the values in the VOE based on comparing the sample sequence number values from the sideband data and the appropriate field here.

Future HW will provide simpler functionality.Serval: 5-bit LMR sample sequence number (in-service only). Cannot be cleared.

Definition at line 798 of file vtss_oam_api.h.

9.101.2.5 rx_lmr_sample_seq_no

```
u32 vtss_oam_lm_counter_t::rx_lmr_sample_seq_no
```

Serval: 5-bit LMM sample sequence number. Cannot be cleared.

Definition at line 799 of file vtss_oam_api.h.

9.101.2.6 rx_ccm_lm_sample_seq_no

```
u32 vtss_oam_lm_counter_t::rx_ccm_lm_sample_seq_no
```

Serval: 5-bit CCM-LM sample sequence number. Cannot be cleared.

Definition at line 800 of file vtss_oam_api.h.

9.101.2.7 lmm

```
u32 vtss_oam_lm_counter_t::lmm
```

LMM: Latest sampled LM RX frame counter. Cannot be cleared.

Definition at line 802 of file vtss_oam_api.h.

9.101.2.8 lmr

```
u32 vtss_oam_lm_counter_t::lmr
```

LMR: Latest sampled LM RX frame counter. Cannot be cleared.

Definition at line 803 of file vtss_oam_api.h.

9.101.2.9 ccm_lm

```
u32 vtss_oam_lm_counter_t::ccm_lm
```

CCM-LM: Latest sampled LM RX frame counter. Cannot be cleared.

Definition at line 804 of file vtss_oam_api.h.

9.101.2.10 lmm_valid

```
BOOL vtss_oam_lm_counter_t::lmm_valid
```

TRUE => value in lmm is valid. Cannot be cleared.

Definition at line 806 of file vtss_oam_api.h.

9.101.2.11 lmr_valid

```
BOOL vtss_oam_lm_counter_t::lmr_valid
```

TRUE => value in lmr is valid. Cannot be cleared.

Definition at line 807 of file vtss_oam_api.h.

9.101.2.12 ccm_lm_valid

```
BOOL vtss_oam_lm_counter_t::ccm_lm_valid
```

TRUE => value in ccm_lm is valid. Cannot be cleared.

Definition at line 808 of file vtss_oam_api.h.

9.101.2.13 lmm_sample_lost

```
BOOL vtss_oam_lm_counter_t::lmm_sample_lost
```

TRUE => one or more lmm sample values have been lost since last poll. Cannot be cleared.

Definition at line 810 of file vtss_oam_api.h.

9.101.2.14 lmr_sample_lost

```
BOOL vtss_oam_lm_counter_t::lmr_sample_lost
```

TRUE => one or more lmr sample values have been lost since last poll. Cannot be cleared.

Definition at line 811 of file vtss_oam_api.h.

9.101.2.15 ccm_lm_sample_lost

```
BOOL vtss_oam_lm_counter_t::ccm_lm_sample_lost
```

TRUE => one or more ccm-lm sample values have been lost since last poll. Cannot be cleared.

Definition at line 812 of file vtss_oam_api.h.

9.101.2.16 up_mep

```
struct { ... } vtss_oam_lm_counter_t::up_mep
```

Up-MEP specific

9.101.2.17 lm_count

```
vtss_oam_rx_tx_counter_t vtss_oam_lm_counter_t::lm_count[VTSS_PRIO_ARRAY_SIZE]
```

One counter-set per priority

Definition at line 814 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.102 vtss_oam_meg_id_t Struct Reference

MEG ID type.

```
#include <vtss_oam_api.h>
```

Data Fields

- u8 data [48]

9.102.1 Detailed Description

MEG ID type.

Definition at line 319 of file vtss_oam_api.h.

9.102.2 Field Documentation

9.102.2.1 data

```
u8 vtss_oam_meg_id_t::data[48]
```

MEG ID bytes

Definition at line 320 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.103 vtss_oam_pdu_seen_status_t Struct Reference

Indicate whether one or more PDUs of various types has been seen since last check. A PDU must pass MEG Level and DMAC checks before being indicated here. The values are cleared after reading.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL generic_seen [VTSS_OAM_GENERIC_OPCODE_CFG_CNT]
- BOOL unknown_seen
- BOOL ltm_seen
- BOOL ltr_seen
- BOOL lmm_seen
- BOOL lmr_seen
- BOOL lbm_seen
- BOOL lbr_seen
- BOOL dmm_seen
- BOOL dmr_seen
- BOOL one_dm_seen
- BOOL ccm_seen
- BOOL ccm_lm_seen

9.103.1 Detailed Description

Indicate whether one or more PDUs of various types has been seen since last check. A PDU must pass MEG Level and DMAC checks before being indicated here. The values are cleared after reading.

Definition at line 692 of file vtss_oam_api.h.

9.103.2 Field Documentation

9.103.2.1 generic_seen

```
BOOL vtss_oam_pdu_seen_status_t::generic_seen[VTSS_OAM_GENERIC_OPCODE_CFG_CNT]
```

Generic opcodes

Definition at line 693 of file vtss_oam_api.h.

9.103.2.2 unknown_seen

```
BOOL vtss_oam_pdu_seen_status_t::unknown_seen
```

Unknown opcode

Definition at line 694 of file vtss_oam_api.h.

9.103.2.3 ltm_seen

```
BOOL vtss_oam_pdu_seen_status_t::ltm_seen
```

LTM

Definition at line 695 of file vtss_oam_api.h.

9.103.2.4 ltr_seen

```
BOOL vtss_oam_pdu_seen_status_t::ltr_seen
```

LTR

Definition at line 696 of file vtss_oam_api.h.

9.103.2.5 lmm_seen

```
BOOL vtss_oam_pdu_seen_status_t::lmm_seen
```

LMM

Definition at line 697 of file vtss_oam_api.h.

9.103.2.6 lmr_seen

```
BOOL vtss_oam_pdu_seen_status_t::lmr_seen
```

LMR

Definition at line 698 of file vtss_oam_api.h.

9.103.2.7 lbm_seen

```
BOOL vtss_oam_pdu_seen_status_t::lbm_seen
```

LBM

Definition at line 699 of file vtss_oam_api.h.

9.103.2.8 lbr_seen

```
BOOL vtss_oam_pdu_seen_status_t::lbr_seen
```

LBR

Definition at line 700 of file vtss_oam_api.h.

9.103.2.9 dmm_seen

```
BOOL vtss_oam_pdu_seen_status_t::dmm_seen
```

DMM

Definition at line 701 of file vtss_oam_api.h.

9.103.2.10 dmr_seen

```
BOOL vtss_oam_pdu_seen_status_t::dmr_seen
```

DMR

Definition at line 702 of file vtss_oam_api.h.

9.103.2.11 one_dm_seen

```
BOOL vtss_oam_pdu_seen_status_t::one_dm_seen
```

1DM

Definition at line 703 of file vtss_oam_api.h.

9.103.2.12 ccm_seen

```
BOOL vtss_oam_pdu_seen_status_t::ccm_seen
```

CCM, including CCM-LM

Definition at line 704 of file vtss_oam_api.h.

9.103.2.13 ccm_lm_seen

```
BOOL vtss_oam_pdu_seen_status_t::ccm_lm_seen
```

CCM-LM (only)

Definition at line 705 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.104 vtss_oam_proc_conf_t Struct Reference

PDU processing checks.

```
#include <vtss_oam_api.h>
```

Data Fields

- u32 meg_level
- vtss_oam_dmac_check_t dmac_check_type
- BOOL ccm_check_only
- BOOL copy_next_only
- BOOL copy_on_ccm_err
- BOOL copy_on_mel_too_low_err
- BOOL copy_on_ccm_more_than_one_tlv
- BOOL copy_on_dmac_err

9.104.1 Detailed Description

PDU processing checks.

Definition at line 520 of file vtss_oam_api.h.

9.104.2 Field Documentation

9.104.2.1 meg_level

```
u32 vtss_oam_proc_conf_t::meg_level
```

MEG Level (MEL)

Definition at line 521 of file vtss_oam_api.h.

9.104.2.2 dmac_check_type

```
vtss_oam_dmac_check_t vtss_oam_proc_conf_t::dmac_check_type
```

Kind of DMAC check to perform

Definition at line 522 of file vtss_oam_api.h.

9.104.2.3 ccm_check_only

```
BOOL vtss_oam_proc_conf_t::ccm_check_only
```

TRUE => Only check CCM frames for valid MEL/valid DMAC/version==0; FALSE => check all OAM frames

Definition at line 523 of file vtss_oam_api.h.

9.104.2.4 copy_next_only

```
BOOL vtss_oam_proc_conf_t::copy_next_only
```

TRUE => For most of the following 'copy_on_...' flags: Copy next PDU only ("hit me once"); FALSE => copy all

Definition at line 525 of file vtss_oam_api.h.

9.104.2.5 copy_on_ccm_err

```
BOOL vtss_oam_proc_conf_t::copy_on_ccm_err
```

TRUE => Copy to CPU if CCM validation fails (honors copy_next_only)

Definition at line 527 of file vtss_oam_api.h.

9.104.2.6 copy_on_mel_too_low_err

```
BOOL vtss_oam_proc_conf_t::copy_on_mel_too_low_err
```

TRUE => Copy to CPU on MEG Level error (honors copy_next_only)

Definition at line 528 of file vtss_oam_api.h.

9.104.2.7 copy_on_ccm_more_than_one_tlv

```
BOOL vtss_oam_proc_conf_t::copy_on_ccm_more_than_one_tlv
```

TRUE => Copy to CPU when CCM PDU with more than one TLV arrives (Serval: Only used when copy_next_only == TRUE)

Definition at line 531 of file vtss_oam_api.h.

9.104.2.8 copy_on_dmac_err

```
BOOL vtss_oam_proc_conf_t::copy_on_dmac_err
```

TRUE => Copy to CPU on DMAC error (no "hit me once")

Definition at line 532 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.105 vtss_oam_proc_status_t Struct Reference

Processing status: change indications and status values for most recently processed PDU(s). ..._err: Status for the most recently processed PDU. ..._seen: This has happened at least once since last poll. Cleared after reading.

```
#include <vtss_oam_api.h>
```

Data Fields

- u32 rx_ccm_seq_no
- u32 tx_next_ccm_seq_no
- u32 rx_lbr_transaction_id
- u32 rx_tst_seq_no
- u32 tx_next_lbm_transaction_id
- BOOL rx_meg_level_err
- BOOL rx_meg_level_err_seen
- BOOL rx_dmac_err_seen
- BOOL tx_meg_level_err_seen

9.105.1 Detailed Description

Processing status: change indications and status values for most recently processed PDU(s). ..._err: Status for the most recently processed PDU. ..._seen: This has happened at least once since last poll. Cleared after reading.

Definition at line 725 of file vtss_oam_api.h.

9.105.2 Field Documentation

9.105.2.1 rx_ccm_seq_no

```
u32 vtss_oam_proc_status_t::rx_ccm_seq_no
```

Most recently received CCM/CCM-LM sequence number. Note: For Serval-1 and up-MEP this value only contains the lower 16 bits of the sequence number.

Definition at line 727 of file vtss_oam_api.h.

9.105.2.2 tx_next_ccm_seq_no

```
u32 vtss_oam_proc_status_t::tx_next_ccm_seq_no
```

Next CCM/CCM-LM sequence number to be used (the value is updated by HW upon TX, if so enabled)

Definition at line 732 of file vtss_oam_api.h.

9.105.2.3 rx_lbr_transaction_id

```
u32 vtss_oam_proc_status_t::rx_lbr_transaction_id
```

NOTE that for Serval rx_tst_seq_no overlays rx_lbr_transaction_id due to HW register re-use. But logically they're distinct and future HW is expected to reflect this.Most recently received LBR transaction ID (NOTE: On Serval, overlays rx_tst_seq_no)

Definition at line 739 of file vtss_oam_api.h.

9.105.2.4 rx_tst_seq_no

```
u32 vtss_oam_proc_status_t::rx_tst_seq_no
```

Most recently received TST sequence number (NOTE: On Serval, overlays rx_lbr_transaction_id)

Definition at line 740 of file vtss_oam_api.h.

9.105.2.5 tx_next_lbm_transaction_id

```
u32 vtss_oam_proc_status_t::tx_next_lbm_transaction_id
```

Next LBM transaction ID to be used (the value is updated by HW upon TX, if so enabled)

Definition at line 746 of file vtss_oam_api.h.

9.105.2.6 rx_meg_level_err

```
BOOL vtss_oam_proc_status_t::rx_meg_level_err
```

Status for most recently RX'd frame

Definition at line 748 of file vtss_oam_api.h.

9.105.2.7 rx_meg_level_err_seen

```
BOOL vtss_oam_proc_status_t::rx_meg_level_err_seen
```

TRUE => MEG Level (MEL) test failed

Definition at line 750 of file vtss_oam_api.h.

9.105.2.8 rx_dmac_err_seen

```
BOOL vtss_oam_proc_status_t::rx_dmac_err_seen
```

TRUE => DMAC test failed

Definition at line 751 of file vtss_oam_api.h.

9.105.2.9 tx_meg_level_err_seen

```
BOOL vtss_oam_proc_status_t::tx_meg_level_err_seen
```

TRUE => TX MEG Level test failed

Definition at line 752 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.106 vtss_oam_rx_tx_counter_t Struct Reference

Simple RX/TX counter structure.

```
#include <vtss_oam_api.h>
```

Data Fields

- u64 rx
- u64 tx

9.106.1 Detailed Description

Simple RX/TX counter structure.

Some counters can be cleared, others not. If a counter cannot be cleared, by calling vtss_oam_voe_counter_clear() this is indicated in the comments for the counter.

Definition at line 775 of file vtss_oam_api.h.

9.106.2 Field Documentation

9.106.2.1 rx

```
u64 vtss_oam_rx_tx_counter_t::rx
```

RX count

Definition at line 776 of file vtss_oam_api.h.

9.106.2.2 tx

```
u64 vtss_oam_rx_tx_counter_t::tx
```

TX count

Definition at line 777 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.107 vtss_oam_sel_counter_t Struct Reference

RX/TX OAM PDU counters. By default, all OAM PDU types are counted as 'non-selected', but by setting the appropriate count_as_selected field in the configuration structures, a PDU type can be "moved" to the 'selected' counter.

```
#include <vtss_oam_api.h>
```

Data Fields

- vtss_oam_rx_tx_counter_t selected_frames
- vtss_oam_rx_tx_counter_t nonselected_frames

9.107.1 Detailed Description

RX/TX OAM PDU counters. By default, all OAM PDU types are counted as 'non-selected', but by setting the appropriate count_as_selected field in the configuration structures, a PDU type can be "moved" to the 'selected' counter.

Definition at line 840 of file vtss_oam_api.h.

9.107.2 Field Documentation

9.107.2.1 selected_frames

```
vtss_oam_rx_tx_counter_t vtss_oam_sel_counter_t::selected_frames
```

Count of selected OAM frames

Definition at line 841 of file vtss_oam_api.h.

9.107.2.2 nonselected_frames

```
vtss_oam_rx_tx_counter_t vtss_oam_sel_counter_t::nonselected_frames
```

Count of non-selected OAM frames

Definition at line 842 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.108 vtss_oam_ts_id_s Struct Reference

parameter for requesting an oam timestamp

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

Data Fields

- u32 voe_id
- u32 voe_sq

9.108.1 Detailed Description

parameter for requesting an oam timestamp

Definition at line 585 of file vtss_ts_api.h.

9.108.2 Field Documentation

9.108.2.1 voe_id

```
u32 vtss_oam_ts_id_s::voe_id
```

VOE instance (Timestamp) identifier

Definition at line 586 of file vtss_ts_api.h.

9.108.2.2 voe_sq

```
u32 vtss_oam_ts_id_s::voe_sq
```

VOE (Timestamp) sequence no

Definition at line 587 of file vtss_ts_api.h.

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

- vtss_api/include/vtss_ts_api.h

9.109 vtss_oam_ts_timestamp_s Struct Reference

parameter for returning an oam timestamp

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

Data Fields

- u32 ts
- vtss_port_no_t port_no
- BOOL ts_valid

9.109.1 Detailed Description

parameter for returning an oam timestamp

Definition at line 593 of file vtss_ts_api.h.

9.109.2 Field Documentation

9.109.2.1 ts

```
u32 vtss_oam_ts_timestamp_s::ts
```

Timestamp value (ns + sec∗10^9) mod 2^32

Definition at line 594 of file vtss_ts_api.h.

9.109.2.2 port_no

```
vtss_port_no_t vtss_oam_ts_timestamp_s::port_no
```

port number

Definition at line 595 of file vtss_ts_api.h.

9.109.2.3 ts_valid

```
BOOL vtss_oam_ts_timestamp_s::ts_valid
```

Timestamp is valid (can be not valid if no timestamp is received for the requested {voe_id, voe_sq}

Definition at line 596 of file vtss_ts_api.h.

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

- vtss_api/include/vtss_ts_api.h

9.110 vtss_oam_tst_counter_t Struct Reference

TST counter.

```
#include <vtss_oam_api.h>
```

Data Fields

- u64 rx_tst
- u64 tx_tst
- u64 rx_tst_trans_id_err

9.110.1 Detailed Description

TST counter.

Definition at line 864 of file vtss_oam_api.h.

9.110.2 Field Documentation

9.110.2.1 rx_tst

```
u64 vtss_oam_tst_counter_t::rx_tst
```

Count of RX TST PDUs

Definition at line 865 of file vtss_oam_api.h.

9.110.2.2 tx_tst

```
u64 vtss_oam_tst_counter_t::tx_tst
```

Count of TX TST PDUs

Definition at line 866 of file vtss_oam_api.h.

9.110.2.3 rx_tst_trans_id_err

```
u64 vtss_oam_tst_counter_t::rx_tst_trans_id_err
```

Count of RX'd TST PDUs with invalid transaction ID

Definition at line 867 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.111 vtss_oam_voe_alloc_cfg_t Struct Reference

Extra data used by vtss_oam_voe_alloc(). Only relevant for port VOEs.

```
#include <vtss_oam_api.h>
```

Data Fields

- u32 phys_port

9.111.1 Detailed Description

Extra data used by vtss_oam_voe_alloc(). Only relevant for port VOEs.

Definition at line 290 of file vtss_oam_api.h.

9.111.2 Field Documentation

9.111.2.1 phys_port

```
u32 vtss_oam_voe_alloc_cfg_t::phys_port
```

For port-VOE: Physical port index

Definition at line 291 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.112 vtss_oam_voe_ccm_conf_t Struct Reference

CCM configuration.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- BOOL copy_to_cpu
- BOOL forward
- BOOL count_as_selected
- BOOL count_as_data
- u32 mepid
- vtss_oam_meg_id_t megid
- vtss_oam_voe_auto_seq_no_conf_t tx_seq_no_auto_upd_op
- u32 tx_seq_no
- BOOL rx_seq_no_check
- u32 rx_seq_no
- u32 rx_priority
- vtss_oam_period_t rx_period

9.112.1 Detailed Description

CCM configuration.

Definition at line 344 of file vtss_oam_api.h.

9.112.2 Field Documentation

9.112.2.1 enable

```
BOOL vtss_oam_voe_ccm_conf_t::enable
```

TRUE => process CMM in hardware

Definition at line 345 of file vtss_oam_api.h.

9.112.2.2 copy_to_cpu

```
BOOL vtss_oam_voe_ccm_conf_t::copy_to_cpu
```

TRUE => send good CCM PDUs to CPU. All or "hitme once" is determined by vtss_oam_proc_conf_t.copy_next_only

Definition at line 346 of file vtss_oam_api.h.

9.112.2.3 forward

```
BOOL vtss_oam_voe_ccm_conf_t::forward
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 347 of file vtss_oam_api.h.

9.112.2.4 count_as_selected

```
BOOL vtss_oam_voe_ccm_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 348 of file vtss_oam_api.h.

9.112.2.5 count_as_data

```
BOOL vtss_oam_voe_ccm_conf_t::count_as_data
```

TRUE => count frames as data for LM

Definition at line 349 of file vtss_oam_api.h.

9.112.2.6 mepid

```
u32 vtss_oam_voe_ccm_conf_t::mepid
```

CCM MEP ID

Definition at line 350 of file vtss_oam_api.h.

9.112.2.7 megid

```
vtss_oam_meg_id_t vtss_oam_voe_ccm_conf_t::megid
```

CCM MEG ID

Definition at line 351 of file vtss_oam_api.h.

9.112.2.8 tx_seq_no_auto_upd_op

```
vtss_oam_voe_auto_seq_no_conf_t vtss_oam_voe_ccm_conf_t::tx_seq_no_auto_upd_op
```

Type of automatic sequence number update to perform at TX

Definition at line 352 of file vtss_oam_api.h.

9.112.2.9 tx_seq_no

```
u32 vtss_oam_voe_ccm_conf_t::tx_seq_no
```

Next sequence number to use at TX

Definition at line 353 of file vtss_oam_api.h.

9.112.2.10 rx_seq_no_check

```
BOOL vtss_oam_voe_ccm_conf_t::rx_seq_no_check
```

TRUE => check incoming CCM sequence number against value in rx_seq_no

Definition at line 354 of file vtss_oam_api.h.

9.112.2.11 rx_seq_no

```
u32 vtss_oam_voe_ccm_conf_t::rx_seq_no
```

Next sequence number expected at RX

Definition at line 355 of file vtss_oam_api.h.

9.112.2.12 rx_priority

```
u32 vtss_oam_voe_ccm_conf_t::rx_priority
```

Expected CCM priority

Definition at line 356 of file vtss_oam_api.h.

9.112.2.13 rx_period

```
vtss_oam_period_t vtss_oam_voe_ccm_conf_t::rx_period
```

Expected CCM period. Also used for Loss Of Connectivity scan

Definition at line 357 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.113 vtss_oam_voe_ccm_lm_conf_t Struct Reference

CCM-LM (dual-ended LM) configuration. NOTE that this configuration extends and augments the CCM configuration in struct vtss_oam_voe_ccm_conf_t. Thus, in order to configure CCM-LM, both structs need to be filled in.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- BOOL copy_to_cpu
- BOOL forward
- BOOL count_as_selected
- vtss_oam_voe_lm_counter_conf_t count
- vtss_oam_period_t period

9.113.1 Detailed Description

CCM-LM (dual-ended LM) configuration. NOTE that this configuration extends and augments the CCM configuration in struct vtss_oam_voe_ccm_conf_t. Thus, in order to configure CCM-LM, both structs need to be filled in.

Definition at line 366 of file vtss_oam_api.h.

9.113.2 Field Documentation

9.113.2.1 enable

```
BOOL vtss_oam_voe_ccm_lm_conf_t::enable
```

TRUE => process CMM-LM in hardware (requires that CCM is HW-enabled, too)

Definition at line 367 of file vtss_oam_api.h.

9.113.2.2 copy_to_cpu

```
BOOL vtss_oam_voe_ccm_lm_conf_t::copy_to_cpu
```

TRUE => send good CCM-LM PDUs to CPU. All or "hitme once" is determined by vtss_oam_proc_conf_t.copy_next_only

Definition at line 368 of file vtss_oam_api.h.

9.113.2.3 forward

```
BOOL vtss_oam_voe_ccm_lm_conf_t::forward
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 369 of file vtss_oam_api.h.

9.113.2.4 count_as_selected

```
BOOL vtss_oam_voe_ccm_lm_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 370 of file vtss_oam_api.h.

9.113.2.5 count

`vtss_oam_voe_lm_counter_conf_t vtss_oam_voe_ccm_lm_conf_t::count`

LM counter configuration

Definition at line 371 of file vtss_oam_api.h.

9.113.2.6 period

`vtss_oam_period_t vtss_oam_voe_ccm_lm_conf_t::period`

Down-MEP only: LM injection period counter. NOTE: Only 100ms and 1sec values are valid

Definition at line 372 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.114 vtss_oam_voe_conf_t Struct Reference

Main VOE configuration structure.

```
#include <vtss_oam_api.h>
```

Data Fields

- vtss_oam_voe_type_t voe_type
- vtss_mac_t unicast_mac
- vtss_oam_mep_type_t mep_type
- BOOL svc_to_path
- u32 svc_to_path_idx_w
- u32 svc_to_path_idx_p
- u32 loop_isdx_w
- u32 loop_isdx_p
- u32 loop_portidx_p
- BOOL sdlb_enable
- u32 sdlb_idx
- vtss_oam_proc_conf_t proc
- vtss_oam_voe_generic_conf_t generic [VTSS_OAM_GENERIC_OPCODE_CFG_CNT]
- vtss_oam_voe_unknown_conf_t unknown
- vtss_oam_voe_ccm_conf_t ccm
- vtss_oam_voe_ccm_lm_conf_t ccm_lm
- vtss_oam_voe_se_lm_conf_t single_ended_lm
- vtss_oam_voe_lb_conf_t lb
- vtss_oam_voe_tst_conf_t tst
- vtss_oam_voe_dm_conf_t dm
- vtss_oam_voe_lt_conf_t lt
- vtss_oam_voe_up_mep_conf_t upmep

9.114.1 Detailed Description

Main VOE configuration structure.

Definition at line 542 of file vtss_oam_api.h.

9.114.2 Field Documentation

9.114.2.1 voe_type

`vtss_oam_voe_type_t vtss_oam_voe_conf_t::voe_type`

Read-Only: Service/Path or Port VOE type

Definition at line 543 of file vtss_oam_api.h.

9.114.2.2 unicast_mac

`vtss_mac_t vtss_oam_voe_conf_t::unicast_mac`

This VOE's unicast MAC

Definition at line 544 of file vtss_oam_api.h.

9.114.2.3 mep_type

`vtss_oam_mep_type_t vtss_oam_voe_conf_t::mep_type`

Up-MEP, Down-MEP, or MIP

Definition at line 546 of file vtss_oam_api.h.

9.114.2.4 svc_to_path

`BOOL vtss_oam_voe_conf_t::svc_to_path`

TRUE => count Service frames on Path VOE

Definition at line 548 of file vtss_oam_api.h.

9.114.2.5 svc_to_path_idx_w

```
u32 vtss_oam_voe_conf_t::svc_to_path_idx_w
```

Service VOE only: Index of associated Path VOE, working

Definition at line 549 of file vtss_oam_api.h.

9.114.2.6 svc_to_path_idx_p

```
u32 vtss_oam_voe_conf_t::svc_to_path_idx_p
```

Service VOE only: Index of associated Path VOE, protected

Definition at line 550 of file vtss_oam_api.h.

9.114.2.7 loop_isdx_w

```
u32 vtss_oam_voe_conf_t::loop_isdx_w
```

Loop (LB, DM) Service index, working

Definition at line 552 of file vtss_oam_api.h.

9.114.2.8 loop_isdx_p

```
u32 vtss_oam_voe_conf_t::loop_isdx_p
```

Loop (LB, DM) Service index, protected

Definition at line 553 of file vtss_oam_api.h.

9.114.2.9 loop_portidx_p

```
u32 vtss_oam_voe_conf_t::loop_portidx_p
```

Loop (LB, DM) port index, protected

Definition at line 554 of file vtss_oam_api.h.

9.114.2.10 sdlb_enable

```
BOOL vtss_oam_voe_conf_t::sdlb_enable
```

TRUE => Use another S-DLB (sdlb_idx) for OAM frames counted as such (not as data)

Definition at line 556 of file vtss_oam_api.h.

9.114.2.11 sdlb_idx

```
u32 vtss_oam_voe_conf_t::sdlb_idx
```

Index of S-DLB to use

Definition at line 557 of file vtss_oam_api.h.

9.114.2.12 proc

```
vtss_oam_proc_conf_t vtss_oam_voe_conf_t::proc
```

Overall PDU processing setup

Definition at line 559 of file vtss_oam_api.h.

9.114.2.13 generic

```
vtss_oam_voe_generic_conf_t vtss_oam_voe_conf_t::generic[VTSS_OAM_GENERIC_OPCODE_CFG_CNT]
```

Generic opcodes

Definition at line 562 of file vtss_oam_api.h.

9.114.2.14 unknown

```
vtss_oam_voe_unknown_conf_t vtss_oam_voe_conf_t::unknown
```

Unknown opcodes

Definition at line 563 of file vtss_oam_api.h.

9.114.2.15 ccm

`vtss_oam_voe_ccm_conf_t vtss_oam_voe_conf_t::ccm`

CCM

Definition at line 564 of file vtss_oam_api.h.

9.114.2.16 ccm_lm

`vtss_oam_voe_ccm_lm_conf_t vtss_oam_voe_conf_t::ccm_lm`

CCM-LM

Definition at line 565 of file vtss_oam_api.h.

9.114.2.17 single_ended_lm

`vtss_oam_voe_se_lm_conf_t vtss_oam_voe_conf_t::single_ended_lm`

Single-ended LM

Definition at line 566 of file vtss_oam_api.h.

9.114.2.18 lb

`vtss_oam_voe_lb_conf_t vtss_oam_voe_conf_t::lb`

LB

Definition at line 567 of file vtss_oam_api.h.

9.114.2.19 tst

`vtss_oam_voe_tst_conf_t vtss_oam_voe_conf_t::tst`

TST

Definition at line 568 of file vtss_oam_api.h.

9.114.2.20 dm

`vtss_oam_voe_dm_conf_t vtss_oam_voe_conf_t::dm`

DM

Definition at line 569 of file vtss_oam_api.h.

9.114.2.21 lt

`vtss_oam_voe_lt_conf_t vtss_oam_voe_conf_t::lt`

LT

Definition at line 570 of file vtss_oam_api.h.

9.114.2.22 upmep

`vtss_oam_voe_up_mep_conf_t vtss_oam_voe_conf_t::upmep`

Up-MEP

Definition at line 573 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.115 vtss_oam_voe_counter_t Struct Reference

Per-VOE counters.

```
#include <vtss_oam_api.h>
```

Data Fields

- vtss_oam_ccm_counter_t ccm
- vtss_oam_lm_counter_t lm
- vtss_oam_lb_counter_t lb
- vtss_oam_tst_counter_t tst
- vtss_oam_sel_counter_t sel

9.115.1 Detailed Description

Per-VOE counters.

Definition at line 871 of file vtss_oam_api.h.

9.115.2 Field Documentation

9.115.2.1 ccm

`vtss_oam_ccm_counter_t vtss_oam_voe_counter_t::ccm`

CCM counters

Definition at line 872 of file vtss_oam_api.h.

9.115.2.2 lm

`vtss_oam_lm_counter_t vtss_oam_voe_counter_t::lm`

LM counters

Definition at line 873 of file vtss_oam_api.h.

9.115.2.3 lb

`vtss_oam_lb_counter_t vtss_oam_voe_counter_t::lb`

LB counters

Definition at line 874 of file vtss_oam_api.h.

9.115.2.4 tst

`vtss_oam_tst_counter_t vtss_oam_voe_counter_t::tst`

TST counters

Definition at line 875 of file vtss_oam_api.h.

9.115.2.5 sel

```
vtss_oam_sel_counter_t vtss_oam_voe_counter_t::sel
```

Selected/non-selected counters

Definition at line 876 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.116 vtss_oam_voe_dm_conf_t Struct Reference

One- and Two-way DM (1DM / DMM) configuration.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable_dmm
- BOOL enable_1dm
- BOOL copy_1dm_to_cpu
- BOOL copy_dmm_to_cpu
- BOOL copy_dmr_to_cpu
- BOOL forward_1dm
- BOOL forward_dmm
- BOOL forward_dmr
- BOOL count_as_selected
- BOOL count_as_data

9.116.1 Detailed Description

One- and Two-way DM (1DM / DMM) configuration.

Definition at line 438 of file vtss_oam_api.h.

9.116.2 Field Documentation

9.116.2.1 enable_dmm

```
BOOL vtss_oam_voe_dm_conf_t::enable_dmm
```

TRUE => process DMM+DMR in hardware

Definition at line 439 of file vtss_oam_api.h.

9.116.2.2 enable_1dm

```
BOOL vtss_oam_voe_dm_conf_t::enable_1dm
```

TRUE => process 1DM in hardware

Definition at line 440 of file vtss_oam_api.h.

9.116.2.3 copy_1dm_to_cpu

```
BOOL vtss_oam_voe_dm_conf_t::copy_1dm_to_cpu
```

TRUE => send all 1DM PDUs to CPU

Definition at line 441 of file vtss_oam_api.h.

9.116.2.4 copy_dmm_to_cpu

```
BOOL vtss_oam_voe_dm_conf_t::copy_dmm_to_cpu
```

TRUE => send all DMM PDUs to CPU

Definition at line 442 of file vtss_oam_api.h.

9.116.2.5 copy_dmr_to_cpu

```
BOOL vtss_oam_voe_dm_conf_t::copy_dmr_to_cpu
```

TRUE => send all DMR PDUs to CPU

Definition at line 443 of file vtss_oam_api.h.

9.116.2.6 forward_1dm

```
BOOL vtss_oam_voe_dm_conf_t::forward_1dm
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 444 of file vtss_oam_api.h.

9.116.2.7 forward_dmm

```
BOOL vtss_oam_voe_dm_conf_t::forward_dmm
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 445 of file vtss_oam_api.h.

9.116.2.8 forward_dmr

```
BOOL vtss_oam_voe_dm_conf_t::forward_dmr
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 446 of file vtss_oam_api.h.

9.116.2.9 count_as_selected

```
BOOL vtss_oam_voe_dm_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 447 of file vtss_oam_api.h.

9.116.2.10 count_as_data

```
BOOL vtss_oam_voe_dm_conf_t::count_as_data
```

TRUE => count frames as data for LM

Definition at line 448 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.117 vtss_oam_voe_generic_conf_t Struct Reference

Generic OAM opcode configuration.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- BOOL copy_to_cpu
- BOOL forward
- BOOL count_as_selected
- BOOL count_as_data

9.117.1 Detailed Description

Generic OAM opcode configuration.

Note that the opcode settings configured here are aligned with those in vtss_oam_vop_conf_t.pdu_type.generic[], i.e. if, say, opcode 40 is configured in index 3 there, the corresponding VOE settings are in index 3 in vtss_oam_↩voe_conf_t.pdu_type.generic[] as well.

Definition at line 474 of file vtss_oam_api.h.

9.117.2 Field Documentation

9.117.2.1 enable

```
BOOL vtss_oam_voe_generic_conf_t::enable
```

TRUE => process opcode in hardware

Definition at line 475 of file vtss_oam_api.h.

9.117.2.2 copy_to_cpu

```
BOOL vtss_oam_voe_generic_conf_t::copy_to_cpu
```

TRUE => send all PDUs with the specified opcode to CPU

Definition at line 476 of file vtss_oam_api.h.

9.117.2.3 forward

```
BOOL vtss_oam_voe_generic_conf_t::forward
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 477 of file vtss_oam_api.h.

9.117.2.4 count_as_selected

```
BOOL vtss_oam_voe_generic_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 478 of file vtss_oam_api.h.

9.117.2.5 count_as_data

```
BOOL vtss_oam_voe_generic_conf_t::count_as_data
```

TRUE => count frames as data for LM

Definition at line 479 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.118 vtss_oam_voe_lb_conf_t Struct Reference

LB configuration. Note that the tx_... members are ignored if LB is already active. If they are to be changed, LB must be disabled and re-enabled with the new values. In other words, the values are only applied at enable-time.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- BOOL copy_lbm_to_cpu
- BOOL copy_lbr_to_cpu
- BOOL forward_lbm
- BOOL forward_lbr
- BOOL count_as_selected
- BOOL count_as_data
- BOOL tx_update_transaction_id
- u32 tx_transaction_id
- u32 rx_transaction_id

9.118.1 Detailed Description

LB configuration. Note that the tx_... members are ignored if LB is already active. If they are to be changed, LB must be disabled and re-enabled with the new values. In other words, the values are only applied at enable-time.

Definition at line 397 of file vtss_oam_api.h.

9.118.2 Field Documentation

9.118.2.1 enable

```
BOOL vtss_oam_voe_lb_conf_t::enable
```

TRUE => process LB in hardware

Definition at line 398 of file vtss_oam_api.h.

9.118.2.2 copy_lbm_to_cpu

```
BOOL vtss_oam_voe_lb_conf_t::copy_lbm_to_cpu
```

TRUE => send all LBM PDUs to CPU

Definition at line 399 of file vtss_oam_api.h.

9.118.2.3 copy_lbr_to_cpu

```
BOOL vtss_oam_voe_lb_conf_t::copy_lbr_to_cpu
```

TRUE => send all LBR PDUs to CPU

Definition at line 400 of file vtss_oam_api.h.

9.118.2.4 forward_lbm

```
BOOL vtss_oam_voe_lb_conf_t::forward_lbm
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 401 of file vtss_oam_api.h.

9.118.2.5 forward_lbr

```
BOOL vtss_oam_voe_lb_conf_t::forward_lbr
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 402 of file vtss_oam_api.h.

9.118.2.6 count_as_selected

```
BOOL vtss_oam_voe_lb_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 403 of file vtss_oam_api.h.

9.118.2.7 count_as_data

```
BOOL vtss_oam_voe_lb_conf_t::count_as_data
```

TRUE => count frames as data for LM

Definition at line 404 of file vtss_oam_api.h.

9.118.2.8 tx_update_transaction_id

```
BOOL vtss_oam_voe_lb_conf_t::tx_update_transaction_id
```

TRUE => update transaction ID on TX – unless LB is already active

Definition at line 405 of file vtss_oam_api.h.

9.118.2.9 tx_transaction_id

```
u32 vtss_oam_voe_lb_conf_t::tx_transaction_id
```

LBM Transaction ID to use on TX (if tx_update_transaction_id is TRUE) – unless LB is already active

Definition at line 406 of file vtss_oam_api.h.

9.118.2.10 rx_transaction_id

```
u32 vtss_oam_voe_lb_conf_t::rx_transaction_id
```

LBR Transaction ID - 1 to expect on RX – unless LB is already active

Definition at line 407 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.119 vtss_oam_voe_lm_counter_conf_t Struct Reference

LM counter configuration.

```
#include <vtss_oam_api.h>
```

Data Fields

- u32 priority_mask
- BOOL yellow

9.119.1 Detailed Description

LM counter configuration.

Definition at line 325 of file vtss_oam_api.h.

9.119.2 Field Documentation

9.119.2.1 priority_mask

```
u32 vtss_oam_voe_lm_counter_conf_t::priority_mask
```

Per-priority counter enable. Zero-bit means count in prio. 7

Definition at line 326 of file vtss_oam_api.h.

9.119.2.2 yellow

```
BOOL vtss_oam_voe_lm_counter_conf_t::yellow
```

TRUE => include yellow frames in count

Definition at line 327 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.120 vtss_oam_voe_lt_conf_t Struct Reference

LT configuration.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- BOOL copy_ltm_to_cpu
- BOOL copy_ltr_to_cpu
- BOOL forward_ltm
- BOOL forward_ltr
- BOOL count_as_selected
- BOOL count_as_data

9.120.1 Detailed Description

LT configuration.

Definition at line 455 of file vtss_oam_api.h.

9.120.2 Field Documentation

9.120.2.1 enable

```
BOOL vtss_oam_voe_lt_conf_t::enable
```

TRUE => process LB in hardware

Definition at line 456 of file vtss_oam_api.h.

9.120.2.2 copy_ltm_to_cpu

```
BOOL vtss_oam_voe_lt_conf_t::copy_ltm_to_cpu
```

TRUE => send all LTM PDUs to CPU

Definition at line 457 of file vtss_oam_api.h.

9.120.2.3 copy_ltr_to_cpu

```
BOOL vtss_oam_voe_lt_conf_t::copy_ltr_to_cpu
```

TRUE => send all LTR PDUs to CPU

Definition at line 458 of file vtss_oam_api.h.

9.120.2.4 forward_ltm

```
BOOL vtss_oam_voe_lt_conf_t::forward_ltm
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 459 of file vtss_oam_api.h.

9.120.2.5 forward_ltr

```
BOOL vtss_oam_voe_lt_conf_t::forward_ltr
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 460 of file vtss_oam_api.h.

9.120.2.6 count_as_selected

```
BOOL vtss_oam_voe_lt_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 461 of file vtss_oam_api.h.

9.120.2.7 count_as_data

```
BOOL vtss_oam_voe_lt_conf_t::count_as_data
```

TRUE => count frames as data for LM

Definition at line 462 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.121 vtss_oam_voe_se_lm_conf_t Struct Reference

Single-ended LM configuration.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- BOOL copy_lmm_to_cpu
- BOOL copy_lmr_to_cpu
- BOOL forward_lmm
- BOOL forward_lmr
- BOOL count_as_selected
- BOOL count_as_data
- vtss_oam_voe_lm_counter_conf_t count

9.121.1 Detailed Description

Single-ended LM configuration.

Definition at line 379 of file vtss_oam_api.h.

9.121.2 Field Documentation

9.121.2.1 enable

```
BOOL vtss_oam_voe_se_lm_conf_t::enable
```

TRUE => process LM in hardware

Definition at line 380 of file vtss_oam_api.h.

9.121.2.2 copy_lmm_to_cpu

```
BOOL vtss_oam_voe_se_lm_conf_t::copy_lmm_to_cpu
```

TRUE => send all LMM PDUs to CPU

Definition at line 381 of file vtss_oam_api.h.

9.121.2.3 copy_lmr_to_cpu

```
BOOL vtss_oam_voe_se_lm_conf_t::copy_lmr_to_cpu
```

TRUE => send all LMR PDUs to CPU

Definition at line 382 of file vtss_oam_api.h.

9.121.2.4 forward_lmm

```
BOOL vtss_oam_voe_se_lm_conf_t::forward_lmm
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 383 of file vtss_oam_api.h.

9.121.2.5 forward_lmr

```
BOOL vtss_oam_voe_se_lm_conf_t::forward_lmr
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 384 of file vtss_oam_api.h.

9.121.2.6 count_as_selected

```
BOOL vtss_oam_voe_se_lm_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 385 of file vtss_oam_api.h.

9.121.2.7 count_as_data

```
BOOL vtss_oam_voe_se_lm_conf_t::count_as_data
```

TRUE => count frames as data for LM

Definition at line 386 of file vtss_oam_api.h.

9.121.2.8 count

```
vtss_oam_voe_lm_counter_conf_t vtss_oam_voe_se_lm_conf_t::count
```

LM counter configuration

Definition at line 387 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.122 vtss_oam_voe_tst_conf_t Struct Reference

TST configuration. Note that the tx_... members are ignored if TST is already active. If they are to be changed, TST must be disabled and re-enabled with the new values. In other words, the values are only applied at enable-time.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- BOOL copy_to_cpu
- BOOL forward
- BOOL count_as_selected
- BOOL count_as_data
- BOOL tx_seq_no_auto_update
- u32 tx_seq_no
- u32 rx_seq_no

9.122.1 Detailed Description

TST configuration. Note that the tx_... members are ignored if TST is already active. If they are to be changed, TST must be disabled and re-enabled with the new values. In other words, the values are only applied at enable-time.

Definition at line 417 of file vtss_oam_api.h.

9.122.2 Field Documentation

9.122.2.1 enable

```
BOOL vtss_oam_voe_tst_conf_t::enable
```

TRUE => process TST in hardware

Definition at line 418 of file vtss_oam_api.h.

9.122.2.2 copy_to_cpu

```
BOOL vtss_oam_voe_tst_conf_t::copy_to_cpu
```

NOTE that for Serval the below fields overload fields from vtss_oam_voe_lb_conf_t A client should therefore *not* configure both TST and LB transaction simultaneously (have enable = TRUE for both).TRUE => send all TST PDUs to CPU

Definition at line 425 of file vtss_oam_api.h.

9.122.2.3 forward

```
BOOL vtss_oam_voe_tst_conf_t::forward
```

TRUE => forward all OAM frames of this type to the extraction destination

Definition at line 426 of file vtss_oam_api.h.

9.122.2.4 count_as_selected

```
BOOL vtss_oam_voe_tst_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 427 of file vtss_oam_api.h.

9.122.2.5 count_as_data

```
BOOL vtss_oam_voe_tst_conf_t::count_as_data
```

TRUE => count frames as data for LM

Definition at line 428 of file vtss_oam_api.h.

9.122.2.6 tx_seq_no_auto_update

```
BOOL vtss_oam_voe_tst_conf_t::tx_seq_no_auto_update
```

TRUE => Automatically update TX TST sequence number – unless TST is already active

Definition at line 429 of file vtss_oam_api.h.

9.122.2.7 tx_seq_no

```
u32 vtss_oam_voe_tst_conf_t::tx_seq_no
```

Sequence number to use for first TST TX – unless TST is already active

Definition at line 430 of file vtss_oam_api.h.

9.122.2.8 rx_seq_no

```
u32 vtss_oam_voe_tst_conf_t::rx_seq_no
```

Sequence number - 1 to expect – unless TST is already active

Definition at line 431 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.123 vtss_oam_voe_unknown_conf_t Struct Reference

Unknown opcode configuration. An unknown opcode is one that isn't a) explicitly supported (e.g. CCM or LTM) or b) configured as a generic opcode.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable
- BOOL copy_to_cpu
- BOOL count_as_selected
- BOOL count_as_data

9.123.1 Detailed Description

Unknown opcode configuration. An unknown opcode is one that isn't a) explicitly supported (e.g. CCM or LTM) or b) configured as a generic opcode.

Definition at line 488 of file vtss_oam_api.h.

9.123.2 Field Documentation

9.123.2.1 enable

```
BOOL vtss_oam_voe_unknown_conf_t::enable
```

TRUE => process opcode in hardware

Definition at line 489 of file vtss_oam_api.h.

9.123.2.2 copy_to_cpu

```
BOOL vtss_oam_voe_unknown_conf_t::copy_to_cpu
```

TRUE => send all PDUs with unknown opcodes to CPU

Definition at line 490 of file vtss_oam_api.h.

9.123.2.3 count_as_selected

```
BOOL vtss_oam_voe_unknown_conf_t::count_as_selected
```

TRUE => count frames as selected

Definition at line 491 of file vtss_oam_api.h.

9.123.2.4 count_as_data

```
BOOL vtss_oam_voe_unknown_conf_t::count_as_data
```

TRUE => count frames as data for LM

Definition at line 492 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.124 vtss_oam_voe_up_mep_conf_t Struct Reference

Up-MEP configuration.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL discard_rx
- BOOL loopback
- u32 port

9.124.1 Detailed Description

Up-MEP configuration.

Definition at line 499 of file vtss_oam_api.h.

9.124.2 Field Documentation

9.124.2.1 discard_rx

```
BOOL vtss_oam_voe_up_mep_conf_t::discard_rx
```

TRUE => all non-OAM RX data is discarded

Definition at line 500 of file vtss_oam_api.h.

9.124.2.2 loopback

```
BOOL vtss_oam_voe_up_mep_conf_t::loopback
```

TRUE => loop OAM mode; FALSE => in-service mode

Definition at line 501 of file vtss_oam_api.h.

9.124.2.3 port

```
u32 vtss_oam_voe_up_mep_conf_t::port
```

Front port where up-MEP is located

Definition at line 502 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.125 vtss_oam_vop_conf_t Struct Reference

VOP configuration. Once the VOP is configured, VOEs can be configured.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL enable_all_voe
- BOOL ccm_lm_enable_rx_fcf_in_reserved_field
- BOOL down_mep_lmr_proprietary_fcf_use
- vtss_mac_t common_multicast_dmac
- u32 external_cpu_portmask
- u32 sdlb_cpy_copy_idx
- vtss_oam_vop_pdu_type_conf_t pdu_type

9.125.1 Detailed Description

VOP configuration. Once the VOP is configured, VOEs can be configured.

Definition at line 206 of file vtss_oam_api.h.

9.125.2 Field Documentation

9.125.2.1 enable_all_voe

```
BOOL vtss_oam_vop_conf_t::enable_all_voe
```

Master enable/disable all VOEs

Definition at line 207 of file vtss_oam_api.h.

9.125.2.2 ccm_lm_enable_rx_fcf_in_reserved_field

```
BOOL vtss_oam_vop_conf_t::ccm_lm_enable_rx_fcf_in_reserved_field
```

TRUE => Use reserved field in RX CCM-LM PDUs for RX FCf

Definition at line 208 of file vtss_oam_api.h.

9.125.2.3 down_mep_lmr_proprietary_fcf_use

```
BOOL vtss_oam_vop_conf_t::down_mep_lmr_proprietary_fcf_use
```

TRUE => Use proprietary

Definition at line 209 of file vtss_oam_api.h.

9.125.2.4 common_multicast_dmac

```
vtss_mac_t vtss_oam_vop_conf_t::common_multicast_dmac
```

Common MC DMAC for all VOEs

Definition at line 210 of file vtss_oam_api.h.

9.125.2.5 external_cpu_portmask

```
u32 vtss_oam_vop_conf_t::external_cpu_portmask
```

Front port(s) to use for external CPU/FPGA

Definition at line 211 of file vtss_oam_api.h.

9.125.2.6 sdlb_cpy_copy_idx

```
u32 vtss_oam_vop_conf_t::sdlb_cpy_copy_idx
```

Index of S-DLB to use for OAM CPU frame copy

Definition at line 212 of file vtss_oam_api.h.

9.125.2.7 pdu_type

```
vtss_oam_vop_pdu_type_conf_t vtss_oam_vop_conf_t::pdu_type
```

Per-PDU type configuration

Definition at line 213 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.126 vtss_oam_vop_extract_conf_t Struct Reference

Configuration for CPU/frontport extraction. Configure if a particular OAM PDU type is extracted to the CPU, or to a front port.

```
#include <vtss_oam_api.h>
```

Data Fields

- BOOL to_front
- vtss_packet_rx_queue_t rx_queue

9.126.1 Detailed Description

Configuration for CPU/frontport extraction. Configure if a particular OAM PDU type is extracted to the CPU, or to a front port.

Definition at line 171 of file vtss_oam_api.h.

9.126.2 Field Documentation

9.126.2.1 to_front

```
BOOL vtss_oam_vop_extract_conf_t::to_front
```

TRUE => front port; false: CPU queue

Definition at line 172 of file vtss_oam_api.h.

9.126.2.2 rx_queue

```
vtss_packet_rx_queue_t vtss_oam_vop_extract_conf_t::rx_queue
```

Index of CPU queue or external port

Definition at line 173 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.127 vtss_oam_vop_generic_opcode_conf_t Struct Reference

Configuration for generic opcodes. Similar to vtss_oam_vop_extract_conf_t, this structure configures extraction for a generic opcode, and allows for DMAC checking for incoming OAM PDUs with that opcode value.

```
#include <vtss_oam_api.h>
```

Data Fields

- u32 opcode
- BOOL check_dmac
- vtss_oam_vop_extract_conf_t extract

9.127.1 Detailed Description

Configuration for generic opcodes. Similar to vtss_oam_vop_extract_conf_t, this structure configures extraction for a generic opcode, and allows for DMAC checking for incoming OAM PDUs with that opcode value.

Definition at line 181 of file vtss_oam_api.h.

9.127.2 Field Documentation

9.127.2.1 opcode

```
u32 vtss_oam_vop_generic_opcode_conf_t::opcode
```

Opcode

Definition at line 182 of file vtss_oam_api.h.

9.127.2.2 check_dmac

```
BOOL vtss_oam_vop_generic_opcode_conf_t::check_dmac
```

TRUE => check DMAC; FALSE => don't

Definition at line 183 of file vtss_oam_api.h.

9.127.2.3 extract

```
vtss_oam_vop_extract_conf_t vtss_oam_vop_generic_opcode_conf_t::extract
```

Extraction settings

Definition at line 184 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.128 vtss_oam_vop_pdu_type_conf_t Struct Reference

Global per-PDU type configuration.

```
#include <vtss_oam_api.h>
```

Data Fields

- vtss_oam_vop_generic_opcode_conf_t generic [VTSS_OAM_GENERIC_OPCODE_CFG_CNT]
- vtss_oam_vop_extract_conf_t ccm
- vtss_oam_vop_extract_conf_t ccm_lm
- vtss_oam_vop_extract_conf_t lt
- vtss_oam_vop_extract_conf_t dmm
- vtss_oam_vop_extract_conf_t dmr
- vtss_oam_vop_extract_conf_t lmm
- vtss_oam_vop_extract_conf_t lmr
- vtss_oam_vop_extract_conf_t lbm
- vtss_oam_vop_extract_conf_t lbr
- vtss_oam_vop_extract_conf_t err
- vtss_oam_vop_extract_conf_t other

9.128.1 Detailed Description

Global per-PDU type configuration.

Definition at line 189 of file vtss_oam_api.h.

9.128.2 Field Documentation

9.128.2.1 generic

```
vtss_oam_vop_generic_opcode_conf_t vtss_oam_vop_pdu_type_conf_t::generic[VTSS_OAM_GENERIC_OPCODE_CFG_CNT]
```

Generic opcodes

Definition at line 190 of file vtss_oam_api.h.

9.128.2.2 ccm

```
vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::ccm
```

CCM (not CCM-LM)

Definition at line 191 of file vtss_oam_api.h.

9.128.2.3 ccm_lm

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::ccm_lm`

CCM-LM

Definition at line 192 of file vtss_oam_api.h.

9.128.2.4 lt

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::lt`

LTM, LTR - common settings

Definition at line 193 of file vtss_oam_api.h.

9.128.2.5 dmm

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::dmm`

DMM

Definition at line 194 of file vtss_oam_api.h.

9.128.2.6 dmr

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::dmr`

DMR

Definition at line 195 of file vtss_oam_api.h.

9.128.2.7 lmm

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::lmm`

LMM

Definition at line 196 of file vtss_oam_api.h.

9.128.2.8 lmr

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::lmr`

LMR

Definition at line 197 of file vtss_oam_api.h.

9.128.2.9 lbm

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::lbm`

LBM

Definition at line 198 of file vtss_oam_api.h.

9.128.2.10 lbr

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::lbr`

LBR

Definition at line 199 of file vtss_oam_api.h.

9.128.2.11 err

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::err`

Errored frames

Definition at line 200 of file vtss_oam_api.h.

9.128.2.12 other

`vtss_oam_vop_extract_conf_t vtss_oam_vop_pdu_type_conf_t::other`

Other frames

Definition at line 201 of file vtss_oam_api.h.

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

- vtss_api/include/vtss_oam_api.h

9.129 vtss_os_timestamp_t Struct Reference

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

Data Fields

- unsigned int hw_cnt

9.129.1 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.

9.129.2 Field Documentation

9.129.2.1 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

9.130 vtss_packet_dma_conf_t Struct Reference

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

Data Fields

- BOOL dma_enable [VTSS_QUEUE_ARRAY_SIZE]

9.130.1 Detailed Description

The packet DMA configuration.

Definition at line 2124 of file vtss_packet_api.h.

9.130.2 Field Documentation

9.130.2.1 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

9.131 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

9.131.1 Detailed Description

Information about frame.

Definition at line 346 of file vtss_packet_api.h.

9.131.2 Field Documentation

9.131.2.1 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.

9.131.2.2 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.

9.131.2.3 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.

9.131.2.4 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.

9.131.2.5 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

9.132 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

9.132.1 Detailed Description

Packet information for each port.

Definition at line 410 of file vtss_packet_api.h.

9.132.2 Field Documentation

9.132.2.1 filter

```
vtss_packet_filter_t vtss_packet_port_filter_t::filter
```

Packet filtering

Definition at line 411 of file vtss_packet_api.h.

9.132.2.2 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

9.133 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

9.133.1 Detailed Description

Port info structure.

Definition at line 390 of file vtss_packet_api.h.

9.133.2 Field Documentation

9.133.2.1 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.

9.133.2.2 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.

9.133.2.3 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.

9.133.2.4 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

9.134 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]
- vtss_bitrate_t shaper_rate

9.134.1 Detailed Description

CPU Rx configuration.

Definition at line 121 of file vtss_packet_api.h.

9.134.2 Field Documentation

9.134.2.1 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.

9.134.2.2 reg

`vtss_packet_rx_reg_t vtss_packet_rx_conf_t::reg`

Packet registration

Definition at line 123 of file vtss_packet_api.h.

9.134.2.3 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.

9.134.2.4 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.

9.134.2.5 shaper_rate

`vtss_bitrate_t vtss_packet_rx_conf_t::shaper_rate`

CPU port shaper bitrate in kbps

Definition at line 129 of file vtss_packet_api.h.

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

- vtss_api/include/vtss_packet_api.h

9.135 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

9.135.1 Detailed Description

System frame header describing received frame.

Definition at line 216 of file vtss_packet_api.h.

9.135.2 Field Documentation

9.135.2.1 length

```
u32 vtss_packet_rx_header_t::length
```

Frame length excluding CRC

Definition at line 218 of file vtss_packet_api.h.

9.135.2.2 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.

9.135.2.3 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.

9.135.2.4 learn

```
BOOL vtss_packet_rx_header_t::learn
```

TRUE if learn frame

Definition at line 221 of file vtss_packet_api.h.

9.135.2.5 arrived_tagged

```
BOOL vtss_packet_rx_header_t::arrived_tagged
```

TRUE is frame was tagged

Definition at line 222 of file vtss_packet_api.h.

9.135.2.6 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

9.136 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

9.136.1 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.

9.136.2 Field Documentation

9.136.2.1 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.

9.136.2.2 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.

9.136.2.3 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.

9.136.2.4 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.

9.136.2.5 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 recveived 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.

9.136.2.6 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.

9.136.2.7 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.

9.136.2.8 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.

9.136.2.9 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.

9.136.2.10 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.

9.136.2.11 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.

9.136.2.12 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.

9.136.2.13 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.

9.136.2.14 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.

9.136.2.15 hw_tstamp

```
u32 vtss_packet_rx_info_t::hw_tstamp
```

The frame's ingress timestamp.

Jaguar1: Frames getting copied to the CPU for SFlow reasons can never have a valid hw_tstamp (such frames will be indicated through hw_tstamp_decoded == FALSE). Note that the hw_tstamp_decoded will indicate TRUE for all other frames, but this is only reliable if the following register are set-up: ANA_AC:PS_COMMON:MISC_CTRL.↩ OAM_RX_FRM_EXTEND_ENA == 0 ANA_AC:PS_COMMON:MISC_CTRL.OAM_RX_TSTAMP_IN_FCS_ENA == 1 ASM:CFG:ETH_CFG.ETH_PRE_MODE == 1 DEV1G/DEV25G:DEV_CFG_STATUS:DEV_PTP_CFG.PTP_ENA == 1 DEV10G:DEV_CFG_STATUS:DEV_PTP_CFG.PTP_CFG_ENA == 1 DEVCPU_GCB:PTP_CFG:PTP_MIS↩ C_CFG.PTP_ENA == 1

Serval: Two-step PTP frames have the 32-bit Rx timestamp saved in this field.

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

Definition at line 1318 of file vtss_packet_api.h.

9.136.2.16 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.

9.136.2.17 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_TY↩PE_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.

9.136.2.18 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.

9.136.2.19 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.

9.136.2.20 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.

9.136.2.21 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

9.137 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

9.137.1 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.

9.137.2 Field Documentation

9.137.2.1 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.

9.137.2.2 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.

9.137.2.3 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.

9.137.2.4 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.

9.137.2.5 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 (sflow-info)
Serval2: Y (sflow-info)
ServalT: Y (sflow-info)
```

Definition at line 851 of file vtss_packet_api.h.

9.137.2.6 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.

9.137.2.7 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

9.138 vtss_packet_rx_port_conf_t Struct Reference

Packet registration per port.

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

Data Fields

- vtss_packet_reg_type_t ipmc_ctrl_reg
- vtss_packet_reg_type_t igmp_reg
- vtss_packet_reg_type_t mld_reg
- vtss_packet_reg_type_t bpdu_reg [16]
- vtss_packet_reg_type_t garp_reg [16]

9.138.1 Detailed Description

Packet registration per port.

Definition at line 764 of file types.h.

9.138.2 Field Documentation

9.138.2.1 ipmc_ctrl_reg

```
vtss_packet_reg_type_t vtss_packet_rx_port_conf_t::ipmc_ctrl_reg
```

IP MC Control, NORMAL/FORWARD/CPU_COPY supported

Definition at line 766 of file types.h.

9.138.2.2 igmp_reg

```
vtss_packet_reg_type_t vtss_packet_rx_port_conf_t::igmp_reg
```

IGMP, NORMAL/FORWARD/CPU_ONLY supported

Definition at line 767 of file types.h.

9.138.2.3 mld_reg

```
vtss_packet_reg_type_t vtss_packet_rx_port_conf_t::mld_reg
```

MLD, NORMAL/FORWARD/CPU_ONLY supported

Definition at line 768 of file types.h.

9.138.2.4 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.

9.138.2.5 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

9.139 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

9.139.1 Detailed Description

CPU Rx queue configuration.

Definition at line 85 of file vtss_packet_api.h.

9.139.2 Field Documentation

9.139.2.1 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.

9.139.2.2 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

9.140 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

9.140.1 Detailed Description

CPU Rx queue map.

Definition at line 103 of file vtss_packet_api.h.

9.140.2 Field Documentation

9.140.2.1 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.

9.140.2.2 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.

9.140.2.3 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.

9.140.2.4 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.

9.140.2.5 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.

9.140.2.6 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.

9.140.2.7 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.

9.140.2.8 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.

9.140.2.9 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

9.141 vtss_packet_rx_queue_npi_conf_t Struct Reference

CPU Rx queue NPI configuration.

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

Data Fields

- BOOL enable

9.141.1 Detailed Description

CPU Rx queue NPI configuration.

Definition at line 75 of file vtss_packet_api.h.

9.141.2 Field Documentation

9.141.2.1 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

9.142 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

9.142.1 Detailed Description

CPU Rx packet registration.

Definition at line 93 of file vtss_packet_api.h.

9.142.2 Field Documentation

9.142.2.1 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.

9.142.2.2 garp_cpu_only

```
BOOL vtss_packet_rx_reg_t::garp_cpu_only[16]
```

Redirect GARPs (DMAC 01-80-C2-00-00-2X)

Definition at line 96 of file vtss_packet_api.h.

9.142.2.3 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.

9.142.2.4 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.

9.142.2.5 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

9.143 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]

9.143.1 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.

9.143.2 Field Documentation

9.143.2.1 length

```
u32 vtss_packet_tx_ifh_t::length
```

Length of compiled IFH (in bytes)

Definition at line 2073 of file vtss_packet_api.h.

9.143.2.2 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

9.144 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
- vtss_afi_id_t afi_id

9.144.1 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.

9.144.2 Field Documentation

9.144.2.1 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.

9.144.2.2 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.

9.144.2.3 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.

9.144.2.4 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.

9.144.2.5 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.

9.144.2.6 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.

9.144.2.7 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.

9.144.2.8 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.

9.144.2.9 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.

9.144.2.10 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.

9.144.2.11 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.

9.144.2.12 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.

9.144.2.13 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.

9.144.2.14 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.

9.144.2.15 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.

9.144.2.16 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.

9.144.2.17 afi_id

```
vtss_afi_id_t vtss_packet_tx_info_t::afi_id
```

Automatic Frame Injector ID.

This field has two different meanings depending on architecture. Serval: The ID of a previously allocated AFI resource.

Set to VTSS_AFI_ID_NONE if you don't want a frame injected periodically.

If using the FDMA to initiate periodic transmission of frames frames, the value of this field is not not used for anything. The FDMA driver has its own Tx properties that define whether to periodically inject a frame.

Jaguar2/Serval2/ServalT: This field is really not an ID, but a kind of boolean that indicates that the frame should be captured by the switch core's AFI block upon injection. When set to VTSS_AFI_ID_NONE, the frame will not be captured by the switch core, otherwise it will. Use the vtss_afi_fast_inj_frm_hijack() or vtss_afi_slow_inj_frm_↩hijack() functions after successful injection to further configure the frame.

Validity: A F

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

Definition at line 1974 of file vtss_packet_api.h.

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

- vtss_api/include/vtss_packet_api.h

9.145 vtss_phy_10g_fifo_sync_t Struct Reference

10G OOS workaround options

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

Data Fields

- BOOL bypass_in_api
- BOOL skip_rev_check
- vtss_debug_printf_t pr

9.145.1 Detailed Description

10G OOS workaround options

Definition at line 2090 of file vtss_phy_ts_api.h.

9.145.2 Field Documentation

9.145.2.1 bypass_in_api

```
BOOL vtss_phy_10g_fifo_sync_t::bypass_in_api
```

clear bypass in API

Definition at line 2091 of file vtss_phy_ts_api.h.

9.145.2.2 skip_rev_check

```
BOOL vtss_phy_10g_fifo_sync_t::skip_rev_check
```

To force execution irrespective of revision

Definition at line 2092 of file vtss_phy_ts_api.h.

9.145.2.3 pr

```
vtss_debug_printf_t vtss_phy_10g_fifo_sync_t::pr
```

Pass print function to get the algorithm execution logs

Definition at line 2093 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.146 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

9.146.1 Detailed Description

PHY auto negotiation advertisement.

Definition at line 309 of file vtss_phy_api.h.

9.146.2 Field Documentation

9.146.2.1 speed_10m_hdx

```
BOOL vtss_phy_aneg_t::speed_10m_hdx
```

10Mbps, half duplex

Definition at line 310 of file vtss_phy_api.h.

9.146.2.2 speed_10m_fdx

```
BOOL vtss_phy_aneg_t::speed_10m_fdx
```

10Mbps, full duplex

Definition at line 311 of file vtss_phy_api.h.

9.146.2.3 speed_100m_hdx

```
BOOL vtss_phy_aneg_t::speed_100m_hdx
```

100Mbps, half duplex

Definition at line 312 of file vtss_phy_api.h.

9.146.2.4 speed_100m_fdx

```
BOOL vtss_phy_aneg_t::speed_100m_fdx
```

100Mbps, full duplex

Definition at line 313 of file vtss_phy_api.h.

9.146.2.5 speed_1g_fdx

```
BOOL vtss_phy_aneg_t::speed_1g_fdx
```

1000Mpbs, full duplex

Definition at line 314 of file vtss_phy_api.h.

9.146.2.6 speed_1g_hdx

```
BOOL vtss_phy_aneg_t::speed_1g_hdx
```

1000Mpbs, full duplex

Definition at line 315 of file vtss_phy_api.h.

9.146.2.7 symmetric_pause

```
BOOL vtss_phy_aneg_t::symmetric_pause
```

Symmetric pause

Definition at line 316 of file vtss_phy_api.h.

9.146.2.8 asymmetric_pause

```
BOOL vtss_phy_aneg_t::asymmetric_pause
```

Asymmetric pause

Definition at line 317 of file vtss_phy_api.h.

9.146.2.9 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

9.147 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

9.147.1 Detailed Description

PHY clock configuration.

Definition at line 648 of file vtss_phy_api.h.

9.147.2 Field Documentation

9.147.2.1 src

```
vtss_phy_clk_source_t vtss_phy_clock_conf_t::src
```

Clock source

Definition at line 649 of file vtss_phy_api.h.

9.147.2.2 freq

```
vtss_phy_freq_t vtss_phy_clock_conf_t::freq
```

Clock requency

Definition at line 650 of file vtss_phy_api.h.

9.147.2.3 squelch

```
vtss_phy_clk_squelch vtss_phy_clock_conf_t::squelch
```

Clock squelsh 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

9.148 vtss_phy_conf_1g_t Struct Reference

PHY 1G configuration.

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

Data Fields

- struct {
  BOOL cfg
  BOOL val
  } master

9.148.1 Detailed Description

PHY 1G configuration.

Definition at line 504 of file vtss_phy_api.h.

9.148.2 Field Documentation

9.148.2.1 cfg

```
BOOL vtss_phy_conf_1g_t::cfg
```

Manual Master/Slave Config. 1=enabled

Definition at line 506 of file vtss_phy_api.h.

9.148.2.2 val

```
BOOL vtss_phy_conf_1g_t::val
```

Master/Slave Config value, 1=Master

Definition at line 507 of file vtss_phy_api.h.

9.148.2.3 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

9.149 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_serd_pcs_cntl_t mac_if_pcs
- vtss_phy_media_serd_pcs_cntl_t media_if_pcs
- vtss_phy_media_force_ams_sel_t force_ams_sel
- BOOL skip_coma

9.149.1 Detailed Description

PHY configuration.

Definition at line 395 of file vtss_phy_api.h.

9.149.2 Field Documentation

9.149.2.1 mode

```
vtss_phy_mode_t vtss_phy_conf_t::mode
```

PHY mode

Definition at line 396 of file vtss_phy_api.h.

9.149.2.2 forced

`vtss_phy_forced_t vtss_phy_conf_t::forced`

Forced mode configuration

Definition at line 397 of file vtss_phy_api.h.

9.149.2.3 aneg

`vtss_phy_aneg_t vtss_phy_conf_t::aneg`

Auto-negotiation mode configuration

Definition at line 398 of file vtss_phy_api.h.

9.149.2.4 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.

9.149.2.5 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.

9.149.2.6 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.

9.149.2.7 unidir

```
vtss_phy_unidirectional_t vtss_phy_conf_t::unidir
```

Unidirectional Configuration

Definition at line 402 of file vtss_phy_api.h.

9.149.2.8 mac_if_pcs

```
vtss_phy_mac_serd_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.

9.149.2.9 media_if_pcs

```
vtss_phy_media_serd_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.

9.149.2.10 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.

9.149.2.11 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

9.150 vtss_phy_daisy_chain_conf_t Struct Reference

SPI daisy chain configuration.

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

Data Fields

- BOOL spi_daisy_input
- BOOL spi_daisy_output

9.150.1 Detailed Description

SPI daisy chain configuration.

Definition at line 535 of file vtss_phy_ts_api.h.

9.150.2 Field Documentation

9.150.2.1 spi_daisy_input

```
BOOL vtss_phy_daisy_chain_conf_t::spi_daisy_input
```

Enable SPI daisy-chain input port

Definition at line 536 of file vtss_phy_ts_api.h.

9.150.2.2 spi_daisy_output

```
BOOL vtss_phy_daisy_chain_conf_t::spi_daisy_output
```

Enable SPI daisy-chain output port

Definition at line 537 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.151 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

9.151.1 Detailed Description

EEE configuration.

Definition at line 987 of file vtss_phy_api.h.

9.151.2 Field Documentation

9.151.2.1 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.

9.151.2.2 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

9.152 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

9.152.1 Detailed Description

enhanced LED control

Definition at line 1010 of file vtss_phy_api.h.

9.152.2 Field Documentation

9.152.2.1 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.

9.152.2.2 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.

9.152.2.3 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.

9.152.2.4 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

9.153 vtss_phy_forced_t Struct Reference

PHY forced mode configuration.

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

Data Fields

- vtss_port_speed_t speed
- BOOL fdx

9.153.1 Detailed Description

PHY forced mode configuration.

Definition at line 303 of file vtss_phy_api.h.

9.153.2 Field Documentation

9.153.2.1 speed

```
vtss_port_speed_t vtss_phy_forced_t::speed
```

Speed

Definition at line 304 of file vtss_phy_api.h.

9.153.2.2 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

9.154 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

9.154.1 Detailed Description

LED model selection.

Definition at line 136 of file vtss_phy_api.h.

9.154.2 Field Documentation

9.154.2.1 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.

9.154.2.2 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

9.155 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

9.155.1 Detailed Description

1G Phy loopbacks

Definition at line 1384 of file vtss_phy_api.h.

9.155.2 Field Documentation

9.155.2.1 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.

9.155.2.2 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.

9.155.2.3 connector_enable

```
BOOL vtss_phy_loopback_t::connector_enable
```

Enable/Disable loopback connector loopback

Definition at line 1387 of file vtss_phy_api.h.

9.155.2.4 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.

9.155.2.5 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.

9.155.2.6 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.

9.155.2.7 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.

9.155.2.8 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.

9.155.2.9 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

9.156 vtss_phy_ltc_freq_synth_s Struct Reference

Frequency systhesis pulse configuration.

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

Data Fields

- BOOL enable
- u8 high_duty_cycle
- u8 low_duty_cycle

9.156.1 Detailed Description

Frequency systhesis pulse configuration.

Definition at line 501 of file vtss_phy_ts_api.h.

9.156.2 Field Documentation

9.156.2.1 enable

```
BOOL vtss_phy_ltc_freq_synth_s::enable
```

Enable/Disable frequency synthesis pulse

Definition at line 502 of file vtss_phy_ts_api.h.

9.156.2.2 high_duty_cycle

```
u8 vtss_phy_ltc_freq_synth_s::high_duty_cycle
```

Number of clock cycles pulse is high

Definition at line 503 of file vtss_phy_ts_api.h.

9.156.2.3 low_duty_cycle

```
u8 vtss_phy_ltc_freq_synth_s::low_duty_cycle
```

Number of clock cycles pulse is low

Definition at line 504 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.157 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

9.157.1 Detailed Description

PHY MAC SerDes PCS Control, Reg16E3.

Definition at line 351 of file vtss_phy_api.h.

9.157.2 Field Documentation

9.157.2.1 disable

```
BOOL vtss_phy_mac_serd_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.

9.157.2.2 restart

```
BOOL vtss_phy_mac_serd_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.

9.157.2.3 pd_enable

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::pd_enable
```

MAC i/f ANEG parallel detect enable

Definition at line 354 of file vtss_phy_api.h.

9.157.2.4 aneg_restart

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::aneg_restart
```

Restart MAC i/f ANEG

Definition at line 355 of file vtss_phy_api.h.

9.157.2.5 force_adv_ability

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::force_adv_ability
```

Force adv. ability from Reg18E3

Definition at line 356 of file vtss_phy_api.h.

9.157.2.6 sgmii_in_pre

```
vtss_phy_mac_serd_pcs_sgmii_pre vtss_phy_mac_serd_pcs_cntl_t::sgmii_in_pre
```

SGMII Input Preamble for 100BaseFX

Definition at line 357 of file vtss_phy_api.h.

9.157.2.7 sgmii_out_pre

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::sgmii_out_pre
```

SGMII Output Preamble

Definition at line 358 of file vtss_phy_api.h.

9.157.2.8 serdes_aneg_ena

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::serdes_aneg_ena
```

MAC SerDes ANEG Enable

Definition at line 359 of file vtss_phy_api.h.

9.157.2.9 serdes_pol_inv_in

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::serdes_pol_inv_in
```

Invert SerDes Polarity at input of MAC

Definition at line 360 of file vtss_phy_api.h.

9.157.2.10 serdes_pol_inv_out

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::serdes_pol_inv_out
```

Invert SerDes Polarity at output of MAC

Definition at line 361 of file vtss_phy_api.h.

9.157.2.11 fast_link_stat_ena

```
BOOL vtss_phy_mac_serd_pcs_cntl_t::fast_link_stat_ena
```

Fast Link Fail Status Enable

Definition at line 362 of file vtss_phy_api.h.

9.157.2.12 inhibit_odd_start

```
BOOL vtss_phy_mac_serd_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

9.158 vtss_phy_media_serd_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

9.158.1 Detailed Description

PHY MEDIA SerDes PCS Control, Reg23E3.

Definition at line 375 of file vtss_phy_api.h.

9.158.2 Field Documentation

9.158.2.1 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.

9.158.2.2 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.

9.158.2.3 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.

9.158.2.4 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.

9.158.2.5 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.

9.158.2.6 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.

9.158.2.7 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.

9.158.2.8 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.

9.158.2.9 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

9.159 vtss_phy_power_conf_t Struct Reference

PHY power configuration.

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

Data Fields

- vtss_phy_power_mode_t mode

9.159.1 Detailed Description

PHY power configuration.

Definition at line 562 of file vtss_phy_api.h.

9.159.2 Field Documentation

9.159.2.1 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

9.160 vtss_phy_power_status_t Struct Reference

PHY power status.

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

Data Fields

- u32 level

9.160.1 Detailed Description

PHY power status.

Definition at line 597 of file vtss_phy_api.h.

9.160.2 Field Documentation

9.160.2.1 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

9.161 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

9.161.1 Detailed Description

PHY reset structure.

Definition at line 218 of file vtss_phy_api.h.

9.161.2 Field Documentation

9.161.2.1 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.

9.161.2.2 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.

9.161.2.3 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.

9.161.2.4 tbi

```
vtss_phy_tbi_conf_t vtss_phy_reset_conf_t::tbi
```

TBI setup

Definition at line 222 of file vtss_phy_api.h.

9.161.2.5 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.

9.161.2.6 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.

9.161.2.7 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

9.162 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

9.162.1 Detailed Description

PHY RGMII configuration.

Definition at line 194 of file vtss_phy_api.h.

9.162.2 Field Documentation

9.162.2.1 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.

9.162.2.2 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

9.163 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

9.163.1 Detailed Description

Phy statistic information.

Definition at line 1348 of file vtss_phy_api.h.

9.163.2 Field Documentation

9.163.2.1 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.

9.163.2.2 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.

9.163.2.3 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.

9.163.2.4 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.

9.163.2.5 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.

9.163.2.6 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.

9.163.2.7 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

9.164 vtss_phy_status_1g_t Struct Reference

PHY 1G status.

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

Data Fields

- BOOL master_cfg_fault
- BOOL master

9.164.1 Detailed Description

PHY 1G status.

Definition at line 538 of file vtss_phy_api.h.

9.164.2 Field Documentation

9.164.2.1 master_cfg_fault

```
BOOL vtss_phy_status_1g_t::master_cfg_fault
```

Master/Slave Configuration fault

Definition at line 539 of file vtss_phy_api.h.

9.164.2.2 master

```
BOOL vtss_phy_status_1g_t::master
```

Master = 1, Slave = 0

Definition at line 540 of file vtss_phy_api.h.

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

- vtss_api/include/vtss_phy_api.h

9.165 vtss_phy_tbi_conf_t Struct Reference

PHY TBI configuration.

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

Data Fields

- BOOL aneg_enable

9.165.1 Detailed Description

PHY TBI configuration.

Definition at line 200 of file vtss_phy_api.h.

9.165.2 Field Documentation

9.165.2.1 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

9.166 vtss_phy_timestamp_t Struct Reference

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

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

Data Fields

- struct {
  u16 high
  u32 low
  } seconds

- u32 nanoseconds

9.166.1 Detailed Description

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

Definition at line 250 of file vtss_phy_ts_api.h.

9.166.2 Field Documentation

9.166.2.1 high

```
u16 vtss_phy_timestamp_t::high
```

bits 32-47 of 48-bit second

Definition at line 252 of file vtss_phy_ts_api.h.

9.166.2.2 low

```
u32 vtss_phy_timestamp_t::low
```

bits 0-31 of 48-bit second

Definition at line 253 of file vtss_phy_ts_api.h.

9.166.2.3 seconds

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

6 bytes second part of Timestamp

9.166.2.4 nanoseconds

```
u32 vtss_phy_timestamp_t::nanoseconds
```

4 bytes nano-sec part of Timestamp

Definition at line 255 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.167 vtss_phy_ts_ach_conf_t Struct Reference

Analyzer ACH comparator configuration options.

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

Data Fields

- struct {
  struct {
    u8 value
    u8 mask
  } version
  struct {
    u16 value
    u16 mask
  } channel_type
  struct {
    u16 value
    u16 mask
  } proto_id
 } comm_opt

9.167.1 Detailed Description

Analyzer ACH comparator configuration options.

Note

ACH uses the IP1 comparator for match. So IP1 and ACH can not be used at the same time.

Definition at line 1099 of file vtss_phy_ts_api.h.

9.167.2 Field Documentation

9.167.2.1 value [1/2]

```
u8 vtss_phy_ts_ach_conf_t::value
```

4-bits version

Definition at line 1102 of file vtss_phy_ts_api.h.

9.167.2.2 mask [1/2]

```
u8 vtss_phy_ts_ach_conf_t::mask
```

Mask

Definition at line 1103 of file vtss_phy_ts_api.h.

9.167.2.3 version

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

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

9.167.2.4 value [2/2]

```
u16 vtss_phy_ts_ach_conf_t::value
```

Channel type value

Protocol Identifier Value

Definition at line 1106 of file vtss_phy_ts_api.h.

9.167.2.5 mask [2/2]

```
u16 vtss_phy_ts_ach_conf_t::mask
```

Channel type mask

Protocl Identifier Mask

Definition at line 1107 of file vtss_phy_ts_api.h.

9.167.2.6 channel_type

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

PW Associated Channel Type in value/mask format

9.167.2.7 proto_id

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

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

9.167.2.8 comm_opt

```
struct { ... } vtss_phy_ts_ach_conf_t::comm_opt
```

ACH common config

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

- vtss_api/include/vtss_phy_ts_api.h

9.168 vtss_phy_ts_alt_clock_mode_s Struct Reference

parameter for setting the alternative clock mode.

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

Data Fields

- BOOL pps_ls_lpbk
- BOOL ls_lpbk
- BOOL ls_pps_lpbk

9.168.1 Detailed Description

parameter for setting the alternative clock mode.

external clock output configuration.

Definition at line 52 of file vtss_phy_ts_api.h.

9.168.2 Field Documentation

9.168.2.1 pps_ls_lpbk

```
BOOL vtss_phy_ts_alt_clock_mode_s::pps_ls_lpbk
```

output PPS is loopback to L/S input pin

Definition at line 53 of file vtss_phy_ts_api.h.

9.168.2.2 ls_lpbk

```
BOOL vtss_phy_ts_alt_clock_mode_s::ls_lpbk
```

L/S act as output pin at 1PPS

Definition at line 54 of file vtss_phy_ts_api.h.

9.168.2.3 ls_pps_lpbk

```
BOOL vtss_phy_ts_alt_clock_mode_s::ls_pps_lpbk
```

L/S connected to PPS out

Definition at line 55 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.169 vtss_phy_ts_eng_init_conf_t Struct Reference

Defines the basic engine parameters passed to the engine_init_conf_get() function.

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

Data Fields

- BOOL eng_used
- vtss_phy_ts_encap_t encap_type
- vtss_phy_ts_engine_flow_match_t flow_match_mode
- u8 flow_st_index
- u8 flow_end_index

9.169.1 Detailed Description

Defines the basic engine parameters passed to the engine_init_conf_get() function.

Definition at line 848 of file vtss_phy_ts_api.h.

9.169.2 Field Documentation

9.169.2.1 eng_used

```
BOOL vtss_phy_ts_eng_init_conf_t::eng_used
```

allocated the engine to application

Definition at line 849 of file vtss_phy_ts_api.h.

9.169.2.2 encap_type

```
vtss_phy_ts_encap_t vtss_phy_ts_eng_init_conf_t::encap_type
```

engine encapsulation

Definition at line 850 of file vtss_phy_ts_api.h.

9.169.2.3 flow_match_mode

```
vtss_phy_ts_engine_flow_match_t vtss_phy_ts_eng_init_conf_t::flow_match_mode
```

strict/non-strict flow match

Definition at line 851 of file vtss_phy_ts_api.h.

9.169.2.4 flow_st_index

```
u8 vtss_phy_ts_eng_init_conf_t::flow_st_index
```

start index of flow

Definition at line 852 of file vtss_phy_ts_api.h.

9.169.2.5 flow_end_index

```
u8 vtss_phy_ts_eng_init_conf_t::flow_end_index
```

end index of flow

Definition at line 853 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.170 vtss_phy_ts_engine_action_t Struct Reference

Engine Action configuration options.

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

Data Fields

- BOOL action_ptp
- BOOL action_gen
- union {
  vtss_phy_ts_ptp_engine_action_t ptp_conf [2]
  vtss_phy_ts_oam_engine_action_t oam_conf [6]
  vtss_phy_ts_generic_action_t gen_conf [6]
  } action

9.170.1 Detailed Description

Engine Action configuration options.

Note

Number of PTP actions in a engine depends on clock mode and delay method, like TC1Step and P2P, it requires 3 ingress rules to be added into PTP flows. There are total 6 flows in PTP and OAM comparator. For each frame type that needs to be timestamped requires one rule (i.e 1 flow). So application should decide the number of actions accordingly. For OAM only 2DM needs 2 flows, others needs 1 flow. The number of PTP/OAM actions config here is the maximum number, application should decide how many are valid for a engine based on clock mode and delay method.

Definition at line 1421 of file vtss_phy_ts_api.h.

9.170.2 Field Documentation

9.170.2.1 action_ptp

```
BOOL vtss_phy_ts_engine_action_t::action_ptp
```

is the action for PTP or OAM

Definition at line 1422 of file vtss_phy_ts_api.h.

9.170.2.2 action_gen

```
BOOL vtss_phy_ts_engine_action_t::action_gen
```

generic action or not

Definition at line 1423 of file vtss_phy_ts_api.h.

9.170.2.3 ptp_conf

```
vtss_phy_ts_ptp_engine_action_t vtss_phy_ts_engine_action_t::ptp_conf[2]
```

Max 2 PTP action per engine

Definition at line 1425 of file vtss_phy_ts_api.h.

9.170.2.4 oam_conf

```
vtss_phy_ts_oam_engine_action_t vtss_phy_ts_engine_action_t::oam_conf[6]
```

Max 6 OAM action per engine

Definition at line 1426 of file vtss_phy_ts_api.h.

9.170.2.5 gen_conf

```
vtss_phy_ts_generic_action_t vtss_phy_ts_engine_action_t::gen_conf[6]
```

Max 6 Generic action per engine

Definition at line 1427 of file vtss_phy_ts_api.h.

9.170.2.6 action

```
union { ...  } vtss_phy_ts_engine_action_t::action
```

PTP/OAM action config

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

- vtss_api/include/vtss_phy_ts_api.h

9.171 vtss_phy_ts_engine_flow_conf_t Struct Reference

Analyzer flow configuration options.

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

Data Fields

- BOOL eng_mode
- vtss_phy_ts_engine_channel_map_t channel_map [8]
- union {
  vtss_phy_ts_ptp_engine_flow_conf_t ptp
  vtss_phy_ts_oam_engine_flow_conf_t oam
  vtss_phy_ts_generic_flow_conf_t gen
  } flow_conf

9.171.1 Detailed Description

Analyzer flow configuration options.

Note

Engine configuration will be parsed to know PTP or OAM flow based on encapsulation type provided during engine allocation.

Definition at line 1178 of file vtss_phy_ts_api.h.

9.171.2 Field Documentation

9.171.2.1 eng_mode

```
BOOL vtss_phy_ts_engine_flow_conf_t::eng_mode
```

engine enable/disable

Definition at line 1179 of file vtss_phy_ts_api.h.

9.171.2.2 channel_map

```
vtss_phy_ts_engine_channel_map_t vtss_phy_ts_engine_flow_conf_t::channel_map[8]
```

maps flows to channel for multi-channel timestamp block. flow_map can be set per comparator in HW

Definition at line 1180 of file vtss_phy_ts_api.h.

9.171.2.3 ptp

`vtss_phy_ts_ptp_engine_flow_conf_t vtss_phy_ts_engine_flow_conf_t::ptp`

PTP engine configuration

Definition at line 1183 of file vtss_phy_ts_api.h.

9.171.2.4 oam

`vtss_phy_ts_oam_engine_flow_conf_t vtss_phy_ts_engine_flow_conf_t::oam`

OAM engine configuration

Definition at line 1184 of file vtss_phy_ts_api.h.

9.171.2.5 gen

`vtss_phy_ts_generic_flow_conf_t vtss_phy_ts_engine_flow_conf_t::gen`

Generic match configuration

Definition at line 1185 of file vtss_phy_ts_api.h.

9.171.2.6 flow_conf

`union { ... } vtss_phy_ts_engine_flow_conf_t::flow_conf`

PTP/OAM flow config

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

- vtss_api/include/vtss_phy_ts_api.h

9.172 vtss_phy_ts_eth_conf_t Struct Reference

Analyzer Ethernet comparator configuration options.

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

Data Fields

- struct {
  - BOOL pbb_en
  - u16 etype
  - u16 tpid

  } comm_opt

- struct {
  - BOOL flow_en
  - u8 addr_match_mode
  - u8 addr_match_select
  - u8 mac_addr [6]
  - BOOL vlan_check
  - u8 num_tag
  - u8 outer_tag_type
  - u8 inner_tag_type
  - u8 tag_range_mode
  - union {
    - struct {
      - u16 upper
      - u16 lower

      } range
    - struct {
      - u16 val
      - u16 mask

      } value

    } outer_tag
  - union {
    - struct {
      - u16 upper
      - u16 lower

      } range
    - struct {
      - u16 val
      - u16 mask

      } value
    - struct {
      - u32 val
      - u32 mask

      } i_tag

    } inner_tag

  } flow_opt [8]

9.172.1 Detailed Description

Analyzer Ethernet comparator configuration options.

Note

Common options apply all the flows within the comparator. Also there are per-flow configuration.

Definition at line 955 of file vtss_phy_ts_api.h.

9.172.2 Field Documentation

9.172.2.1 pbb_en

```
BOOL vtss_phy_ts_eth_conf_t::pbb_en
```

PBB tag present, not applicable for Eth2 comparator

Definition at line 957 of file vtss_phy_ts_api.h.

9.172.2.2 etype

```
u16 vtss_phy_ts_eth_conf_t::etype
```

The value of Ether type to be checked if Ethertype/length field is an Ethertype

Definition at line 958 of file vtss_phy_ts_api.h.

9.172.2.3 tpid

```
u16 vtss_phy_ts_eth_conf_t::tpid
```

VLAN TPID for S or B-tag

Definition at line 959 of file vtss_phy_ts_api.h.

9.172.2.4 comm_opt

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

Ethernet common config

9.172.2.5 flow_en

```
BOOL vtss_phy_ts_eth_conf_t::flow_en
```

flow enable/disable

Definition at line 963 of file vtss_phy_ts_api.h.

9.172.2.6 addr_match_mode

```
u8 vtss_phy_ts_eth_conf_t::addr_match_mode
```

Multiple match can be possible using OR

Definition at line 968 of file vtss_phy_ts_api.h.

9.172.2.7 addr_match_select

```
u8 vtss_phy_ts_eth_conf_t::addr_match_select
```

src or dest addr to be matched

Definition at line 972 of file vtss_phy_ts_api.h.

9.172.2.8 mac_addr

```
u8 vtss_phy_ts_eth_conf_t::mac_addr[6]
```

addr to be matched, src or dest

Definition at line 973 of file vtss_phy_ts_api.h.

9.172.2.9 vlan_check

```
BOOL vtss_phy_ts_eth_conf_t::vlan_check
```

TRUE=>verify configured VLAN tag configuration, FALSE=>parse VLAN tag if any, but don't check, for PBB I-tag is always checked

Definition at line 975 of file vtss_phy_ts_api.h.

9.172.2.10 num_tag

```
u8 vtss_phy_ts_eth_conf_t::num_tag
```

No of Tags (max 2 tag), for PBB at least I-tag should be present

Definition at line 976 of file vtss_phy_ts_api.h.

9.172.2.11 outer_tag_type

`u8 vtss_phy_ts_eth_conf_t::outer_tag_type`

for PBB enabled with 2-tag, this must be B-tag

Definition at line 981 of file vtss_phy_ts_api.h.

9.172.2.12 inner_tag_type

`u8 vtss_phy_ts_eth_conf_t::inner_tag_type`

for PBB this must be I-tag; also for single tag inner_tag is used

Definition at line 982 of file vtss_phy_ts_api.h.

9.172.2.13 tag_range_mode

`u8 vtss_phy_ts_eth_conf_t::tag_range_mode`

for PBB no range check is allowed

Definition at line 986 of file vtss_phy_ts_api.h.

9.172.2.14 upper

`u16 vtss_phy_ts_eth_conf_t::upper`

Upper value for outer tag range

Upper value for inner tag range

Definition at line 989 of file vtss_phy_ts_api.h.

9.172.2.15 lower

`u16 vtss_phy_ts_eth_conf_t::lower`

Lower value for outer tag range

Loower value for inner tag range

Definition at line 990 of file vtss_phy_ts_api.h.

9.172.2.16 range [1/2]

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

tag in range

9.172.2.17 val [1/2]

```
u16 vtss_phy_ts_eth_conf_t::val
```

Value

Definition at line 993 of file vtss_phy_ts_api.h.

9.172.2.18 mask [1/2]

```
u16 vtss_phy_ts_eth_conf_t::mask
```

Mask

Definition at line 994 of file vtss_phy_ts_api.h.

9.172.2.19 value [1/2]

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

tag in value/mask

9.172.2.20 outer_tag

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

Outer tag

9.172.2.21 range [2/2]

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

tag in range

9.172.2.22 value [2/2]

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

tag in value/mask

9.172.2.23 val [2/2]

```
u32 vtss_phy_ts_eth_conf_t::val
```

24-bit I-tag value

Definition at line 1007 of file vtss_phy_ts_api.h.

9.172.2.24 mask [2/2]

```
u32 vtss_phy_ts_eth_conf_t::mask
```

24-bit I-tag mask

Definition at line 1008 of file vtss_phy_ts_api.h.

9.172.2.25 i_tag

```
struct { ... } vtss_phy_ts_eth_conf_t::i_tag
```

I-tag in value/mask. This is applicable for PBB i.e. Eth1 comparator

9.172.2.26 inner_tag

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

Inner Tag

9.172.2.27 flow_opt

```
struct { ... } vtss_phy_ts_eth_conf_t::flow_opt[8]
```

Ethernet per flow config

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

- vtss_api/include/vtss_phy_ts_api.h

9.173 vtss_phy_ts_fifo_conf_t Struct Reference

Defines the params for FIFO SYNC function.

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

Data Fields

- BOOL detect_only
- vtss_phy_ts_engine_t eng_recov
- vtss_phy_ts_engine_t eng_minE
- BOOL skip_rev_check

9.173.1 Detailed Description

Defines the params for FIFO SYNC function.

Definition at line 2141 of file vtss_phy_ts_api.h.

9.173.2 Field Documentation

9.173.2.1 detect_only

```
BOOL vtss_phy_ts_fifo_conf_t::detect_only
```

TS FIFO OOS Detect only, no recovery, Only for Tesla

Definition at line 2142 of file vtss_phy_ts_api.h.

9.173.2.2 eng_recov

```
vtss_phy_ts_engine_t vtss_phy_ts_fifo_conf_t::eng_recov
```

Main Engine used for recovery, Only for Tesla

Definition at line 2143 of file vtss_phy_ts_api.h.

9.173.2.3 eng_minE

```
vtss_phy_ts_engine_t vtss_phy_ts_fifo_conf_t::eng_minE
```

Mini-E Engine used for recovery, Only for Tesla

Definition at line 2144 of file vtss_phy_ts_api.h.

9.173.2.4 skip_rev_check

```
BOOL vtss_phy_ts_fifo_conf_t::skip_rev_check
```

To force execution, regardless of revision

Definition at line 2145 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.174 vtss_phy_ts_fifo_sig_t Struct Reference

Tx TSFIFO entry signature.

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

Data Fields

- vtss_phy_ts_fifo_sig_mask_t sig_mask
- u8 msg_type
- u8 domain_num
- u8 src_port_identity [10]
- u16 sequence_id
- u32 dest_ip
- u32 src_ip
- u8 dest_mac [6]

9.174.1 Detailed Description

Tx TSFIFO entry signature.

Definition at line 669 of file vtss_phy_ts_api.h.

9.174.2 Field Documentation

9.174.2.1 sig_mask

```
vtss_phy_ts_fifo_sig_mask_t vtss_phy_ts_fifo_sig_t::sig_mask
```

valid signature fields

Definition at line 670 of file vtss_phy_ts_api.h.

9.174.2.2 msg_type

```
u8 vtss_phy_ts_fifo_sig_t::msg_type
```

PTP message type

Definition at line 671 of file vtss_phy_ts_api.h.

9.174.2.3 domain_num

```
u8 vtss_phy_ts_fifo_sig_t::domain_num
```

domain number in PTP message

Definition at line 672 of file vtss_phy_ts_api.h.

9.174.2.4 src_port_identity

```
u8 vtss_phy_ts_fifo_sig_t::src_port_identity[10]
```

source port identity in PTP message

Definition at line 673 of file vtss_phy_ts_api.h.

9.174.2.5 sequence_id

```
u16 vtss_phy_ts_fifo_sig_t::sequence_id
```

PTP message sequence ID

Definition at line 674 of file vtss_phy_ts_api.h.

9.174.2.6 dest_ip

```
u32 vtss_phy_ts_fifo_sig_t::dest_ip
```

Destination IP

Definition at line 675 of file vtss_phy_ts_api.h.

9.174.2.7 src_ip

```
u32 vtss_phy_ts_fifo_sig_t::src_ip
```

Source IP

Definition at line 676 of file vtss_phy_ts_api.h.

9.174.2.8 dest_mac

```
u8 vtss_phy_ts_fifo_sig_t::dest_mac[6]
```

Destination MAC

Definition at line 677 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.175 vtss_phy_ts_gen_conf_t Struct Reference

Analyzer Generic data configuration options using IP comparator.

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

Data Fields

- struct {
  u8 flow_offset
  u8 next_prot_offset
  } comm_opt

- struct {
  BOOL flow_en
  u32 data [4]
  u32 mask [4]
  } flow_opt [8]

9.175.1 Detailed Description

Analyzer Generic data configuration options using IP comparator.

Definition at line 1122 of file vtss_phy_ts_api.h.

9.175.2 Field Documentation

9.175.2.1 flow_offset

```
u8 vtss_phy_ts_gen_conf_t::flow_offset
```

Offset of data pattern to match with current comparator

Definition at line 1124 of file vtss_phy_ts_api.h.

9.175.2.2 next_prot_offset

```
u8 vtss_phy_ts_gen_conf_t::next_prot_offset
```

Offset of data pattern to match with next comparator

Definition at line 1125 of file vtss_phy_ts_api.h.

9.175.2.3 comm_opt

```
struct { ... } vtss_phy_ts_gen_conf_t::comm_opt
```

Generic Matching common configuration

9.175.2.4 flow_en

```
BOOL vtss_phy_ts_gen_conf_t::flow_en
```

Enable the flow

Definition at line 1129 of file vtss_phy_ts_api.h.

9.175.2.5 data

```
u32 vtss_phy_ts_gen_conf_t::data[4]
```

Data byte pattern to match

Definition at line 1130 of file vtss_phy_ts_api.h.

9.175.2.6 mask

```
u32 vtss_phy_ts_gen_conf_t::mask[4]
```

Mask of the matching pattern

Definition at line 1131 of file vtss_phy_ts_api.h.

9.175.2.7 flow_opt

```
struct { ... } vtss_phy_ts_gen_conf_t::flow_opt[8]
```

Generic matching config per flow

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

- vtss_api/include/vtss_phy_ts_api.h

9.176 vtss_phy_ts_generic_action_t Struct Reference

Generic Action configuration option.

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

Data Fields

- BOOL enable
- vtss_phy_ts_engine_channel_map_t channel_map
- u8 flow_id
- u32 data [2]
- u32 mask [2]
- vtss_phy_ts_action_format ts_type
- u32 ts_offset

9.176.1 Detailed Description

Generic Action configuration option.

Definition at line 1400 of file vtss_phy_ts_api.h.

9.176.2 Field Documentation

9.176.2.1 enable

```
BOOL vtss_phy_ts_generic_action_t::enable
```

Generic action active/enable or not

Definition at line 1401 of file vtss_phy_ts_api.h.

9.176.2.2 channel_map

```
vtss_phy_ts_engine_channel_map_t vtss_phy_ts_generic_action_t::channel_map
```

maps action to channel for multi-channel timestamp block

Definition at line 1402 of file vtss_phy_ts_api.h.

9.176.2.3 flow_id

```
u8 vtss_phy_ts_generic_action_t::flow_id
```

Flow id to be associated with this action

Definition at line 1403 of file vtss_phy_ts_api.h.

9.176.2.4 data

```
u32 vtss_phy_ts_generic_action_t::data[2]
```

Matching data pattern

Definition at line 1404 of file vtss_phy_ts_api.h.

9.176.2.5 mask

```
u32 vtss_phy_ts_generic_action_t::mask[2]
```

Mask for the matching pattern

Definition at line 1405 of file vtss_phy_ts_api.h.

9.176.2.6 ts_type

```
vtss_phy_ts_action_format vtss_phy_ts_generic_action_t::ts_type
```

Timestamp type 4-byte or 10 byte timestamp

Definition at line 1406 of file vtss_phy_ts_api.h.

9.176.2.7 ts_offset

```
u32 vtss_phy_ts_generic_action_t::ts_offset
```

Timestamp offset

Definition at line 1407 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.177 vtss_phy_ts_generic_flow_conf_t Struct Reference

Generic engine flow configuration options.

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

Data Fields

- vtss_phy_ts_eth_conf_t eth1_opt
- vtss_phy_ts_gen_conf_t gen_opt

9.177.1 Detailed Description

Generic engine flow configuration options.

Definition at line 1168 of file vtss_phy_ts_api.h.

9.177.2 Field Documentation

9.177.2.1 eth1_opt

```
vtss_phy_ts_eth_conf_t vtss_phy_ts_generic_flow_conf_t::eth1_opt
```

Eth-1 comparator

Definition at line 1169 of file vtss_phy_ts_api.h.

9.177.2.2 gen_opt

```
vtss_phy_ts_gen_conf_t vtss_phy_ts_generic_flow_conf_t::gen_opt
```

Generic : It uses IP1 comparator

Definition at line 1170 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.178 vtss_phy_ts_ietf_mpls_ach_oam_conf_t Struct Reference

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

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

Data Fields

- vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t delaym_type
- vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t ts_format
- u8 ds

9.178.1 Detailed Description

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

Definition at line 1368 of file vtss_phy_ts_api.h.

9.178.2 Field Documentation

9.178.2.1 delaym_type

`vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t vtss_phy_ts_ietf_mpls_ach_oam_conf_t::delaym_type`

OAM delay measurement method

Definition at line 1369 of file vtss_phy_ts_api.h.

9.178.2.2 ts_format

`vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t vtss_phy_ts_ietf_mpls_ach_oam_conf_t::ts_format`

OAM DM Timestamp format

Definition at line 1370 of file vtss_phy_ts_api.h.

9.178.2.3 ds

`u8 vtss_phy_ts_ietf_mpls_ach_oam_conf_t::ds`

DSCP value, that corresponds to a traffic class being measured.

Definition at line 1371 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.179 vtss_phy_ts_init_conf_t Struct Reference

Defines the initial parameters to be passed to init function.

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

Data Fields

- vtss_phy_ts_clockfreq_t clk_freq
- vtss_phy_ts_clock_src_t clk_src
- vtss_phy_ts_rxtimestamp_pos_t rx_ts_pos
- vtss_phy_ts_rxtimestamp_len_t rx_ts_len
- vtss_phy_ts_fifo_mode_t tx_fifo_mode
- vtss_phy_ts_fifo_timestamp_len_t tx_ts_len
- BOOL tx_fifo_spi_conf
- u8 tx_fifo_hi_clk_cycs
- u8 tx_fifo_lo_clk_cycs
- vtss_phy_ts_tc_op_mode_t tc_op_mode
- BOOL auto_clear_ls
- BOOL macsec_ena
- BOOL chk_ing_modified
- BOOL one_step_txfifo

9.179.1 Detailed Description

Defines the initial parameters to be passed to init function.

Definition at line 1743 of file vtss_phy_ts_api.h.

9.179.2 Field Documentation

9.179.2.1 clk_freq

`vtss_phy_ts_clockfreq_t vtss_phy_ts_init_conf_t::clk_freq`

reference clock frequency

Definition at line 1744 of file vtss_phy_ts_api.h.

9.179.2.2 clk_src

`vtss_phy_ts_clock_src_t vtss_phy_ts_init_conf_t::clk_src`

clock source

Definition at line 1745 of file vtss_phy_ts_api.h.

9.179.2.3 rx_ts_pos

`vtss_phy_ts_rxtimestamp_pos_t vtss_phy_ts_init_conf_t::rx_ts_pos`

Rx timestamp position

Definition at line 1746 of file vtss_phy_ts_api.h.

9.179.2.4 rx_ts_len

`vtss_phy_ts_rxtimestamp_len_t vtss_phy_ts_init_conf_t::rx_ts_len`

Rx timestamp length

Definition at line 1747 of file vtss_phy_ts_api.h.

9.179.2.5 tx_fifo_mode

```
vtss_phy_ts_fifo_mode_t vtss_phy_ts_init_conf_t::tx_fifo_mode
```

Tx TSFIFO access mode

Definition at line 1748 of file vtss_phy_ts_api.h.

9.179.2.6 tx_ts_len

```
vtss_phy_ts_fifo_timestamp_len_t vtss_phy_ts_init_conf_t::tx_ts_len
```

timestamp size in Tx TSFIFO

Definition at line 1749 of file vtss_phy_ts_api.h.

9.179.2.7 tx_fifo_spi_conf

```
BOOL vtss_phy_ts_init_conf_t::tx_fifo_spi_conf
```

Modify default 1588_spi configuration, applicable only on PHYs with SPI timestamp fifo support

Definition at line 1750 of file vtss_phy_ts_api.h.

9.179.2.8 tx_fifo_hi_clk_cycs

```
u8 vtss_phy_ts_init_conf_t::tx_fifo_hi_clk_cycs
```

Number of clock periods that the spi_clk is high

Definition at line 1751 of file vtss_phy_ts_api.h.

9.179.2.9 tx_fifo_lo_clk_cycs

```
u8 vtss_phy_ts_init_conf_t::tx_fifo_lo_clk_cycs
```

Number of clock periods that the spi_clk is low

Definition at line 1752 of file vtss_phy_ts_api.h.

9.179.2.10 tc_op_mode

```
vtss_phy_ts_tc_op_mode_t vtss_phy_ts_init_conf_t::tc_op_mode
```

TC operating mode

Definition at line 1759 of file vtss_phy_ts_api.h.

9.179.2.11 auto_clear_ls

```
BOOL vtss_phy_ts_init_conf_t::auto_clear_ls
```

Load and Save of LTC are auto cleared

Definition at line 1760 of file vtss_phy_ts_api.h.

9.179.2.12 macsec_ena

```
BOOL vtss_phy_ts_init_conf_t::macsec_ena
```

MACsec is enabled or disabled

Definition at line 1761 of file vtss_phy_ts_api.h.

9.179.2.13 chk_ing_modified

```
BOOL vtss_phy_ts_init_conf_t::chk_ing_modified
```

True if the flag bit needs to be modified in ingress and thus in egress

Definition at line 1762 of file vtss_phy_ts_api.h.

9.179.2.14 one_step_txfifo

```
BOOL vtss_phy_ts_init_conf_t::one_step_txfifo
```

used when transmitting Delay_Req in one step mode. FALSE when correctionfield update is used instead

Definition at line 1763 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.180 vtss_phy_ts_ip_conf_t Struct Reference

Analyzer IP comparator configuration options.

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

Data Fields

- struct {
  - u8 ip_mode
  - u16 sport_val
  - u16 sport_mask
  - u16 dport_val
  - u16 dport_mask

  } comm_opt

- struct {
  - BOOL flow_en
  - u8 match_mode
  - union {
    - struct {
      - u32 addr
      - u32 mask

      } ipv4
    - struct {
      - u32 addr [4]
      - u32 mask [4]

      } ipv6

    } ip_addr

  } flow_opt [8]

9.180.1 Detailed Description

Analyzer IP comparator configuration options.

Note

Common options apply all the flows within the comparator. Also there are per flow configuration.

Definition at line 1019 of file vtss_phy_ts_api.h.

9.180.2 Field Documentation

9.180.2.1 ip_mode

```
u8 vtss_phy_ts_ip_conf_t::ip_mode
```

IPv4, IPv6 if next protocol is not UDP, next UDP fields are not used

Definition at line 1023 of file vtss_phy_ts_api.h.

9.180.2.2 sport_val

```
u16 vtss_phy_ts_ip_conf_t::sport_val
```

UDP source port value

Definition at line 1025 of file vtss_phy_ts_api.h.

9.180.2.3 sport_mask

```
u16 vtss_phy_ts_ip_conf_t::sport_mask
```

UDP source port mask

Definition at line 1026 of file vtss_phy_ts_api.h.

9.180.2.4 dport_val

```
u16 vtss_phy_ts_ip_conf_t::dport_val
```

UDP dest port value

Definition at line 1027 of file vtss_phy_ts_api.h.

9.180.2.5 dport_mask

```
u16 vtss_phy_ts_ip_conf_t::dport_mask
```

UDP dest port mask

Definition at line 1028 of file vtss_phy_ts_api.h.

9.180.2.6 comm_opt

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

IP common config

9.180.2.7 flow_en

```
BOOL vtss_phy_ts_ip_conf_t::flow_en
```

flow enable/disable

Definition at line 1032 of file vtss_phy_ts_api.h.

9.180.2.8 match_mode

```
u8 vtss_phy_ts_ip_conf_t::match_mode
```

match src, dest or either IP address

Definition at line 1036 of file vtss_phy_ts_api.h.

9.180.2.9 addr

```
u32 vtss_phy_ts_ip_conf_t::addr[4]
```

IPv4 address

IPv6 Address

Definition at line 1039 of file vtss_phy_ts_api.h.

9.180.2.10 mask

```
u32 vtss_phy_ts_ip_conf_t::mask[4]
```

IPv4 address mask

IPv6 Mask

Definition at line 1040 of file vtss_phy_ts_api.h.

9.180.2.11 ipv4

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

IPv4 Address

9.180.2.12 ipv6

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

IPv6 Mask

9.180.2.13 ip_addr

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

IPv4/IPv6 address to be matched

9.180.2.14 flow_opt

```
struct { ... } vtss_phy_ts_ip_conf_t::flow_opt[8]
```

IP per flow config

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

- vtss_api/include/vtss_phy_ts_api.h

9.181 vtss_phy_ts_mpls_conf_t Struct Reference

Analyzer MPLS comparator configuration options.

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

Data Fields

- struct {
  BOOL cw_en
  } comm_opt

- struct {
  BOOL flow_en
  u8 stack_depth
  u8 stack_ref_point
  union {
    struct {
      vtss_phy_ts_mpls_lvl_rng_t top
      vtss_phy_ts_mpls_lvl_rng_t frst_lvl_after_top
      vtss_phy_ts_mpls_lvl_rng_t snd_lvl_after_top
      vtss_phy_ts_mpls_lvl_rng_t thrd_lvl_after_top
    } top_down
    struct {
      vtss_phy_ts_mpls_lvl_rng_t end
      vtss_phy_ts_mpls_lvl_rng_t frst_lvl_before_end
      vtss_phy_ts_mpls_lvl_rng_t snd_lvl_before_end
      vtss_phy_ts_mpls_lvl_rng_t thrd_lvl_before_end
    } bottom_up
  } stack_level
  } flow_opt [8]

9.181.1 Detailed Description

Analyzer MPLS comparator configuration options.

Definition at line 1061 of file vtss_phy_ts_api.h.

9.181.2 Field Documentation

9.181.2.1 cw_en

```
BOOL vtss_phy_ts_mpls_conf_t::cw_en
```

flow uses psudowire control word or not

Definition at line 1063 of file vtss_phy_ts_api.h.

9.181.2.2 comm_opt

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

MPLS common config

9.181.2.3 flow_en

```
BOOL vtss_phy_ts_mpls_conf_t::flow_en
```

flow enable/disable

Definition at line 1066 of file vtss_phy_ts_api.h.

9.181.2.4 stack_depth

```
u8 vtss_phy_ts_mpls_conf_t::stack_depth
```

depth of MPLS level; multiple depth match can be possible using OR

Definition at line 1072 of file vtss_phy_ts_api.h.

9.181.2.5 stack_ref_point

```
u8 vtss_phy_ts_mpls_conf_t::stack_ref_point
```

Search direction for label matching: top to bottom or bottom to top

Definition at line 1076 of file vtss_phy_ts_api.h.

9.181.2.6 top

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::top
```

Top level

Definition at line 1079 of file vtss_phy_ts_api.h.

9.181.2.7 frst_lvl_after_top

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::frst_lvl_after_top
```

First label after the top label

Definition at line 1080 of file vtss_phy_ts_api.h.

9.181.2.8 snd_lvl_after_top

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::snd_lvl_after_top
```

Second label after the top label

Definition at line 1081 of file vtss_phy_ts_api.h.

9.181.2.9 thrd_lvl_after_top

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::thrd_lvl_after_top
```

Third label after the top label

Definition at line 1082 of file vtss_phy_ts_api.h.

9.181.2.10 top_down

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

Top down configuration

9.181.2.11 end

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::end
```

End level

Definition at line 1085 of file vtss_phy_ts_api.h.

9.181.2.12 frst_lvl_before_end

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::frst_lvl_before_end
```

First label before the end label

Definition at line 1086 of file vtss_phy_ts_api.h.

9.181.2.13 snd_lvl_before_end

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::snd_lvl_before_end
```

Second label before the end label

Definition at line 1087 of file vtss_phy_ts_api.h.

9.181.2.14 thrd_lvl_before_end

```
vtss_phy_ts_mpls_lvl_rng_t vtss_phy_ts_mpls_conf_t::thrd_lvl_before_end
```

Third label before the end label

Definition at line 1088 of file vtss_phy_ts_api.h.

9.181.2.15 bottom_up

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

Bottom up configuration

9.181.2.16 stack_level

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

4 level values; top_down or bottom_up depends on stack_ref_point

9.181.2.17 flow_opt

```
struct { ... } vtss_phy_ts_mpls_conf_t::flow_opt[8]
```

MPLS per flow config

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

- vtss_api/include/vtss_phy_ts_api.h

9.182 vtss_phy_ts_mpls_lvl_rng_t Struct Reference

MPLS level range.

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

Data Fields

- u32 lower
- u32 upper

9.182.1 Detailed Description

MPLS level range.

Definition at line 1053 of file vtss_phy_ts_api.h.

9.182.2 Field Documentation

9.182.2.1 lower

```
u32 vtss_phy_ts_mpls_lvl_rng_t::lower
```

lower range value

Definition at line 1054 of file vtss_phy_ts_api.h.

9.182.2.2 upper

```
u32 vtss_phy_ts_mpls_lvl_rng_t::upper
```

upper range value

Definition at line 1055 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.183 vtss_phy_ts_nphase_status_t Struct Reference

n-phase status

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

Data Fields

- BOOL enable
- BOOL CALIB_ERR
- BOOL CALIB_DONE

9.183.1 Detailed Description

n-phase status

Definition at line 1885 of file vtss_phy_ts_api.h.

9.183.2 Field Documentation

9.183.2.1 enable

```
BOOL vtss_phy_ts_nphase_status_t::enable
```

Enabled status

Definition at line 1886 of file vtss_phy_ts_api.h.

9.183.2.2 CALIB_ERR

```
BOOL vtss_phy_ts_nphase_status_t::CALIB_ERR
```

Calibration error

Definition at line 1887 of file vtss_phy_ts_api.h.

9.183.2.3 CALIB_DONE

```
BOOL vtss_phy_ts_nphase_status_t::CALIB_DONE
```

Calibration done

Definition at line 1888 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.184 vtss_phy_ts_oam_engine_action_t Struct Reference

OAM Action configuration options.

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

Data Fields

- BOOL enable
- BOOL y1731_en
- vtss_phy_ts_engine_channel_map_t channel_map
- u8 version
- union {
  vtss_phy_ts_y1731_oam_conf_t y1731_oam_conf
  vtss_phy_ts_ietf_mpls_ach_oam_conf_t ietf_oam_conf
  } oam_conf

9.184.1 Detailed Description

OAM Action configuration options.

Note

Timestamp action will be based on OAM delay measurement method.

Definition at line 1378 of file vtss_phy_ts_api.h.

9.184.2 Field Documentation

9.184.2.1 enable

`BOOL vtss_phy_ts_oam_engine_action_t::enable`

OAM action active/enable or not

Definition at line 1379 of file vtss_phy_ts_api.h.

9.184.2.2 y1731_en

`BOOL vtss_phy_ts_oam_engine_action_t::y1731_en`

Y.1731 Message Format Enabled/Disable

Definition at line 1380 of file vtss_phy_ts_api.h.

9.184.2.3 channel_map

`vtss_phy_ts_engine_channel_map_t vtss_phy_ts_oam_engine_action_t::channel_map`

maps action to channel for multi-channel timestamp block

Definition at line 1381 of file vtss_phy_ts_api.h.

9.184.2.4 version

`u8 vtss_phy_ts_oam_engine_action_t::version`

Protocol Version; only 0 is supported

Definition at line 1382 of file vtss_phy_ts_api.h.

9.184.2.5 y1731_oam_conf

`vtss_phy_ts_y1731_oam_conf_t vtss_phy_ts_oam_engine_action_t::y1731_oam_conf`

Y.1731 OAM configuration

Definition at line 1384 of file vtss_phy_ts_api.h.

9.184.2.6 ietf_oam_conf

```
vtss_phy_ts_ietf_mpls_ach_oam_conf_t vtss_phy_ts_oam_engine_action_t::ietf_oam_conf
```

IETF OAM configuration

Definition at line 1385 of file vtss_phy_ts_api.h.

9.184.2.7 oam_conf

```
union { ... } vtss_phy_ts_oam_engine_action_t::oam_conf
```

OAM action config

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

- vtss_api/include/vtss_phy_ts_api.h

9.185 vtss_phy_ts_oam_engine_flow_conf_t Struct Reference

OAM engine flow configuration options.

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

Data Fields

- vtss_phy_ts_eth_conf_t eth1_opt
- vtss_phy_ts_eth_conf_t eth2_opt
- vtss_phy_ts_mpls_conf_t mpls_opt
- vtss_phy_ts_ach_conf_t ach_opt

9.185.1 Detailed Description

OAM engine flow configuration options.

Definition at line 1158 of file vtss_phy_ts_api.h.

9.185.2 Field Documentation

9.185.2.1 eth1_opt

```
vtss_phy_ts_eth_conf_t vtss_phy_ts_oam_engine_flow_conf_t::eth1_opt
```

Eth-1 comparator

Definition at line 1159 of file vtss_phy_ts_api.h.

9.185.2.2 eth2_opt

```
vtss_phy_ts_eth_conf_t vtss_phy_ts_oam_engine_flow_conf_t::eth2_opt
```

Eth-2 comparator; for single ETH encap, Eth-1 is used

Definition at line 1160 of file vtss_phy_ts_api.h.

9.185.2.3 mpls_opt

```
vtss_phy_ts_mpls_conf_t vtss_phy_ts_oam_engine_flow_conf_t::mpls_opt
```

MPLS comparator

Definition at line 1161 of file vtss_phy_ts_api.h.

9.185.2.4 ach_opt

```
vtss_phy_ts_ach_conf_t vtss_phy_ts_oam_engine_flow_conf_t::ach_opt
```

ACH: it uses the IP1 comparator, so IP1 and ACH can not enabled simultaneously

Definition at line 1162 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.186 vtss_phy_ts_pps_config_s Struct Reference

PPS Configuration.

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

Data Fields

- u32 pps_width_adj
- u32 pps_offset
- u32 pps_output_enable

9.186.1 Detailed Description

PPS Configuration.

Definition at line 102 of file vtss_phy_ts_api.h.

9.186.2 Field Documentation

9.186.2.1 pps_width_adj

```
u32 vtss_phy_ts_pps_config_s::pps_width_adj
```

The value of nano second counter upto which 1PPS is held high

Definition at line 103 of file vtss_phy_ts_api.h.

9.186.2.2 pps_offset

```
u32 vtss_phy_ts_pps_config_s::pps_offset
```

PPS pulse offset in nano seconds

Definition at line 104 of file vtss_phy_ts_api.h.

9.186.2.3 pps_output_enable

```
u32 vtss_phy_ts_pps_config_s::pps_output_enable
```

PPS pulse output is enabled for this port

Definition at line 105 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.187 vtss_phy_ts_ptp_conf_t Struct Reference

Analyzer PTP comparator configuration options.

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

Data Fields

- BOOL range_en
- union {
  - struct {
    - u8 val
    - u8 mask
  - } value
  - struct {
    - u8 upper
    - u8 lower
  - } range
- } domain

9.187.1 Detailed Description

Analyzer PTP comparator configuration options.

Definition at line 1268 of file vtss_phy_ts_api.h.

9.187.2 Field Documentation

9.187.2.1 range_en

```
BOOL vtss_phy_ts_ptp_conf_t::range_en
```

PTP domain number in range enable/disable

Definition at line 1269 of file vtss_phy_ts_api.h.

9.187.2.2 val

```
u8 vtss_phy_ts_ptp_conf_t::val
```

PTP domain number value

Definition at line 1272 of file vtss_phy_ts_api.h.

9.187.2.3 mask

```
u8 vtss_phy_ts_ptp_conf_t::mask
```

PTP domain number mask

Definition at line 1273 of file vtss_phy_ts_api.h.

9.187.2.4 value

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

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

9.187.2.5 upper

```
u8 vtss_phy_ts_ptp_conf_t::upper
```

Ranger upper value

Definition at line 1276 of file vtss_phy_ts_api.h.

9.187.2.6 lower

```
u8 vtss_phy_ts_ptp_conf_t::lower
```

Range lower value

Definition at line 1277 of file vtss_phy_ts_api.h.

9.187.2.7 range

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

PTP domain range configuration

9.187.2.8 domain

```
union { ... } vtss_phy_ts_ptp_conf_t::domain
```

PTP domain number configuration

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

- vtss_api/include/vtss_phy_ts_api.h

9.188 vtss_phy_ts_ptp_engine_action_t Struct Reference

Analyzer PTP action configuration options.

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

Data Fields

- BOOL enable
- vtss_phy_ts_engine_channel_map_t channel_map
- vtss_phy_ts_ptp_conf_t ptp_conf
- vtss_phy_ts_ptp_clock_mode_t clk_mode
- vtss_phy_ts_ptp_delaym_type_t delaym_type
- BOOL cf_update

9.188.1 Detailed Description

Analyzer PTP action configuration options.

Note

Timestamp action will be based on clock type and delay measurement method.

Definition at line 1310 of file vtss_phy_ts_api.h.

9.188.2 Field Documentation

9.188.2.1 enable

```
BOOL vtss_phy_ts_ptp_engine_action_t::enable
```

PTP action active/enable or not

Definition at line 1311 of file vtss_phy_ts_api.h.

9.188.2.2 channel_map

```
vtss_phy_ts_engine_channel_map_t vtss_phy_ts_ptp_engine_action_t::channel_map
```

maps action to channel for multi-channel timestamp block

Definition at line 1312 of file vtss_phy_ts_api.h.

9.188.2.3 ptp_conf

`vtss_phy_ts_ptp_conf_t vtss_phy_ts_ptp_engine_action_t::ptp_conf`

PTP configuration

Definition at line 1313 of file vtss_phy_ts_api.h.

9.188.2.4 clk_mode

`vtss_phy_ts_ptp_clock_mode_t vtss_phy_ts_ptp_engine_action_t::clk_mode`

clock mode: bc1step, bc2step, tc1step or tc2step

Definition at line 1314 of file vtss_phy_ts_api.h.

9.188.2.5 delaym_type

`vtss_phy_ts_ptp_delaym_type_t vtss_phy_ts_ptp_engine_action_t::delaym_type`

delay measurement method: P2P, E2E

Definition at line 1315 of file vtss_phy_ts_api.h.

9.188.2.6 cf_update

`BOOL vtss_phy_ts_ptp_engine_action_t::cf_update`

correction field update for bc1step

Definition at line 1316 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.189 vtss_phy_ts_ptp_engine_flow_conf_t Struct Reference

PTP engine flow configuration options.

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

Data Fields

- vtss_phy_ts_eth_conf_t eth1_opt
- vtss_phy_ts_eth_conf_t eth2_opt
- vtss_phy_ts_ip_conf_t ip1_opt
- vtss_phy_ts_ip_conf_t ip2_opt
- vtss_phy_ts_mpls_conf_t mpls_opt
- vtss_phy_ts_ach_conf_t ach_opt

9.189.1 Detailed Description

PTP engine flow configuration options.

Definition at line 1146 of file vtss_phy_ts_api.h.

9.189.2 Field Documentation

9.189.2.1 eth1_opt

`vtss_phy_ts_eth_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::eth1_opt`

Eth-1 comparator

Definition at line 1147 of file vtss_phy_ts_api.h.

9.189.2.2 eth2_opt

`vtss_phy_ts_eth_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::eth2_opt`

Eth-2 comparator; for single ETH encap, Eth-1 is used

Definition at line 1148 of file vtss_phy_ts_api.h.

9.189.2.3 ip1_opt

`vtss_phy_ts_ip_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::ip1_opt`

IP-1 comparator

Definition at line 1149 of file vtss_phy_ts_api.h.

9.189.2.4 ip2_opt

`vtss_phy_ts_ip_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::ip2_opt`

IP-2 comparator; for single IP encap, IP-1 is used

Definition at line 1150 of file vtss_phy_ts_api.h.

9.189.2.5 mpls_opt

`vtss_phy_ts_mpls_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::mpls_opt`

MPLS comparator

Definition at line 1151 of file vtss_phy_ts_api.h.

9.189.2.6 ach_opt

`vtss_phy_ts_ach_conf_t vtss_phy_ts_ptp_engine_flow_conf_t::ach_opt`

ACH: it uses the IP1 comparator, so IP1 and ACH can not enabled simultaneously

Definition at line 1152 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.190 vtss_phy_ts_sertod_conf_t Struct Reference

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

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

Data Fields

- BOOL ip_enable
- BOOL op_enable
- BOOL ls_inv

9.190.1 Detailed Description

PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)

Definition at line 330 of file vtss_phy_ts_api.h.

9.190.2 Field Documentation

9.190.2.1 ip_enable

```
BOOL vtss_phy_ts_sertod_conf_t::ip_enable
```

Serial ToD Input Enable

Definition at line 331 of file vtss_phy_ts_api.h.

9.190.2.2 op_enable

```
BOOL vtss_phy_ts_sertod_conf_t::op_enable
```

Serial ToD Output Enable

Definition at line 332 of file vtss_phy_ts_api.h.

9.190.2.3 ls_inv

```
BOOL vtss_phy_ts_sertod_conf_t::ls_inv
```

Invert the polarity of Load Save

Definition at line 333 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.191 vtss_phy_ts_stats_t Struct Reference

Timestamping Statistics.

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

Data Fields

- u32 ingr_pream_shrink_err
- u32 egr_pream_shrink_err
- u32 ingr_fcs_err
- u32 egr_fcs_err
- u32 ingr_frm_mod_cnt
- u32 egr_frm_mod_cnt
- u32 ts_fifo_tx_cnt
- u32 ts_fifo_drop_cnt

9.191.1 Detailed Description

Timestamping Statistics.

Note

Use vtss_phy_ts_stats_get() to retrieve current statistics.

Definition at line 1574 of file vtss_phy_ts_api.h.

9.191.2 Field Documentation

9.191.2.1 ingr_pream_shrink_err

```
u32 vtss_phy_ts_stats_t::ingr_pream_shrink_err
```

Frames with preambles too short to shrink

Definition at line 1575 of file vtss_phy_ts_api.h.

9.191.2.2 egr_pream_shrink_err

```
u32 vtss_phy_ts_stats_t::egr_pream_shrink_err
```

Frames with preambles too short to shrink

Definition at line 1576 of file vtss_phy_ts_api.h.

9.191.2.3 ingr_fcs_err

```
u32 vtss_phy_ts_stats_t::ingr_fcs_err
```

Timestamp block received frame with FCS error in ingress

Definition at line 1577 of file vtss_phy_ts_api.h.

9.191.2.4 egr_fcs_err

```
u32 vtss_phy_ts_stats_t::egr_fcs_err
```

Timestamp block received frame with FCS error in egress

Definition at line 1578 of file vtss_phy_ts_api.h.

9.191.2.5 ingr_frm_mod_cnt

```
u32 vtss_phy_ts_stats_t::ingr_frm_mod_cnt
```

No of frames modified by timestamp block (rewritter) in ingress

Definition at line 1579 of file vtss_phy_ts_api.h.

9.191.2.6 egr_frm_mod_cnt

```
u32 vtss_phy_ts_stats_t::egr_frm_mod_cnt
```

No of frames modified by timestamp block (rewritter) in egress

Definition at line 1580 of file vtss_phy_ts_api.h.

9.191.2.7 ts_fifo_tx_cnt

```
u32 vtss_phy_ts_stats_t::ts_fifo_tx_cnt
```

the number of timestamps transmitted to the interface

Definition at line 1581 of file vtss_phy_ts_api.h.

9.191.2.8 ts_fifo_drop_cnt

```
u32 vtss_phy_ts_stats_t::ts_fifo_drop_cnt
```

Count of dropped Timestamps not enqueued to the Tx TSFIFO

Definition at line 1582 of file vtss_phy_ts_api.h.

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

- vtss_api/include/vtss_phy_ts_api.h

9.192 vtss_phy_ts_y1731_oam_conf_t Struct Reference

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

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

Data Fields

- BOOL range_en
- vtss_phy_ts_y1731_oam_delaym_type_t delaym_type
- union {
  - struct {
    - u8 val
    - u8 mask
  - } value
  - struct {
    - u8 upper
    - u8 lower
  - } range
- } meg_level

9.192.1 Detailed Description

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

Definition at line 1342 of file vtss_phy_ts_api.h.

9.192.2 Field Documentation

9.192.2.1 range_en

```
BOOL vtss_phy_ts_y1731_oam_conf_t::range_en
```

OAM MEG level in range enable/disable

Definition at line 1343 of file vtss_phy_ts_api.h.

9.192.2.2 delaym_type

```
vtss_phy_ts_y1731_oam_delaym_type_t vtss_phy_ts_y1731_oam_conf_t::delaym_type
```

OAM delay measurement method

Definition at line 1344 of file vtss_phy_ts_api.h.

9.192.2.3 val

```
u8 vtss_phy_ts_y1731_oam_conf_t::val
```

Value

Definition at line 1347 of file vtss_phy_ts_api.h.

9.192.2.4 mask

```
u8 vtss_phy_ts_y1731_oam_conf_t::mask
```

Mask

Definition at line 1348 of file vtss_phy_ts_api.h.

9.192.2.5 value

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

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

9.192.2.6 upper

```
u8 vtss_phy_ts_y1731_oam_conf_t::upper
```

Range Upper value

Definition at line 1351 of file vtss_phy_ts_api.h.

9.192.2.7 lower

```
u8 vtss_phy_ts_y1731_oam_conf_t::lower
```

Range lower value

Definition at line 1352 of file vtss_phy_ts_api.h.

9.192.2.8 range

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

Range configuration

9.192.2.9 meg_level

```
union { ... } vtss_phy_ts_y1731_oam_conf_t::meg_level
```

OAM MEG level config

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

- vtss_api/include/vtss_phy_ts_api.h

9.193 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

9.193.1 Detailed Description

Phy type information.

Definition at line 143 of file vtss_phy_api.h.

9.193.2 Field Documentation

9.193.2.1 part_number

```
u16 vtss_phy_type_t::part_number
```

Part number

Definition at line 145 of file vtss_phy_api.h.

9.193.2.2 revision

```
u16 vtss_phy_type_t::revision
```

Chip revision

Definition at line 146 of file vtss_phy_api.h.

9.193.2.3 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.

9.193.2.4 channel_id

```
u16 vtss_phy_type_t::channel_id
```

Channel id

Definition at line 148 of file vtss_phy_api.h.

9.193.2.5 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.

9.193.2.6 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

9.194 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]

9.194.1 Detailed Description

VeriPHY result.

Definition at line 958 of file vtss_phy_api.h.

9.194.2 Field Documentation

9.194.2.1 link

```
BOOL vtss_phy_veriphy_result_t::link
```

Link status

Definition at line 959 of file vtss_phy_api.h.

9.194.2.2 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.

9.194.2.3 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

9.195 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

9.195.1 Detailed Description

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

Definition at line 1680 of file vtss_phy_api.h.

9.195.2 Field Documentation

9.195.2.1 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.

9.195.2.2 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.

9.195.2.3 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.

9.195.2.4 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.

9.195.2.5 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

9.196 vtss_pi_conf_t Struct Reference

PI configuration.

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

Data Fields

- u32 cs_wait_ns

9.196.1 Detailed Description

PI configuration.

Definition at line 322 of file vtss_init_api.h.

9.196.2 Field Documentation

9.196.2.1 cs_wait_ns

```
u32 vtss_pi_conf_t::cs_wait_ns
```

Minimum CS wait time in nanoseconds

Definition at line 327 of file vtss_init_api.h.

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

- vtss_api/include/vtss_init_api.h

9.197 vtss_policer_ext_t Struct Reference

Policer Extensions.

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

Data Fields

- BOOL frame_rate
- BOOL flow_control

9.197.1 Detailed Description

Policer Extensions.

Definition at line 198 of file vtss_qos_api.h.

9.197.2 Field Documentation

9.197.2.1 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.

9.197.2.2 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

9.198 vtss_policer_t Struct Reference

Policer.

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

Data Fields

- vtss_burst_level_t level
- vtss_bitrate_t rate

9.198.1 Detailed Description

Policer.

Definition at line 185 of file vtss_qos_api.h.

9.198.2 Field Documentation

9.198.2.1 level

```
vtss_burst_level_t vtss_policer_t::level
```

Burst level

Definition at line 187 of file vtss_qos_api.h.

9.198.2.2 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

9.199 vtss_port_bridge_counters_t Struct Reference

Port bridge counter structure (RFC 4188)

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

Data Fields

- vtss_port_counter_t dot1dTpPortInDiscards

9.199.1 Detailed Description

Port bridge counter structure (RFC 4188)

Definition at line 201 of file port.h.

9.199.2 Field Documentation

9.199.2.1 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

9.200 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

9.200.1 Detailed Description

Advertisement control data for Clause 37 aneg.

Definition at line 127 of file vtss_port_api.h.

9.200.2 Field Documentation

9.200.2.1 fdx

```
BOOL vtss_port_clause_37_adv_t::fdx
```

(FD)

Definition at line 129 of file vtss_port_api.h.

9.200.2.2 hdx

```
BOOL vtss_port_clause_37_adv_t::hdx
```

(HD)

Definition at line 130 of file vtss_port_api.h.

9.200.2.3 symmetric_pause

```
BOOL vtss_port_clause_37_adv_t::symmetric_pause
```

(PS1)

Definition at line 131 of file vtss_port_api.h.

9.200.2.4 asymmetric_pause

```
BOOL vtss_port_clause_37_adv_t::asymmetric_pause
```

(PS2)

Definition at line 132 of file vtss_port_api.h.

9.200.2.5 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.

9.200.2.6 acknowledge

```
BOOL vtss_port_clause_37_adv_t::acknowledge
```

(Ack)

Definition at line 134 of file vtss_port_api.h.

9.200.2.7 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

9.201 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

9.201.1 Detailed Description

Auto-negotiation control parameter struct.

Definition at line 152 of file vtss_port_api.h.

9.201.2 Field Documentation

9.201.2.1 enable

```
BOOL vtss_port_clause_37_control_t::enable
```

Enable of Autoneg

Definition at line 154 of file vtss_port_api.h.

9.201.2.2 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

9.202 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

9.202.1 Detailed Description

Port configuration structure.

Definition at line 268 of file vtss_port_api.h.

9.202.2 Field Documentation

9.202.2.1 if_type

```
vtss_port_interface_t vtss_port_conf_t::if_type
```

Interface type

Definition at line 270 of file vtss_port_api.h.

9.202.2.2 sd_enable

```
BOOL vtss_port_conf_t::sd_enable
```

Signal detect enable

Definition at line 271 of file vtss_port_api.h.

9.202.2.3 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.

9.202.2.4 sd_internal

```
BOOL vtss_port_conf_t::sd_internal
```

Internal signal detect selection

Definition at line 273 of file vtss_port_api.h.

9.202.2.5 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.

9.202.2.6 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.

9.202.2.7 speed

```
vtss_port_speed_t vtss_port_conf_t::speed
```

Port speed

Definition at line 276 of file vtss_port_api.h.

9.202.2.8 fdx

```
BOOL vtss_port_conf_t::fdx
```

Full duplex mode

Definition at line 277 of file vtss_port_api.h.

9.202.2.9 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.

9.202.2.10 max_frame_length

```
u32 vtss_port_conf_t::max_frame_length
```

Maximum frame length

Definition at line 279 of file vtss_port_api.h.

9.202.2.11 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.

9.202.2.12 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.

9.202.2.13 exc_col_cont

```
BOOL vtss_port_conf_t::exc_col_cont
```

Excessive collision continuation

Definition at line 282 of file vtss_port_api.h.

9.202.2.14 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.

9.202.2.15 xaui_tx_lane_flip

```
BOOL vtss_port_conf_t::xaui_tx_lane_flip
```

Xaui Tx lane flip

Definition at line 284 of file vtss_port_api.h.

9.202.2.16 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.

9.202.2.17 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

9.203 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

9.203.1 Detailed Description

Port counter structure.

Definition at line 220 of file port.h.

9.203.2 Field Documentation

9.203.2.1 rmon

`vtss_port_rmon_counters_t vtss_port_counters_t::rmon`

RMON counters

Definition at line 222 of file port.h.

9.203.2.2 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.

9.203.2.3 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.

9.203.2.4 bridge

`vtss_port_bridge_counters_t vtss_port_counters_t::bridge`

Bridge counters

Definition at line 227 of file port.h.

9.203.2.5 prop

`vtss_port_proprietary_counters_t vtss_port_counters_t::prop`

Proprietary counters

Definition at line 230 of file port.h.

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

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

9.204 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

9.204.1 Detailed Description

Ethernet-like Interface counter structure (RFC 3635)

Definition at line 163 of file port.h.

9.204.2 Field Documentation

9.204.2.1 dot3InPauseFrames

```
vtss_port_counter_t vtss_port_ethernet_like_counters_t::dot3InPauseFrames
```

Rx pause

Definition at line 171 of file port.h.

9.204.2.2 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

9.205 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]

9.205.1 Detailed Description

Flow control setup.

Definition at line 189 of file vtss_port_api.h.

9.205.2 Field Documentation

9.205.2.1 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.

9.205.2.2 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.

9.205.2.3 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.

9.205.2.4 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

9.206 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

9.206.1 Detailed Description

Inter frame gap structure.

Definition at line 206 of file vtss_port_api.h.

9.206.2 Field Documentation

9.206.2.1 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.

9.206.2.2 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.

9.206.2.3 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

9.207 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

9.207.1 Detailed Description

Interfaces Group counter structure (RFC 2863)

Definition at line 144 of file port.h.

9.207.2 Field Documentation

9.207.2.1 ifInOctets

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInOctets
```

Rx octets

Definition at line 145 of file port.h.

9.207.2.2 ifInUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInUcastPkts
```

Rx unicasts

Definition at line 146 of file port.h.

9.207.2.3 ifInMulticastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInMulticastPkts
```

Rx multicasts

Definition at line 147 of file port.h.

9.207.2.4 ifInBroadcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInBroadcastPkts
```

Rx broadcasts

Definition at line 148 of file port.h.

9.207.2.5 ifInNUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInNUcastPkts
```

Rx non-unicasts

Definition at line 149 of file port.h.

9.207.2.6 ifInDiscards

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInDiscards
```

Rx discards

Definition at line 150 of file port.h.

9.207.2.7 ifInErrors

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifInErrors
```

Rx errors

Definition at line 151 of file port.h.

9.207.2.8 ifOutOctets

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutOctets
```

Tx octets

Definition at line 153 of file port.h.

9.207.2.9 ifOutUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutUcastPkts
```

Tx unicasts

Definition at line 154 of file port.h.

9.207.2.10 ifOutMulticastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutMulticastPkts
```

Tx multicasts

Definition at line 155 of file port.h.

9.207.2.11 ifOutBroadcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutBroadcastPkts
```

Tx broadcasts

Definition at line 156 of file port.h.

9.207.2.12 ifOutNUcastPkts

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutNUcastPkts
```

Tx non-unicasts

Definition at line 157 of file port.h.

9.207.2.13 ifOutDiscards

```
vtss_port_counter_t vtss_port_if_group_counters_t::ifOutDiscards
```

Tx discards

Definition at line 158 of file port.h.

9.207.2.14 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

9.208 vtss_port_ifh_t Struct Reference

Port Internal Frame Header structure.

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

Data Fields

- BOOL ena_inj_header
- BOOL ena_xtr_header

9.208.1 Detailed Description

Port Internal Frame Header structure.

Definition at line 434 of file vtss_port_api.h.

9.208.2 Field Documentation

9.208.2.1 ena_inj_header

```
BOOL vtss_port_ifh_t::ena_inj_header
```

At ingress expect long prefix followed by an internal frame header

Definition at line 437 of file vtss_port_api.h.

9.208.2.2 ena_xtr_header

```
BOOL vtss_port_ifh_t::ena_xtr_header
```

At egress prepend long prefix followed by the internal frame header

Definition at line 438 of file vtss_port_api.h.

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

- vtss_api/include/vtss_port_api.h

9.209 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

9.209.1 Detailed Description

Port map structure.

Definition at line 72 of file vtss_port_api.h.

9.209.2 Field Documentation

9.209.2.1 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.

9.209.2.2 chip_no

```
vtss_chip_no_t vtss_port_map_t::chip_no
```

Chip number, multi-chip targets

Definition at line 75 of file vtss_port_api.h.

9.209.2.3 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.

9.209.2.4 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.

9.209.2.5 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

9.210 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]

9.210.1 Detailed Description

Port proprietary counter structure.

Definition at line 207 of file port.h.

9.210.2 Field Documentation

9.210.2.1 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.

9.210.2.2 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

9.211 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

9.211.1 Detailed Description

RMON counter structure (RFC 2819)

Definition at line 109 of file port.h.

9.211.2 Field Documentation

9.211.2.1 rx_etherStatsDropEvents

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsDropEvents
```

Rx drop events

Definition at line 110 of file port.h.

9.211.2.2 rx_etherStatsOctets

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOctets
```

Rx octets

Definition at line 111 of file port.h.

9.211.2.3 rx_etherStatsPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsPkts
```

Rx packets

Definition at line 112 of file port.h.

9.211.2.4 rx_etherStatsBroadcastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsBroadcastPkts
```

Rx broadcasts

Definition at line 113 of file port.h.

9.211.2.5 rx_etherStatsMulticastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsMulticastPkts
```

Rx multicasts

Definition at line 114 of file port.h.

9.211.2.6 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.

9.211.2.7 rx_etherStatsUndersizePkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsUndersizePkts
```

Rx undersize packets

Definition at line 116 of file port.h.

9.211.2.8 rx_etherStatsOversizePkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsOversizePkts
```

Rx oversize packets

Definition at line 117 of file port.h.

9.211.2.9 rx_etherStatsFragments

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsFragments
```

Rx fragments

Definition at line 118 of file port.h.

9.211.2.10 rx_etherStatsJabbers

```
vtss_port_counter_t vtss_port_rmon_counters_t::rx_etherStatsJabbers
```

Rx jabbers

Definition at line 119 of file port.h.

9.211.2.11 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.

9.211.2.12 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.

9.211.2.13 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.

9.211.2.14 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.

9.211.2.15 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.

9.211.2.16 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.

9.211.2.17 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.

9.211.2.18 tx_etherStatsDropEvents

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsDropEvents
```

Tx drop events

Definition at line 128 of file port.h.

9.211.2.19 tx_etherStatsOctets

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsOctets
```

Tx octets

Definition at line 129 of file port.h.

9.211.2.20 tx_etherStatsPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsPkts
```

Tx packets

Definition at line 130 of file port.h.

9.211.2.21 tx_etherStatsBroadcastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsBroadcastPkts
```

Tx broadcasts

Definition at line 131 of file port.h.

9.211.2.22 tx_etherStatsMulticastPkts

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsMulticastPkts
```

Tx multicasts

Definition at line 132 of file port.h.

9.211.2.23 tx_etherStatsCollisions

```
vtss_port_counter_t vtss_port_rmon_counters_t::tx_etherStatsCollisions
```

Tx collisions

Definition at line 133 of file port.h.

9.211.2.24 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.

9.211.2.25 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.

9.211.2.26 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.

9.211.2.27 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.

9.211.2.28 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.

9.211.2.29 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.

9.211.2.30 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

9.212 vtss_port_serdes_conf_t Struct Reference

SFI Serdes configuration.

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

Data Fields

- BOOL sfp_dac

9.212.1 Detailed Description

SFI Serdes configuration.

Definition at line 261 of file vtss_port_api.h.

9.212.2 Field Documentation

9.212.2.1 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

9.213 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

9.213.1 Detailed Description

Advertisement control data for SGMII aneg.

Definition at line 139 of file vtss_port_api.h.

9.213.2 Field Documentation

9.213.2.1 link

```
BOOL vtss_port_sgmii_aneg_t::link
```

LP link status

Definition at line 141 of file vtss_port_api.h.

9.213.2.2 fdx

```
BOOL vtss_port_sgmii_aneg_t::fdx
```

FD

Definition at line 142 of file vtss_port_api.h.

9.213.2.3 hdx

```
BOOL vtss_port_sgmii_aneg_t::hdx
```

HD

Definition at line 143 of file vtss_port_api.h.

9.213.2.4 speed_10M

```
BOOL vtss_port_sgmii_aneg_t::speed_10M
```

speed 10 advertised

Definition at line 144 of file vtss_port_api.h.

9.213.2.5 speed_100M

```
BOOL vtss_port_sgmii_aneg_t::speed_100M
```

speed 100 advertised

Definition at line 145 of file vtss_port_api.h.

9.213.2.6 speed_1G

```
BOOL vtss_port_sgmii_aneg_t::speed_1G
```

speed 1G advertised

Definition at line 146 of file vtss_port_api.h.

9.213.2.7 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

9.214 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

9.214.1 Detailed Description

Port status parameter struct.

Definition at line 295 of file port.h.

9.214.2 Field Documentation

9.214.2.1 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.

9.214.2.2 link

```
BOOL vtss_port_status_t::link
```

Link is up. Remaining fields only valid if TRUE

Definition at line 298 of file port.h.

9.214.2.3 speed

```
vtss_port_speed_t vtss_port_status_t::speed
```

Speed

Definition at line 299 of file port.h.

9.214.2.4 fdx

```
BOOL vtss_port_status_t::fdx
```

Full duplex

Definition at line 300 of file port.h.

9.214.2.5 remote_fault

```
BOOL vtss_port_status_t::remote_fault
```

Remote fault signalled

Definition at line 301 of file port.h.

9.214.2.6 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.

9.214.2.7 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.

9.214.2.8 aneg

```
vtss_aneg_t vtss_port_status_t::aneg
```

Auto negotiation result

Definition at line 307 of file port.h.

9.214.2.9 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.

9.214.2.10 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.

9.214.2.11 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

9.215 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

9.215.1 Detailed Description

QCE action.

Definition at line 566 of file vtss_qos_api.h.

9.215.2 Field Documentation

9.215.2.1 prio_enable

```
BOOL vtss_qce_action_t::prio_enable
```

Enable priority classification

Definition at line 568 of file vtss_qos_api.h.

9.215.2.2 prio

```
vtss_prio_t vtss_qce_action_t::prio
```

Priority value

Definition at line 569 of file vtss_qos_api.h.

9.215.2.3 dp_enable

```
BOOL vtss_qce_action_t::dp_enable
```

Enable DP classification

Definition at line 570 of file vtss_qos_api.h.

9.215.2.4 dp

```
vtss_dp_level_t vtss_qce_action_t::dp
```

DP value

Definition at line 571 of file vtss_qos_api.h.

9.215.2.5 dscp_enable

```
BOOL vtss_qce_action_t::dscp_enable
```

Enable DSCP classification

Definition at line 572 of file vtss_qos_api.h.

9.215.2.6 dscp

```
vtss_dscp_t vtss_qce_action_t::dscp
```

DSCP value

Definition at line 573 of file vtss_qos_api.h.

9.215.2.7 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.

9.215.2.8 pcp

```
vtss_tagprio_t vtss_qce_action_t::pcp
```

PCP value

Definition at line 576 of file vtss_qos_api.h.

9.215.2.9 dei

```
vtss_dei_t vtss_qce_action_t::dei
```

DEI value

Definition at line 577 of file vtss_qos_api.h.

9.215.2.10 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.

9.215.2.11 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

9.216 vtss_qce_frame_etype_t Struct Reference

Frame data for VTSS_QCE_TYPE_ETYPE.

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

Data Fields

- vtss_vcap_u16_t etype
- vtss_vcap_u32_t data

9.216.1 Detailed Description

Frame data for VTSS_QCE_TYPE_ETYPE.

Definition at line 494 of file vtss_qos_api.h.

9.216.2 Field Documentation

9.216.2.1 etype

```
vtss_vcap_u16_t vtss_qce_frame_etype_t::etype
```

Ethernet Type value

Definition at line 496 of file vtss_qos_api.h.

9.216.2.2 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

9.217 vtss_qce_frame_ipv4_t Struct Reference

Frame data for VTSS_QCE_TYPE_IPV4.

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

Data Fields

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

9.217.1 Detailed Description

Frame data for VTSS_QCE_TYPE_IPV4.

Definition at line 513 of file vtss_qos_api.h.

9.217.2 Field Documentation

9.217.2.1 fragment

`vtss_vcap_bit_t vtss_qce_frame_ipv4_t::fragment`

Fragment

Definition at line 515 of file vtss_qos_api.h.

9.217.2.2 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.

9.217.2.3 proto

`vtss_vcap_u8_t vtss_qce_frame_ipv4_t::proto`

Protocol

Definition at line 517 of file vtss_qos_api.h.

9.217.2.4 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.

9.217.2.5 dip

`vtss_vcap_ip_t vtss_qce_frame_ipv4_t::dip`

Destination IP address - Serval: key_type = ip_addr and mac_ip_addr

Definition at line 520 of file vtss_qos_api.h.

9.217.2.6 sport

```
vtss_vcap_vr_t vtss_qce_frame_ipv4_t::sport
```

UDP/TCP: Source port - Serval: key_type = normal, ip_addr and mac_ip_addr

Definition at line 522 of file vtss_qos_api.h.

9.217.2.7 dport

```
vtss_vcap_vr_t vtss_qce_frame_ipv4_t::dport
```

UDP/TCP: Destination port - Serval: key_type = double_tag, ip_addr and mac_ip_addr

Definition at line 523 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

9.218 vtss_qce_frame_ipv6_t Struct Reference

Frame data for VTSS_QCE_TYPE_IPV6.

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

Data Fields

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

9.218.1 Detailed Description

Frame data for VTSS_QCE_TYPE_IPV6.

Definition at line 527 of file vtss_qos_api.h.

9.218.2 Field Documentation

9.218.2.1 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.

9.218.2.2 proto

```
vtss_vcap_u8_t vtss_qce_frame_ipv6_t::proto
```

Protocol

Definition at line 530 of file vtss_qos_api.h.

9.218.2.3 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.

9.218.2.4 dip

```
vtss_vcap_u128_t vtss_qce_frame_ipv6_t::dip
```

Destination IP address - 64 LSB on Serval when key_type = mac_ip_addr

Definition at line 533 of file vtss_qos_api.h.

9.218.2.5 sport

```
vtss_vcap_vr_t vtss_qce_frame_ipv6_t::sport
```

UDP/TCP: Source port - Serval: key_type = normal, ip_addr and mac_ip_addr

Definition at line 535 of file vtss_qos_api.h.

9.218.2.6 dport

```
vtss_vcap_vr_t vtss_qce_frame_ipv6_t::dport
```

UDP/TCP: Destination port - Serval: key_type = double_tag, ip_addr and mac_ip_addr

Definition at line 536 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

9.219 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

9.219.1 Detailed Description

Frame data for VTSS_QCE_TYPE_LLC.

Definition at line 501 of file vtss_qos_api.h.

9.219.2 Field Documentation

9.219.2.1 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

9.220 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

9.220.1 Detailed Description

Frame data for VTSS_QCE_TYPE_SNAP.

Definition at line 507 of file vtss_qos_api.h.

9.220.2 Field Documentation

9.220.2.1 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

9.221 vtss_qce_key_t Struct Reference

QCE key.

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

Data Fields

- BOOL port_list [VTSS_PORT_ARRAY_SIZE]
- vtss_qce_mac_t mac
- vtss_qce_tag_t tag
- vtss_qce_tag_t inner_tag
- vtss_qce_type_t type
- union {
  vtss_qce_frame_etype_t etype
  vtss_qce_frame_llc_t llc
  vtss_qce_frame_snap_t snap
  vtss_qce_frame_ipv4_t ipv4
  vtss_qce_frame_ipv6_t ipv6
  } frame

9.221.1 Detailed Description

QCE key.

Definition at line 542 of file vtss_qos_api.h.

9.221.2 Field Documentation

9.221.2.1 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.

9.221.2.2 mac

```
vtss_qce_mac_t vtss_qce_key_t::mac
```

MAC

Definition at line 545 of file vtss_qos_api.h.

9.221.2.3 tag

`vtss_qce_tag_t vtss_qce_key_t::tag`

Tag

Definition at line 546 of file vtss_qos_api.h.

9.221.2.4 inner_tag

`vtss_qce_tag_t vtss_qce_key_t::inner_tag`

Inner tag

Definition at line 548 of file vtss_qos_api.h.

9.221.2.5 type

`vtss_qce_type_t vtss_qce_key_t::type`

Frame type

Definition at line 550 of file vtss_qos_api.h.

9.221.2.6 etype

`vtss_qce_frame_etype_t vtss_qce_key_t::etype`

VTSS_QCE_TYPE_ETYPE

Definition at line 555 of file vtss_qos_api.h.

9.221.2.7 llc

`vtss_qce_frame_llc_t vtss_qce_key_t::llc`

VTSS_QCE_TYPE_LLC

Definition at line 556 of file vtss_qos_api.h.

9.221.2.8 snap

```
vtss_qce_frame_snap_t vtss_qce_key_t::snap
```

VTSS_QCE_TYPE_SNAP

Definition at line 557 of file vtss_qos_api.h.

9.221.2.9 ipv4

```
vtss_qce_frame_ipv4_t vtss_qce_key_t::ipv4
```

VTSS_QCE_TYPE_IPV4

Definition at line 558 of file vtss_qos_api.h.

9.221.2.10 ipv6

```
vtss_qce_frame_ipv6_t vtss_qce_key_t::ipv6
```

VTSS_QCE_TYPE_IPV6

Definition at line 559 of file vtss_qos_api.h.

9.221.2.11 frame

```
union { ... } vtss_qce_key_t::frame
```

Frame type specific data

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

- vtss_api/include/vtss_qos_api.h

9.222 vtss_qce_mac_t Struct Reference

QCE MAC information.

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

Data Fields

- vtss_vcap_bit_t dmac_mc
- vtss_vcap_bit_t dmac_bc
- vtss_vcap_u48_t dmac
- vtss_vcap_u48_t smac

9.222.1 Detailed Description

QCE MAC information.

Definition at line 471 of file vtss_qos_api.h.

9.222.2 Field Documentation

9.222.2.1 dmac_mc

```
vtss_vcap_bit_t vtss_qce_mac_t::dmac_mc
```

Multicast DMAC

Definition at line 473 of file vtss_qos_api.h.

9.222.2.2 dmac_bc

```
vtss_vcap_bit_t vtss_qce_mac_t::dmac_bc
```

Broadcast DMAC

Definition at line 474 of file vtss_qos_api.h.

9.222.2.3 dmac

```
vtss_vcap_u48_t vtss_qce_mac_t::dmac
```

DMAC - Serval: key_type = mac_ip_addr

Definition at line 476 of file vtss_qos_api.h.

9.222.2.4 smac

```
vtss_vcap_u48_t vtss_qce_mac_t::smac
```

SMAC - Only the 24 most significant bits (OUI) are supported on Jaguar1, rest are wildcards

Definition at line 478 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

9.223 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

9.223.1 Detailed Description

QoS Control Entry.

Definition at line 588 of file vtss_qos_api.h.

9.223.2 Field Documentation

9.223.2.1 id

```
vtss_qce_id_t vtss_qce_t::id
```

Entry ID

Definition at line 590 of file vtss_qos_api.h.

9.223.2.2 key

```
vtss_qce_key_t vtss_qce_t::key
```

QCE key

Definition at line 591 of file vtss_qos_api.h.

9.223.2.3 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

9.224 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

9.224.1 Detailed Description

QCE tag information.

Definition at line 482 of file vtss_qos_api.h.

9.224.2 Field Documentation

9.224.2.1 vid

```
vtss_vcap_vr_t vtss_qce_tag_t::vid
```

VLAN ID (12 bit)

Definition at line 484 of file vtss_qos_api.h.

9.224.2.2 pcp

```
vtss_vcap_u8_t vtss_qce_tag_t::pcp
```

PCP (3 bit)

Definition at line 485 of file vtss_qos_api.h.

9.224.2.3 dei

```
vtss_vcap_bit_t vtss_qce_tag_t::dei
```

DEI

Definition at line 486 of file vtss_qos_api.h.

9.224.2.4 tagged

```
vtss_vcap_bit_t vtss_qce_tag_t::tagged
```

Tagged/untagged frame

Definition at line 487 of file vtss_qos_api.h.

9.224.2.5 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

9.225 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
- vtss_red_v2_t red_v2 [VTSS_QUEUE_ARRAY_SIZE][2]

9.225.1 Detailed Description

All parameters below are defined per chip.

Definition at line 91 of file vtss_qos_api.h.

9.225.2 Field Documentation

9.225.2.1 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.

9.225.2.2 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.

9.225.2.3 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.

9.225.2.4 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.

9.225.2.5 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.

9.225.2.6 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.

9.225.2.7 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.

9.225.2.8 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.

9.225.2.9 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.

9.225.2.10 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.

9.225.2.11 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.

9.225.2.12 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.

9.225.2.13 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.

9.225.2.14 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.

9.225.2.15 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.

9.225.2.16 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.

9.225.2.17 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.

9.225.2.18 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.

9.225.2.19 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.

9.225.2.20 red_v2

```
vtss_red_v2_t vtss_qos_conf_t::red_v2[VTSS_QUEUE_ARRAY_SIZE][2]
```

Random Early Detection - per queue (0..7), per dpl (0..1)

Definition at line 143 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

9.226 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
- u8 dwrr_cnt
- vtss_pct_t queue_pct [VTSS_QUEUE_ARRAY_SIZE]
- BOOL dmac_dip
- vtss_vcap_key_type_t key_type

9.226.1 Detailed Description

QoS setup per port.

Definition at line 344 of file vtss_qos_api.h.

9.226.2 Field Documentation

9.226.2.1 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.

9.226.2.2 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.

9.226.2.3 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.

9.226.2.4 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.

9.226.2.5 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.

9.226.2.6 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.

9.226.2.7 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.

9.226.2.8 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.

9.226.2.9 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.

9.226.2.10 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.

9.226.2.11 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.

9.226.2.12 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.

9.226.2.13 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.

9.226.2.14 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.

9.226.2.15 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.

9.226.2.16 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.

9.226.2.17 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.

9.226.2.18 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.

9.226.2.19 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.

9.226.2.20 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.

9.226.2.21 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.

9.226.2.22 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.

9.226.2.23 dwrr_enable

```
BOOL vtss_qos_port_conf_t::dwrr_enable
```

Enable Weighted fairness queueing

Definition at line 400 of file vtss_qos_api.h.

9.226.2.24 dwrr_cnt

```
u8 vtss_qos_port_conf_t::dwrr_cnt
```

Number of queues, starting from queue 0, running in DWRR mode

Definition at line 402 of file vtss_qos_api.h.

9.226.2.25 queue_pct

```
vtss_pct_t vtss_qos_port_conf_t::queue_pct[VTSS_QUEUE_ARRAY_SIZE]
```

Queue percentages

Definition at line 404 of file vtss_qos_api.h.

9.226.2.26 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.

9.226.2.27 key_type

```
vtss_vcap_key_type_t vtss_qos_port_conf_t::key_type
```

Key type for received frames

Definition at line 412 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

9.227 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

9.227.1 Detailed Description

Structure for Get PHY RC-PLL status.

Definition at line 1742 of file vtss_phy_api.h.

9.227.2 Field Documentation

9.227.2.1 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.

9.227.2.2 cal_error

```
u8 vtss_rcpll_status_t::cal_error
```

Calibration Error indication

Definition at line 1744 of file vtss_phy_api.h.

9.227.2.3 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

9.228 vtss_red_v2_t Struct Reference

Random Early Detection configuration struct version 2 (per queue, per dpl - switch global)

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

Data Fields

- BOOL enable
- vtss_pct_t min_fl
- vtss_pct_t max
- vtss_wred_v2_max_t max_unit

9.228.1 Detailed Description

Random Early Detection configuration struct version 2 (per queue, per dpl - switch global)

Definition at line 73 of file vtss_qos_api.h.

9.228.2 Field Documentation

9.228.2.1 enable

```
BOOL vtss_red_v2_t::enable
```

Enable/disable RED

Definition at line 75 of file vtss_qos_api.h.

9.228.2.2 min_fl

```
vtss_pct_t vtss_red_v2_t::min_fl
```

Minimum fill level

Definition at line 76 of file vtss_qos_api.h.

9.228.2.3 max

```
vtss_pct_t vtss_red_v2_t::max
```

Maximum drop probability or fill level - selected by max_unit

Definition at line 77 of file vtss_qos_api.h.

9.228.2.4 max_unit

```
vtss_wred_v2_max_t vtss_red_v2_t::max_unit
```

Selects the unit for max

Definition at line 78 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

9.229 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

9.229.1 Detailed Description

Restart status.

Definition at line 608 of file vtss_init_api.h.

9.229.2 Field Documentation

9.229.2.1 restart

```
vtss_restart_t vtss_restart_status_t::restart
```

Previous restart mode

Definition at line 609 of file vtss_init_api.h.

9.229.2.2 prev_version

```
vtss_version_t vtss_restart_status_t::prev_version
```

Previous API version

Definition at line 610 of file vtss_init_api.h.

9.229.2.3 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

9.230 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
  } route

- vtss_vid_t vlan

9.230.1 Detailed Description

Routing entry.

Definition at line 868 of file types.h.

9.230.2 Field Documentation

9.230.2.1 type

```
vtss_routing_entry_type_t vtss_routing_entry_t::type
```

Type of route

Definition at line 871 of file types.h.

9.230.2.2 ipv4_uc

```
vtss_ipv4_uc_t vtss_routing_entry_t::ipv4_uc
```

IPv6 unicast route

Definition at line 875 of file types.h.

9.230.2.3 ipv6_uc

```
vtss_ipv6_uc_t vtss_routing_entry_t::ipv6_uc
```

IPv6 unicast route

Definition at line 878 of file types.h.

9.230.2.4 route

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

Route

9.230.2.5 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

9.231 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]

9.231.1 Detailed Description

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

Definition at line 1664 of file vtss_phy_api.h.

9.231.2 Field Documentation

9.231.2.1 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

9.232 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

9.232.1 Detailed Description

Serdes macro configuration.

Definition at line 363 of file vtss_init_api.h.

9.232.2 Field Documentation

9.232.2.1 serdes1g_vdd

```
vtss_vdd_t vtss_serdes_macro_conf_t::serdes1g_vdd
```

Serdes1g supply

Definition at line 364 of file vtss_init_api.h.

9.232.2.2 serdes6g_vdd

```
vtss_vdd_t vtss_serdes_macro_conf_t::serdes6g_vdd
```

Serdes6g supply

Definition at line 365 of file vtss_init_api.h.

9.232.2.3 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.

9.232.2.4 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

9.233 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

9.233.1 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.

9.233.2 Field Documentation

9.233.2.1 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.

9.233.2.2 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

9.234 vtss_sgpio_conf_t Struct Reference

SGPIO 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]

9.234.1 Detailed Description

SGPIO configuration for a group.

Definition at line 797 of file vtss_misc_api.h.

9.234.2 Field Documentation

9.234.2.1 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.

9.234.2.2 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.

9.234.2.3 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

9.235 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]

9.235.1 Detailed Description

SGPIO port configuration.

Definition at line 789 of file vtss_misc_api.h.

9.235.2 Field Documentation

9.235.2.1 enabled

```
BOOL vtss_sgpio_port_conf_t::enabled
```

Port enabled/disabled

Definition at line 791 of file vtss_misc_api.h.

9.235.2.2 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.

9.235.2.3 int_pol_high

```
BOOL vtss_sgpio_port_conf_t::int_pol_high[4]
```

SGPIO interrupt polarity for each bit - TRUE - Interrupt when SGPIO pin high, FALSE - Interrupt when SGPIO pin low.

Definition at line 793 of file vtss_misc_api.h.

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

- vtss_api/include/vtss_misc_api.h

9.236 vtss_sgpio_port_data_t Struct Reference

SGPIO read data for a port.

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

Data Fields

- BOOL value [4]

9.236.1 Detailed Description

SGPIO read data for a port.

Definition at line 835 of file vtss_misc_api.h.

9.236.2 Field Documentation

9.236.2.1 value

```
BOOL vtss_sgpio_port_data_t::value[4]
```

Data for each and bit

Definition at line 837 of file vtss_misc_api.h.

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

- vtss_api/include/vtss_misc_api.h

9.237 vtss_shaper_t Struct Reference

Shaper.

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

Data Fields

- vtss_burst_level_t level
- vtss_bitrate_t rate
- vtss_burst_level_t ebs
- vtss_bitrate_t eir
- vtss_shaper_mode_t mode

9.237.1 Detailed Description

Shaper.

Definition at line 298 of file vtss_qos_api.h.

9.237.2 Field Documentation

9.237.2.1 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.

9.237.2.2 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.

9.237.2.3 ebs

`vtss_burst_level_t vtss_shaper_t::ebs`

EBS (Excess Burst Size). Unit: bytes

Definition at line 303 of file vtss_qos_api.h.

9.237.2.4 eir

`vtss_bitrate_t vtss_shaper_t::eir`

EIR (Excess Information Rate). Unit: kbps. Use VTSS_BITRATE_DISABLED to disable DLB

Definition at line 304 of file vtss_qos_api.h.

9.237.2.5 mode

```
vtss_shaper_mode_t vtss_shaper_t::mode
```

RT (Rate type). Shaper rate type configuration: 0 = Line-rate, 1 = Data-rate

Definition at line 307 of file vtss_qos_api.h.

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

- vtss_api/include/vtss_qos_api.h

9.238 vtss_synce_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

9.238.1 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.

9.238.2 Field Documentation

9.238.2.1 port_no

```
vtss_port_no_t vtss_synce_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.

9.238.2.2 squelsh

```
BOOL vtss_synce_clock_in_t::squelsh
```

Enable/disable of automatic squelsh

Definition at line 110 of file vtss_sync_api.h.

9.238.2.3 enable

```
BOOL vtss_synce_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

9.239 vtss_synce_clock_out_t Struct Reference

Struct containing configuration for a recovered clock output port.

```
#include <vtss_sync_api.h>
```

Data Fields

- vtss_synce_divider_t divider
- BOOL enable

9.239.1 Detailed Description

Struct containing configuration for a recovered clock output port.

Definition at line 73 of file vtss_sync_api.h.

9.239.2 Field Documentation

9.239.2.1 divider

```
vtss_synce_divider_t vtss_synce_clock_out_t::divider
```

Selection the clock division. This should be set to VTSS_SYNCE_DIVIDER_1 if recovered clock is comming from internal PHY

Definition at line 75 of file vtss_sync_api.h.

9.239.2.2 enable

```
BOOL vtss_synce_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

9.240 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

9.240.1 Detailed Description

Tag Control Information (according to IEEE 802.1Q)

Definition at line 206 of file vtss_packet_api.h.

9.240.2 Field Documentation

9.240.2.1 vid

```
vtss_vid_t vtss_tci_t::vid
```

VLAN ID

Definition at line 208 of file vtss_packet_api.h.

9.240.2.2 cfi

```
BOOL vtss_tci_t::cfi
```

Canonical Format Indicator

Definition at line 209 of file vtss_packet_api.h.

9.240.2.3 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

9.241 vtss_timeofday_t Struct Reference

Time of day structure.

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

Data Fields

- u32 sec
- time_t sec

9.241.1 Detailed Description

Time of day structure.

Definition at line 59 of file vtss_os_ecos.h.

9.241.2 Field Documentation

9.241.2.1 sec [1/2]

```
u32 vtss_timeofday_t::sec
```

Time of day in seconds

Definition at line 60 of file vtss_os_ecos.h.

9.241.2.2 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

9.242 vtss_timestamp_t Struct Reference

Time stamp in seconds and nanoseconds.

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

Data Fields

- u16 sec_msb
- u32 seconds
- u32 nanoseconds

9.242.1 Detailed Description

Time stamp in seconds and nanoseconds.

Definition at line 1212 of file types.h.

9.242.2 Field Documentation

9.242.2.1 sec_msb

```
u16 vtss_timestamp_t::sec_msb
```

Seconds msb

Definition at line 1213 of file types.h.

9.242.2.2 seconds

```
u32 vtss_timestamp_t::seconds
```

Seconds

Definition at line 1214 of file types.h.

9.242.2.3 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

9.243 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]

9.243.1 Detailed Description

Trace group configuration.

Definition at line 97 of file vtss_misc_api.h.

9.243.2 Field Documentation

9.243.2.1 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

9.244 vtss_ts_alt_clock_mode_t Struct Reference

parameter for setting the alternative clock mode.

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

Data Fields

- BOOL one_pps_out
- BOOL one_pps_in
- BOOL save
- BOOL load

9.244.1 Detailed Description

parameter for setting the alternative clock mode.

external clock output configuration.

Definition at line 222 of file vtss_ts_api.h.

9.244.2 Field Documentation

9.244.2.1 one_pps_out

```
BOOL vtss_ts_alt_clock_mode_t::one_pps_out
```

Enable 1pps output

Definition at line 223 of file vtss_ts_api.h.

9.244.2.2 one_pps_in

```
BOOL vtss_ts_alt_clock_mode_t::one_pps_in
```

Enable 1pps input

Definition at line 224 of file vtss_ts_api.h.

9.244.2.3 save

```
BOOL vtss_ts_alt_clock_mode_t::save
```

Save actual time counter at next 1 PPS input

Definition at line 225 of file vtss_ts_api.h.

9.244.2.4 load

```
BOOL vtss_ts_alt_clock_mode_t::load
```

Load actual time counter with at next 1 PPS input

Definition at line 226 of file vtss_ts_api.h.

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

- vtss_api/include/vtss_ts_api.h

9.245 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

9.245.1 Detailed Description

external clock output configuration.

Definition at line 289 of file vtss_ts_api.h.

9.245.2 Field Documentation

9.245.2.1 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.

9.245.2.2 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.

9.245.2.3 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

9.246 vtss_ts_id_t Struct Reference

Timestamp identifier.

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

Data Fields

- u32 ts_id

9.246.1 Detailed Description

Timestamp identifier.

Definition at line 527 of file vtss_ts_api.h.

9.246.2 Field Documentation

9.246.2.1 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

9.247 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

9.247.1 Detailed Description

Hardware timestamping format mode for internal ports.

Definition at line 498 of file vtss_ts_api.h.

9.247.2 Field Documentation

9.247.2.1 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

9.248 vtss_ts_operation_mode_t Struct Reference

Timestamp operation.

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

Data Fields

- vtss_ts_mode_t mode

9.248.1 Detailed Description

Timestamp operation.

Definition at line 451 of file vtss_ts_api.h.

9.248.2 Field Documentation

9.248.2.1 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

9.249 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)

9.249.1 Detailed Description

Timestamp allocation.

Definition at line 613 of file vtss_ts_api.h.

9.249.2 Field Documentation

9.249.2.1 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.

9.249.2.2 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.

9.249.2.3 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

9.250 vtss_ts_timestamp_t Struct Reference

Timestamp structure.

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

Data Fields

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

9.250.1 Detailed Description

Timestamp structure.

Definition at line 532 of file vtss_ts_api.h.

9.250.2 Field Documentation

9.250.2.1 ts

```
u32 vtss_ts_timestamp_t::ts
```

Timestamp value

Definition at line 533 of file vtss_ts_api.h.

9.250.2.2 id

```
u32 vtss_ts_timestamp_t::id
```

Timestamp identifier

Definition at line 534 of file vtss_ts_api.h.

9.250.2.3 context

```
void* vtss_ts_timestamp_t::context
```

Application specific context

Definition at line 535 of file vtss_ts_api.h.

9.250.2.4 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

9.251 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

9.251.1 Detailed Description

VCAP IPv4 address value and mask.

Definition at line 968 of file types.h.

9.251.2 Field Documentation

9.251.2.1 value

```
vtss_ip_t vtss_vcap_ip_t::value
```

Value

Definition at line 970 of file types.h.

9.251.2.2 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

9.252 vtss_vcap_port_conf_t Struct Reference

VCAP port configuration.

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

Data Fields

- vtss_vcap_key_type_t key_type_is1_1
- BOOL dmac_dip_1

9.252.1 Detailed Description

VCAP port configuration.

Definition at line 1179 of file vtss_l2_api.h.

9.252.2 Field Documentation

9.252.2.1 key_type_is1_1

```
vtss_vcap_key_type_t vtss_vcap_port_conf_t::key_type_is1_1
```

Key type for second IS1 lookup

Definition at line 1180 of file vtss_l2_api.h.

9.252.2.2 dmac_dip_1

```
BOOL vtss_vcap_port_conf_t::dmac_dip_1
```

Enable DMAC/DIP matching in second lookup (default SMAC/SIP)

Definition at line 1181 of file vtss_l2_api.h.

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

- vtss_api/include/vtss_l2_api.h

9.253 vtss_vcap_u128_t Struct Reference

VCAP 128 bit value and mask.

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

Data Fields

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

9.253.1 Detailed Description

VCAP 128 bit value and mask.

Definition at line 954 of file types.h.

9.253.2 Field Documentation

9.253.2.1 value

```
u8 vtss_vcap_u128_t::value[16]
```

Value

Definition at line 956 of file types.h.

9.253.2.2 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

9.254 vtss_vcap_u16_t Struct Reference

VCAP 16 bit value and mask.

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

Data Fields

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

9.254.1 Detailed Description

VCAP 16 bit value and mask.

Definition at line 919 of file types.h.

9.254.2 Field Documentation

9.254.2.1 value

```
u8 vtss_vcap_u16_t::value[2]
```

Value

Definition at line 921 of file types.h.

9.254.2.2 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

9.255 vtss_vcap_u24_t Struct Reference

VCAP 24 bit value and mask.

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

Data Fields

- u8 value [3]
- u8 mask [3]

9.255.1 Detailed Description

VCAP 24 bit value and mask.

Definition at line 926 of file types.h.

9.255.2 Field Documentation

9.255.2.1 value

```
u8 vtss_vcap_u24_t::value[3]
```

Value

Definition at line 928 of file types.h.

9.255.2.2 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

9.256 vtss_vcap_u32_t Struct Reference

VCAP 32 bit value and mask.

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

Data Fields

- u8 value [4]
- u8 mask [4]

9.256.1 Detailed Description

VCAP 32 bit value and mask.

Definition at line 933 of file types.h.

9.256.2 Field Documentation

9.256.2.1 value

```
u8 vtss_vcap_u32_t::value[4]
```

Value

Definition at line 935 of file types.h.

9.256.2.2 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

9.257 vtss_vcap_u40_t Struct Reference

VCAP 40 bit value and mask.

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

Data Fields

- u8 value [5]
- u8 mask [5]

9.257.1 Detailed Description

VCAP 40 bit value and mask.

Definition at line 940 of file types.h.

9.257.2 Field Documentation

9.257.2.1 value

```
u8 vtss_vcap_u40_t::value[5]
```

Value

Definition at line 942 of file types.h.

9.257.2.2 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

9.258 vtss_vcap_u48_t Struct Reference

VCAP 48 bit value and mask.

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

Data Fields

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

9.258.1 Detailed Description

VCAP 48 bit value and mask.

Definition at line 947 of file types.h.

9.258.2 Field Documentation

9.258.2.1 value

```
u8 vtss_vcap_u48_t::value[6]
```

Value

Definition at line 949 of file types.h.

9.258.2.2 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

9.259 vtss_vcap_u8_t Struct Reference

VCAP 8 bit value and mask.

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

Data Fields

- u8 value
- u8 mask

9.259.1 Detailed Description

VCAP 8 bit value and mask.

Definition at line 912 of file types.h.

9.259.2 Field Documentation

9.259.2.1 value

```
u8 vtss_vcap_u8_t::value
```

Value

Definition at line 914 of file types.h.

9.259.2.2 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

9.260 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

9.260.1 Detailed Description

VCAP UDP/TCP port range.

Definition at line 975 of file types.h.

9.260.2 Field Documentation

9.260.2.1 in_range

```
BOOL vtss_vcap_udp_tcp_t::in_range
```

Port in range match

Definition at line 977 of file types.h.

9.260.2.2 low

```
vtss_udp_tcp_t vtss_vcap_udp_tcp_t::low
```

Port low value

Definition at line 978 of file types.h.

9.260.2.3 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

9.261 vtss_vcap_vid_t Struct Reference

VCAP VLAN ID value and mask.

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

Data Fields

- u16 value
- u16 mask

9.261.1 Detailed Description

VCAP VLAN ID value and mask.

Definition at line 961 of file types.h.

9.261.2 Field Documentation

9.261.2.1 value

```
u16 vtss_vcap_vid_t::value
```

Value

Definition at line 963 of file types.h.

9.261.2.2 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

9.262 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

9.262.1 Detailed Description

VCAP universal value or range.

Definition at line 994 of file types.h.

9.262.2 Field Documentation

9.262.2.1 type

`vtss_vcap_vr_type_t vtss_vcap_vr_t::type`

Type

Definition at line 996 of file types.h.

9.262.2.2 value

`vtss_vcap_vr_value_t vtss_vcap_vr_t::value`

Value

Definition at line 1001 of file types.h.

9.262.2.3 mask

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::mask
```

Mask, cleared bits are wildcards

Definition at line 1002 of file types.h.

9.262.2.4 v

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

type == VTSS_VCAP_VR_TYPE_VALUE_MASK

9.262.2.5 low

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::low
```

Low value

Definition at line 1006 of file types.h.

9.262.2.6 high

```
vtss_vcap_vr_value_t vtss_vcap_vr_t::high
```

High value

Definition at line 1007 of file types.h.

9.262.2.7 r

```
struct { ... } vtss_vcap_vr_t::r
```

type == VTSS_VCAP_VR_TYPE_RANGE_XXXXXX

9.262.2.8 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

9.263 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

9.263.1 Detailed Description

VCE Action.

Definition at line 1006 of file vtss_l2_api.h.

9.263.2 Field Documentation

9.263.2.1 vid

```
vtss_vid_t vtss_vce_action_t::vid
```

Classified VLAN ID

Definition at line 1008 of file vtss_l2_api.h.

9.263.2.2 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

9.264 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

9.264.1 Detailed Description

Frame data for VTSS_VCE_TYPE_ETYPE.

Definition at line 948 of file vtss_l2_api.h.

9.264.2 Field Documentation

9.264.2.1 etype

```
vtss_vcap_u16_t vtss_vce_frame_etype_t::etype
```

Ethernet Type value

Definition at line 950 of file vtss_l2_api.h.

9.264.2.2 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

9.265 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

9.265.1 Detailed Description

Frame data for VTSS_VCE_TYPE_IPV4.

Definition at line 967 of file vtss_l2_api.h.

9.265.2 Field Documentation

9.265.2.1 fragment

```
vtss_vcap_bit_t vtss_vce_frame_ipv4_t::fragment
```

Fragment

Definition at line 969 of file vtss_l2_api.h.

9.265.2.2 options

```
vtss_vcap_bit_t vtss_vce_frame_ipv4_t::options
```

Header options

Definition at line 970 of file vtss_l2_api.h.

9.265.2.3 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.

9.265.2.4 proto

```
vtss_vcap_u8_t vtss_vce_frame_ipv4_t::proto
```

Protocol

Definition at line 972 of file vtss_l2_api.h.

9.265.2.5 sip

```
vtss_vcap_ip_t vtss_vce_frame_ipv4_t::sip
```

Source IP address

Definition at line 973 of file vtss_l2_api.h.

9.265.2.6 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

9.266 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

9.266.1 Detailed Description

Frame data for VTSS_VCE_TYPE_IPV6.

Definition at line 978 of file vtss_l2_api.h.

9.266.2 Field Documentation

9.266.2.1 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.

9.266.2.2 proto

`vtss_vcap_u8_t vtss_vce_frame_ipv6_t::proto`

Protocol

Definition at line 981 of file vtss_l2_api.h.

9.266.2.3 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.

9.266.2.4 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

9.267 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

9.267.1 Detailed Description

Frame data for VTSS_VCE_TYPE_LLC.

Definition at line 955 of file vtss_l2_api.h.

9.267.2 Field Documentation

9.267.2.1 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

9.268 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

9.268.1 Detailed Description

Frame data for VTSS_VCE_TYPE_SNAP.

Definition at line 961 of file vtss_l2_api.h.

9.268.2 Field Documentation

9.268.2.1 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

9.269 vtss_vce_key_t Struct Reference

VCE Key.

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

Data Fields

- BOOL port_list [VTSS_PORT_ARRAY_SIZE]
- vtss_vce_mac_t mac
- vtss_vce_tag_t tag
- vtss_vce_type_t type
- union {
  - vtss_vce_frame_etype_t etype
  - vtss_vce_frame_llc_t llc
  - vtss_vce_frame_snap_t snap
  - vtss_vce_frame_ipv4_t ipv4
  - vtss_vce_frame_ipv6_t ipv6

  } frame

9.269.1 Detailed Description

VCE Key.

Definition at line 987 of file vtss_l2_api.h.

9.269.2 Field Documentation

9.269.2.1 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.

9.269.2.2 mac

```
vtss_vce_mac_t vtss_vce_key_t::mac
```

MAC header

Definition at line 990 of file vtss_l2_api.h.

9.269.2.3 tag

```
vtss_vce_tag_t vtss_vce_key_t::tag
```

Tag

Definition at line 991 of file vtss_l2_api.h.

9.269.2.4 type

```
vtss_vce_type_t vtss_vce_key_t::type
```

VCE frame type

Definition at line 992 of file vtss_l2_api.h.

9.269.2.5 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.

9.269.2.6 llc

`vtss_vce_frame_llc_t vtss_vce_key_t::llc`

VTSS_VCE_TYPE_LLC

Definition at line 998 of file vtss_l2_api.h.

9.269.2.7 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.

9.269.2.8 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.

9.269.2.9 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.

9.269.2.10 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

9.270 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

9.270.1 Detailed Description

VCE MAC header information.

Definition at line 930 of file vtss_l2_api.h.

9.270.2 Field Documentation

9.270.2.1 dmac_mc

```
vtss_vcap_bit_t vtss_vce_mac_t::dmac_mc
```

Multicast DMAC

Definition at line 932 of file vtss_l2_api.h.

9.270.2.2 dmac_bc

```
vtss_vcap_bit_t vtss_vce_mac_t::dmac_bc
```

Broadcast DMAC

Definition at line 933 of file vtss_l2_api.h.

9.270.2.3 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

9.271 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

9.271.1 Detailed Description

VLAN Control Entry.

Definition at line 1013 of file vtss_l2_api.h.

9.271.2 Field Documentation

9.271.2.1 id

```
vtss_vce_id_t vtss_vce_t::id
```

VCE ID

Definition at line 1015 of file vtss_l2_api.h.

9.271.2.2 key

```
vtss_vce_key_t vtss_vce_t::key
```

VCE Key

Definition at line 1016 of file vtss_l2_api.h.

9.271.2.3 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

9.272 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

9.272.1 Detailed Description

VCE tag information.

Definition at line 938 of file vtss_l2_api.h.

9.272.2 Field Documentation

9.272.2.1 vid

`vtss_vcap_vid_t vtss_vce_tag_t::vid`

VLAN ID (12 bit)

Definition at line 940 of file vtss_l2_api.h.

9.272.2.2 pcp

`vtss_vcap_u8_t vtss_vce_tag_t::pcp`

PCP (3 bit)

Definition at line 941 of file vtss_l2_api.h.

9.272.2.3 dei

`vtss_vcap_bit_t vtss_vce_tag_t::dei`

DEI

Definition at line 942 of file vtss_l2_api.h.

9.272.2.4 tagged

`vtss_vcap_bit_t vtss_vce_tag_t::tagged`

Tagged/untagged frame

Definition at line 943 of file vtss_l2_api.h.

9.272.2.5 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

9.273 vtss_vcl_port_conf_t Struct Reference

VCL port configuration.

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

Data Fields

- BOOL dmac_dip

9.273.1 Detailed Description

VCL port configuration.

Definition at line 878 of file vtss_l2_api.h.

9.273.2 Field Documentation

9.273.2.1 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

9.274 vtss_vid_mac_t Struct Reference

MAC Address in specific VLAN.

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

Data Fields

- vtss_vid_t vid
- vtss_mac_t mac

9.274.1 Detailed Description

MAC Address in specific VLAN.

Definition at line 652 of file types.h.

9.274.2 Field Documentation

9.274.2.1 vid

`vtss_vid_t vtss_vid_mac_t::vid`

VLAN ID

Definition at line 654 of file types.h.

9.274.2.2 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

9.275 vtss_vlan_conf_t Struct Reference

VLAN configuration.

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

Data Fields

- vtss_etype_t s_etype

9.275.1 Detailed Description

VLAN configuration.

Definition at line 625 of file vtss_l2_api.h.

9.275.2 Field Documentation

9.275.2.1 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

9.276 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
- vtss_etype_t s_etype

9.276.1 Detailed Description

VLAN port configuration.

Definition at line 664 of file vtss_l2_api.h.

9.276.2 Field Documentation

9.276.2.1 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.

9.276.2.2 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.

9.276.2.3 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.

9.276.2.4 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.

9.276.2.5 ingress_filter

`BOOL vtss_vlan_port_conf_t::ingress_filter`

Ingress filtering

Definition at line 676 of file vtss_l2_api.h.

9.276.2.6 s_etype

```
vtss_etype_t vtss_vlan_port_conf_t::s_etype
```

Alternative S-tag Ethernet Type, if non-zero

Definition at line 678 of file vtss_l2_api.h.

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

- vtss_api/include/vtss_l2_api.h

9.277 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

9.277.1 Detailed Description

VLAN tag with "arbitrary" TPID.

Definition at line 632 of file types.h.

9.277.2 Field Documentation

9.277.2.1 tpid

```
vtss_etype_t vtss_vlan_tag_t::tpid
```

Tag Protocol Identifier

Definition at line 633 of file types.h.

9.277.2.2 pcp

```
vtss_tagprio_t vtss_vlan_tag_t::pcp
```

Priority Code Point

Definition at line 634 of file types.h.

9.277.2.3 dei

```
BOOL vtss_vlan_tag_t::dei
```

Drop Eligible Indicator

Definition at line 635 of file types.h.

9.277.2.4 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

9.278 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

9.278.1 Detailed Description

VLAN translation group-to-VLAN configuration.

Definition at line 1104 of file vtss_l2_api.h.

9.278.2 Field Documentation

9.278.2.1 group_id

```
u16 vtss_vlan_trans_grp2vlan_conf_t::group_id
```

Group ID

Definition at line 1105 of file vtss_l2_api.h.

9.278.2.2 vid

```
vtss_vid_t vtss_vlan_trans_grp2vlan_conf_t::vid
```

VLAN ID

Definition at line 1106 of file vtss_l2_api.h.

9.278.2.3 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

9.279 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]

9.279.1 Detailed Description

VLAN translation port-to-group configuration.

Definition at line 1098 of file vtss_l2_api.h.

9.279.2 Field Documentation

9.279.2.1 group_id

```
u16 vtss_vlan_trans_port2grp_conf_t::group_id
```

Group ID

Definition at line 1099 of file vtss_l2_api.h.

9.279.2.2 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

9.280 vtss_vlan_vid_conf_t Struct Reference

VLAN ID configuration.

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

Data Fields

- BOOL learning
- BOOL mirror
- vtss_vid_t fid

9.280.1 Detailed Description

VLAN ID configuration.

Definition at line 740 of file vtss_l2_api.h.

9.280.2 Field Documentation

9.280.2.1 learning

```
BOOL vtss_vlan_vid_conf_t::learning
```

Enable/disable learning

Definition at line 742 of file vtss_l2_api.h.

9.280.2.2 mirror

```
BOOL vtss_vlan_vid_conf_t::mirror
```

Enable/disable mirroring

Definition at line 743 of file vtss_l2_api.h.

9.280.2.3 fid

```
vtss_vid_t vtss_vlan_vid_conf_t::fid
```

Forwarding ID for SVL/IVL control

Definition at line 745 of file vtss_l2_api.h.

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

- vtss_api/include/vtss_l2_api.h

9.281 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]

9.281.1 Detailed Description

Structure for Wake-On-LAN MAC Address.

Definition at line 1656 of file vtss_phy_api.h.

9.281.2 Field Documentation

9.281.2.1 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 10

File Documentation

10.1 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_sflow_type_t { VTSS_SFLOW_TYPE_NONE = 0, VTSS_SFLOW_TYPE_RX, VTSS_SFLOW_TYPE_TX, VTSS_SFLOW_TYPE_ALL }

  sFlow sampler type.

10.1.1 Detailed Description

Layer 2 Public API Header for l2.

This header file describes public Layer 2 datatypes

10.1.2 Enumeration Type Documentation

10.1.2.1 vtss_sflow_type_t

```
enum vtss_sflow_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.

10.2 vtss_api/include/vtss/api/options.h File Reference

Features and options.

Macros

- #define VTSS_ARCH_SERVAL
- #define VTSS_ARCH_SERVAL_CE
- #define VTSS_ARCH_SERVAL_CPU
- #define VTSS_FEATURE_WARM_START
- #define VTSS_FEATURE_MISC
- #define VTSS_FEATURE_PORT_CONTROL
- #define VTSS_FEATURE_PORT_IFH
- #define VTSS_FEATURE_CLAUSE_37
- #define VTSS_FEATURE_EXC_COL_CONT
- #define VTSS_FEATURE_PORT_CNT_BRIDGE
- #define VTSS_FEATURE_PFC
- #define VTSS_FEATURE_QOS
- #define VTSS_FEATURE_QCL
- #define VTSS_FEATURE_QCL_V2
- #define VTSS_FEATURE_QCL_DMAC_DIP
- #define VTSS_FEATURE_QCL_KEY_TYPE
- #define VTSS_FEATURE_QCL_KEY_S_TAG
- #define VTSS_FEATURE_QCL_KEY_INNER_TAG
- #define VTSS_FEATURE_QCL_KEY_DMAC
- #define VTSS_FEATURE_QCL_KEY_DIP
- #define VTSS_FEATURE_QCL_PCP_DEI_ACTION
- #define VTSS_FEATURE_QCL_POLICY_ACTION
- #define VTSS_FEATURE_QOS_POLICER_UC_SWITCH
- #define VTSS_FEATURE_QOS_POLICER_MC_SWITCH
- #define VTSS_FEATURE_QOS_POLICER_BC_SWITCH
- #define VTSS_FEATURE_QOS_PORT_POLICER_EXT
- #define VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS
- #define VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC
- #define VTSS_FEATURE_QOS_QUEUE_TX
- #define VTSS_FEATURE_QOS_QUEUE_POLICER
- #define VTSS_FEATURE_QOS_SCHEDULER_V2
- #define VTSS_FEATURE_QOS_SCHEDULER_DWRR_CNT
- #define VTSS_FEATURE_QOS_TAG_REMARK_V2
- #define VTSS_FEATURE_QOS_CLASSIFICATION_V2
- #define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS

- #define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB
- #define VTSS_FEATURE_QOS_EGRESS_SHAPERS_DLB
- #define VTSS_FEATURE_QOS_EGRESS_SHAPERS_RT
- #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_QOS_WRED_V2
- #define VTSS_FEATURE_QOS_POLICER_DLB
- #define VTSS_FEATURE_QOS_CPU_PORT_SHAPER
- #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_VLAN_SVL
- #define VTSS_FEATURE_MAC_AGE_AUTO
- #define VTSS_FEATURE_MAC_CPU_QUEUE
- #define VTSS_FEATURE_EEE
- #define VTSS_FEATURE_VCAP
- #define VTSS_FEATURE_ACL
- #define VTSS_FEATURE_ACL_V2
- #define VTSS_FEATURE_VCL
- #define VTSS_FEATURE_NPI
- #define VTSS_FEATURE_LED_POW_REDUC
- #define VTSS_FEATURE_VLAN_TRANSLATION
- #define VTSS_FEATURE_SFLOW
- #define VTSS_FEATURE_MIRROR_CPU
- #define VTSS_FEATURE_IRQ_CONTROL
- #define VTSS_FEATURE_SERDES_MACRO_SETTINGS
- #define TESLA_ING_TS_ERRFIX
- #define VIPER_B_FIFO_RESET
- #define VTSS_TS_FIFO_SYNC_LOOPBACK
- #define VTSS_TS_FIFO_MEDIA_SWAP_SYNC
- #define VTSS_PHY_TS_SPI_CLK_THRU_PPS0
- #define VTSS_FEATURE_INTERRUPTS
- #define VTSS_FEATURE_SERDES_MACRO_SETTINGS
- #define VTSS_FEATURE_SYNCE
- #define VTSS_FEATURE_SERIAL_GPIO
- #define VTSS_FEATURE_FAN
- #define VTSS_FEATURE_PHY_TIMESTAMP
- #define VTSS_FEATURE_TIMESTAMP
- #define VTSS_FEATURE_TIMESTAMP_ONE_STEP
- #define VTSS_FEATURE_TIMESTAMP_LATENCY_COMP
- #define VTSS_FEATURE_TIMESTAMP_ORG_TIME
- #define VTSS_FEATURE_TIMESTAMP_P2P_DELAY_COMP
- #define VTSS_FEATURE_TIMESTAMP_ASYMMETRY_COMP
- #define VTSS_FEATURE_PTP_RS422
- #define VTSS_OPT_VCORE_III 1
- #define VTSS_FEATURE_FDMA
- #define VTSS_FEATURE_EVC

- #define VTSS_FEATURE_E_TREE
- #define VTSS_FEATURE_OAM
- #define VTSS_FEATURE_AFI_SWC
- #define VTSS_AFI_V1
- #define VTSS_FEATURE_MPLS
- #define VTSS_FEATURE_HQOS
- #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_VERIPHY 1
- #define VTSS_FEATURE_WARM_START
- #define VTSS_FEATURE_VCAP

10.2.1 Detailed Description

Features and options.

This header file describes target features and compile-time options

10.2.2 Macro Definition Documentation

10.2.2.1 VTSS_ARCH_SERVAL

```
#define VTSS_ARCH_SERVAL
```

Serval architecture

Definition at line 415 of file options.h.

10.2.2.2 VTSS_ARCH_SERVAL_CE

```
#define VTSS_ARCH_SERVAL_CE
```

Serval CE architecture

Definition at line 416 of file options.h.

10.2.2.3 VTSS_ARCH_SERVAL_CPU

```
#define VTSS_ARCH_SERVAL_CPU
```

Serval CPU system architecture

Definition at line 417 of file options.h.

10.2.2.4 VTSS_FEATURE_WARM_START [1/2]

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 617 of file options.h.

10.2.2.5 VTSS_FEATURE_MISC

```
#define VTSS_FEATURE_MISC
```

Miscellaneous

Definition at line 428 of file options.h.

10.2.2.6 VTSS_FEATURE_PORT_CONTROL

```
#define VTSS_FEATURE_PORT_CONTROL
```

Port control

Definition at line 429 of file options.h.

10.2.2.7 VTSS_FEATURE_PORT_IFH

```
#define VTSS_FEATURE_PORT_IFH
```

Port IFH control

Definition at line 430 of file options.h.

10.2.2.8 VTSS_FEATURE_CLAUSE_37

```
#define VTSS_FEATURE_CLAUSE_37
```

IEEE 802.3 clause 37 auto-negotiation

Definition at line 431 of file options.h.

10.2.2.9 VTSS_FEATURE_EXC_COL_CONT

```
#define VTSS_FEATURE_EXC_COL_CONT
```

Excessive collision continuation

Definition at line 432 of file options.h.

10.2.2.10 VTSS_FEATURE_PORT_CNT_BRIDGE

```
#define VTSS_FEATURE_PORT_CNT_BRIDGE
```

Bridge counters

Definition at line 433 of file options.h.

10.2.2.11 VTSS_FEATURE_PFC

```
#define VTSS_FEATURE_PFC
```

802.1Qbb Priority Flow Control

Definition at line 434 of file options.h.

10.2.2.12 VTSS_FEATURE_QOS

```
#define VTSS_FEATURE_QOS
```

QoS

Definition at line 435 of file options.h.

10.2.2.13 VTSS_FEATURE_QCL

```
#define VTSS_FEATURE_QCL
```

QoS: QoS Control Lists

Definition at line 436 of file options.h.

10.2.2.14 VTSS_FEATURE_QCL_V2

```
#define VTSS_FEATURE_QCL_V2
```

QoS: QoS Control Lists, V2 features

Definition at line 437 of file options.h.

10.2.2.15 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 438 of file options.h.

10.2.2.16 VTSS_FEATURE_QCL_KEY_TYPE

```
#define VTSS_FEATURE_QCL_KEY_TYPE
```

QoS: QoS Control Lists, different key types per port

Definition at line 439 of file options.h.

10.2.2.17 VTSS_FEATURE_QCL_KEY_S_TAG

```
#define VTSS_FEATURE_QCL_KEY_S_TAG
```

QoS: QoS Control Lists has S tag support

Definition at line 440 of file options.h.

10.2.2.18 VTSS_FEATURE_QCL_KEY_INNER_TAG

```
#define VTSS_FEATURE_QCL_KEY_INNER_TAG
```

QoS: QoS Control Lists has inner tag

Definition at line 441 of file options.h.

10.2.2.19 VTSS_FEATURE_QCL_KEY_DMAC

```
#define VTSS_FEATURE_QCL_KEY_DMAC
```

QoS: QoS Control Lists has destination MAC address

Definition at line 442 of file options.h.

10.2.2.20 VTSS_FEATURE_QCL_KEY_DIP

```
#define VTSS_FEATURE_QCL_KEY_DIP
```

QoS: QoS Control Lists has destination IP address

Definition at line 443 of file options.h.

10.2.2.21 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 444 of file options.h.

10.2.2.22 VTSS_FEATURE_QCL_POLICY_ACTION

```
#define VTSS_FEATURE_QCL_POLICY_ACTION
```

QoS: QoS Control Lists has policy action

Definition at line 445 of file options.h.

10.2.2.23 VTSS_FEATURE_QOS_POLICER_UC_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_UC_SWITCH
```

QoS: Unicast policer per switch

Definition at line 446 of file options.h.

10.2.2.24 VTSS_FEATURE_QOS_POLICER_MC_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_MC_SWITCH
```

QoS: Multicast policer per switch

Definition at line 447 of file options.h.

10.2.2.25 VTSS_FEATURE_QOS_POLICER_BC_SWITCH

```
#define VTSS_FEATURE_QOS_POLICER_BC_SWITCH
```

QoS: Broadcast policer per switch

Definition at line 448 of file options.h.

10.2.2.26 VTSS_FEATURE_QOS_PORT_POLICER_EXT

```
#define VTSS_FEATURE_QOS_PORT_POLICER_EXT
```

QoS: Port Policer Extensions

Definition at line 449 of file options.h.

10.2.2.27 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 450 of file options.h.

10.2.2.28 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 451 of file options.h.

10.2.2.29 VTSS_FEATURE_QOS_QUEUE_TX

```
#define VTSS_FEATURE_QOS_QUEUE_TX
```

QoS: Has TX Queue support

Definition at line 452 of file options.h.

10.2.2.30 VTSS_FEATURE_QOS_QUEUE_POLICER

```
#define VTSS_FEATURE_QOS_QUEUE_POLICER
```

QoS: Has Ingress Queue Policers

Definition at line 453 of file options.h.

10.2.2.31 VTSS_FEATURE_QOS_SCHEDULER_V2

```
#define VTSS_FEATURE_QOS_SCHEDULER_V2
```

QoS: 2. version of scheduler

Definition at line 454 of file options.h.

10.2.2.32 VTSS_FEATURE_QOS_SCHEDULER_DWRR_CNT

```
#define VTSS_FEATURE_QOS_SCHEDULER_DWRR_CNT
```

QoS: Scheduler supports variable number of DWRR inputs

Definition at line 455 of file options.h.

10.2.2.33 VTSS_FEATURE_QOS_TAG_REMARK_V2

```
#define VTSS_FEATURE_QOS_TAG_REMARK_V2
```

QoS: 2. version of tag priority remarking

Definition at line 456 of file options.h.

10.2.2.34 VTSS_FEATURE_QOS_CLASSIFICATION_V2

```
#define VTSS_FEATURE_QOS_CLASSIFICATION_V2
```

QoS: 2. version of classification

Definition at line 457 of file options.h.

10.2.2.35 VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS

```
#define VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS
```

QoS: Has Egress Queue Shapers

Definition at line 458 of file options.h.

10.2.2.36 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 459 of file options.h.

10.2.2.37 VTSS_FEATURE_QOS_EGRESS_SHAPERS_DLB

```
#define VTSS_FEATURE_QOS_EGRESS_SHAPERS_DLB
```

QoS: Egress shapers has DLB support

Definition at line 460 of file options.h.

10.2.2.38 VTSS_FEATURE_QOS_EGRESS_SHAPERS_RT

```
#define VTSS_FEATURE_QOS_EGRESS_SHAPERS_RT
```

Qos: Egress shapers have rate type support - line or date rate

Definition at line 461 of file options.h.

10.2.2.39 VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE

```
#define VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE
```

QoS: DSCP classification is DP aware

Definition at line 462 of file options.h.

10.2.2.40 VTSS_FEATURE_QOS_DSCP_REMARK

```
#define VTSS_FEATURE_QOS_DSCP_REMARK
```

QoS: Has DSCP remarking

Definition at line 463 of file options.h.

10.2.2.41 VTSS_FEATURE_QOS_DSCP_REMARK_V2

```
#define VTSS_FEATURE_QOS_DSCP_REMARK_V2
```

QoS: 2. version of DSCP remarking

Definition at line 464 of file options.h.

10.2.2.42 VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWARE

```
#define VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWARE
```

QoS: DSCP remarking is DP aware

Definition at line 465 of file options.h.

10.2.2.43 VTSS_FEATURE_QOS_WRED_V2

```
#define VTSS_FEATURE_QOS_WRED_V2
```

QoS: WRED global - per queue (0..7), per dpl (0..1)

Definition at line 466 of file options.h.

10.2.2.44 VTSS_FEATURE_QOS_POLICER_DLB

```
#define VTSS_FEATURE_QOS_POLICER_DLB
```

DLB policers

Definition at line 467 of file options.h.

10.2.2.45 VTSS_FEATURE_QOS_CPU_PORT_SHAPER

```
#define VTSS_FEATURE_QOS_CPU_PORT_SHAPER
```

QoS: Has CPU port shaper

Definition at line 468 of file options.h.

10.2.2.46 VTSS_FEATURE_PACKET

```
#define VTSS_FEATURE_PACKET
```

CPU Rx/Tx frame configuration

Definition at line 469 of file options.h.

10.2.2.47 VTSS_FEATURE_PACKET_TX

```
#define VTSS_FEATURE_PACKET_TX
```

CPU Tx frame

Definition at line 470 of file options.h.

10.2.2.48 VTSS_FEATURE_PACKET_RX

```
#define VTSS_FEATURE_PACKET_RX
```

CPU Rx frame

Definition at line 471 of file options.h.

10.2.2.49 VTSS_FEATURE_PACKET_GROUPING

```
#define VTSS_FEATURE_PACKET_GROUPING
```

Extraction and injection occurs through extraction and injection groups rather than queues.

Definition at line 472 of file options.h.

10.2.2.50 VTSS_FEATURE_PACKET_PORT_REG

```
#define VTSS_FEATURE_PACKET_PORT_REG
```

Packet registration per port

Definition at line 473 of file options.h.

10.2.2.51 VTSS_FEATURE_LAYER2

```
#define VTSS_FEATURE_LAYER2
```

Layer 2 (switching)

Definition at line 474 of file options.h.

10.2.2.52 VTSS_FEATURE_PVLAN

```
#define VTSS_FEATURE_PVLAN
```

Private VLANs

Definition at line 475 of file options.h.

10.2.2.53 VTSS_FEATURE_VLAN_PORT_V2

```
#define VTSS_FEATURE_VLAN_PORT_V2
```

VLAN port configuration, V2 features

Definition at line 476 of file options.h.

10.2.2.54 VTSS_FEATURE_VLAN_TX_TAG

```
#define VTSS_FEATURE_VLAN_TX_TAG
```

VLAN tagging per (VID, port)

Definition at line 477 of file options.h.

10.2.2.55 VTSS_FEATURE_VLAN_SVL

```
#define VTSS_FEATURE_VLAN_SVL
```

Shared VLAN Learning

Definition at line 478 of file options.h.

10.2.2.56 VTSS_FEATURE_MAC_AGE_AUTO

```
#define VTSS_FEATURE_MAC_AGE_AUTO
```

Automatic MAC address ageing

Definition at line 479 of file options.h.

10.2.2.57 VTSS_FEATURE_MAC_CPU_QUEUE

```
#define VTSS_FEATURE_MAC_CPU_QUEUE
```

CPU queue per MAC address

Definition at line 480 of file options.h.

10.2.2.58 VTSS_FEATURE_EEE

```
#define VTSS_FEATURE_EEE
```

Energy Efficient Ethernet

Definition at line 481 of file options.h.

10.2.2.59 VTSS_FEATURE_VCAP [1/2]

```
#define VTSS_FEATURE_VCAP
```

VCAP

Definition at line 637 of file options.h.

10.2.2.60 VTSS_FEATURE_ACL

```
#define VTSS_FEATURE_ACL
```

Access Control Lists

Definition at line 483 of file options.h.

10.2.2.61 VTSS_FEATURE_ACL_V2

```
#define VTSS_FEATURE_ACL_V2
```

Access Control Lists, V2 features

Definition at line 484 of file options.h.

10.2.2.62 VTSS_FEATURE_VCL

```
#define VTSS_FEATURE_VCL
```

VLAN Control Lists

Definition at line 485 of file options.h.

10.2.2.63 VTSS_FEATURE_NPI

```
#define VTSS_FEATURE_NPI
```

NPI port

Definition at line 486 of file options.h.

10.2.2.64 VTSS_FEATURE_LED_POW_REDUC

```
#define VTSS_FEATURE_LED_POW_REDUC
```

LED power reduction

Definition at line 487 of file options.h.

10.2.2.65 VTSS_FEATURE_VLAN_TRANSLATION

```
#define VTSS_FEATURE_VLAN_TRANSLATION
```

VLAN Translation

Definition at line 488 of file options.h.

10.2.2.66 VTSS_FEATURE_SFLOW

```
#define VTSS_FEATURE_SFLOW
```

sFlow feature

Definition at line 489 of file options.h.

10.2.2.67 VTSS_FEATURE_MIRROR_CPU

```
#define VTSS_FEATURE_MIRROR_CPU
```

CPU mirroring

Definition at line 490 of file options.h.

10.2.2.68 VTSS_FEATURE_IRQ_CONTROL

```
#define VTSS_FEATURE_IRQ_CONTROL
```

General IRQ support

Definition at line 491 of file options.h.

10.2.2.69 VTSS_FEATURE_SERDES_MACRO_SETTINGS [1/2]

```
#define VTSS_FEATURE_SERDES_MACRO_SETTINGS
```

Hooks for Serdes Macro configuration

Definition at line 504 of file options.h.

10.2.2.70 TESLA_ING_TS_ERRFIX

```
#define TESLA_ING_TS_ERRFIX
```

PHY Timestamp FIFO out of sync Support

Definition at line 493 of file options.h.

10.2.2.71 VIPER_B_FIFO_RESET

```
#define VIPER_B_FIFO_RESET
```

Viper B 1588 FIFO sync

Definition at line 494 of file options.h.

10.2.2.72 VTSS_TS_FIFO_SYNC_LOOPBACK

```
#define VTSS_TS_FIFO_SYNC_LOOPBACK
```

PHY Timestamp FIFO OOS Check, Run Automatically if FE Looback transition to OFF

Definition at line 497 of file options.h.

10.2.2.73 VTSS_TS_FIFO_MEDIA_SWAP_SYNC

```
#define VTSS_TS_FIFO_MEDIA_SWAP_SYNC
```

PHY Timestamp FIFO OOS CHECK for correction of TSFIFO, in MEDIA i/f Swap case

Definition at line 498 of file options.h.

10.2.2.74 VTSS_PHY_TS_SPI_CLK_THRU_PPS0

```
#define VTSS_PHY_TS_SPI_CLK_THRU_PPS0
```

Use 1588_PPS0 as New SPI_CLK, applicable only to 8574-15; old SPI_CLK pin will not be used anymore

Definition at line 499 of file options.h.

10.2.2.75 VTSS_FEATURE_INTERRUPTS

```
#define VTSS_FEATURE_INTERRUPTS
```

Port Interrupt support

Definition at line 503 of file options.h.

10.2.2.76 VTSS_FEATURE_SERDES_MACRO_SETTINGS [2/2]

```
#define VTSS_FEATURE_SERDES_MACRO_SETTINGS
```

Hooks for Serdes Macro configuration

Definition at line 504 of file options.h.

10.2.2.77 VTSS_FEATURE_SYNCE

```
#define VTSS_FEATURE_SYNCE
```

SYNCE - L1 syncronization feature

Definition at line 505 of file options.h.

10.2.2.78 VTSS_FEATURE_SERIAL_GPIO

```
#define VTSS_FEATURE_SERIAL_GPIO
```

Serial GPIO control

Definition at line 506 of file options.h.

10.2.2.79 VTSS_FEATURE_FAN

```
#define VTSS_FEATURE_FAN
```

Fan control

Definition at line 507 of file options.h.

10.2.2.80 VTSS_FEATURE_PHY_TIMESTAMP

```
#define VTSS_FEATURE_PHY_TIMESTAMP
```

PHY timestamp feature (for PTP/OAM)

Definition at line 508 of file options.h.

10.2.2.81 VTSS_FEATURE_TIMESTAMP

```
#define VTSS_FEATURE_TIMESTAMP
```

Packet timestamp feature (for PTP and OAM)

Definition at line 509 of file options.h.

10.2.2.82 VTSS_FEATURE_TIMESTAMP_ONE_STEP

```
#define VTSS_FEATURE_TIMESTAMP_ONE_STEP
```

ONESTEP timestamp hardware support

Definition at line 510 of file options.h.

10.2.2.83 VTSS_FEATURE_TIMESTAMP_LATENCY_COMP

```
#define VTSS_FEATURE_TIMESTAMP_LATENCY_COMP
```

Ingress and egress latency compensation hardwarce support

Definition at line 511 of file options.h.

10.2.2.84 VTSS_FEATURE_TIMESTAMP_ORG_TIME

```
#define VTSS_FEATURE_TIMESTAMP_ORG_TIME
```

OriginTimestamp update hardware support

Definition at line 512 of file options.h.

10.2.2.85 VTSS_FEATURE_TIMESTAMP_P2P_DELAY_COMP

```
#define VTSS_FEATURE_TIMESTAMP_P2P_DELAY_COMP
```

Peer-to-peer path delay compensation hardware support

Definition at line 513 of file options.h.

10.2.2.86 VTSS_FEATURE_TIMESTAMP_ASYMMETRY_COMP

```
#define VTSS_FEATURE_TIMESTAMP_ASYMMETRY_COMP
```

Path delay asymmetry compensation hardware support

Definition at line 514 of file options.h.

10.2.2.87 VTSS_FEATURE_PTP_RS422

```
#define VTSS_FEATURE_PTP_RS422
```

Support for the RS422 serial/1PPS interface

Definition at line 515 of file options.h.

10.2.2.88 VTSS_OPT_VCORE_III

```
#define VTSS_OPT_VCORE_III 1
```

Internal VCore-III (MIPS) CPU enabled by default

Definition at line 517 of file options.h.

10.2.2.89 VTSS_FEATURE_FDMA

```
#define VTSS_FEATURE_FDMA
```

Frame DMA

Definition at line 520 of file options.h.

10.2.2.90 VTSS_FEATURE_EVC

```
#define VTSS_FEATURE_EVC
```

Ethernet Virtual Connections

Definition at line 525 of file options.h.

10.2.2.91 VTSS_FEATURE_E_TREE

```
#define VTSS_FEATURE_E_TREE
```

EVC E-Tree

Definition at line 526 of file options.h.

10.2.2.92 VTSS_FEATURE_OAM

```
#define VTSS_FEATURE_OAM
```

Y.1731/IEEE802.1ag OAM

Definition at line 527 of file options.h.

10.2.2.93 VTSS_FEATURE_AFI_SWC

```
#define VTSS_FEATURE_AFI_SWC
```

Switch-core-based Automatic Frame Injection

Definition at line 528 of file options.h.

10.2.2.94 VTSS_AFI_V1

```
#define VTSS_AFI_V1
```

AFI API version 1

Definition at line 529 of file options.h.

10.2.2.95 VTSS_FEATURE_MPLS

```
#define VTSS_FEATURE_MPLS
```

MPLS / MPLS-TP

Definition at line 534 of file options.h.

10.2.2.96 VTSS_FEATURE_HQOS

```
#define VTSS_FEATURE_HQOS
```

Hierarchical Quality of Service

Definition at line 535 of file options.h.

10.2.2.97 VTSS_CHIP_CU_PHY

```
#define VTSS_CHIP_CU_PHY
```

Cobber PHY chip

Definition at line 540 of file options.h.

10.2.2.98 VTSS_OPT_TRACE

```
#define VTSS_OPT_TRACE 1
```

Trace enabled by default

Definition at line 548 of file options.h.

10.2.2.99 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.

10.2.2.100 VTSS_OPT_FDMA_DEBUG

```
#define VTSS_OPT_FDMA_DEBUG 0
```

FDMA debug disabled by default

Definition at line 599 of file options.h.

10.2.2.101 VTSS_OPT_VAUI_EQ_CTRL

```
#define VTSS_OPT_VAUI_EQ_CTRL 6
```

Default equalization control

Definition at line 605 of file options.h.

10.2.2.102 VTSS_OPT_PORT_COUNT

```
#define VTSS_OPT_PORT_COUNT 0
```

Use all target ports by default

Definition at line 609 of file options.h.

10.2.2.103 VTSS_PHY_OPT_VERIPHY

```
#define VTSS_PHY_OPT_VERIPHY 1
```

VeriPHY enabled by default

Definition at line 613 of file options.h.

10.2.2.104 VTSS_FEATURE_WARM_START [2/2]

```
#define VTSS_FEATURE_WARM_START
```

Warm start

Definition at line 617 of file options.h.

10.2.2.105 VTSS_FEATURE_VCAP [2/2]

```
#define VTSS_FEATURE_VCAP
```

VCAP

Definition at line 637 of file options.h.

10.3 vtss_api/include/vtss/api/phy.h File Reference

PHY Public API Header.

```
#include <vtss/api/types.h>
```

Macros

- #define VTSS_PHY_POWER_ACTIPHY_BIT 0
- #define VTSS_PHY_POWER_DYNAMIC_BIT 1

Enumerations

- enum vtss_phy_power_mode_t { VTSS_PHY_POWER_NOMINAL = 0, VTSS_PHY_POWER_ACTIPHY = 1 << VTSS_PHY_POWER_ACTIPHY_BIT, VTSS_PHY_POWER_DYNAMIC = 1 << VTSS_PHY_POWE↩R_DYNAMIC_BIT, VTSS_PHY_POWER_ENABLED = VTSS_PHY_POWER_ACTIPHY + VTSS_PHY_P↩OWER_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 = 14, VTSS_VERIPHY_STATUS_COUPL_D = 15, VTSS_VERIPHY_STATUS_UNKNOWN = 16, VTSS_VERIPHY_STATUS_RUNNING = 17 }

  VeriPHY status.

10.3.1 Detailed Description

PHY Public API Header.

This header file describes public PHY data-types

10.3.2 Macro Definition Documentation

10.3.2.1 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.

10.3.2.2 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.

10.3.3 Enumeration Type Documentation

10.3.3.1 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.

10.3.3.2 vtss_phy_veriphy_status_t

```
enum vtss_phy_veriphy_status_t
```

VeriPHY status.

Enumerator

VTSS_VERIPHY_STATUS_OK	Correctly terminated pair
VTSS_VERIPHY_STATUS_OPEN	Open pair
VTSS_VERIPHY_STATUS_SHORT	Short pair
VTSS_VERIPHY_STATUS_ABNORM	Abnormal termination
VTSS_VERIPHY_STATUS_SHORT_A	Cross-pair short to pair A
VTSS_VERIPHY_STATUS_SHORT_B	Cross-pair short to pair B
VTSS_VERIPHY_STATUS_SHORT_C	Cross-pair short to pair C
VTSS_VERIPHY_STATUS_SHORT_D	Cross-pair short to pair D
VTSS_VERIPHY_STATUS_COUPL_A	Abnormal cross-pair coupling, pair A
VTSS_VERIPHY_STATUS_COUPL_B	Abnormal cross-pair coupling, pair B
VTSS_VERIPHY_STATUS_COUPL_C	Abnormal cross-pair coupling, pair C
VTSS_VERIPHY_STATUS_COUPL_D	Abnormal cross-pair coupling, pair D
VTSS_VERIPHY_STATUS_UNKNOWN	Unknown - VeriPhy never started ?
VTSS_VERIPHY_STATUS_RUNNING	VeriPhy is still running - No result yet

Definition at line 54 of file phy.h.

10.4 vtss_api/include/vtss/api/port.h File Reference

Port Public API Header.

```
#include <vtss/api/types.h>
#include <vtss/api/phy.h>
```

Data Structures

- struct vtss_port_rmon_counters_t

  RMON counter structure (RFC 2819)
- struct vtss_port_if_group_counters_t

  Interfaces Group counter structure (RFC 2863)
- struct vtss_port_ethernet_like_counters_t

  Ethernet-like Interface counter structure (RFC 3635)
- struct vtss_port_bridge_counters_t

  Port bridge counter structure (RFC 4188)
- struct vtss_port_proprietary_counters_t

  Port proprietary counter structure.

- struct vtss_port_counters_t

  Port counter structure.
- struct port_custom_conf_t

  Port configuration.
- struct vtss_port_status_t

  Port status parameter struct.

Macros

- #define PORT_CAP_NONE 0x00000000
- #define PORT_CAP_AUTONEG 0x00000001
- #define PORT_CAP_10M_HDX 0x00000002
- #define PORT_CAP_10M_FDX 0x00000004
- #define PORT_CAP_100M_HDX 0x00000008
- #define PORT_CAP_100M_FDX 0x00000010
- #define PORT_CAP_1G_FDX 0x00000020
- #define PORT_CAP_2_5G_FDX 0x00000040
- #define PORT_CAP_5G_FDX 0x00000080
- #define PORT_CAP_10G_FDX 0x00000100
- #define PORT_CAP_FLOW_CTRL 0x00001000
- #define PORT_CAP_COPPER 0x00002000
- #define PORT_CAP_FIBER 0x00004000
- #define PORT_CAP_DUAL_COPPER 0x00008000
- #define PORT_CAP_DUAL_FIBER 0x00010000
- #define PORT_CAP_SD_ENABLE 0x00020000
- #define PORT_CAP_SD_HIGH 0x00040000
- #define PORT_CAP_SD_INTERNAL 0x00080000
- #define PORT_CAP_DUAL_FIBER_100FX 0x00100000
- #define PORT_CAP_XAUI_LANE_FLIP 0x00200000
- #define PORT_CAP_VTSS_10G_PHY 0x00400000
- #define PORT_CAP_SFP_DETECT 0x00800000
- #define PORT_CAP_STACKING 0x01000000
- #define PORT_CAP_DUAL_SFP_DETECT 0x02000000
- #define PORT_CAP_SFP_ONLY 0x04000000
- #define PORT_CAP_DUAL_COPPER_100FX 0x08000000
- #define PORT_CAP_HDX (PORT_CAP_10M_HDX | PORT_CAP_100M_HDX)
- #define PORT_CAP_TRI_SPEED_FDX (PORT_CAP_AUTONEG | PORT_CAP_1G_FDX | PORT_CAP_100M_FDX | PORT_CAP_10M_FDX | PORT_CAP_FLOW_CTRL)
- #define PORT_CAP_TRI_SPEED (PORT_CAP_TRI_SPEED_FDX | PORT_CAP_HDX)
- #define PORT_CAP_1G_PHY (PORT_CAP_COPPER | PORT_CAP_FIBER | PORT_CAP_DUAL_COPPER | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_FIBER_100FX)
- #define PORT_CAP_TRI_SPEED_COPPER (PORT_CAP_TRI_SPEED | PORT_CAP_COPPER)
- #define PORT_CAP_TRI_SPEED_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_FIBER)
- #define PORT_CAP_TRI_SPEED_DUAL_COPPER (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_COPPER)
- #define PORT_CAP_TRI_SPEED_DUAL_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_FIBER)
- #define PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX (PORT_CAP_TRI_SPEED | PORT_CAP_DUAL_FIBER_100FX)
- #define PORT_CAP_ANY_FIBER (PORT_CAP_FIBER | PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER | PORT_CAP_SFP_DETECT)
- #define PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED (PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER)
- #define PORT_CAP_SPEED_DUAL_ANY_FIBER (PORT_CAP_DUAL_COPPER | PORT_CAP_DUAL_FIBER| PORT_CAP_DUAL_FIBER_100FX | PORT_CAP_DUAL_SFP_DETECT)
- #define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER

- #define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXED_SFP_SPEED (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_SPEED)
- #define PORT_CAP_DUAL_FIBER_1000X (PORT_CAP_DUAL_FIBER | PORT_CAP_DUAL_COPPER)
- #define PORT_CAP_SFP_1G (PORT_CAP_AUTONEG | PORT_CAP_100M_FDX | PORT_CAP_1G_FDX | PORT_CAP_FLOW_CTRL | PORT_CAP_SFP_ONLY)
- #define PORT_CAP_SFP_2_5G (PORT_CAP_SFP_1G | PORT_CAP_2_5G_FDX)
- #define PORT_CAP_SFP_SD_HIGH (PORT_CAP_SD_ENABLE | PORT_CAP_SD_HIGH | PORT_CAP_SD_INTERNAL | PORT_CAP_SFP_DETECT | PORT_CAP_SFP_ONLY)
- #define PORT_CAP_2_5G_TRI_SPEED_FDX (PORT_CAP_AUTONEG | PORT_CAP_2_5G_FDX | PORT_CAP_1G_FDX | PORT_CAP_100M_FDX | PORT_CAP_FLOW_CTRL)
- #define PORT_CAP_2_5G_TRI_SPEED (PORT_CAP_2_5G_TRI_SPEED_FDX | PORT_CAP_100M_HDX)
- #define PORT_CAP_2_5G_TRI_SPEED_COPPER (PORT_CAP_2_5G_TRI_SPEED | PORT_CAP_COPPER)

Typedefs

- typedef u32 port_cap_t
- typedef u64 vtss_port_counter_t

  Counter type.

Enumerations

- enum vtss_port_speed_t {
  VTSS_SPEED_UNDEFINED, VTSS_SPEED_10M, VTSS_SPEED_100M, VTSS_SPEED_1G, VTSS_SPEED_2500M, VTSS_SPEED_5G, VTSS_SPEED_10G, VTSS_SPEED_12G }

  Port speed.
- enum vtss_fiber_port_speed_t {
  VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_DISABLED = 0, VTSS_SPEED_FIBER_100FX = 2, VTSS_SPEED_FIBER_1000X = 3, VTSS_SPEED_FIBER_AUTO = 4, VTSS_SPEED_FIBER_DISABLED = 5 }

  Fiber Port speed.

10.4.1 Detailed Description

Port Public API Header.

This header file describes public port data-types

10.4.2 Macro Definition Documentation

10.4.2.1 PORT_CAP_NONE

```
#define PORT_CAP_NONE 0x00000000
```

No capabilities

Definition at line 46 of file port.h.

10.4.2.2 PORT_CAP_AUTONEG

```
#define PORT_CAP_AUTONEG 0x00000001
```

Auto negotiation

Definition at line 47 of file port.h.

10.4.2.3 PORT_CAP_10M_HDX

```
#define PORT_CAP_10M_HDX 0x00000002
```

10 Mbps, half duplex

Definition at line 48 of file port.h.

10.4.2.4 PORT_CAP_10M_FDX

```
#define PORT_CAP_10M_FDX 0x00000004
```

10 Mbps, full duplex

Definition at line 49 of file port.h.

10.4.2.5 PORT_CAP_100M_HDX

```
#define PORT_CAP_100M_HDX 0x00000008
```

100 Mbps, half duplex

Definition at line 50 of file port.h.

10.4.2.6 PORT_CAP_100M_FDX

```
#define PORT_CAP_100M_FDX 0x00000010
```

100 Mbps, full duplex

Definition at line 51 of file port.h.

10.4.2.7 PORT_CAP_1G_FDX

```
#define PORT_CAP_1G_FDX 0x00000020
```

1 Gbps, full duplex

Definition at line 52 of file port.h.

10.4.2.8 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.

10.4.2.9 PORT_CAP_5G_FDX

```
#define PORT_CAP_5G_FDX 0x00000080
```

5Gbps, full duplex

Definition at line 54 of file port.h.

10.4.2.10 PORT_CAP_10G_FDX

```
#define PORT_CAP_10G_FDX 0x00000100
```

10Gbps, full duplex

Definition at line 55 of file port.h.

10.4.2.11 PORT_CAP_FLOW_CTRL

```
#define PORT_CAP_FLOW_CTRL 0x00001000
```

Flow control

Definition at line 56 of file port.h.

10.4.2.12 PORT_CAP_COPPER

```
#define PORT_CAP_COPPER 0x00002000
```

Copper media

Definition at line 57 of file port.h.

10.4.2.13 PORT_CAP_FIBER

```
#define PORT_CAP_FIBER 0x00004000
```

Fiber media

Definition at line 58 of file port.h.

10.4.2.14 PORT_CAP_DUAL_COPPER

```
#define PORT_CAP_DUAL_COPPER 0x00008000
```

Dual media, copper preferred

Definition at line 59 of file port.h.

10.4.2.15 PORT_CAP_DUAL_FIBER

```
#define PORT_CAP_DUAL_FIBER 0x00010000
```

Dual media, fiber preferred

Definition at line 60 of file port.h.

10.4.2.16 PORT_CAP_SD_ENABLE

```
#define PORT_CAP_SD_ENABLE 0x00020000
```

Signal Detect enabled

Definition at line 61 of file port.h.

10.4.2.17 PORT_CAP_SD_HIGH

```
#define PORT_CAP_SD_HIGH 0x00040000
```

Signal Detect active high

Definition at line 62 of file port.h.

10.4.2.18 PORT_CAP_SD_INTERNAL

```
#define PORT_CAP_SD_INTERNAL 0x00080000
```

Signal Detect select internal

Definition at line 63 of file port.h.

10.4.2.19 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.

10.4.2.20 PORT_CAP_XAUI_LANE_FLIP

```
#define PORT_CAP_XAUI_LANE_FLIP 0x00200000
```

Flip the XAUI lanes

Definition at line 65 of file port.h.

10.4.2.21 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.

10.4.2.22 PORT_CAP_SFP_DETECT

```
#define PORT_CAP_SFP_DETECT 0x00800000
```

Auto detect the SFP module

Definition at line 67 of file port.h.

10.4.2.23 PORT_CAP_STACKING

```
#define PORT_CAP_STACKING 0x01000000
```

Stack port candidate

Definition at line 68 of file port.h.

10.4.2.24 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.

10.4.2.25 PORT_CAP_SFP_ONLY

```
#define PORT_CAP_SFP_ONLY 0x04000000
```

SFP only port (not dual media)

Definition at line 70 of file port.h.

10.4.2.26 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.

10.4.2.27 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.

10.4.2.28 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.

10.4.2.29 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.

10.4.2.30 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.

10.4.2.31 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.

10.4.2.32 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.

10.4.2.33 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.

10.4.2.34 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.

10.4.2.35 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.

10.4.2.36 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.

10.4.2.37 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.

10.4.2.38 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.

10.4.2.39 PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER

```
#define PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER (PORT_CAP_TRI_SPEED | PORT_CAP_SPEED_DUAL_ANY_FIBER)
```

Copper 5 Fiber mode, auto detection supported

Definition at line 87 of file port.h.

10.4.2.40 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.

10.4.2.41 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.

10.4.2.42 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.

10.4.2.43 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.

10.4.2.44 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.

10.4.2.45 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.

10.4.2.46 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.

10.4.2.47 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.

10.4.3 Typedef Documentation

10.4.3.1 port_cap_t

```
typedef u32 port_cap_t
```

Bit-mask containing the port capabilities

Definition at line 99 of file port.h.

10.4.4 Enumeration Type Documentation

10.4.4.1 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.

10.4.4.2 vtss_fiber_port_speed_t

```
enum vtss_fiber_port_speed_t
```

Fiber Port speed.

Enumerator

VTSS_SPEED_FIBER_NOT_SUPPORTED_OR_↩ DISABLED	Fiber not supported/ Fiber port disabled
VTSS_SPEED_FIBER_100FX	100BASE-FX
VTSS_SPEED_FIBER_1000X	1000BASE-X
VTSS_SPEED_FIBER_AUTO	Auto detection
VTSS_SPEED_FIBER_DISABLED	Obsolete - use VTSS_SPEED_FIBER_NOT_SUPP↩ ORTED_OR_DISABLED instead

Definition at line 255 of file port.h.

10.5 vtss_api/include/vtss/api/types.h File Reference

Generic types API.

```
#include <vtss/api/options.h>
#include <sys/bsdtypes.h>
```

Data Structures

- struct vtss_aneg_t

  Auto negotiation struct.
- struct vtss_vlan_tag_t
- struct vtss_mac_t

  MAC Address.
- struct vtss_vid_mac_t

  MAC Address in specific VLAN.
- struct vtss_packet_rx_port_conf_t

  Packet registration per port.
- struct vtss_ipv6_t

  IPv6 address/mask.
- struct vtss_ip_addr_t

  Either an IPv4 or IPv6 address.
- struct vtss_ipv4_network_t

  IPv4 network.
- struct vtss_ipv6_network_t

  IPv6 network.
- struct vtss_ip_network_t

  IPv6 network.
- struct vtss_ipv4_uc_t

IPv4 unicast routing entry.

- struct vtss_ipv6_uc_t

  IPv6 routing entry.
- struct vtss_routing_entry_t

  Routing entry.
- struct vtss_l3_counters_t

  Routing interface statics counter.
- struct vtss_vcap_u8_t

  VCAP 8 bit value and mask.
- struct vtss_vcap_u16_t

  VCAP 16 bit value and mask.
- struct vtss_vcap_u24_t

  VCAP 24 bit value and mask.
- struct vtss_vcap_u32_t

  VCAP 32 bit value and mask.
- struct vtss_vcap_u40_t

  VCAP 40 bit value and mask.
- struct vtss_vcap_u48_t

  VCAP 48 bit value and mask.
- struct vtss_vcap_u128_t

  VCAP 128 bit value and mask.
- struct vtss_vcap_vid_t

  VCAP VLAN ID value and mask.
- struct vtss_vcap_ip_t

  VCAP IPv4 address value and mask.
- struct vtss_vcap_udp_tcp_t

  VCAP UDP/TCP port range.
- struct vtss_vcap_vr_t

  VCAP universal value or range.
- struct vtss_counter_pair_t

  Counter pair.
- struct vtss_evc_counters_t

  EVC/ECE counters.
- struct vtss_mpls_xc_counters_t

  MPLS cross-connect counters.
- 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 11
- #define VTSS_PORTS VTSS_PORT_COUNT
- #define VTSS_PORT_NO_NONE (0xffffffff)
- #define VTSS_PORT_NO_CPU (0xfffffffe)
- #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)0xFFF)
- #define VTSS_VIDS ((const vtss_vid_t)4096)
- #define VTSS_VID_ALL ((const vtss_vid_t)0x1000)
- #define VTSS_ETYPE_VTSS 0x8880
- #define VTSS_MAC_ADDR_SZ_BYTES 6
- #define MAC_ADDR_BROADCAST {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}
- #define VTSS_EVCS 1024
- #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 63
- #define VTSS_ACL_POLICIES (VTSS_ACL_POLICY_NO_MAX + 1)
- #define VTSS_ACL_POLICY_NO_START VTSS_ACL_POLICY_NO_MIN
- #define VTSS_ACL_POLICY_NO_END (VTSS_ACL_POLICY_NO_START + VTSS_ACL_POLICIES)
- #define VTSS_HQOS_COUNT 256
- #define VTSS_HQOS_ID_NONE 0xffff
- #define VTSS_ONE_MIA 1000000000
- #define VTSS_ONE_MILL 1000000
- #define VTSS_MAX_TIMEINTERVAL 0x7fffffffffffffffLL
- #define VTSS_INTERVAL_SEC(t) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MIA))
- #define VTSS_INTERVAL_MS(t) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MILL))
- #define VTSS_INTERVAL_US(t) ((i32)VTSS_DIV64((t)>>16, 1000))
- #define VTSS_INTERVAL_NS(t) ((i32)VTSS_MOD64((t)>>16, VTSS_ONE_MIA))
- #define VTSS_INTERVAL_PS(t) (((((i32)(t & 0xffff))∗1000)+0x8000)/0x10000)
- #define VTSS_SEC_NS_INTERVAL(s, n) (((vtss_timeinterval_t)(n)+(vtss_timeinterval_t)(s)∗VTSS_ONE_MIA)<<16)
- #define VTSS_CLOCK_IDENTITY_LENGTH 8
- #define VTSS_SYNCE_CLK_PORT_ARRAY_SIZE 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 occured. The API will never clear the variable. If is up to the application to clear the variable, when the event has been handled.
- typedef u32 vtss_packet_rate_t

  Policer packet rate in PPS.
- typedef u32 vtss_port_no_t

Port Number.

- typedef u32 vtss_phys_port_no_t

  Physical port number.
- typedef u32 vtss_prio_t

  Priority number.
- typedef u32 vtss_queue_t

  Queue number.
- typedef u32 vtss_tagprio_t

  Tag Priority or Priority Code Point (PCP)
- typedef BOOL vtss_dei_t

  Drop Eligible Indicator (DEI)
- typedef u8 vtss_dp_level_t

  Drop Precedence Level (DPL)
- typedef u8 vtss_pct_t

  Percentage, 0-100.
- typedef u32 vtss_bitrate_t

  Policer/Shaper bit rate in kbps (1000 bits per second). The rate will be rounded to the nearest value supported by the chip.
- typedef u32 vtss_burst_level_t

  Policer/shaper burst level in bytes. The level will be rounded to the nearest value supported by the chip.
- typedef u8 vtss_dscp_t

  DSCP value (0-63)
- typedef u32 vtss_qce_id_t

  QoS Control Entry ID.
- typedef u16 vtss_evc_policer_id_t

  EVC policer index.
- typedef u32 vtss_wred_group_t

  WRED group number.
- typedef u16 vtss_vid_t

  VLAN Identifier.
- typedef u16 vtss_etype_t

  Ethernet Type.
- typedef u8 vtss_mac_addr_t[VTSS_MAC_ADDR_SZ_BYTES]
- typedef u16 vtss_evc_id_t

  EVC ID.
- typedef u32 vtss_isdx_t
- typedef u32 vtss_aggr_no_t

  Aggregation Number.
- typedef u32 vtss_glag_no_t

  Description: GLAG number.
- typedef u32 vtss_packet_rx_queue_t

  Description: CPU Rx queue number.
- typedef u32 vtss_packet_rx_grp_t

  Description: CPU Rx group number.
- typedef u32 vtss_packet_tx_grp_t

  Description: CPU Tx group number.
- typedef u16 vtss_udp_tcp_t

  Description: UDP/TCP port number.
- typedef u32 vtss_ip_t

  IPv4 address/mask.
- typedef vtss_ip_t vtss_ipv4_t

IPv4 address/mask.

- typedef u32 vtss_prefix_size_t

  Prefix size.
- typedef u16 vtss_vcap_vr_value_t

  VCAP universal value or range type.
- typedef u32 vtss_acl_policer_no_t

  ACL policer number.
- typedef u32 vtss_acl_policy_no_t

  ACL policy number.
- typedef u32 vtss_ece_id_t

  EVC Control Entry (ECE) ID.
- typedef u64 vtss_counter_t

  Counter.
- typedef u16 vtss_hqos_id_t

  HQoS entry identifier (HQoS ID)
- typedef i64 vtss_clk_adj_rate_t

  Clock adjustment rate in parts per billion (ppb) ∗ 1<<16. Range is +-2∗∗47 ppb For example, 8.25 ppb is expressed as 0x0000.0000.0008.4000.
- typedef i64 vtss_timeinterval_t

  Time interval in ns ∗ 1<<16 range +-2∗∗47 ns = 140737 sec = 39 hours For example, 2.5 ns is expressed as 0x0000.0000.0002.8000.
- typedef u8 vtss_clock_identity[VTSS_CLOCK_IDENTITY_LENGTH]

  PTP clock unique identifier.

Enumerations

- enum {
  VTSS_RC_OK = 0, VTSS_RC_ERROR = -1, VTSS_RC_INV_STATE = -2, VTSS_RC_INCOMPLETE = -3,
  VTSS_RC_ERR_CLK_CONF_NOT_SUPPORTED = -6, VTSS_RC_ERR_KR_CONF_NOT_SUPPORTED
  = -7, VTSS_RC_ERR_KR_CONF_INVALID_PARAMETER = -8, VTSS_RC_ERR_PHY_BASE_NO_NOT_FOUND
  = -50,
  VTSS_RC_ERR_PHY_6G_MACRO_SETUP = -51, VTSS_RC_ERR_PHY_MEDIA_IF_NOT_SUPPORTED
  = -52, VTSS_RC_ERR_PHY_CLK_CONF_NOT_SUPPORTED = -53, VTSS_RC_ERR_PHY_GPIO_ALT_MODE_NOT_SUPP
  = -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_SUPPORTE
  = -58,
  VTSS_RC_ERR_PHY_RCPLL_NOT_SUPPORTED = -59, VTSS_RC_ERR_MACSEC_INVALID_SCI_MACADDR
  = -60, VTSS_RC_ERR_MACSEC_NOT_ENABLED = -61, VTSS_RC_ERR_MACSEC_SECY_ALREADY_IN_USE
  = -63,
  VTSS_RC_ERR_MACSEC_NO_SECY_FOUND = -64, VTSS_RC_ERR_MACSEC_NO_SECY_VACANCY
  = -65, VTSS_RC_ERR_MACSEC_INVALID_VALIDATE_FRM = -66, VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_MA
  = -67,
  VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_SA_FLOW = -68, VTSS_RC_ERR_MACSEC_COULD_NOT_ENA_SA
  = -69, VTSS_RC_ERR_MACSEC_COULD_NOT_SET_SA = -70, VTSS_RC_ERR_MACSEC_INVALID_BYPASS_HDR_LEN
  = -71,
  VTSS_RC_ERR_MACSEC_SC_NOT_FOUND = -72, VTSS_RC_ERR_MACSEC_NO_CTRL_FRM_MATCH
  = -73, VTSS_RC_ERR_MACSEC_COULD_NOT_SET_PATTERN = -74, VTSS_RC_ERR_MACSEC_TIMEOUT_ISSUE
  = -75,
  VTSS_RC_ERR_MACSEC_COULD_NOT_EMPTY_EGRESS = -76, VTSS_RC_ERR_MACSEC_AN_NOT_CREATED
  = -77, VTSS_RC_ERR_MACSEC_COULD_NOT_EMPTY_INGRESS = -78, VTSS_RC_ERR_MACSEC_TX_SC_NOT_EXIST
  = -80,
  VTSS_RC_ERR_MACSEC_COULD_NOT_DISABLE_SA = -81, VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_RX_SA
  = -82, VTSS_RC_ERR_MACSEC_COULD_NOT_DEL_TX_SA = -83, VTSS_RC_ERR_MACSEC_PATTERN_NOT_SET

= -84,
VTSS_RC_ERR_MACSEC_HW_RESOURCE_EXHUSTED = -85, VTSS_RC_ERR_MACSEC_SCI_ALREADY_EXISTS
= -86, VTSS_RC_ERR_MACSEC_SC_RESOURCE_NOT_FOUND = -87, VTSS_RC_ERR_MACSEC_RX_AN_ALREADY_IN_
= -88,
VTSS_RC_ERR_MACSEC_EMPTY_RECORD = -89, VTSS_RC_ERR_MACSEC_COULD_NOT_PRG_XFORM
= -90, VTSS_RC_ERR_MACSEC_COULD_NOT_TOGGLE_SA = -91, VTSS_RC_ERR_MACSEC_TX_AN_ALREADY_IN_US
= -92,
VTSS_RC_ERR_MACSEC_ALL_AVAILABLE_SA_IN_USE = -93, VTSS_RC_ERR_MACSEC_MATCH_DISABLE
= -94, VTSS_RC_ERR_MACSEC_ALL_CP_RULES_IN_USE = -95, VTSS_RC_ERR_MACSEC_PATTERN_PRIO_NOT_VAL
= -96,
VTSS_RC_ERR_MACSEC_BUFFER_TOO_SMALL = -97, VTSS_RC_ERR_MACSEC_FRAME_TOO_LONG
= -98, VTSS_RC_ERR_MACSEC_FRAME_TRUNCATED = -99, VTSS_RC_ERR_MACSEC_PHY_POWERED_DOWN
= -100,
VTSS_RC_ERR_MACSEC_PHY_NOT_MACSEC_CAPABLE = -101, VTSS_RC_ERR_MACSEC_AN_NOT_EXIST
= -102, VTSS_RC_ERR_MACSEC_NO_PATTERN_CFG = -103, VTSS_RC_ERR_MACSEC_MAX_MTU =
-105,
VTSS_RC_ERR_MACSEC_UNEXPECT_CP_MODE = -106, VTSS_RC_ERR_MACSEC_COULD_NOT_DISABLE_AN
= -107, VTSS_RC_ERR_MACSEC_RULE_OUT_OF_RANGE = -108, VTSS_RC_ERR_MACSEC_RULE_NOT_EXIST
= -109,
VTSS_RC_ERR_MACSEC_CSR_READ = -110, VTSS_RC_ERR_MACSEC_CSR_WRITE = -111,
VTSS_RC_ERR_PHY_6G_RCPLL_ON_BASE_PORT_ONLY = -112, VTSS_RC_ERR_INVALID_NULL_PTR
= -200 }

Error codes.

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

  Memory allocation flags.

- enum vtss_port_interface_t {
VTSS_PORT_INTERFACE_NO_CONNECTION, VTSS_PORT_INTERFACE_LOOPBACK, VTSS_PORT_INTERFACE_INTE
VTSS_PORT_INTERFACE_MII,
VTSS_PORT_INTERFACE_GMII, VTSS_PORT_INTERFACE_RGMII, VTSS_PORT_INTERFACE_TBI,
VTSS_PORT_INTERFACE_RTBI,
VTSS_PORT_INTERFACE_SGMII, VTSS_PORT_INTERFACE_SGMII_CISCO, VTSS_PORT_INTERFACE_SERDES,
VTSS_PORT_INTERFACE_VAUI,
VTSS_PORT_INTERFACE_100FX, VTSS_PORT_INTERFACE_XAUI, VTSS_PORT_INTERFACE_RXAUI,
VTSS_PORT_INTERFACE_XGMII,
VTSS_PORT_INTERFACE_SPI4, VTSS_PORT_INTERFACE_QSGMII, VTSS_PORT_INTERFACE_SFI }

  The different interfaces for connecting MAC and PHY.

- enum vtss_serdes_mode_t {
VTSS_SERDES_MODE_DISABLE, VTSS_SERDES_MODE_XAUI_12G, VTSS_SERDES_MODE_XAUI,
VTSS_SERDES_MODE_RXAUI,
VTSS_SERDES_MODE_RXAUI_12G, VTSS_SERDES_MODE_2G5, VTSS_SERDES_MODE_QSGMII,
VTSS_SERDES_MODE_SGMII,
VTSS_SERDES_MODE_100FX, VTSS_SERDES_MODE_1000BaseX, VTSS_SERDES_MODE_SFI,
VTSS_SERDES_MODE_SFI_DAC,
VTSS_SERDES_MODE_IDLE }

  Serdes macro mode.

- enum vtss_storm_policer_mode_t { VTSS_STORM_POLICER_MODE_PORTS_AND_CPU, VTSS_STORM_POLICER_MODE
VTSS_STORM_POLICER_MODE_CPU_ONLY }

  Storm policer mode configuration.

- enum vtss_policer_type_t { VTSS_POLICER_TYPE_MEF, VTSS_POLICER_TYPE_SINGLE }

  Dual leaky buckets policer configuration.

- enum vtss_vlan_frame_t { VTSS_VLAN_FRAME_ALL, VTSS_VLAN_FRAME_TAGGED, VTSS_VLAN_FRAME_UNTAGGED
}

  VLAN acceptable frame type.

- enum vtss_packet_reg_type_t {
VTSS_PACKET_REG_NORMAL, VTSS_PACKET_REG_FORWARD, VTSS_PACKET_REG_DISCARD,

VTSS_PACKET_REG_CPU_COPY, VTSS_PACKET_REG_CPU_ONLY }

Packet registration type.

- enum vtss_vdd_t { VTSS_VDD_1V0, VTSS_VDD_1V2 }

  VDD power supply.
- enum vtss_ip_type_t { VTSS_IP_TYPE_NONE = 0, VTSS_IP_TYPE_IPV4 = 1, VTSS_IP_TYPE_IPV6 = 2 }

  IP address type.
- enum vtss_routing_entry_type_t { **VTSS_ROUTING_ENTRY_TYPE_INVALID** = 0, **VTSS_ROUTING_EN↩TRY_TYPE_IPV6_UC** = 1, **VTSS_ROUTING_ENTRY_TYPE_IPV4_MC** = 2, **VTSS_ROUTING_ENTRY_↩TYPE_IPV4_UC** = 3 }

  Routing entry type.
- enum vtss_vcap_bit_t { VTSS_VCAP_BIT_ANY, VTSS_VCAP_BIT_0, VTSS_VCAP_BIT_1 }

  VCAP 1 bit.
- enum vtss_vcap_vr_type_t { VTSS_VCAP_VR_TYPE_VALUE_MASK, VTSS_VCAP_VR_TYPE_RANGE_INCLUSIVE, VTSS_VCAP_VR_TYPE_RANGE_EXCLUSIVE }

  Value/Range type.
- enum vtss_vcap_key_type_t { VTSS_VCAP_KEY_TYPE_NORMAL, VTSS_VCAP_KEY_TYPE_DOUBLE_TAG, VTSS_VCAP_KEY_TYPE_IP_ADDR, VTSS_VCAP_KEY_TYPE_MAC_IP_ADDR }

  VCAP key type.
- enum vtss_ece_dir_t { VTSS_ECE_DIR_BOTH, VTSS_ECE_DIR_UNI_TO_NNI, VTSS_ECE_DIR_NNI_TO_UNI }

  ECE direction.
- enum vtss_ece_pop_tag_t { VTSS_ECE_POP_TAG_0, VTSS_ECE_POP_TAG_1, VTSS_ECE_POP_TAG_2 }

  Ingress tag popping.
- enum vtss_ece_rule_t { VTSS_ECE_RULE_BOTH, VTSS_ECE_RULE_RX, VTSS_ECE_RULE_TX }

  ECE rule types.
- enum vtss_ece_tx_lookup_t { VTSS_ECE_TX_LOOKUP_VID, VTSS_ECE_TX_LOOKUP_VID_PCP, VTSS_ECE_TX_LOOKUP_ISDX }

  ECE egress lookup types.
- enum vtss_ece_pcp_mode_t { VTSS_ECE_PCP_MODE_CLASSIFIED, VTSS_ECE_PCP_MODE_FIXED, VTSS_ECE_PCP_MODE_MAPPED }

  PCP mode.
- enum vtss_ece_dei_mode_t { VTSS_ECE_DEI_MODE_CLASSIFIED, VTSS_ECE_DEI_MODE_FIXED, VTSS_ECE_DEI_MODE_DP }

  DEI mode.
- enum vtss_ece_inner_tag_type_t { VTSS_ECE_INNER_TAG_NONE, VTSS_ECE_INNER_TAG_C, VTSS_ECE_INNER_TAG_S, VTSS_ECE_INNER_TAG_S_CUSTOM }

  ECE inner tag type.
- enum vtss_hqos_sch_mode_t { VTSS_HQOS_SCH_MODE_NORMAL, VTSS_HQOS_SCH_MODE_BASIC, VTSS_HQOS_SCH_MODE_HIERARCHICAL }

  HQoS port scheduling mode.

10.5.1 Detailed Description

Generic types API.

This header file describes generic types used in the API

10.5.2 Macro Definition Documentation

10.5.2.1 PRIu64

```
#define PRIu64 "llu"
```

Fallback un-signed 64-bit formatting string

Definition at line 111 of file types.h.

10.5.2.2 PRIi64

```
#define PRIi64 "lli"
```

Fallback signed 64-bit formatting string

Definition at line 115 of file types.h.

10.5.2.3 PRIx64

```
#define PRIx64 "llx"
```

Fallback hex 64-bit formatting string

Definition at line 119 of file types.h.

10.5.2.4 VTSS_BIT64

```
#define VTSS_BIT64(
              x ) (1ULL << (x))
```

Set one bit in a 64-bit mask

Definition at line 122 of file types.h.

10.5.2.5 VTSS_BITMASK64

```
#define VTSS_BITMASK64(
              x ) ((1ULL << (x)) - 1)
```

Get a bitmask consisting of x ones

Definition at line 123 of file types.h.

10.5.2.6 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.

10.5.2.7 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.

10.5.2.8 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.

10.5.2.9 TRUE

```
#define TRUE 1
```

True boolean value

Definition at line 129 of file types.h.

10.5.2.10 FALSE

```
#define FALSE 0
```

False boolean value

Definition at line 132 of file types.h.

10.5.2.11 VTSS_PACKET_RATE_DISABLED

```
#define VTSS_PACKET_RATE_DISABLED 0xffffffff
```

Special value for disabling packet policer

Definition at line 238 of file types.h.

10.5.2.12 VTSS_PORT_COUNT [1/2]

```
#define VTSS_PORT_COUNT 1
```

Default number of ports

Number of ports

Definition at line 301 of file types.h.

10.5.2.13 VTSS_PORT_COUNT [2/2]

```
#define VTSS_PORT_COUNT 11
```

Default number of ports

Number of ports

Definition at line 301 of file types.h.

10.5.2.14 VTSS_PORTS

```
#define VTSS_PORTS VTSS_PORT_COUNT
```

Number of ports

Definition at line 444 of file types.h.

10.5.2.15 VTSS_PORT_NO_NONE

```
#define VTSS_PORT_NO_NONE (0xffffffff)
```

Port number none

Definition at line 448 of file types.h.

10.5.2.16 VTSS_PORT_NO_CPU

```
#define VTSS_PORT_NO_CPU (0xfffffffe)
```

Port number for CPU for special purposes

Definition at line 449 of file types.h.

10.5.2.17 VTSS_PORT_NO_START

```
#define VTSS_PORT_NO_START (0)
```

Port start number

Definition at line 450 of file types.h.

10.5.2.18 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.

10.5.2.19 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.

10.5.2.20 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.

10.5.2.21 VTSS_PRIOS

```
#define VTSS_PRIOS 8
```

Number of priorities

Definition at line 515 of file types.h.

10.5.2.22 VTSS_PRIO_NO_NONE

```
#define VTSS_PRIO_NO_NONE 0xffffffff
```

Priority number none (= undefined)

Definition at line 516 of file types.h.

10.5.2.23 VTSS_PRIO_START

```
#define VTSS_PRIO_START 0
```

Priority start number (lowest)

Definition at line 517 of file types.h.

10.5.2.24 VTSS_PRIO_END

```
#define VTSS_PRIO_END (VTSS_PRIO_START + VTSS_PRIOS)
```

Priority end number

Definition at line 518 of file types.h.

10.5.2.25 VTSS_PRIO_ARRAY_SIZE

```
#define VTSS_PRIO_ARRAY_SIZE VTSS_PRIO_END
```

Priority number array size

Definition at line 519 of file types.h.

10.5.2.26 VTSS_QUEUES

```
#define VTSS_QUEUES VTSS_PRIOS
```

Number of queues

Definition at line 523 of file types.h.

10.5.2.27 VTSS_QUEUE_START

```
#define VTSS_QUEUE_START 0
```

Queue start number

Definition at line 524 of file types.h.

10.5.2.28 VTSS_QUEUE_END

```
#define VTSS_QUEUE_END (VTSS_QUEUE_START + VTSS_QUEUES)
```

Queue end number

Definition at line 525 of file types.h.

10.5.2.29 VTSS_QUEUE_ARRAY_SIZE

```
#define VTSS_QUEUE_ARRAY_SIZE VTSS_QUEUE_END
```

Queue number array size

Definition at line 526 of file types.h.

10.5.2.30 VTSS_PCPS

```
#define VTSS_PCPS 8
```

Number of PCP values

Definition at line 530 of file types.h.

10.5.2.31 VTSS_PCP_START

```
#define VTSS_PCP_START 0
```

PCP start number

Definition at line 531 of file types.h.

10.5.2.32 VTSS_PCP_END

```
#define VTSS_PCP_END (VTSS_PCP_START + VTSS_PCPS)
```

PCP end number

Definition at line 532 of file types.h.

10.5.2.33 VTSS_PCP_ARRAY_SIZE

```
#define VTSS_PCP_ARRAY_SIZE VTSS_PCP_END
```

PCP array size

Definition at line 533 of file types.h.

10.5.2.34 VTSS_DEIS

```
#define VTSS_DEIS 2
```

Number of DEI values

Definition at line 537 of file types.h.

10.5.2.35 VTSS_DEI_START

```
#define VTSS_DEI_START 0
```

DEI start number

Definition at line 538 of file types.h.

10.5.2.36 VTSS_DEI_END

```
#define VTSS_DEI_END (VTSS_DEI_START + VTSS_DEIS)
```

DEI end number

Definition at line 539 of file types.h.

10.5.2.37 VTSS_DEI_ARRAY_SIZE

```
#define VTSS_DEI_ARRAY_SIZE VTSS_DEI_END
```

DEI array size

Definition at line 540 of file types.h.

10.5.2.38 VTSS_DPLS

```
#define VTSS_DPLS 2
```

Default number of drop precedence levels

Definition at line 544 of file types.h.

10.5.2.39 VTSS_DPL_START

```
#define VTSS_DPL_START 0
```

DPL start number

Definition at line 551 of file types.h.

10.5.2.40 VTSS_DPL_END

```
#define VTSS_DPL_END (VTSS_DPL_START + VTSS_DPLS)
```

DPL end number

Definition at line 552 of file types.h.

10.5.2.41 VTSS_DPL_ARRAY_SIZE

```
#define VTSS_DPL_ARRAY_SIZE VTSS_DPL_END
```

DPL array size

Definition at line 553 of file types.h.

10.5.2.42 VTSS_BITRATE_DISABLED

```
#define VTSS_BITRATE_DISABLED 0xffffffff
```

Bitrate disabled

Definition at line 563 of file types.h.

10.5.2.43 VTSS_VID_NULL

```
#define VTSS_VID_NULL ((const vtss_vid_t)0)
```

NULL VLAN ID

Definition at line 609 of file types.h.

10.5.2.44 VTSS_VID_DEFAULT

```
#define VTSS_VID_DEFAULT ((const vtss_vid_t)1)
```

Default VLAN ID

Definition at line 610 of file types.h.

10.5.2.45 VTSS_VID_RESERVED

```
#define VTSS_VID_RESERVED ((const vtss_vid_t)0xFFF)
```

Reserved VLAN ID

Definition at line 611 of file types.h.

10.5.2.46 VTSS_VIDS

```
#define VTSS_VIDS ((const vtss_vid_t)4096)
```

Number of VLAN IDs

Definition at line 612 of file types.h.

10.5.2.47 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.

10.5.2.48 VTSS_ETYPE_VTSS

```
#define VTSS_ETYPE_VTSS 0x8880
```

Vitesse Ethernet Type

Definition at line 640 of file types.h.

10.5.2.49 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.

10.5.2.50 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.

10.5.2.51 VTSS_EVCS

```
#define VTSS_EVCS 1024
```

Maximum number of Ethernet Virtual Connections

Definition at line 666 of file types.h.

10.5.2.52 VTSS_ISDX_NONE

```
#define VTSS_ISDX_NONE (0)
```

Ingress Service Index number none

Definition at line 674 of file types.h.

10.5.2.53 VTSS_AGGRS

```
#define VTSS_AGGRS (VTSS_PORTS/2)
```

Number of LLAGs

Definition at line 678 of file types.h.

10.5.2.54 VTSS_AGGR_NO_NONE

```
#define VTSS_AGGR_NO_NONE 0xffffffff
```

Aggregation number none

Definition at line 679 of file types.h.

10.5.2.55 VTSS_AGGR_NO_START

```
#define VTSS_AGGR_NO_START 0
```

Aggregation start number

Definition at line 680 of file types.h.

10.5.2.56 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.

10.5.2.57 VTSS_GLAGS

```
#define VTSS_GLAGS 2
```

Number of GLAGs

Definition at line 689 of file types.h.

10.5.2.58 VTSS_GLAG_NO_NONE

```
#define VTSS_GLAG_NO_NONE 0xffffffff
```

GLAG number none

Definition at line 691 of file types.h.

10.5.2.59 VTSS_GLAG_NO_START

```
#define VTSS_GLAG_NO_START 0
```

GLAG start number

Definition at line 692 of file types.h.

10.5.2.60 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.

10.5.2.61 VTSS_GLAG_PORTS

```
#define VTSS_GLAG_PORTS 8
```

Number of GLAG ports

Definition at line 696 of file types.h.

10.5.2.62 VTSS_GLAG_PORT_START

```
#define VTSS_GLAG_PORT_START 0
```

GLAG port start number

Definition at line 697 of file types.h.

10.5.2.63 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.

10.5.2.64 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.

10.5.2.65 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.

10.5.2.66 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.

10.5.2.67 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.

10.5.2.68 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.

10.5.2.69 VTSS_PACKET_RX_QUEUE_START

```
#define VTSS_PACKET_RX_QUEUE_START (0)
```

Rx queue start number

Definition at line 746 of file types.h.

10.5.2.70 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.

10.5.2.71 VTSS_ACL_POLICERS

```
#define VTSS_ACL_POLICERS 16
```

Number of ACL policers

Definition at line 1028 of file types.h.

10.5.2.72 VTSS_ACL_POLICER_NO_START

```
#define VTSS_ACL_POLICER_NO_START 0
```

ACL policer start number

Definition at line 1029 of file types.h.

10.5.2.73 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.

10.5.2.74 VTSS_ACL_POLICY_NO_NONE

```
#define VTSS_ACL_POLICY_NO_NONE 0xffffffff
```

ACLs disabled on port

Definition at line 1034 of file types.h.

10.5.2.75 VTSS_ACL_POLICY_NO_MIN

```
#define VTSS_ACL_POLICY_NO_MIN 0
```

ACLs policy minimum number

Definition at line 1035 of file types.h.

10.5.2.76 VTSS_ACL_POLICY_NO_MAX

```
#define VTSS_ACL_POLICY_NO_MAX 63
```

ACLs policy maximum number

Definition at line 1039 of file types.h.

10.5.2.77 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.

10.5.2.78 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.

10.5.2.79 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.

10.5.2.80 VTSS_HQOS_COUNT

```
#define VTSS_HQOS_COUNT 256
```

Maximum number of HQoS entries

Definition at line 1165 of file types.h.

10.5.2.81 VTSS_HQOS_ID_NONE

```
#define VTSS_HQOS_ID_NONE 0xffff
```

Special HQoS ID value

Definition at line 1167 of file types.h.

10.5.2.82 VTSS_ONE_MIA

```
#define VTSS_ONE_MIA 1000000000
```

One billion

Definition at line 1198 of file types.h.

10.5.2.83 VTSS_ONE_MILL

```
#define VTSS_ONE_MILL 1000000
```

One million

Definition at line 1199 of file types.h.

10.5.2.84 VTSS_MAX_TIMEINTERVAL

```
#define VTSS_MAX_TIMEINTERVAL 0x7fffffffffffffffLL
```

Maximum time interval

Definition at line 1200 of file types.h.

10.5.2.85 VTSS_INTERVAL_SEC

```
#define VTSS_INTERVAL_SEC(
            t ) ((i32)VTSS_DIV64((t)>>16, VTSS_ONE_MIA))
```

One Second time interval

Definition at line 1202 of file types.h.

10.5.2.86 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.

10.5.2.87 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.

10.5.2.88 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.

10.5.2.89 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.

10.5.2.90 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.

10.5.2.91 VTSS_CLOCK_IDENTITY_LENGTH

```
#define VTSS_CLOCK_IDENTITY_LENGTH 8
```

Length of unique PTP identifier

Definition at line 1218 of file types.h.

10.5.2.92 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.

10.5.3 Typedef Documentation

10.5.3.1 i8

```
typedef char i8
```

Fallback Integer types.

8-bit signed

Definition at line 68 of file types.h.

10.5.3.2 i16

```
typedef signed short i16
```

16-bit signed

Definition at line 69 of file types.h.

10.5.3.3 i32

```
typedef signed int i32
```

32-bit signed

Definition at line 70 of file types.h.

10.5.3.4 i64

```
typedef signed long long i64
```

64-bit signed

Definition at line 71 of file types.h.

10.5.3.5 u8

```
typedef unsigned char u8
```

8-bit unsigned

Definition at line 73 of file types.h.

10.5.3.6 u16

```
typedef unsigned short u16
```

16-bit unsigned

Definition at line 74 of file types.h.

10.5.3.7 u32

```
typedef unsigned int u32
```

32-bit unsigned

Definition at line 75 of file types.h.

10.5.3.8 u64

```
typedef unsigned long long u64
```

64-bit unsigned

Definition at line 76 of file types.h.

10.5.3.9 BOOL

```
typedef unsigned char BOOL
```

Boolean implemented as 8-bit unsigned

Definition at line 78 of file types.h.

10.5.3.10 uintptr_t

```
typedef unsigned int uintptr_t
```

Unsigned integer big enough to hold pointers

Definition at line 79 of file types.h.

10.5.3.11 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.

10.5.3.12 vtss_isdx_t

```
typedef u32 vtss_isdx_t
```

Ingress Service Index type

Definition at line 673 of file types.h.

10.5.3.13 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.

10.5.3.14 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.

10.5.4 Enumeration Type Documentation

Enumerator

10.5.4.1 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 SynceE)
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_SUPPOR↩TED	PHY does not support the selected media mode
VTSS_RC_ERR_PHY_CLK_CONF_NOT_SUPPO↩RTED	The PHY doesn't support clock configuration (for SynceE)
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_SUPPOR↩TED	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_S↩UPPORTED	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_MACA↩DDR	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

Enumerator

VTSS_RC_ERR_MACSEC_COULD_NOT_SET_SA	Could not set SA to in use
VTSS_RC_ERR_MACSEC_INVALID_BYPASS_H↩DR_LEN	Invalid header bypass length
VTSS_RC_ERR_MACSEC_SC_NOT_FOUND	Could not find SC (from sci)
VTSS_RC_ERR_MACSEC_NO_CTRL_FRM_MA↩TCH	No control frame match
VTSS_RC_ERR_MACSEC_COULD_NOT_SET_P↩ATTERN	Could no set bypass pattern for CP rule
VTSS_RC_ERR_MACSEC_TIMEOUT_ISSUE	Internal timeout issue, bailing out
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

Enumerator

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
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.

10.5.4.2 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.

10.5.4.3 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.

10.5.4.4 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

Enumerator

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.

10.5.4.5 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.

10.5.4.6 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.

10.5.4.7 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.

10.5.4.8 vtss_packet_reg_type_t

`enum vtss_packet_reg_type_t`

Packet registration type.

Enumerator

VTSS_PACKET_REG_NORMAL	Global registration configuration is used
VTSS_PACKET_REG_FORWARD	Forward normally
VTSS_PACKET_REG_DISCARD	Discard
VTSS_PACKET_REG_CPU_COPY	Copy to CPU
VTSS_PACKET_REG_CPU_ONLY	Redirect to CPU

Definition at line 753 of file types.h.

10.5.4.9 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.

10.5.4.10 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.

10.5.4.11 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.

10.5.4.12 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.

10.5.4.13 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.

10.5.4.14 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.

10.5.4.15 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.

10.5.4.16 vtss_ece_rule_t

`enum vtss_ece_rule_t`

ECE rule types.

Enumerator

VTSS_ECE_RULE_BOTH	Ingress and egress rules
VTSS_ECE_RULE_RX	Ingress rules
VTSS_ECE_RULE_TX	Egress rules

Definition at line 1075 of file types.h.

10.5.4.17 vtss_ece_tx_lookup_t

```
enum vtss_ece_tx_lookup_t
```

ECE egress lookup types.

Enumerator

VTSS_ECE_TX_LOOKUP_VID	VID lookup
VTSS_ECE_TX_LOOKUP_VID_PCP	(VID, PCP) lookup
VTSS_ECE_TX_LOOKUP_ISDX	ISDX lookup

Definition at line 1083 of file types.h.

10.5.4.18 vtss_ece_pcp_mode_t

```
enum vtss_ece_pcp_mode_t
```

PCP mode.

Enumerator

VTSS_ECE_PCP_MODE_CLASSIFIED	Classified PCP
VTSS_ECE_PCP_MODE_FIXED	Fixed PCP
VTSS_ECE_PCP_MODE_MAPPED	PCP based on mapped (QOS, DP)

Definition at line 1091 of file types.h.

10.5.4.19 vtss_ece_dei_mode_t

```
enum vtss_ece_dei_mode_t
```

DEI mode.

Enumerator

VTSS_ECE_DEI_MODE_CLASSIFIED	Classified DEI
VTSS_ECE_DEI_MODE_FIXED	Fixed DEI
VTSS_ECE_DEI_MODE_DP	DP-based DEI

Definition at line 1099 of file types.h.

10.5.4.20 vtss_ece_inner_tag_type_t

```
enum vtss_ece_inner_tag_type_t
```

ECE inner tag type.

Enumerator

VTSS_ECE_INNER_TAG_NONE	No inner tag
VTSS_ECE_INNER_TAG_C	Inner tag is C-tag
VTSS_ECE_INNER_TAG_S	Inner tag is S-tag
VTSS_ECE_INNER_TAG_S_CUSTOM	Inner tag is S-custom tag

Definition at line 1118 of file types.h.

10.5.4.21 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.

10.6 vtss_api/include/vtss_ae_api.h File Reference

ae API

```
#include <vtss/api/types.h>
```

10.6.1 Detailed Description

ae API

10.7 vtss_api/include/vtss_afi_api.h File Reference

AFI API.

```
#include <vtss/api/options.h>
#include <vtss/api/types.h>
```

10.7.1 Detailed Description

AFI API.

This header file describes Automatic Frame Injector functions.

10.8 vtss_api/include/vtss_aneg_api.h File Reference

ANEG API.

```
#include <vtss/api/types.h>
```

10.8.1 Detailed Description

ANEG API.

10.9 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_afi_api.h>
#include <vtss_security_api.h>
#include <vtss_l2_api.h>
#include <vtss_oam_api.h>
#include <vtss_evc_api.h>
#include <vtss_fdma_api.h>
#include <vtss_mpls_api.h>
#include <vtss_hqos_api.h>
#include <vtss_sync_api.h>
#include <vtss_phy_ts_api.h>
#include <vtss_ts_api.h>
```

10.9.1 Detailed Description

Vitesse API main header file.

This is the only header file which must be included by the application

10.10 vtss_api/include/vtss_evc_api.h File Reference

EVC API.

```
#include <vtss/api/types.h>
#include "vtss_mpls_api.h"
```

Data Structures

- struct vtss_evc_port_conf_t

  EVC port configuration.
- struct vtss_evc_pb_conf_t

  PB specific EVC configuration.
- struct vtss_evc_mpls_pw_info_t

  MPLS Pseudo Wire configuration, used when attaching a PW to an EVC.
- struct vtss_evc_mpls_tp_conf_t

  MPLS 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_etype_t

  ECE Ethernet Type information.
- struct vtss_ece_frame_llc_t

  ECE LLC information.
- struct vtss_ece_frame_snap_t

  ECE SNAP information.
- struct vtss_ece_frame_ipv4_t

  ECE IPv4 information.
- struct vtss_ece_frame_ipv6_t

  ECE IPv6 information.
- struct vtss_ece_key_t

  ECE key.
- struct vtss_ece_outer_tag_t

  ECE outer tag.
- struct vtss_ece_inner_tag_t

  ECE inner tag.
- struct vtss_ece_action_t

  ECE action.
- struct vtss_ece_t

  EVC Control Entry.
- struct vtss_evc_oam_port_conf_t

  EVC OAM port configuration.
- struct vtss_mce_tag_t

  MCE tag information.
- struct vtss_mce_key_t

  MCE key.
- struct vtss_mce_outer_tag_t

  ECE outer tag.
- struct vtss_mce_action_t

  MCE action.
- struct vtss_mce_t

  MEP Control Entry.
- struct vtss_mce_port_info_t

  MEP Control Entry port information.

Macros

- #define VTSS_EVC_POLICERS 1022
- #define VTSS_EVC_POLICER_ID_DISCARD 4094
- #define VTSS_EVC_POLICER_ID_NONE 4095
- #define VTSS_EVC_POLICER_ID_EVC 4096
- #define VTSS_EVC_ID_NONE 0xffff
- #define VTSS_EVC_MPLS_PW_CNT VTSS_PORTS
- #define VTSS_ECE_ID_LAST 0
- #define VTSS_MCE_ID_LAST 0
- #define VTSS_MCE_ISDX_NONE 0xFFFFFFFF
- #define VTSS_MCE_ISDX_NEW 0xFFFFFFFE
- #define VTSS_ISDX_CPU_TX 1023
- #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_ece_type_t { VTSS_ECE_TYPE_ANY, VTSS_ECE_TYPE_ETYPE, VTSS_ECE_TYPE_LLC, VTSS_ECE_TYPE_SNAP, 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.
- enum vtss_mce_pcp_mode_t { **VTSS_MCE_PCP_MODE_FIXED**, **VTSS_MCE_PCP_MODE_MAPPED** }

  MCE PCP mode.
- enum vtss_mce_dei_mode_t { **VTSS_MCE_DEI_MODE_FIXED**, **VTSS_MCE_DEI_MODE_DP** }

  MCE PCP mode.
- enum vtss_mce_tx_lookup_t { VTSS_MCE_TX_LOOKUP_VID, VTSS_MCE_TX_LOOKUP_ISDX, VTSS_MCE_TX_LOOKUP_ISDX_PCP }

  MCE egress lookup types.
- enum vtss_mce_oam_detect_t { VTSS_MCE_OAM_DETECT_NONE, VTSS_MCE_OAM_DETECT_UNTAGGED, VTSS_MCE_OAM_DETECT_SINGLE_TAGGED, VTSS_MCE_OAM_DETECT_DOUBLE_TAGGED }

  MCE OAM detection signalled to VOE.
- enum vtss_mce_rule_t { VTSS_MCE_RULE_BOTH, VTSS_MCE_RULE_RX }

  MCE rule types.

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_evc_hqos_map_get (const vtss_inst_t inst, const vtss_evc_id_t evc_id, const vtss_port_no_t port_no, vtss_hqos_id_t *const hqos_id)

  Get the HQoS ID mapping of a port on an EVC.

- vtss_rc vtss_evc_hqos_map_set (const vtss_inst_t inst, const vtss_evc_id_t evc_id, const vtss_port_no_t port_no, const vtss_hqos_id_t hqos_id)

  Map a port on an EVC to an HQoS ID.
- vtss_rc vtss_ece_init (const vtss_inst_t inst, const vtss_ece_type_t type, vtss_ece_t *const ece)

  Initialize ECE to default values.
- vtss_rc vtss_ece_add (const vtss_inst_t inst, const vtss_ece_id_t ece_id, const vtss_ece_t *const ece)

  Add/modify ECE.
- vtss_rc vtss_ece_del (const vtss_inst_t inst, const vtss_ece_id_t ece_id)

  Delete ECE.
- vtss_rc vtss_evc_oam_port_conf_get (const vtss_inst_t inst, const vtss_evc_id_t evc_id, const vtss_port_no_t port_no, vtss_evc_oam_port_conf_t *const conf)

  Get OAM port configuration for (EVC, port)
- vtss_rc vtss_evc_oam_port_conf_set (const vtss_inst_t inst, const vtss_evc_id_t evc_id, const vtss_port_no_t port_no, const vtss_evc_oam_port_conf_t *const conf)

  Set OAM port configuration for (EVC, port)
- 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.
- vtss_rc vtss_mce_port_info_get (const vtss_inst_t inst, const vtss_mce_id_t mce_id, const vtss_port_no_t port_no, vtss_mce_port_info_t *const info)

  Get MCE info.
- vtss_rc vtss_evc_counters_get (const vtss_inst_t inst, const vtss_evc_id_t evc_id, const vtss_port_no_t port_no, vtss_evc_counters_t *const counters)

  Get EVC counters for a port.
- vtss_rc vtss_evc_counters_clear (const vtss_inst_t inst, const vtss_evc_id_t evc_id, const vtss_port_no_t port_no)

  Clear EVC counters for a port.
- vtss_rc vtss_ece_counters_get (const vtss_inst_t inst, const vtss_ece_id_t ece_id, const vtss_port_no_t port_no, vtss_evc_counters_t *const counters)

  Get ECE counters for a port.
- vtss_rc vtss_ece_counters_clear (const vtss_inst_t inst, const vtss_ece_id_t ece_id, const vtss_port_no_t port_no)

  Clear ECE counters for a port.
- vtss_rc vtss_mce_counters_get (const vtss_inst_t inst, const vtss_mce_id_t mce_id, const vtss_port_no_t port_no, vtss_evc_counters_t *const counters)

  Get MCE counters for a port.
- vtss_rc vtss_mce_counters_clear (const vtss_inst_t inst, const vtss_mce_id_t mce_id, const vtss_port_no_t port_no)

  Clear MCE counters for a port.

10.10.1 Detailed Description

EVC API.

This header file describes EVC functions

10.10.2 Macro Definition Documentation

10.10.2.1 VTSS_EVC_POLICERS

```
#define VTSS_EVC_POLICERS 1022
```

Maximum number of EVC policers

Definition at line 197 of file vtss_evc_api.h.

10.10.2.2 VTSS_EVC_POLICER_ID_DISCARD

```
#define VTSS_EVC_POLICER_ID_DISCARD 4094
```

EVC/ECE: Policer discards all frames

Definition at line 205 of file vtss_evc_api.h.

10.10.2.3 VTSS_EVC_POLICER_ID_NONE

```
#define VTSS_EVC_POLICER_ID_NONE 4095
```

EVC/ECE: Policer forwards all frames

Definition at line 206 of file vtss_evc_api.h.

10.10.2.4 VTSS_EVC_POLICER_ID_EVC

```
#define VTSS_EVC_POLICER_ID_EVC 4096
```

ECE only: Use EVC policer

Definition at line 207 of file vtss_evc_api.h.

10.10.2.5 VTSS_EVC_ID_NONE

```
#define VTSS_EVC_ID_NONE 0xffff
```

Special EVC ID value

Definition at line 243 of file vtss_evc_api.h.

10.10.2.6 VTSS_EVC_MPLS_PW_CNT

```
#define VTSS_EVC_MPLS_PW_CNT VTSS_PORTS
```

Maximum number of PWs that can be attached to one EVC

Definition at line 292 of file vtss_evc_api.h.

10.10.2.7 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.

10.10.2.8 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.

10.10.2.9 VTSS_MCE_ISDX_NONE

```
#define VTSS_MCE_ISDX_NONE 0xFFFFFFFF
```

Allocate no ISDX

Definition at line 752 of file vtss_evc_api.h.

10.10.2.10 VTSS_MCE_ISDX_NEW

```
#define VTSS_MCE_ISDX_NEW 0xFFFFFFFE
```

Allocate new ISDX

Definition at line 753 of file vtss_evc_api.h.

10.10.2.11 VTSS_ISDX_CPU_TX

```
#define VTSS_ISDX_CPU_TX 1023
```

ISDX used for CPU transmissions

Definition at line 754 of file vtss_evc_api.h.

10.10.2.12 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.

10.10.3 Enumeration Type Documentation

10.10.3.1 vtss_ece_type_t

```
enum vtss_ece_type_t
```

ECE frame type.

Enumerator

VTSS_ECE_TYPE_ANY	Any frame type
VTSS_ECE_TYPE_ETYPE	Ethernet Type
VTSS_ECE_TYPE_LLC	LLC
VTSS_ECE_TYPE_SNAP	SNAP
VTSS_ECE_TYPE_IPV4	IPv4
VTSS_ECE_TYPE_IPV6	IPv6

Definition at line 400 of file vtss_evc_api.h.

10.10.3.2 vtss_ece_port_t

```
enum vtss_ece_port_t
```

ECE port type.

Enumerator

VTSS_ECE_PORT_NONE	Port not included
VTSS_ECE_PORT_ROOT	Root UNI port

Definition at line 413 of file vtss_evc_api.h.

10.10.3.3 vtss_mce_tx_lookup_t

```
enum vtss_mce_tx_lookup_t
```

MCE egress lookup types.

Enumerator

VTSS_MCE_TX_LOOKUP_VID	VID lookup
VTSS_MCE_TX_LOOKUP_ISDX	ISDX lookup
VTSS_MCE_TX_LOOKUP_ISDX_PCP	(ISDX, PCP) lookup

Definition at line 729 of file vtss_evc_api.h.

10.10.3.4 vtss_mce_oam_detect_t

```
enum vtss_mce_oam_detect_t
```

MCE OAM detection signalled to VOE.

Enumerator

VTSS_MCE_OAM_DETECT_NONE	No OAM detection
VTSS_MCE_OAM_DETECT_UNTAGGED	Untagged OAM detection
VTSS_MCE_OAM_DETECT_SINGLE_TAGGED	Single tagged OAM detection
VTSS_MCE_OAM_DETECT_DOUBLE_TAGGED	Double tagged OAM detection

Definition at line 736 of file vtss_evc_api.h.

10.10.3.5 vtss_mce_rule_t

```
enum vtss_mce_rule_t
```

MCE rule types.

Enumerator

VTSS_MCE_RULE_BOTH	Ingress and egress rules
VTSS_MCE_RULE_RX	Ingress rules

Definition at line 744 of file vtss_evc_api.h.

10.10.4 Function Documentation

10.10.4.1 vtss_evc_port_conf_get()

```
vtss_rc vtss_evc_port_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_evc_port_conf_t *const conf )
```

Get EVC port configuration.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] EVC port configuration structure.

Returns

Return code.

10.10.4.2 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] EVC port configuration structure.

Returns

Return code.

10.10.4.3 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

inst	[IN] Target instance reference.
policer↩ _id	[IN] Policer ID.
conf	[OUT] Policer configuration.

Returns

Return code.

10.10.4.4 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

inst	[IN] Target instance reference.
policer↩ _id	[IN] Policer ID.
conf	[IN] Policer configuration.

Returns

Return code.

10.10.4.5 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

inst	[IN] Target instance reference.
evc↩ _id	[IN] EVC ID.
conf	[IN] EVC configuration.

Returns

Return code.

10.10.4.6 vtss_evc_del()

```
vtss_rc vtss_evc_del (
            const vtss_inst_t inst,
            const vtss_evc_id_t evc_id )
```

Delete EVC.

Parameters

inst	[IN] Target instance reference.
evc↩ _id	[IN] EVC ID.

Returns

Return code.

10.10.4.7 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

inst	[IN] Target instance reference.
evc↩ _id	[IN] EVC ID.
conf	[OUT] EVC configuration.

Returns

Return code.

10.10.4.8 vtss_evc_hqos_map_get()

```
vtss_rc vtss_evc_hqos_map_get (
            const vtss_inst_t inst,
            const vtss_evc_id_t evc_id,
            const vtss_port_no_t port_no,
            vtss_hqos_id_t *const hqos_id )
```

Get the HQoS ID mapping of a port on an EVC.

Parameters

inst	[IN] Target instance reference.
evc_id	[IN] EVC ID.
port_no	[IN] Port number.
hqos↩ _id	[OUT] HQoS ID.

Returns

Return code.

10.10.4.9 vtss_evc_hqos_map_set()

```
vtss_rc vtss_evc_hqos_map_set (
            const vtss_inst_t inst,
            const vtss_evc_id_t evc_id,
            const vtss_port_no_t port_no,
            const vtss_hqos_id_t hqos_id )
```

Map a port on an EVC to an HQoS ID.

Parameters

inst	[IN] Target instance reference.
evc_id	[IN] EVC ID.
port_no	[IN] Port number.
hqos↩ _id	[IN] HQoS ID.

Returns

Return code.

10.10.4.10 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

inst	[IN] Target instance reference.
type	[IN] ECE type.
ece	[OUT] ECE structure.

Returns

Return code.

10.10.4.11 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

inst	[IN] Target instance reference.
ece↩ _id	[IN] ECE ID. The ECE will be added before the entry with this ID. VTSS_ECE_ID_LAST is reserved for inserting last.
ece	[IN] ECE structure.

Returns

Return code.

10.10.4.12 vtss_ece_del()

```
vtss_rc vtss_ece_del (
            const vtss_inst_t inst,
            const vtss_ece_id_t ece_id )
```

Delete ECE.

Parameters

inst	[IN] Target instance reference.
ece↩ _id	[IN] ECE ID.

Returns

Return code.

10.10.4.13 vtss_evc_oam_port_conf_get()

```
vtss_rc vtss_evc_oam_port_conf_get (
            const vtss_inst_t inst,
            const vtss_evc_id_t evc_id,
            const vtss_port_no_t port_no,
            vtss_evc_oam_port_conf_t *const conf )
```

Get OAM port configuration for (EVC, port)

Parameters

inst	[IN] Target instance reference.
evc_id	[IN] EVC ID.
port_no	[IN] Port number.
conf	[OUT] OAM port configuration.

Returns

Return code.

10.10.4.14 vtss_evc_oam_port_conf_set()

```
vtss_rc vtss_evc_oam_port_conf_set (
            const vtss_inst_t inst,
            const vtss_evc_id_t evc_id,
            const vtss_port_no_t port_no,
            const vtss_evc_oam_port_conf_t *const conf )
```

Set OAM port configuration for (EVC, port)

Parameters

inst	[IN] Target instance reference.
evc_id	[IN] EVC ID.
port_no	[IN] Port number.
conf	[IN] OAM port configuration.

Returns

Return code.

10.10.4.15 vtss_mce_init()

```
vtss_rc vtss_mce_init (
            const vtss_inst_t inst,
            vtss_mce_t *const mce )
```

Initialize MCE to default values.

Parameters

inst	[IN] Target instance reference.
mce	[OUT] MCE structure.

Returns

Return code.

10.10.4.16 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

inst	[IN] Target instance reference.
mce↩ _id	[IN] MCE ID. The MCE will be added before the entry with this ID. VTSS_MCE_ID_LAST is reserved for inserting last.
mce	[IN] MCE structure.

Returns

Return code.

10.10.4.17 vtss_mce_del()

```
vtss_rc vtss_mce_del (
            const vtss_inst_t inst,
            const vtss_mce_id_t mce_id )
```

Delete MCE.

Parameters

inst	[IN] Target instance reference.
mce↩ _id	[IN] MCE ID.

Returns

Return code.

10.10.4.18 vtss_mce_port_info_get()

```
vtss_rc vtss_mce_port_info_get (
            const vtss_inst_t inst,
            const vtss_mce_id_t mce_id,
            const vtss_port_no_t port_no,
            vtss_mce_port_info_t *const info )
```

Get MCE info.

Parameters

inst	[IN] Target instance reference.
mce↩ _id	[IN] MCE ID.
port_no	[IN] Port number.
info	[OUT] MCE information.

Returns

Return code.

10.10.4.19 vtss_evc_counters_get()

```
vtss_rc vtss_evc_counters_get (
            const vtss_inst_t inst,
            const vtss_evc_id_t evc_id,
            const vtss_port_no_t port_no,
            vtss_evc_counters_t *const counters )
```

Get EVC counters for a port.

Parameters

inst	[IN] Target instance reference.
evc_id	[IN] EVC ID.
port_no	[IN] Port number.
counters	[OUT] EVC counters.

Returns

Return code.

10.10.4.20 vtss_evc_counters_clear()

```
vtss_rc vtss_evc_counters_clear (
            const vtss_inst_t inst,
            const vtss_evc_id_t evc_id,
            const vtss_port_no_t port_no )
```

Clear EVC counters for a port.

Parameters

inst	[IN] Target instance reference.
evc_id	[IN] EVC ID.
port_no	[IN] Port number.

Returns

Return code.

10.10.4.21 vtss_ece_counters_get()

```
vtss_rc vtss_ece_counters_get (
            const vtss_inst_t inst,
            const vtss_ece_id_t ece_id,
            const vtss_port_no_t port_no,
            vtss_evc_counters_t *const counters )
```

Get ECE counters for a port.

Parameters

inst	[IN] Target instance reference.
ece_id	[IN] ECE ID.
port_no	[IN] Port number.
counters	[OUT] ECE counters.

Returns

Return code.

10.10.4.22 vtss_ece_counters_clear()

```
vtss_rc vtss_ece_counters_clear (
            const vtss_inst_t inst,
            const vtss_ece_id_t ece_id,
            const vtss_port_no_t port_no )
```

Clear ECE counters for a port.

Parameters

inst	[IN] Target instance reference.
ece_id	[IN] ECE ID.
port_no	[IN] Port number.

Returns

Return code.

10.10.4.23 vtss_mce_counters_get()

```
vtss_rc vtss_mce_counters_get (
            const vtss_inst_t inst,
```

```
            const vtss_mce_id_t mce_id,
            const vtss_port_no_t port_no,
            vtss_evc_counters_t *const counters )
```

Get MCE counters for a port.

Parameters

inst	[IN] Target instance reference.
mce_id	[IN] MCE ID.
port_no	[IN] Port number.
counters	[OUT] MCE counters.

Returns

Return code.

10.10.4.24 vtss_mce_counters_clear()

```
vtss_rc vtss_mce_counters_clear (
            const vtss_inst_t inst,
            const vtss_mce_id_t mce_id,
            const vtss_port_no_t port_no )
```

Clear MCE counters for a port.

Parameters

inst	[IN] Target instance reference.
mce↩ _id	[IN] MCE ID.
port_no	[IN] Port number.

Returns

Return code.

10.11 vtss_api/include/vtss_fdma_api.h File Reference

Frame DMA API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct tag_vtss_fdma_list

  Software DCB structure.
- struct vtss_fdma_tx_info_t

  FDMA Injection Properties.
- struct vtss_fdma_ch_cfg_t

  Channel configuration structure.
- struct vtss_fdma_cfg_t

  FDMA configuration structure.
- struct vtss_fdma_throttle_cfg_t

Macros

- #define VTSS_FDMA_CH_CNT 4
- #define VTSS_PHYS_PORT_CNT 11
- #define VTSS_FDMA_DCB_SIZE_BYTES 16
- #define VTSS_FDMA_HDR_SIZE_BYTES 20
- #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)

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 }

  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_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 cntxt)

  FDMA Interrupt Handler.

10.11.1 Detailed Description

Frame DMA API.

10.11.2 Macro Definition Documentation

10.11.2.1 VTSS_FDMA_CH_CNT

```
#define VTSS_FDMA_CH_CNT 4
```

Number of DMA Channels. Fixed layout

Definition at line 207 of file vtss_fdma_api.h.

10.11.2.2 VTSS_PHYS_PORT_CNT

```
#define VTSS_PHYS_PORT_CNT 11
```

Number of Port Modules, excluding CPU Port Module.

Definition at line 208 of file vtss_fdma_api.h.

10.11.2.3 VTSS_FDMA_DCB_SIZE_BYTES

```
#define VTSS_FDMA_DCB_SIZE_BYTES 16
```

Number of bytes in a DCB.

Definition at line 209 of file vtss_fdma_api.h.

10.11.2.4 VTSS_FDMA_HDR_SIZE_BYTES

```
#define VTSS_FDMA_HDR_SIZE_BYTES 20
```

Max(XTR_HDR_SZ, INJ_HDR_SZ). Worst-case is INJ (16 for IFH + 4 for VLAN tag (if @switch_frm == TRUE)

Definition at line 210 of file vtss_fdma_api.h.

10.11.2.5 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.

10.11.2.6 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.

10.11.2.7 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.

10.11.2.8 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.

10.11.2.9 VTSS_FDMA_MIN_DATA_PER_NON_SOF_DCB_BYTES

```
#define VTSS_FDMA_MIN_DATA_PER_NON_SOF_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.

10.11.2.10 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.

10.11.2.11 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.

10.11.2.12 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.

10.11.2.13 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.

10.11.3 Typedef Documentation

10.11.3.1 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 scenaria, 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_SIZ↩E_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.

10.11.3.2 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.

10.11.4 Enumeration Type Documentation

10.11.4.1 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.

10.11.4.2 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.

Definition at line 1547 of file vtss_fdma_api.h.

10.11.4.3 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.

10.11.5 Function Documentation

10.11.5.1 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

inst	[IN]: Target instance reference.

Returns

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

10.11.5.2 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

inst	[IN]: Target instance reference.
cfg	[IN]: FDMA configuration.

Returns

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

10.11.5.3 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

inst	[IN]: Target instance reference.
list	[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.

10.11.5.4 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 invokation 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

inst	[IN]: Target instance reference.
list	[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.
fdma_info	[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.
tx_info	[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.

10.11.5.5 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

inst	[IN]: Target instance
fdma_info	[IN]: Structure to initialize

Returns

VTSS_RC_OK unless fdma_info == NULL.

10.11.5.6 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

inst	[IN]: Target instance
frm_ptr	[IN]: Frame pointer.

Returns

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

10.11.5.7 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_INCOM↩PLETE. 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

inst	[IN]: Target instance reference.
dcb_cnt	[IN]: Number of DCBs.
dcb_type	[IN]: DCB type requested.
list	[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.

10.11.5.8 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

inst	[IN]: Target instance reference.
cfg	[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.

10.11.5.9 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

inst	[IN]: Target instance reference.
cfg	[IN]: New throttling configuration.

Returns

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

10.11.5.10 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

inst	[IN]: Target instance reference.

Returns

VTSS_RC_OK - always.

10.11.5.11 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

inst	[IN]: Target instance reference.

Returns

VTSS_RC_OK on success.
VTSS_RC_ERROR if supplied parameter was erroneous.

10.11.5.12 vtss_fdma_irq_handler()

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

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 whenver 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 extxracted 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. ISRs 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 scendario 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

inst	[IN]: Target instance reference.
cntxt	[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.

10.12 vtss_api/include/vtss_gfp_api.h File Reference

GFP API.

```
#include <vtss/api/types.h>
```

10.12.1 Detailed Description

GFP API.

10.13 vtss_api/include/vtss_hqos_api.h File Reference

HQoS API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct vtss_hqos_port_conf_t

  HQoS port configuration.
- struct vtss_hqos_conf_t

  Egress QoS setup per HQoS ID.

Functions

- vtss_rc vtss_hqos_port_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_hqos_port_conf_t *const conf)

  Get HQoS port configuration.
- vtss_rc vtss_hqos_port_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_hqos_port_conf_t *const conf)

  Set HQoS port configuration.
- vtss_rc vtss_hqos_add (const vtss_inst_t inst, const vtss_hqos_id_t hqos_id, const vtss_hqos_conf_t *const conf)

  Add or set HQoS entry.
- vtss_rc vtss_hqos_del (const vtss_inst_t inst, const vtss_hqos_id_t hqos_id)

  Delete HQoS entry.
- vtss_rc vtss_hqos_get (const vtss_inst_t inst, const vtss_hqos_id_t hqos_id, vtss_hqos_conf_t *const conf)

  Get HQoS ID configuration.
- vtss_rc vtss_hqos_min_rate_calc (const vtss_inst_t inst, const vtss_hqos_id_t hqos_id, vtss_bitrate_t *const min_rate_calc)

  Calculate guaranteed bandwidth for an HQoS ID.
- vtss_rc vtss_hqos_port_min_rate_calc (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_bitrate_t *const min_rate_calc)

  Calculate non-service guaranteed bandwidth.

10.13.1 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.

10.13.2 Function Documentation

10.13.2.1 vtss_hqos_port_conf_get()

```
vtss_rc vtss_hqos_port_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_hqos_port_conf_t *const conf )
```

Get HQoS port configuration.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] HQoS port configuration structure.

Returns

Return code.

10.13.2.2 vtss_hqos_port_conf_set()

```
vtss_rc vtss_hqos_port_conf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_hqos_port_conf_t *const conf )
```

Set HQoS port configuration.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] HQoS port configuration structure.

Returns

Return code.

10.13.2.3 vtss_hqos_add()

```
vtss_rc vtss_hqos_add (
            const vtss_inst_t inst,
            const vtss_hqos_id_t hqos_id,
            const vtss_hqos_conf_t *const conf )
```

Add or set HQoS entry.

Parameters

inst	[IN] Target instance reference.
hqos↩ _id	[IN] HQoS ID.
conf	[IN] HQoS ID configuration.

Returns

Return code.

10.13.2.4 vtss_hqos_del()

```
vtss_rc vtss_hqos_del (
            const vtss_inst_t inst,
            const vtss_hqos_id_t hqos_id )
```

Delete HQoS entry.

Parameters

inst	[IN] Target instance reference.
hqos↩ _id	[IN] HQoS ID.

Returns

Return code.

10.13.2.5 vtss_hqos_get()

```
vtss_rc vtss_hqos_get (
            const vtss_inst_t inst,
            const vtss_hqos_id_t hqos_id,
            vtss_hqos_conf_t *const conf )
```

Get HQoS ID configuration.

Parameters

inst	[IN] Target instance reference.
hqos↩ _id	[IN] HQoS ID.
conf	[OUT] HQoS ID configuration.

Returns

Return code.

10.13.2.6 vtss_hqos_min_rate_calc()

```
vtss_rc vtss_hqos_min_rate_calc (
            const vtss_inst_t inst,
            const vtss_hqos_id_t hqos_id,
            vtss_bitrate_t *const min_rate_calc )
```

Calculate guaranteed bandwidth for an HQoS ID.

Parameters

inst	[IN] Target instance reference.
hqos_id	[IN] HQoS ID.
min_rate_calc	[OUT] Calculated guaranteed bandwidth.

Returns

Return code.

10.13.2.7 vtss_hqos_port_min_rate_calc()

```
vtss_rc vtss_hqos_port_min_rate_calc (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_bitrate_t *const min_rate_calc )
```

Calculate non-service guaranteed bandwidth.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
min_rate_calc	[OUT] Calculated guaranteed bandwidth.

Returns

Return code.

10.14 vtss_api/include/vtss_i2c_api.h File Reference

I2C API.

```
#include <vtss/api/types.h>
```

10.14.1 Detailed Description

I2C API.

10.15 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 bitsize, 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_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)

10.15.1 Detailed Description

Initialization API.

This header file describes functions used to create and initialize targets

10.15.2 Macro Definition Documentation

10.15.2.1 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.

10.15.3 Typedef Documentation

10.15.3.1 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

chip_no	[IN] Chip number, for targets with multiple chips
addr	[IN] Register address
value	[OUT] Register value

Returns

Return code.

Definition at line 122 of file vtss_init_api.h.

10.15.3.2 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

chip_no	[IN] Chip number, for targets with multiple chips
addr	[IN] Register address
value	[IN] Register value

Returns

Return code.

Definition at line 135 of file vtss_init_api.h.

10.15.3.3 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

port_no	[IN] Port number
i2c_addr	[IN] I2C device address
addr	[IN] Register address
data	[OUT] Pointer the register(s) data value.
cnt	[IN] Number of registers to read
i2c_clk_sel	[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.

10.15.3.4 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

port_no	[IN] Port number
i2c_addr	[IN] I2C device address
data	[OUT] Pointer the data to be written.
cnt	[IN] Number of data bytes to write
i2c_clk_sel	[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.

10.15.3.5 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

inst	[IN] Vitesse API instance.
port_no	[IN] Port number.
bitsize	[IN] Size (in bytes) of bitstream following this parameter.
data	[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.

10.15.3.6 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

inst	[IN] Vitesse API instance.
port_no	[IN] Port number.
read	[IN] Read/Write.
dev	[IN] MMD device number.
reg_num	[IN] Register offset.
data	[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.

10.15.3.7 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

inst	[IN] Vitesse API instance.
port_no	[IN] Port number.
read	[IN] Read/Write.
dev	[IN] MMD device number.
reg_num	[IN] Register offset.
data	[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.

10.15.3.8 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
addr	[IN] Register address (0-31)
value	[OUT] Register value

Returns

Return code.

Definition at line 242 of file vtss_init_api.h.

10.15.3.9 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
addr	[IN] Register address (0-31)
value	[IN] Register value

Returns

Return code.

Definition at line 257 of file vtss_init_api.h.

10.15.3.10 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
mmd	[IN] MMD address (0-31)
addr	[IN] Register address (0-65535)
value	[OUT] Register value

Returns

Return code.

Definition at line 273 of file vtss_init_api.h.

10.15.3.11 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
mmd	[IN] MMD address (0-31)
addr	[IN] Start register address (0-65535)
buf	[OUT] The register values (pointer provided by user)
count	[IN] Number of register reads (increment register reads)

Returns

Return code.

Definition at line 291 of file vtss_init_api.h.

10.15.3.12 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
mmd	[IN] MMD address (0-31)
addr	[IN] Start register address (0-65535)
buf	[IN] The register value

Returns

Return code.

Definition at line 309 of file vtss_init_api.h.

10.15.4 Enumeration Type Documentation

10.15.4.1 vtss_target_type_t

```
enum vtss_target_type_t
```

Target chip type.

Enumerator

VTSS_TARGET_CU_PHY	Cu PHY family
VTSS_TARGET_10G_PHY	10G PHY family
VTSS_TARGET_SPARX_III_11	SparX-III-11 SME switch
VTSS_TARGET_SERVAL_LITE	Serval Lite CE switch
VTSS_TARGET_SERVAL	Serval CE switch
VTSS_TARGET_SEVILLE	Seville switch
VTSS_TARGET_SPARX_III_10_UM	SparxIII-10 unmanaged switch
VTSS_TARGET_SPARX_III_17_UM	SparxIII-17 unmanaged switch
VTSS_TARGET_SPARX_III_25_UM	SparxIII-25 unmanaged switch
VTSS_TARGET_CARACAL_LITE	Caracal-Lite CE switch
VTSS_TARGET_SPARX_III_10	SparxIII-10 switch
VTSS_TARGET_SPARX_III_18	SparxIII-18 switch
VTSS_TARGET_SPARX_III_24	SparxIII-24 switch
VTSS_TARGET_SPARX_III_26	SparxIII-26 switch
VTSS_TARGET_SPARX_III_10_01	SparxIII-10-01 switch
VTSS_TARGET_CARACAL_1	Caracal-1 CE switch
VTSS_TARGET_CARACAL_2	Caracal-2 CE switch
VTSS_TARGET_JAGUAR_1	Jaguar-1 CE switch
VTSS_TARGET_LYNX_1	LynX-1 CE switch
VTSS_TARGET_E_STAX_III_48	E-StaX-III-48
VTSS_TARGET_E_STAX_III_68	E-StaX-III-68
VTSS_TARGET_E_STAX_III_24_DUAL	Dual E-StaX-III-24
VTSS_TARGET_E_STAX_III_68_DUAL	Dual E-StaX-III-68
VTSS_TARGET_DAYTONA	Daytona FEC OTN Phy
VTSS_TARGET_TALLADEGA	Talladega FEC OTN Phy
VTSS_TARGET_SERVAL_2	Serval-2 CE switch
VTSS_TARGET_LYNX_2	LynX-2 CE switch
VTSS_TARGET_JAGUAR_2	Jaguar-2 CE switch
VTSS_TARGET_SPARX_IV_52	Sparx-IV-52 switch
VTSS_TARGET_SPARX_IV_44	Sparx-IV-44 switch
VTSS_TARGET_SPARX_IV_80	Sparx-IV-80 switch
VTSS_TARGET_SPARX_IV_90	Sparx-IV-80 switch

Definition at line 42 of file vtss_init_api.h.

10.15.4.2 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.

10.15.5 Function Documentation

10.15.5.1 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

target	[IN] Target name
create	[IN] Create structure

Returns

Return code.

10.15.5.2 vtss_inst_create()

```
vtss_rc vtss_inst_create (
            const vtss_inst_create_t *const create,
            vtss_inst_t *const inst )
```

Create target instance.

Parameters

create	[IN] Create structure
inst	[OUT] Target instance reference.

Returns

Return code.

10.15.5.3 vtss_inst_destroy()

```
vtss_rc vtss_inst_destroy (
            const vtss_inst_t inst )
```

Destroy target instance.

Parameters

inst	[IN] Target instance reference.

Returns

Return code.

10.15.5.4 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

inst	[IN] Target instance reference
conf	[OUT] Initialization configuration

Returns

Return code.

10.15.5.5 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

inst	[IN] Target instance reference
conf	[IN] Initialization configuration

Returns

Return code.

10.15.5.6 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

inst	[IN] Target instance reference

Returns

Return code.

10.15.5.7 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

inst	[IN] Target instance reference
status	[OUT] Restart status

Returns

Return code.

10.15.5.8 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

inst	[IN] Target instance reference
restart	[OUT] Restart mode

Returns

Return code.

10.15.5.9 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

inst	[IN] Target instance reference
restart	[IN] Restart mode

Returns

Return code.

10.16 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_vcap_port_conf_t

  VCAP port 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_GROU - 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_FORWAR }

  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_vcap_port_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_vcap_port_conf_t *const conf)

  Get VCAP port configuration.
- vtss_rc vtss_vcap_port_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_vcap_port_conf_t *const conf)

  Get VCAP port configuration.
- vtss_rc vtss_isolated_vlan_get (const vtss_inst_t inst, const vtss_vid_t vid, BOOL *const isolated)

  Get enable/disable port isolation for VLAN.
- vtss_rc vtss_isolated_vlan_set (const vtss_inst_t inst, const vtss_vid_t vid, const BOOL isolated)

  Set enable/disable port isolation for VLAN.
- vtss_rc vtss_isolated_port_members_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get the isolated port member set.
- vtss_rc vtss_isolated_port_members_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set the isolated port member set.
- vtss_rc vtss_pvlan_port_members_get (const vtss_inst_t inst, const vtss_pvlan_no_t pvlan_no, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get Private VLAN membership.
- vtss_rc vtss_pvlan_port_members_set (const vtss_inst_t inst, const vtss_pvlan_no_t pvlan_no, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set Private VLAN membership.
- vtss_rc vtss_apvlan_port_members_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL member[VTSS_PORT_ARRAY_SIZE])

Get Asymmetric Private VLAN membership.

- vtss_rc vtss_apvlan_port_members_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set Asymmetric Private VLAN membership.
- vtss_rc vtss_dgroup_port_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_dgroup_port_conf_t *const conf)

  Get Destination Group configuration for port.
- vtss_rc vtss_dgroup_port_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_dgroup_port_conf_t *const conf)

  Set Destination Group configuration for port.
- vtss_rc vtss_sflow_port_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_sflow_port_conf_t *const conf)

  Get port sFlow configuration.
- vtss_rc vtss_sflow_port_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_sflow_port_conf_t *const conf)

  Set port sFlow configuration.
- vtss_rc vtss_sflow_sampling_rate_convert (const vtss_inst_t inst, const BOOL power2, const u32 rate_in, u32 *const rate_out)

  Convert desired sample rate to supported sample rate.
- vtss_rc vtss_aggr_port_members_get (const vtss_inst_t inst, const vtss_aggr_no_t aggr_no, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get aggregation port members.
- vtss_rc vtss_aggr_port_members_set (const vtss_inst_t inst, const vtss_aggr_no_t aggr_no, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set aggregation port members.
- vtss_rc vtss_aggr_mode_get (const vtss_inst_t inst, vtss_aggr_mode_t *const mode)

  Get aggregation traffic distribution mode.
- vtss_rc vtss_aggr_mode_set (const vtss_inst_t inst, const vtss_aggr_mode_t *const mode)

  Set aggregation traffic distribution mode.
- vtss_rc vtss_mirror_conf_get (const vtss_inst_t inst, vtss_mirror_conf_t *const conf)

  Get the mirror configuration.
- vtss_rc vtss_mirror_conf_set (const vtss_inst_t inst, const vtss_mirror_conf_t *const conf)

  Set the mirror configuration.
- vtss_rc vtss_mirror_monitor_port_get (const vtss_inst_t inst, vtss_port_no_t *const port_no)

  Get the mirror monitor port.
- vtss_rc vtss_mirror_monitor_port_set (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Set the mirror monitor port.
- vtss_rc vtss_mirror_ingress_ports_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get the mirror ingress ports.
- vtss_rc vtss_mirror_ingress_ports_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set the mirror ingress ports.
- vtss_rc vtss_mirror_egress_ports_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get the mirror egress ports.
- vtss_rc vtss_mirror_egress_ports_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set the mirror egress ports.
- vtss_rc vtss_mirror_cpu_ingress_get (const vtss_inst_t inst, BOOL *member)

  Get the mirror CPU ingress.
- vtss_rc vtss_mirror_cpu_ingress_set (const vtss_inst_t inst, const BOOL member)

  Set CPU ingress mirroring.
- vtss_rc vtss_mirror_cpu_egress_get (const vtss_inst_t inst, BOOL *member)

  Get the mirror CPU egress.
- vtss_rc vtss_mirror_cpu_egress_set (const vtss_inst_t inst, const BOOL member)

Set the mirror CPU egress.

- vtss_rc vtss_uc_flood_members_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get unicast flood members.
- vtss_rc vtss_uc_flood_members_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set unicast flood members.
- vtss_rc vtss_mc_flood_members_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get multicast flood members.
- vtss_rc vtss_mc_flood_members_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set multicast flood members.
- vtss_rc vtss_ipv4_mc_flood_members_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get IPv4 multicast flood members.
- vtss_rc vtss_ipv4_mc_flood_members_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set IPv4 multicast flood members.
- vtss_rc vtss_ipv6_mc_flood_members_get (const vtss_inst_t inst, BOOL member[VTSS_PORT_ARRAY_SIZE])

  Get IPv6 multicast flood members.
- vtss_rc vtss_ipv6_mc_flood_members_set (const vtss_inst_t inst, const BOOL member[VTSS_PORT_ARRAY_SIZE])

  Set IPv6 multicast flood members.
- vtss_rc vtss_ipv6_mc_ctrl_flood_get (const vtss_inst_t inst, BOOL *const scope)

  Get IPv6 multicast control flooding mode.
- vtss_rc vtss_ipv6_mc_ctrl_flood_set (const vtss_inst_t inst, const BOOL scope)

  Set IPv6 multicast control flooding mode. This controls whether unknown Link-Local scope IPv6 multicasts (FF02:↩ :/16) are flooded to all ports or to the ports in the IPv6 multicast flood mask.
- vtss_rc vtss_eps_port_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_eps_port_conf_t *const conf)

  Get EPS port configuration.
- vtss_rc vtss_eps_port_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_eps_port_conf_t *const conf)

  Set EPS port configuration.
- vtss_rc vtss_eps_port_selector_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_eps_selector_t *const selector)

  Get EPS port selector.
- vtss_rc vtss_eps_port_selector_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_eps_selector_t selector)

  Set EPS port selector.
- vtss_rc vtss_erps_vlan_member_get (const vtss_inst_t inst, const vtss_erpi_t erpi, const vtss_vid_t vid, BOOL *const member)

  Get ERPS member state for a VLAN.
- vtss_rc vtss_erps_vlan_member_set (const vtss_inst_t inst, const vtss_erpi_t erpi, const vtss_vid_t vid, const BOOL member)

  Set ERPS member state for a VLAN.
- vtss_rc vtss_erps_port_state_get (const vtss_inst_t inst, const vtss_erpi_t erpi, const vtss_port_no_t port↩ _no, vtss_erps_state_t *const state)

  Get ERPS state for ERPS instance and port.
- vtss_rc vtss_erps_port_state_set (const vtss_inst_t inst, const vtss_erpi_t erpi, const vtss_port_no_t port_no, const vtss_erps_state_t state)

  Set ERPS state for ERPS instance and port.

10.16.1 Detailed Description

Layer 2 API.

This header file describes Layer 2 switching functions

10.16.2 Macro Definition Documentation

10.16.2.1 VTSS_MAC_ADDRS

```
#define VTSS_MAC_ADDRS 8192
```

Number of MAC addresses

Definition at line 95 of file vtss_l2_api.h.

10.16.2.2 VTSS_MSTIS

```
#define VTSS_MSTIS (65)
```

Number of MSTP instances

Definition at line 514 of file vtss_l2_api.h.

10.16.2.3 VTSS_MSTI_START

```
#define VTSS_MSTI_START (0)
```

MSTI start number

Definition at line 515 of file vtss_l2_api.h.

10.16.2.4 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.

10.16.2.5 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.

10.16.2.6 VTSS_VCL_IDS

```
#define VTSS_VCL_IDS 256
```

Number of VCLs

Definition at line 922 of file vtss_l2_api.h.

10.16.2.7 VTSS_VCL_ID_START

```
#define VTSS_VCL_ID_START 0
```

VCL ID start number

Definition at line 923 of file vtss_l2_api.h.

10.16.2.8 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.

10.16.2.9 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.

10.16.2.10 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.

10.16.2.11 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.

10.16.2.12 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.

10.16.2.13 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.

10.16.2.14 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.

10.16.2.15 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.

10.16.2.16 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.

10.16.2.17 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.

10.16.2.18 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.

10.16.2.19 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.

10.16.2.20 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.

10.16.2.21 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.

10.16.2.22 VTSS_PVLANS

```
#define VTSS_PVLANS (VTSS_PORTS)
```

Number of PVLANs

Definition at line 1304 of file vtss_l2_api.h.

10.16.2.23 VTSS_PVLAN_NO_START

```
#define VTSS_PVLAN_NO_START (0)
```

PVLAN start number

Definition at line 1305 of file vtss_l2_api.h.

10.16.2.24 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.

10.16.2.25 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.

10.16.2.26 VTSS_PVLAN_NO_DEFAULT

```
#define VTSS_PVLAN_NO_DEFAULT (0)
```

Default PVLAN

Definition at line 1308 of file vtss_l2_api.h.

10.16.2.27 VTSS_ERPIS

```
#define VTSS_ERPIS (64)
```

Number of ERPS instances

Definition at line 2228 of file vtss_l2_api.h.

10.16.2.28 VTSS_ERPI_START

```
#define VTSS_ERPI_START (0)
```

ERPI start number

Definition at line 2229 of file vtss_l2_api.h.

10.16.2.29 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.

10.16.2.30 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.

10.16.3 Typedef Documentation

10.16.3.1 vtss_vt_id_t

```
typedef u64 vtss_vt_id_t
```

VLAN Translation ID

Definition at line 1095 of file vtss_l2_api.h.

10.16.4 Enumeration Type Documentation

10.16.4.1 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.

10.16.4.2 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.

10.16.4.3 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.

10.16.4.4 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.

10.16.4.5 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.

10.16.4.6 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.

10.16.4.7 vtss_erps_state_t

```
enum vtss_erps_state_t
```

ERPS state.

Enumerator

VTSS_ERPS_STATE_FORWARDING	Forwarding
VTSS_ERPS_STATE_DISCARDING	Discarding

Definition at line 2266 of file vtss_l2_api.h.

10.16.5 Function Documentation

10.16.5.1 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

inst	[IN] Target instance reference.
entry	[IN] MAC address entry structure.

Returns

Return code.

10.16.5.2 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

inst	[IN] Target instance reference.
vid_mac	[IN] VLAN ID and MAC address structure.

Returns

Return code.

10.16.5.3 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

inst	[IN] Target instance reference.
vid_mac	[IN] VLAN ID and MAC address.
entry	[OUT] MAC address entry.

Returns

Return code.

10.16.5.4 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

inst	[IN] Target instance reference.
vid_mac	[IN] VLAN ID and MAC address.
entry	[OUT] MAC address entry.

Returns

Return code.

10.16.5.5 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

inst	[IN] Target instance reference.
age_time	[OUT] MAC age time in seconds. Value zero disables aging.

Returns

Return code.

10.16.5.6 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

inst	[IN] Target instance reference.
age_time	[IN] MAC age time in seconds. Value zero disables aging.

Returns

Return code.

10.16.5.7 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

inst	[IN] Target instance reference.

Returns

Return code.

10.16.5.8 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.

Returns

Return code.

10.16.5.9 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

inst	[IN] Target instance reference.

Returns

Return code.

10.16.5.10 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.

Returns

Return code.

10.16.5.11 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.

Returns

Return code.

10.16.5.12 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
vid	[IN] VLAN ID.

Returns

Return code.

10.16.5.13 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

inst	[IN] Target instance reference.
status	[OUT] MAC address table status.

Returns

Return code.

10.16.5.14 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
mode	[OUT] Learn mode.

Returns

Return code.

10.16.5.15 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
mode	[IN] Learn mode.

Returns

Return code.

10.16.5.16 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
state	[OUT] Port state, TRUE if link is up.

Returns

Return code.

10.16.5.17 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
state	[IN] Port state, TRUE if link is up.

Returns

Return code.

10.16.5.18 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
state	[OUT] STP state.

Returns

Return code.

10.16.5.19 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
state	[IN] STP state.

Returns

Return code.

10.16.5.20 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
msti	[OUT] MSTP instance.

Returns

Return code.

10.16.5.21 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
msti	[IN] MSTP instance.

Returns

Return code.

10.16.5.22 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
msti	[IN] MSTP instance.
state	[OUT] MSTP state.

Returns

Return code.

10.16.5.23 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
msti	[IN] MSTP instance.
state	[IN] MSTP state.

Returns

Return code.

10.16.5.24 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

inst	[IN] Target instance reference.
conf	[OUT] VLAN configuration structure.

Returns

Return code.

10.16.5.25 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

inst	[IN] Target instance reference.
conf	[IN] VLAN configuration structure.

Returns

Return code.

10.16.5.26 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] VLAN port configuration structure.

Returns

Return code.

10.16.5.27 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] VLAN port configuration structure.

Returns

Return code.

10.16.5.28 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
member	[OUT] VLAN port member list.

Returns

Return code.

10.16.5.29 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
member	[IN] VLAN port member list.

Returns

Return code.

10.16.5.30 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
conf	[OUT] VLAN configuration.

Returns

Return code.

10.16.5.31 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
conf	[IN] VLAN configuration.

Returns

Return code.

10.16.5.32 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
tx_tag	[OUT] Tx tagging list.

Returns

Return code.

10.16.5.33 vtss_vlan_tx_tag_set()

```
vtss_rc vtss_vlan_tx_tag_set (
            const vtss_inst_t inst,
            const vtss_vid_t vid,
            const vtss_vlan_tx_tag_t tx_tag[VTSS_PORT_ARRAY_SIZE] )
```

Get VLAN Tx tagging configuration.

Parameters

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
tx_tag	[IN] Tx tagging list.

Returns

Return code.

10.16.5.34 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] VCL port configuration structure.

Returns

Return code.

10.16.5.35 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] VCL port configuration structure.

Returns

Return code.

10.16.5.36 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

inst	[IN] Target instance reference.
type	[IN] VCE type.
vce	[OUT] VCE structure.

Returns

Return code.

10.16.5.37 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

inst	[IN] Target instance reference.
vce↩ _id	[IN] VCE ID. The VCE will be added before the entry with this ID. VTSS_VCE_ID_LAST is reserved for inserting last.
vce	[IN] VCE structure.

Returns

Return code.

10.16.5.38 vtss_vce_del()

```
vtss_rc vtss_vce_del (
            const vtss_inst_t inst,
            const vtss_vce_id_t vce_id )
```

Delete VCE.

Parameters

inst	[IN] Target instance reference.
vce↩ _id	[IN] VCE ID.

Returns

Return code.

10.16.5.39 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

inst	[IN] Target instance reference.
group↩ _id	[IN] Group ID.
vid	[IN] VLAN ID.
trans_vid	[IN] Translated VLAN ID.

Returns

Return code.

10.16.5.40 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

inst	[IN] Target instance reference.
group↩ _id	[IN] Group ID.
vid	[IN] VLAN ID.

Returns

Return code.

10.16.5.41 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

inst	[IN] Target instance reference.
conf	[INOUT] Pointer to vtss_vlan_trans_grp2vlan_conf_t. Input group_id in the conf structure
next	[IN] Flag to indicate next entry.

Returns

Return code.

10.16.5.42 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

inst	[IN] Target instance reference.
conf	[IN] Pointer to vtss_vlan_trans_port2grp_conf_t.

Returns

Return code.

10.16.5.43 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

inst	[IN] Target instance reference.
conf	[INOUT] Pointer to vtss_vlan_trans_port2grp_conf_t.
next	[IN] Flag to indicate next entry.

Returns

Return code.

10.16.5.44 vtss_vcap_port_conf_get()

```
vtss_rc vtss_vcap_port_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_vcap_port_conf_t *const conf )
```

Get VCAP port configuration.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] VCAP port configuration structure.

Returns

Return code.

10.16.5.45 vtss_vcap_port_conf_set()

```
vtss_rc vtss_vcap_port_conf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_vcap_port_conf_t *const conf )
```

Get VCAP port configuration.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] VCAP port configuration structure.

Returns

Return code.

10.16.5.46 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
isolated	[OUT] VLAN isolation enable/disable option.

Returns

Return code.

10.16.5.47 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID.
isolated	[IN] VLAN isolation enable/disable option.

Returns

Return code.

10.16.5.48 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

inst	[IN] Target instance reference.
member	[OUT] Isolated port member list.

Returns

Return code.

10.16.5.49 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

inst	[IN] Target instance reference.
member	[IN] Isolated port member list.

Returns

Return code.

10.16.5.50 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

inst	[IN] Target instance reference.
pvlan_no	[IN] Private VLAN group number.
member	[OUT] Private VLAN port member list.

Returns

Return code.

10.16.5.51 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

inst	[IN] Target instance reference.
pvlan_no	[IN] Private VLAN group number.
member	[IN] Private VLAN port member list.

Returns

Return code.

10.16.5.52 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

inst	[IN] Target instance reference.
port_no	[IN] Ingress port.
member	[OUT] Asymmetric Private VLAN port member list.

Returns

Return code.

10.16.5.53 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

inst	[IN] Target instance reference.
port_no	[IN] Ingress port.
member	[IN] Asymmetric Private VLAN port member list.

Returns

Return code.

10.16.5.54 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] Destination group port configuration structure.

Returns

Return code.

10.16.5.55 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] Destination group port configuration structure.

Returns

Return code.

10.16.5.56 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

inst	[IN] Target instance reference.
port_no	[IN] Port number (a.k.a. data source).
conf	[OUT] sFlow sampler configuration.

Returns

Return code.

10.16.5.57 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

inst	[IN] Target instance reference.
port_no	[IN] Port number (a.k.a. data source).
conf	[IN] sFlow sampler configuration.

Returns

Return code.

10.16.5.58 vtss_sflow_sampling_rate_convert()

```
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.

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

inst	[IN] Target instance reference.
power2	[IN] Only return sampling rates in powers of two.
rate_in	[IN] Desired sample rate
rate_out	[OUT] Realizable sample rate

Returns

Return code.

10.16.5.59 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

inst	[IN] Target instance reference.
aggr_no	[IN] Aggregation number.
member	[OUT] Aggregation port member list.

Returns

Return code.

10.16.5.60 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

inst	[IN] Target instance reference.
aggr_no	[IN] Aggregation number.
member	[IN] Aggregation port member list.

Returns

Return code.

10.16.5.61 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

inst	[IN] Target instance reference.
mode	[OUT] Distribution mode structure.

Returns

Return code.

10.16.5.62 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

inst	[IN] Target instance reference.
mode	[IN] Distribution mode structure.

Returns

Return code.

10.16.5.63 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

inst	[IN] Target instance reference.
conf	[OUT] Mirror configuration.

Returns

Return code.

10.16.5.64 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

inst	[IN] Target instance reference.
conf	[IN] Mirror configuration.

Returns

Return code.

10.16.5.65 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

inst	[IN] Target instance reference.
port_no	[OUT] Port number.

Returns

Return code.

10.16.5.66 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

inst	[IN] Target instance reference.
port_no	[IN] Port number or VTSS_PORT_NO_NONE.

Returns

Return code.

10.16.5.67 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

inst	[IN] Target instance reference.
member	[OUT] Port member list. If a port is enabled in this array, frames received on the port are mirrored.

Returns

Return code.

10.16.5.68 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

inst	[IN] Target instance reference.
member	[IN] Port member list. If a port is enabled in this array, frames received on the port are mirrored.

Returns

Return code.

10.16.5.69 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

inst	[IN] Target instance reference.
member	[OUT] Port member list. If a port is enabled in this array, frames transmitted on the port are mirrored.

Returns

Return code.

10.16.5.70 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

inst	[IN] Target instance reference.
member	[IN] Port member list. If a port is enabled in this array, frames transmitted on the port are mirrored.

Returns

Return code.

10.16.5.71 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

inst	[IN] Target instance reference.
member	[OUT] CPU member. If a CPU is enabled, frames received to the CPU port are mirrored.

Returns

Return code.

10.16.5.72 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

inst	[IN] Target instance reference.
member	[IN] CPU member. If a CPU is enabled, frames received by the CPU port are mirrored.

Returns

Return code.

10.16.5.73 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

inst	[IN] Target instance reference.
member	[OUT] CPU member. If a CPU is enabled, frames transmitted by the CPU port are mirrored.

Returns

Return code.

10.16.5.74 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

inst	[IN] Target instance reference.
member	[IN] CPU member. If a CPU is enabled, frames transmitted by the CPU the port are mirrored.

Returns

Return code.

10.16.5.75 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

inst	[IN] Target instance reference.
member	[OUT] Port member list.

Returns

Return code.

10.16.5.76 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

inst	[IN] Target instance reference.
member	[IN] Port member list.

Returns

Return code.

10.16.5.77 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

inst	[IN] Target instance reference.
member	[OUT] Port member list.

Returns

Return code.

10.16.5.78 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

inst	[IN] Target instance reference.
member	[IN] Port member list.

Returns

Return code.

10.16.5.79 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

inst	[IN] Target instance reference.
member	[OUT] Port member list. Ports connected to IPv4 multicast routers should be enabled.

Returns

Return code.

10.16.5.80 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

inst	[IN] Target instance reference.
member	[IN] Port member list. Ports connected to IPv4 multicast routers should be enabled.

Returns

Return code.

10.16.5.81 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

inst	[IN] Target instance reference.
member	[OUT] Port member list. Ports connected to IPv6 multicast routers should be enabled.

Returns

Return code.

10.16.5.82 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

inst	[IN] Target instance reference.
member	[IN] Port member list. Ports connected to IPv6 multicast routers should be enabled.

Returns

Return code.

10.16.5.83 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

inst	[IN] Target instance reference.
scope	[OUT] IPv6 multicast control flood. 0: Flood to all ports. 1: Flood to IPv6 multicast flood members.

Returns

Return code.

10.16.5.84 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

inst	[IN] Target instance reference.
scope	[IN] IPv6 multicast control flood. 0: Flood to all ports. 1: Flood to IPv6 multicast flood members.

Returns

Return code.

10.16.5.85 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

inst	[IN] Target instance reference.
port_no	[IN] Working port.
conf	[OUT] Protection configuration.

Returns

Return code.

10.16.5.86 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

inst	[IN] Target instance reference.
port_no	[IN] Working port.
conf	[IN] Protection configuration.

Returns

Return code.

10.16.5.87 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

inst	[IN] Target instance reference.
port_no	[IN] Working port.
selector	[OUT] Selector.

Returns

Return code.

10.16.5.88 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

inst	[IN] Target instance reference.
port_no	[IN] Working port.
selector	[IN] Selector.

Returns

Return code.

10.16.5.89 vtss_erps_vlan_member_get()

```
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.

Parameters

inst	[IN] Target instance reference.
erpi	[IN] ERPS instance.
vid	[IN] VLAN ID.
member	[OUT] Membership, TRUE if VLAN is included in ERPS instance.

Returns

Return code.

10.16.5.90 vtss_erps_vlan_member_set()

```
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.

Parameters

inst	[IN] Target instance reference.
erpi	[IN] ERPS instance.
vid	[IN] VLAN ID.
member	[IN] Membership, TRUE if VLAN is included in ERPS instance.

Returns

Return code.

10.16.5.91 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

inst	[IN] Target instance reference.
erpi	[IN] ERPS instance.
port_no	[IN] Port number.
state	[OUT] ERPS state.

Returns

Return code.

10.16.5.92 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
erpi	[IN] ERPS instance.
state	[IN] ERPS state.

Returns

Return code.

10.17 vtss_api/include/vtss_l3_api.h File Reference

L3 routing API.

```
#include <vtss/api/types.h>
```

10.17.1 Detailed Description

L3 routing API.

This header file describes L3 IPv4/IPv6 hardware assisted routing functions.

10.18 vtss_api/include/vtss_mac10g_api.h File Reference

MAC10G API.

```
#include <vtss/api/types.h>
```

10.18.1 Detailed Description

MAC10G API.

10.19 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_COUN }

  Trace groups.
- enum vtss_trace_level_t { VTSS_TRACE_LEVEL_NONE, VTSS_TRACE_LEVEL_ERROR, VTSS_TRACE_LEVEL_INFO, VTSS_TRACE_LEVEL_DEBU 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_POR VTSS_DEBUG_GROUP_PORT_CNT, VTSS_DEBUG_GROUP_PHY, VTSS_DEBUG_GROUP_VLAN, VTSS_DEBUG_GROUP_PVLAN, VTSS_DEBUG_GROUP_MAC_TABLE, VTSS_DEBUG_GROUP_ACL, VTSS_DEBUG_GROUP_QOS, VTSS_DEBUG_GROUP_AGGR, VTSS_DEBUG_GROUP_GLAG, VTSS_DEBUG_GROUP_STP, VTSS_DEBUG_GROUP_MIRROR, VTSS_DEBUG_GROUP_EVC, VTSS_DEBUG_GROUP_ERPS, VTSS_DEBUG_GROUP_EPS, VTSS_DEBUG_GROUP_PACKET, VTSS_DEBUG_GROUP_FDMA, VTSS_DEBUG_GROUP_TS, VTSS_DEBUG_GROUP_PHY_TS, VTSS_DEBUG_GROUP_WM, VTSS_DEBUG_GROUP_LR VTSS_DEBUG_GROUP_IPMC, VTSS_DEBUG_GROUP_STACK, VTSS_DEBUG_GROUP_CMEF, VTSS_DEBUG_GROUP_HOST, VTSS_DEBUG_GROUP_MPLS, VTSS_DEBUG_GROUP_MPLS_OAM, VTSS_DEBUG_GROUP_HQOS, VTSS_DEBUG_GROUP_VXLAT, VTSS_DEBUG_GROUP_OAM, VTSS_DEBUG_GROUP_SER_GPIO, VTSS_DEBUG_GROUP_L3, VTSS_DEBUG_GROUP_AFI, VTSS_DEBUG_GROUP_MACSEC, VTSS_DEBUG_GROUP_COUNT }

  Debug function group.

- enum vtss_ptp_event_type_t { **VTSS_PTP_SYNC_EV** = (1 << 0), **VTSS_PTP_EXT_SYNC_EV** = (1 << 1), **VTSS_PTP_CLK_ADJ_EV** = (1 << 2), **VTSS_PTP_TX_TSTAMP_EV** = (1 << 3), **VTSS_PTP_EXT_1_SYNC_EV** =(1 << 4) }

  Define event (interrupt) types relatesd to PTP in the switch chips.

- enum vtss_dev_all_event_type_t { **VTSS_DEV_ALL_TX_TSTAMP_EV** = (1 << 0), **VTSS_DEV_ALL_LI↩NK_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**, **V↩TSS_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 prntf, 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 periodicly.

- 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 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.

- 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.

10.19.1 Detailed Description

Miscellaneous API.

This header file describes miscellaneous API functions

10.19.2 Macro Definition Documentation

10.19.2.1 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.

10.19.2.2 VTSS_OS_TIMESTAMP

```
#define VTSS_OS_TIMESTAMP(
            timestamp )
```

Value:

```
do {                                                                    \
    /* Currently no need to lock scheduler, since it's only          */ \
    /* called from a function, where the sceduler is already locked. */ \
    /* cyg_scheduler_lock(__FILE__, __LINE__); */                                  \
    (timestamp)->hw_cnt = vtss_tod_get_ns_cnt();                        \
    /* cyg_scheduler_unlock(__FILE__, __LINE__); */                                \
  } while(0);
```

VTSS_OS_TIMESTAMP() provides the implementation that will fill in the timestamp.

Definition at line 1076 of file vtss_misc_api.h.

10.19.3 Typedef Documentation

10.19.3.1 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.

10.19.4 Enumeration Type Documentation

10.19.4.1 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.

10.19.4.2 vtss_trace_group_t

```
enum vtss_trace_group_t
```

Trace groups.

Enumerator

VTSS_TRACE_GROUP_DEFAULT	Default trace group
VTSS_TRACE_GROUP_PORT	Port control
VTSS_TRACE_GROUP_PHY	PHY control
VTSS_TRACE_GROUP_PACKET	Packet control
VTSS_TRACE_GROUP_AFI	AFI
VTSS_TRACE_GROUP_QOS	Quality of Service
VTSS_TRACE_GROUP_L2	Layer 2
VTSS_TRACE_GROUP_L3	Layer 3
VTSS_TRACE_GROUP_SECURITY	Security
VTSS_TRACE_GROUP_EVC	Ethernet Virtual Connections
VTSS_TRACE_GROUP_FDMA_NORMAL	Frame DMA Extraction and Injection when interrupts/scheduler is enabled
VTSS_TRACE_GROUP_FDMA_IRQ	Frame DMA when interrupts/scheduler is disabled Daytona layers are placed before the PHY layer, othewise they are not shown in the CLI commands
VTSS_TRACE_GROUP_REG_CHECK	Register access errors (must be able to print when interrupts/scheduler is disabled)
VTSS_TRACE_GROUP_MPLS	MPLS
VTSS_TRACE_GROUP_HQOS	Hierarchical Quality of Service
VTSS_TRACE_GROUP_MACSEC	MACSEC control
VTSS_TRACE_GROUP_VCAP	VCAP
VTSS_TRACE_GROUP_OAM	OAM
VTSS_TRACE_GROUP_TS	Timestamping
VTSS_TRACE_GROUP_COUNT	Number of trace groups

Definition at line 52 of file vtss_misc_api.h.

10.19.4.3 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.

10.19.4.4 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.

10.19.4.5 vtss_debug_group_t

```
enum vtss_debug_group_t
```

Debug function group.

Enumerator

VTSS_DEBUG_GROUP_ALL	All groups
VTSS_DEBUG_GROUP_INIT	Initialization
VTSS_DEBUG_GROUP_MISC	Miscellaneous
VTSS_DEBUG_GROUP_PORT	Port configuration
VTSS_DEBUG_GROUP_PORT_CNT	Port counters
VTSS_DEBUG_GROUP_PHY	PHY
VTSS_DEBUG_GROUP_VLAN	VLAN
VTSS_DEBUG_GROUP_PVLAN	PVLAN
VTSS_DEBUG_GROUP_MAC_TABLE	MAC address table
VTSS_DEBUG_GROUP_ACL	ACL

Enumerator

VTSS_DEBUG_GROUP_QOS	QoS
VTSS_DEBUG_GROUP_AGGR	Link aggregation
VTSS_DEBUG_GROUP_GLAG	Global link aggregation
VTSS_DEBUG_GROUP_STP	Spanning Tree
VTSS_DEBUG_GROUP_MIRROR	Mirroring
VTSS_DEBUG_GROUP_EVC	EVC
VTSS_DEBUG_GROUP_ERPS	ERPS
VTSS_DEBUG_GROUP_EPS	EPS
VTSS_DEBUG_GROUP_PACKET	Packet control
VTSS_DEBUG_GROUP_FDMA	FDMA
VTSS_DEBUG_GROUP_TS	TS: TimeStamping
VTSS_DEBUG_GROUP_PHY_TS	PHY_TS: PHY TimeStamping
VTSS_DEBUG_GROUP_WM	WaterMarks
VTSS_DEBUG_GROUP_LRN	LRN:COMMON
VTSS_DEBUG_GROUP_IPMC	IP Multicast
VTSS_DEBUG_GROUP_STACK	Stacking
VTSS_DEBUG_GROUP_CMEF	Congestion Management
VTSS_DEBUG_GROUP_HOST	CE-MAX Host configuration
VTSS_DEBUG_GROUP_MPLS	MPLS
VTSS_DEBUG_GROUP_MPLS_OAM	MPLS OAM
VTSS_DEBUG_GROUP_HQOS	Hierarchical Quality of Service
VTSS_DEBUG_GROUP_VXLAT	VLAN Translation
VTSS_DEBUG_GROUP_OAM	OAM, incl. VOEs/VOP
VTSS_DEBUG_GROUP_SER_GPIO	Serial GPIO configuration
VTSS_DEBUG_GROUP_L3	L3 services
VTSS_DEBUG_GROUP_AFI	Automatic Frame Injector
VTSS_DEBUG_GROUP_MACSEC	802.1AE MacSec
VTSS_DEBUG_GROUP_COUNT	Number of groups

Definition at line 184 of file vtss_misc_api.h.

10.19.4.6 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.

10.19.4.7 vtss_sgpio_mode_t

```
enum vtss_sgpio_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.

10.19.4.8 vtss_sgpio_bmode_t

```
enum vtss_sgpio_bmode_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.

10.19.4.9 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.

10.19.4.10 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.

10.19.5 Function Documentation

10.19.5.1 vtss_trace_conf_get()

```
vtss_rc vtss_trace_conf_get (
            const vtss_trace_group_t group,
            vtss_trace_conf_t *const conf )
```

Get trace configuration.

Parameters

group	[IN] Trace group
conf	[OUT] Trace group configuration.

Returns

Return code.

10.19.5.2 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

group	[IN] Trace group
conf	[IN] Trace group configuration.

Returns

Return code.

10.19.5.3 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

layer	[IN] Trace layer
group	[IN] Trace group
level	[IN] Trace level
file	[IN] File name string
line	[IN] Line number in file
function	[IN] Function name string
format	[IN] Print format string

Returns

Nothing.

10.19.5.4 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

layer	[IN] Trace layer
group	[IN] Trace group
level	[IN] Trace level
file	[IN] The file from where the trace were called.
line	[IN] The line from where the trace were called.
function	[IN] The function from where the trace were called.
byte_p	[IN] Pointer to start of area to print
byte_cnt	[IN] Number of bytes to print

Returns

Nothing.

10.19.5.5 vtss_debug_info_get()

```
vtss_rc vtss_debug_info_get (
            vtss_debug_info_t *const info )
```

Get default debug information structure.

Parameters

info	[OUT] Debug information

Returns

Return code.

10.19.5.6 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

inst	[IN] Target instance reference.
prntf	[IN] Debug printf function.
info	[IN] Debug information

Returns

Return code.

10.19.5.7 vtss_callout_lock()

```
void vtss_callout_lock (
            const vtss_api_lock_t *const lock )
```

Lock API access.

Parameters

lock	[IN] Lock information

10.19.5.8 vtss_callout_unlock()

```
void vtss_callout_unlock (
            const vtss_api_lock_t *const lock )
```

Unlock API access.

Parameters

lock	[IN] Lock information

10.19.5.9 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

inst	[IN] Target instance reference.
lock	[IN] Lock information

Returns

Return code.

10.19.5.10 vtss_debug_unlock()

```
vtss_rc vtss_debug_unlock (
            const vtss_inst_t inst,
            vtss_debug_lock_t *const lock )
```

Debug unlock API access.

Parameters

inst	[IN] Target instance reference.
lock	[IN] Lock information

Returns

Return code.

10.19.5.11 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
addr	[IN] Address to read. Format depends on target.
value	[OUT] Register value.

Returns

Return code.

10.19.5.12 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
addr	[IN] Address to read. Format depends on target.
value	[IN] Register value.

Returns

Return code.

10.19.5.13 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
addr	[IN] Address to read. Format depends on target.
value	[IN] Register value.
mask	[IN] Register mask, only bits enabled are changed.

Returns

Return code.

10.19.5.14 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

inst	[IN] Target instance reference.
ext	[IN] EXT number (0-1).

Returns

Return code.

10.19.5.15 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

inst	[IN] Target instance reference.
chip↩ _id	[IN] Pointer to chip ID structure.

Returns

Return code.

10.19.5.16 vtss_poll_1sec()

```
vtss_rc vtss_poll_1sec (
            const vtss_inst_t inst )
```

Polling function called every second.

Parameters

inst	[IN] Target instance reference.

Returns

Return code.

10.19.5.17 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 periodicly.

Parameters

inst	[IN] Target instance reference.
ev_mask	[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.

10.19.5.18 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

inst	[IN] Target instance reference.
enable	[IN] Enable or disable events
ev_mask	[IN] Event type(s) to control (mask)

Returns

Return code.

10.19.5.19 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

inst	[IN] Target instance reference.
poll_type	[IN] Polling type
ev_mask	[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.

10.19.5.20 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
enable	[IN] Enable or disable events.
ev_mask	[IN] Event type(s) to control (mask).

Returns

Return code.

10.19.5.21 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
gpio_no	[IN] GPIO pin number.
mode	[IN] GPIO mode.

Returns

Return code.

10.19.5.22 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
gpio_no	[IN] GPIO pin number.
output	[IN] TRUE if output, FALSE if input.

Returns

Return code.

DEPRECATED. Use vtss_gpio_mode_set() instead.

10.19.5.23 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
gpio_no	[IN] GPIO pin number.
value	[OUT] TRUE if pin is high, FALSE if it is low.

Returns

Return code.

10.19.5.24 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
gpio_no	[IN] GPIO pin number.
value	[IN] TRUE to set pin high, FALSE to set pin low.

Returns

Return code.

10.19.5.25 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
events	[OUT] Event indication for each GPIO pin - must point to VTSS_GPIOS of BOOL.

Returns

Return code.

10.19.5.26 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

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
gpio_no	[IN] GPIO pin number.
enable	[IN] Enable or disable event.

Returns

Return code.

10.19.5.27 vtss_sgpio_conf_get()

```
vtss_rc vtss_sgpio_conf_get (
            const vtss_inst_t inst,
            const vtss_chip_no_t chip_no,
            const vtss_sgpio_group_t group,
            vtss_sgpio_conf_t *const conf )
```

Get SGPIO configuration.

Parameters

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
group	[IN] SGPIO group.
conf	[OUT] SGPIO configuration.

Returns

Return code.

10.19.5.28 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 SGPIO configuration.

Parameters

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
group	[IN] SGPIO group.
conf	[IN] SGPIO configuration.

Returns

Return code.

10.19.5.29 vtss_sgpio_read()

```
vtss_rc vtss_sgpio_read (
            const vtss_inst_t inst,
            const vtss_chip_no_t chip_no,
            const vtss_sgpio_group_t group,
            vtss_sgpio_port_data_t data[VTSS_SGPIO_PORTS] )
```

Read SGPIO data.

Parameters

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
group	[IN] SGPIO group.
data	[OUT] SGPIO data.

Returns

Return code.

10.19.5.30 vtss_sgpio_event_poll()

```
vtss_rc vtss_sgpio_event_poll (
            const vtss_inst_t inst,
            const vtss_chip_no_t chip_no,
            const vtss_sgpio_group_t group,
            const u32 bit,
            BOOL *const events )
```

Get SGPIO event indication.

Parameters

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
group	[IN] SGPIO group.
bit	[IN] SGPIO port bit (0-3).
events	[OUT] Event indication for each port for the selected bit - must point to VTSS_SGPIO_PORTS of BOOL.

Returns

Return code.

10.19.5.31 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 SGPIO event enable.

Parameters

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
group	[IN] SGPIO group.
port	[IN] SGPIO port (0-31).
bit	[IN] SGPIO port bit (0-3).
enable	[IN] Event for each port for the selected bit is enabled or disabled.

Returns

Return code.

10.19.5.32 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

inst	[IN] Target instance reference.
mask	[IN] Interrupt mask - Configures the interrupts for the bits set in the mask.
polarity	[IN] Polarity - Interrupt polarity.
enable	[IN] Enable - 1 = enable, 0 = disable.

Returns

Return code.

10.19.5.33 vtss_intr_mask_set()

```
vtss_rc vtss_intr_mask_set (
            const vtss_inst_t inst,
            vtss_intr_t * mask )
```

Set the interrupt mask.

Parameters

inst	[IN] Target instance reference.
mask	[IN] Pointer to mask structure.

Returns

Return code.

10.19.5.34 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

inst	[IN] Target instance reference.
status	[OUT] Pointer to a structure with status of all enabled interrupt sources.

Returns

Return code.

10.19.5.35 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

inst	[IN] Target instance reference.

Returns

Return code.

10.19.5.36 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

inst	[IN] Target instance reference.
irq	[IN] Interrupt source.
conf	[OUT] IRQ configuration.

Returns

Return code.

10.19.5.37 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

inst	[IN] Target instance reference.
irq	[IN] Interrupt source.
conf	[IN] IRQ configuration.

Returns

Return code.

10.19.5.38 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

inst	[IN] Target instance reference.
status	[OUT] IRQ status.

Returns

Return code.

10.19.5.39 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

inst	[IN] Target instance reference.
irq	[IN] Interrupt source.
enable	[IN] Enable or disable source.

Returns

Return code.

10.19.5.40 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

10.19.5.41 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

cb	pointer to callback function

10.19.5.42 vtss_temp_sensor_init()

```
vtss_rc vtss_temp_sensor_init (
            const vtss_inst_t inst,
            const BOOL enable )
```

Initialize the temperature sensor.

Parameters

inst	[IN] Target instance reference
enable	[IN] Set to true if sensor shall be active else false

Returns

Return code.

10.19.5.43 vtss_temp_sensor_get()

```
vtss_rc vtss_temp_sensor_get (
            const vtss_inst_t inst,
            i16 * temperature )
```

Read temperature sensor value.

Parameters

inst	[IN] Target instance reference
temperature	[OUT] Temperature from sensor (range from -46 to 135 degC)

Returns

Return code.

10.19.5.44 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

inst	[IN] Target instance reference
fan_spec	[IN] Fan specification
rotation_count	[OUT] Number of fan rotation countered for the last second.

Returns

Return code.

10.19.5.45 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

inst	[IN] Target instance reference
lvl	[IN] Level. 0 = Fan off, 255 = fan fully on

Returns

Return code.

10.19.5.46 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

inst	[IN] Target instance reference
spec	[IN] Fan specifications

Returns

Return code.

10.19.5.47 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

inst	[IN] Target instance reference
lvl	[IN] Level. 0 = Fan off, 255 = fan fully on

Returns

Return code.

10.19.5.48 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

inst	[IN] Target instance reference
port_no	[IN] Port number
eee_conf	[IN] EEE configuration

Returns

Return code.

10.19.5.49 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

inst	[IN] Target instance reference
port_no	[IN] Port number
eee_state	[IN] New port state

Returns

Return code.

10.19.5.50 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

inst	[IN] Target instance reference
port_no	[IN] Port number
eee_counter	[INOUT] Structure indicating which counters to get, and the returned counter value.

Returns

Return code.

10.19.5.51 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

inst	[IN] Target instance reference.
enable	[IN] Enable or disable register access checking (ref. counted).

Returns

Return code.

10.20 vtss_api/include/vtss_mpls_api.h File Reference

MPLS API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct vtss_mpls_segment_status_t

  Segment status. Contains various kinds of status/state information for a segment.
- struct vtss_mpls_label_t

  MPLS label. The TTL value is only relevant for LER out-segments.
- struct vtss_mpls_qos_to_tc_map_entry_t

  QoS-to-TC map entry.
- struct vtss_mpls_tc_to_qos_map_entry_t

  TC-to-(QoS,DP) map entry.

- struct vtss_mpls_tc_conf_t

  TC/QoS map configuration.
- struct vtss_mpls_oam_conf_t

  OAM configuration.
- struct vtss_mpls_l2_t

  Layer-2 configuration.
- struct vtss_mpls_pw_conf_t

  Pseudo-wire configuration.
- struct vtss_mpls_segment_t

  Segment configuration.
- struct vtss_mpls_xc_t

  Cross Connect function for unidirectional LSP/PW.

Macros

- #define VTSS_MPLS_LABEL_VALUE_DONTCARE 0xffffffff
- #define VTSS_MPLS_TC_VALUE_DONTCARE 0xff
- #define VTSS_MPLS_IDX_UNDEFINED (-1)
- #define VTSS_MPLS_IDX_IS_UNDEF(idx) ((idx) == VTSS_MPLS_IDX_UNDEFINED)
- #define VTSS_MPLS_IDX_IS_DEF(idx) ((idx) != VTSS_MPLS_IDX_UNDEFINED)
- #define VTSS_MPLS_IDX_UNDEF(idx) do { (idx) = VTSS_MPLS_IDX_UNDEFINED; } while (0)
- #define VTSS_MPLS_QOS_TO_TC_MAP_CNT 8
- #define VTSS_MPLS_QOS_TO_TC_ENTRY_CNT 8
- #define VTSS_MPLS_TC_TO_QOS_MAP_CNT 8
- #define VTSS_MPLS_TC_TO_QOS_ENTRY_CNT 8

Typedefs

- typedef u32 vtss_mpls_label_value_t
- typedef u8 vtss_mpls_tc_t
- typedef u8 vtss_mpls_cos_t
- typedef u8 vtss_mpls_qos_t
- typedef i16 vtss_mpls_segment_idx_t
- typedef i16 vtss_mpls_xc_idx_t
- typedef i16 vtss_mpls_l2_idx_t

Enumerations

- enum vtss_mll_tagtype_t { VTSS_MPLS_TAGTYPE_UNTAGGED = 0, VTSS_MPLS_TAGTYPE_CTAGGED = 1, VTSS_MPLS_TAGTYPE_STAGGED = 2 }

  Frame tagging. Used both for ingress and egress.
- enum vtss_mpls_segment_state_t { VTSS_MPLS_SEGMENT_STATE_UNCONF, VTSS_MPLS_SEGMENT_STATE_CONF, VTSS_MPLS_SEGMENT_STATE_UP }

  Segment state. This expresses whether a segment is sufficiently configured to be able to operate, and in that case, whether it has been able to allocate the necessary hardware resources.
- enum vtss_mpls_xc_type_t { VTSS_MPLS_XC_TYPE_LSR, VTSS_MPLS_XC_TYPE_LER }

  Cross-connect (XC) types.
- enum vtss_mpls_tunnel_mode_t { VTSS_MPLS_TUNNEL_MODE_PIPE, VTSS_MPLS_TUNNEL_MODE_SHORT_PIPE, VTSS_MPLS_TUNNEL_MODE_UNIFORM }

  DiffServ tunnel modes for TC/TTL.
- enum vtss_mpls_oam_type_t { VTSS_MPLS_OAM_TYPE_NONE, VTSS_MPLS_OAM_TYPE_VCCV1, VTSS_MPLS_OAM_TYPE_VCCV2, VTSS_MPLS_OAM_TYPE_VCCV3, VTSS_MPLS_OAM_TYPE_VCCV4, VTSS_MPLS_OAM_TYPE_GAL_MEP, VTSS_MPLS_OAM_TYPE_GAL_MIP }

  MPLS Pseudo-wire + LSP OAM types.

Functions

- vtss_rc vtss_mpls_l2_alloc (vtss_inst_t inst, vtss_mpls_l2_idx_t *const idx)

  Allocate L2 entry.
- vtss_rc vtss_mpls_l2_free (vtss_inst_t inst, const vtss_mpls_l2_idx_t idx)

  Free L2 entry.
- vtss_rc vtss_mpls_l2_get (vtss_inst_t inst, const vtss_mpls_l2_idx_t idx, vtss_mpls_l2_t *const l2)

  Get L2 entry.
- vtss_rc vtss_mpls_l2_set (vtss_inst_t inst, const vtss_mpls_l2_idx_t idx, const vtss_mpls_l2_t *const l2)

  Set L2 entry.
- vtss_rc vtss_mpls_l2_segment_attach (vtss_inst_t inst, const vtss_mpls_l2_idx_t l2_idx, const vtss_mpls_segment_idx_t seg_idx)

  Attach segment to L2 entry.
- vtss_rc vtss_mpls_l2_segment_detach (vtss_inst_t inst, const vtss_mpls_segment_idx_t seg_idx)

  Detach segment from L2 entry.
- vtss_rc vtss_mpls_segment_alloc (vtss_inst_t inst, const BOOL is_in, vtss_mpls_segment_idx_t *const idx)

  Allocate Segment entry.
- vtss_rc vtss_mpls_segment_free (vtss_inst_t inst, const vtss_mpls_segment_idx_t idx)

  Free segment entry.
- vtss_rc vtss_mpls_segment_get (vtss_inst_t inst, const vtss_mpls_segment_idx_t idx, vtss_mpls_segment_t *const seg)

  Get segment entry.
- vtss_rc vtss_mpls_segment_set (vtss_inst_t inst, const vtss_mpls_segment_idx_t idx, const vtss_mpls_segment_t *const seg)

  Set segment entry.
- vtss_rc vtss_mpls_segment_status_get (vtss_inst_t inst, const vtss_mpls_segment_idx_t idx, vtss_mpls_segment_status_t *const status)

  Get current segment status.
- vtss_rc vtss_mpls_segment_server_attach (vtss_inst_t inst, const vtss_mpls_segment_idx_t idx, const vtss_mpls_segment_idx_t srv_idx)

  Attach segment to server segment.
- vtss_rc vtss_mpls_segment_server_detach (vtss_inst_t inst, const vtss_mpls_segment_idx_t seg_idx)

  Detach segment from server segment.
- vtss_rc vtss_mpls_xc_alloc (vtss_inst_t inst, const vtss_mpls_xc_type_t type, vtss_mpls_xc_idx_t *const idx)

  Allocate XC entry.
- vtss_rc vtss_mpls_xc_free (vtss_inst_t inst, const vtss_mpls_xc_idx_t idx)

  Free XC entry.
- vtss_rc vtss_mpls_xc_get (vtss_inst_t inst, const vtss_mpls_xc_idx_t idx, vtss_mpls_xc_t *const xc)

  Get XC entry.
- vtss_rc vtss_mpls_xc_set (vtss_inst_t inst, const vtss_mpls_xc_idx_t idx, const vtss_mpls_xc_t *const xc)

  Set XC entry.
- vtss_rc vtss_mpls_xc_segment_attach (vtss_inst_t inst, const vtss_mpls_xc_idx_t xc_idx, const vtss_mpls_segment_idx_t seg_idx)

  Attach segment to XC.
- vtss_rc vtss_mpls_xc_segment_detach (vtss_inst_t inst, const vtss_mpls_segment_idx_t seg_idx)

  Detach segment from XC.
- vtss_rc vtss_mpls_tc_conf_get (vtss_inst_t inst, vtss_mpls_tc_conf_t *const conf)

  Retrieve current global TC config.
- vtss_rc vtss_mpls_tc_conf_set (vtss_inst_t inst, const vtss_mpls_tc_conf_t *const conf)

  Set global TC config.
- vtss_rc vtss_mpls_xc_counters_get (vtss_inst_t inst, vtss_mpls_xc_idx_t xc_idx, vtss_mpls_xc_counters_t *const counters)

  Retrieve XC counters.
- vtss_rc vtss_mpls_xc_counters_clear (vtss_inst_t inst, vtss_mpls_xc_idx_t xc_idx)

  Clear XC counters.

10.20.1 Detailed Description

MPLS API.

This header file describes the MPLS functions

10.20.2 Macro Definition Documentation

10.20.2.1 VTSS_MPLS_LABEL_VALUE_DONTCARE

```
#define VTSS_MPLS_LABEL_VALUE_DONTCARE 0xffffffff
```

Don't-care label value

Definition at line 174 of file vtss_mpls_api.h.

10.20.2.2 VTSS_MPLS_TC_VALUE_DONTCARE

```
#define VTSS_MPLS_TC_VALUE_DONTCARE 0xff
```

Don't-care TC value

Definition at line 175 of file vtss_mpls_api.h.

10.20.2.3 VTSS_MPLS_IDX_UNDEFINED

```
#define VTSS_MPLS_IDX_UNDEFINED (-1)
```

Value of an undefined MPLS index

Definition at line 181 of file vtss_mpls_api.h.

10.20.2.4 VTSS_MPLS_IDX_IS_UNDEF

```
#define VTSS_MPLS_IDX_IS_UNDEF(
              idx ) ((idx) == VTSS_MPLS_IDX_UNDEFINED)
```

Test whether an MPLS index is undefined (TRUE) or not (FALSE)

Definition at line 182 of file vtss_mpls_api.h.

10.20.2.5 VTSS_MPLS_IDX_IS_DEF

```
#define VTSS_MPLS_IDX_IS_DEF(
              idx ) ((idx) != VTSS_MPLS_IDX_UNDEFINED)
```

Test whether an MPLS index is defined (TRUE) or not (FALSE)

Definition at line 183 of file vtss_mpls_api.h.

10.20.2.6 VTSS_MPLS_IDX_UNDEF

```
#define VTSS_MPLS_IDX_UNDEF(
              idx ) do { (idx) = VTSS_MPLS_IDX_UNDEFINED; } while (0)
```

Set an MPLS index to undefined

Definition at line 184 of file vtss_mpls_api.h.

10.20.2.7 VTSS_MPLS_QOS_TO_TC_MAP_CNT

```
#define VTSS_MPLS_QOS_TO_TC_MAP_CNT 8
```

8 tables

Definition at line 273 of file vtss_mpls_api.h.

10.20.2.8 VTSS_MPLS_QOS_TO_TC_ENTRY_CNT

```
#define VTSS_MPLS_QOS_TO_TC_ENTRY_CNT 8
```

8 entries per table

Definition at line 274 of file vtss_mpls_api.h.

10.20.2.9 VTSS_MPLS_TC_TO_QOS_MAP_CNT

```
#define VTSS_MPLS_TC_TO_QOS_MAP_CNT 8
```

8 tables

Definition at line 276 of file vtss_mpls_api.h.

10.20.2.10 VTSS_MPLS_TC_TO_QOS_ENTRY_CNT

```
#define VTSS_MPLS_TC_TO_QOS_ENTRY_CNT 8
```

8 entries per table

Definition at line 277 of file vtss_mpls_api.h.

10.20.3 Typedef Documentation

10.20.3.1 vtss_mpls_label_value_t

```
typedef u32 vtss_mpls_label_value_t
```

[0..2^20[or VTSS_MPLS_LABEL_VALUE_DONTCARE for don't care/undefined

Definition at line 186 of file vtss_mpls_api.h.

10.20.3.2 vtss_mpls_tc_t

```
typedef u8 vtss_mpls_tc_t
```

[0..7] or VTSS_MPLS_TC_VALUE_DONTCARE for don't care/undefined

Definition at line 187 of file vtss_mpls_api.h.

10.20.3.3 vtss_mpls_cos_t

```
typedef u8 vtss_mpls_cos_t
```

[0..7]

Definition at line 188 of file vtss_mpls_api.h.

10.20.3.4 vtss_mpls_qos_t

```
typedef u8 vtss_mpls_qos_t
```

[0..7]

Definition at line 189 of file vtss_mpls_api.h.

10.20.3.5 vtss_mpls_segment_idx_t

```
typedef i16 vtss_mpls_segment_idx_t
```

Index into table of vtss_mpls_segment_t

Definition at line 190 of file vtss_mpls_api.h.

10.20.3.6 vtss_mpls_xc_idx_t

```
typedef i16 vtss_mpls_xc_idx_t
```

Index into table of vtss_mpls_xc_t

Definition at line 191 of file vtss_mpls_api.h.

10.20.3.7 vtss_mpls_l2_idx_t

```
typedef i16 vtss_mpls_l2_idx_t
```

Index into table of vtss_mpls_l2_t

Definition at line 192 of file vtss_mpls_api.h.

10.20.4 Enumeration Type Documentation

10.20.4.1 vtss_mll_tagtype_t

```
enum vtss_mll_tagtype_t
```

Frame tagging. Used both for ingress and egress.

Enumerator

VTSS_MPLS_TAGTYPE_UNTAGGED	Frame is untagged
VTSS_MPLS_TAGTYPE_CTAGGED	Frame is C-tagged
VTSS_MPLS_TAGTYPE_STAGGED	Frame is S-tagged

Definition at line 197 of file vtss_mpls_api.h.

10.20.4.2 vtss_mpls_segment_state_t

enum vtss_mpls_segment_state_t

Segment state. This expresses whether a segment is sufficiently configured to be able to operate, and in that case, whether it has been able to allocate the necessary hardware resources.

A segment may fail to acquire HW resources for two main reasons:

1. The underlying resource pool is shared with other components
2. The underlying resource is dynamically partitioned, meaning that the allocation limit for one allocation type is dependent on the number of allocations of other types

Enumerator

VTSS_MPLS_SEGMENT_STATE_UNCONF	Segment is not fully configured
VTSS_MPLS_SEGMENT_STATE_CONF	Segment is sufficiently configured, but has not acquired HW resources
VTSS_MPLS_SEGMENT_STATE_UP	Segment has acquired HW resources

Definition at line 217 of file vtss_mpls_api.h.

10.20.4.3 vtss_mpls_xc_type_t

enum vtss_mpls_xc_type_t

Cross-connect (XC) types.

Note: Once an XC is created, it cannot change type.

Enumerator

VTSS_MPLS_XC_TYPE_LSR	Label swap. In- and out-segment
VTSS_MPLS_XC_TYPE_LER	LSP init/term. In- or out-segment

Definition at line 235 of file vtss_mpls_api.h.

10.20.4.4 vtss_mpls_tunnel_mode_t

enum vtss_mpls_tunnel_mode_t

DiffServ tunnel modes for TC/TTL.

Enumerator

VTSS_MPLS_TUNNEL_MODE_PIPE	Pipe mode
VTSS_MPLS_TUNNEL_MODE_SHORT_PIPE	Short Pipe mode
VTSS_MPLS_TUNNEL_MODE_UNIFORM	Uniform mode

Definition at line 243 of file vtss_mpls_api.h.

10.20.4.5 vtss_mpls_oam_type_t

```
enum vtss_mpls_oam_type_t
```

MPLS Pseudo-wire + LSP OAM types.

Enumerator

VTSS_MPLS_OAM_TYPE_NONE	No OAM
VTSS_MPLS_OAM_TYPE_VCCV1	VCCV1 PW OAM
VTSS_MPLS_OAM_TYPE_VCCV2	VCCV2 PW OAM
VTSS_MPLS_OAM_TYPE_VCCV3	VCCV3 PW OAM
VTSS_MPLS_OAM_TYPE_VCCV4	VCCV4 PW OAM
VTSS_MPLS_OAM_TYPE_GAL_MEP	LSP MEP using GAL
VTSS_MPLS_OAM_TYPE_GAL_MIP	LSP MIP using GAL

Definition at line 252 of file vtss_mpls_api.h.

10.20.5 Function Documentation

10.20.5.1 vtss_mpls_l2_alloc()

```
vtss_rc vtss_mpls_l2_alloc (
            vtss_inst_t inst,
            vtss_mpls_l2_idx_t *const idx )
```

Allocate L2 entry.

The entry contains default values after allocation: Indices are undefined and all other values are zero.

Parameters

inst	[IN] Instance.
idx	[OUT] Index of newly allocated entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if no entries available.

10.20.5.2 vtss_mpls_l2_free()

```
vtss_rc vtss_mpls_l2_free (
            vtss_inst_t inst,
            const vtss_mpls_l2_idx_t idx )
```

Free L2 entry.

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to free.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if entry is still in use.

10.20.5.3 vtss_mpls_l2_get()

```
vtss_rc vtss_mpls_l2_get (
            vtss_inst_t inst,
            const vtss_mpls_l2_idx_t idx,
            vtss_mpls_l2_t *const l2 )
```

Get L2 entry.

Note that it is possible *but not recommended* to retrieve entries that haven't been allocated. In that case the contents are undefined.

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to retrieve.
l2	[OUT] Entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if index is out-of-range.

10.20.5.4 vtss_mpls_l2_set()

```
vtss_rc vtss_mpls_l2_set (
            vtss_inst_t inst,
            const vtss_mpls_l2_idx_t idx,
            const vtss_mpls_l2_t *const l2 )
```

Set L2 entry.

Note that it is possible *but not recommended* to set entries that haven't been allocated. The results are undefined and may lead to system malfunction.

If segments are attached, they will be asked to refresh themselves. This will not cause attempted allocation of new HW resources for segments in state VTSS_MPLS_SEGMENT_STATE_CONF (i.e. segments which have not earlier on been able acquire HW resources).

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to retrieve.
l2	[IN] Entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if index is out-of-range or some of the new values are invalid.

10.20.5.5 vtss_mpls_l2_segment_attach()

```
vtss_rc vtss_mpls_l2_segment_attach (
            vtss_inst_t inst,
            const vtss_mpls_l2_idx_t l2_idx,
            const vtss_mpls_segment_idx_t seg_idx )
```

Attach segment to L2 entry.

If the segment is in state VTSS_MPLS_SEGMENT_STATE_CONF after attachment, it will try to go to state VTS↩S_MPLS_SEGMENT_STATE_UP, i.e. allocate HW resources.

If the segment reaches UP and is a server it will try to (recursively) bring each client UP as well.

HW allocation may succeed or fail; use vtss_mpls_segment_status_get() to determine the outcome.

Parameters

inst	[IN] Instance.
l2_idx	[IN] L2 index.
seg_idx	[IN] Segment index to attach.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if parameters are invalid.

10.20.5.6 vtss_mpls_l2_segment_detach()

```
vtss_rc vtss_mpls_l2_segment_detach (
            vtss_inst_t inst,
            const vtss_mpls_segment_idx_t seg_idx )
```

Detach segment from L2 entry.

If the segment is a server it will (recursively) bring all clients out of UP state (and usually back to CONF, but don't depend on that behavior.).

Parameters

inst	[IN] Instance.
seg_idx	[IN] Segment index to detach.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if parameters are invalid.

10.20.5.7 vtss_mpls_segment_alloc()

```
vtss_rc vtss_mpls_segment_alloc (
            vtss_inst_t inst,
            const BOOL is_in,
            vtss_mpls_segment_idx_t *const idx )
```

Allocate Segment entry.

The entry contains default values after allocation: Indices are undefined and all other values except *is_in* are zero/↩ FALSE.

Parameters

inst	[IN] Instance.
is↩ _in	[IN] TRUE: Allocate in-segment; FALSE: out-segment.
idx	[OUT] Index of newly allocated entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if no entries available.

10.20.5.8 vtss_mpls_segment_free()

```
vtss_rc vtss_mpls_segment_free (
            vtss_inst_t inst,
            const vtss_mpls_segment_idx_t idx )
```

Free segment entry.

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to free.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if entry is still in use (attached to XC, attached to server, or has clients).

10.20.5.9 vtss_mpls_segment_get()

```
vtss_rc vtss_mpls_segment_get (
            vtss_inst_t inst,
            const vtss_mpls_segment_idx_t idx,
            vtss_mpls_segment_t *const seg )
```

Get segment entry.

Note that it is possible *but not recommended* to retrieve entries that haven't been allocated. In that case the contents are undefined.

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to retrieve.
seg	[OUT] Entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if index is out-of-range.

10.20.5.10 vtss_mpls_segment_set()

```
vtss_rc vtss_mpls_segment_set (
            vtss_inst_t inst,
```

```
            const vtss_mpls_segment_idx_t idx,
            const vtss_mpls_segment_t *const seg )
```

Set segment entry.

Note that it is possible *but not recommended* to set entries that haven't been allocated. The results are undefined and may lead to system malfunction.

Several values in a vtss_mpls_segment_t are write-once: Once set they cannot be changed. RO/RW for others depend on the current state of the segment, in particular whether it is attached to something or is a server.

If the segment is sufficiently configured, but has been unable to acquire HW resources, the allocation can be re-tried by calling set() again.

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to retrieve.
seg	[IN] Entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if index is out-of-range or some of the new values are invalid. NOTE that success does not imply that the segment is up. Use vtss_mpls_segment_status_get() to determine resulting state.

10.20.5.11 vtss_mpls_segment_status_get()

```
vtss_rc vtss_mpls_segment_status_get (
            vtss_inst_t inst,
            const vtss_mpls_segment_idx_t idx,
            vtss_mpls_segment_status_t *const status )
```

Get current segment status.

Parameters

inst	[IN] Instance.
idx	[IN] Index of segment to retrieve state for.
status	[OUT] Current segment status.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if index is out-of-range.

10.20.5.12 vtss_mpls_segment_server_attach()

```
vtss_rc vtss_mpls_segment_server_attach (
            vtss_inst_t inst,
            const vtss_mpls_segment_idx_t idx,
            const vtss_mpls_segment_idx_t srv_idx )
```

Attach segment to server segment.

If the segment is in state VTSS_MPLS_SEGMENT_STATE_CONF after attachment, it will try to go to state VTS↩S_MPLS_SEGMENT_STATE_UP, i.e. allocate HW resources. (This requires that the server is UP, too.)

If the segment reaches UP and is a server it will try to (recursively) bring each client UP as well.

HW allocation may succeed or fail; use vtss_mpls_segment_status_get() to determine the outcome.

Parameters

inst	[IN] Instance.
idx	[IN] Segment index to attach.
srv_idx	[IN] Server segment index.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if parameters are invalid.

10.20.5.13 vtss_mpls_segment_server_detach()

```
vtss_rc vtss_mpls_segment_server_detach (
            vtss_inst_t inst,
            const vtss_mpls_segment_idx_t seg_idx )
```

Detach segment from server segment.

This will cause the segment to go out of UP state.

If the segment is itself a server it will (recursively) bring all clients out of UP state (and usually back to CONF, but don't depend on that behavior.).

Parameters

inst	[IN] Instance.
seg_idx	[IN] Segment index to detach from server.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if parameters are invalid.

10.20.5.14 vtss_mpls_xc_alloc()

```
vtss_rc vtss_mpls_xc_alloc (
            vtss_inst_t inst,
            const vtss_mpls_xc_type_t type,
            vtss_mpls_xc_idx_t *const idx )
```

Allocate XC entry.

The entry contains default values after allocation: Indices are undefined and all other values except *type* are zero/FALSE.

Parameters

inst	[IN] Instance.
type	[IN] XC type.
idx	[OUT] Index of newly allocated entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if no entries available.

10.20.5.15 vtss_mpls_xc_free()

```
vtss_rc vtss_mpls_xc_free (
            vtss_inst_t inst,
            const vtss_mpls_xc_idx_t idx )
```

Free XC entry.

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to free.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if entry is still in use.

10.20.5.16 vtss_mpls_xc_get()

```
vtss_rc vtss_mpls_xc_get (
            vtss_inst_t inst,
            const vtss_mpls_xc_idx_t idx,
            vtss_mpls_xc_t *const xc )
```

Get XC entry.

Note that it is possible *but not recommended* to retrieve entries that haven't been allocated. In that case the contents are undefined.

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to retrieve.
xc	[OUT] Entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if index is out-of-range.

10.20.5.17 vtss_mpls_xc_set()

```
vtss_rc vtss_mpls_xc_set (
            vtss_inst_t inst,
            const vtss_mpls_xc_idx_t idx,
            const vtss_mpls_xc_t *const xc )
```

Set XC entry.

Note that it is possible *but not recommended* to set entries that haven't been allocated. The results are undefined and may lead to system malfunction.

Parameters

inst	[IN] Instance.
idx	[IN] Index of entry to retrieve.
xc	[IN] Entry.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if index is out-of-range or some of the new values are invalid.

10.20.5.18 vtss_mpls_xc_segment_attach()

```
vtss_rc vtss_mpls_xc_segment_attach (
            vtss_inst_t inst,
            const vtss_mpls_xc_idx_t xc_idx,
            const vtss_mpls_segment_idx_t seg_idx )
```

Attach segment to XC.

Note: If the segment contains inconsistent configuration, the attach operation will fail.

If the segment is in state VTSS_MPLS_SEGMENT_STATE_CONF after attachment, it will try to go to state VTS↩S_MPLS_SEGMENT_STATE_UP, i.e. allocate HW resources.

If the segment reaches UP and is a server it will try to (recursively) bring each client UP as well.

HW allocation may succeed or fail; use vtss_mpls_segment_status_get() to determine the outcome.

Parameters

inst	[IN] Instance.
xc_idx	[IN] XC index.
seg_idx	[IN] Segment index to attach.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if parameters are invalid.

10.20.5.19 vtss_mpls_xc_segment_detach()

```
vtss_rc vtss_mpls_xc_segment_detach (
            vtss_inst_t inst,
            const vtss_mpls_segment_idx_t seg_idx )
```

Detach segment from XC.

This will cause the segment to go out of UP state.

If the segment is itself a server it will (recursively) bring all clients out of UP state (and usually back to CONF, but don't depend on that behavior.).

Parameters

inst	[IN] Instance.
seg_idx	[IN] Segment index to detach from server.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if parameters are invalid.

10.20.5.20 vtss_mpls_tc_conf_get()

```
vtss_rc vtss_mpls_tc_conf_get (
            vtss_inst_t inst,
            vtss_mpls_tc_conf_t *const conf )
```

Retrieve current global TC config.

Parameters

inst	[IN] Instance.
conf	[OUT] Configuration.

Returns

Return code.

10.20.5.21 vtss_mpls_tc_conf_set()

```
vtss_rc vtss_mpls_tc_conf_set (
            vtss_inst_t inst,
            const vtss_mpls_tc_conf_t *const conf )
```

Set global TC config.

NOTE: conf.map[0] is pre-configured and cannot be altered. It provides an identity mapping.

Parameters

inst	[IN] Instance.
conf	[IN] Configuration.

Returns

VTSS_RC_OK when successful; VTSS_RC_ERROR if parameters are invalid.

10.20.5.22 vtss_mpls_xc_counters_get()

```
vtss_rc vtss_mpls_xc_counters_get (
            vtss_inst_t inst,
            vtss_mpls_xc_idx_t xc_idx,
            vtss_mpls_xc_counters_t *const counters )
```

Retrieve XC counters.

Parameters

inst	[IN] Instance.
xc_idx	[IN] XC index.
counters	[OUT] XC counters. Contents are undefined in case of error return code.

Returns

Return code. VTSS_RC_ERROR if counters aren't available/valid.

10.20.5.23 vtss_mpls_xc_counters_clear()

```
vtss_rc vtss_mpls_xc_counters_clear (
            vtss_inst_t inst,
            vtss_mpls_xc_idx_t xc_idx )
```

Clear XC counters.

Parameters

inst	[IN] Instance.
xc_idx	[IN] XC index.

Returns

Return code. VTSS_RC_ERROR if counters aren't available/valid.

10.21 vtss_api/include/vtss_oam_api.h File Reference

OAM API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct vtss_oam_vop_extract_conf_t

  Configuration for CPU/frontport extraction. Configure if a particular OAM PDU type is extracted to the CPU, or to a front port.
- struct vtss_oam_vop_generic_opcode_conf_t

  Configuration for generic opcodes. Similar to vtss_oam_vop_extract_conf_t, this structure configures extraction for a generic opcode, and allows for DMAC checking for incoming OAM PDUs with that opcode value.
- struct vtss_oam_vop_pdu_type_conf_t

  Global per-PDU type configuration.
- struct vtss_oam_vop_conf_t

  VOP configuration. Once the VOP is configured, VOEs can be configured.
- struct vtss_oam_event_mask_t

  Event polling across all VOEs.
- struct vtss_oam_voe_alloc_cfg_t

  Extra data used by vtss_oam_voe_alloc(). Only relevant for port VOEs.
- struct vtss_oam_meg_id_t

  MEG ID type.
- struct vtss_oam_voe_lm_counter_conf_t

  LM counter configuration.
- struct vtss_oam_voe_ccm_conf_t

  CCM configuration.
- struct vtss_oam_voe_ccm_lm_conf_t

  CCM-LM (dual-ended LM) configuration. NOTE that this configuration extends and augments the CCM configuration in struct vtss_oam_voe_ccm_conf_t. Thus, in order to configure CCM-LM, both structs need to be filled in.
- struct vtss_oam_voe_se_lm_conf_t

Single-ended LM configuration.

- struct vtss_oam_voe_lb_conf_t

  LB configuration. Note that the tx_... members are ignored if LB is already active. If they are to be changed, LB must be disabled and re-enabled with the new values. In other words, the values are only applied at enable-time.

- struct vtss_oam_voe_tst_conf_t

  TST configuration. Note that the tx_... members are ignored if TST is already active. If they are to be changed, TST must be disabled and re-enabled with the new values. In other words, the values are only applied at enable-time.

- struct vtss_oam_voe_dm_conf_t

  One- and Two-way DM (1DM / DMM) configuration.

- struct vtss_oam_voe_lt_conf_t

  LT configuration.

- struct vtss_oam_voe_generic_conf_t

  Generic OAM opcode configuration.

- struct vtss_oam_voe_unknown_conf_t

  Unknown opcode configuration. An unknown opcode is one that isn't a) explicitly supported (e.g. CCM or LTM) or b) configured as a generic opcode.

- struct vtss_oam_voe_up_mep_conf_t

  Up-MEP configuration.

- struct vtss_oam_proc_conf_t

  PDU processing checks.

- struct vtss_oam_voe_conf_t

  Main VOE configuration structure.

- struct vtss_oam_ccm_status_t

  Status for most recently processed RX CCM PDU.

- struct vtss_oam_pdu_seen_status_t

  Indicate whether one or more PDUs of various types has been seen since last check. A PDU must pass MEG Level and DMAC checks before being indicated here. The values are cleared after reading.

- struct vtss_oam_proc_status_t

  Processing status: change indications and status values for most recently processed PDU(s). ..._err: Status for the most recently processed PDU. ..._seen: This has happened at least once since last poll. Cleared after reading.

- struct vtss_oam_rx_tx_counter_t

  Simple RX/TX counter structure.

- struct vtss_oam_lm_counter_t

  Up-/Down-MEP LM counters and per-priority RX/TX LM counters.

- struct vtss_oam_ccm_counter_t

  CCM counters.

- struct vtss_oam_sel_counter_t

  RX/TX OAM PDU counters. By default, all OAM PDU types are counted as 'non-selected', but by setting the appropriate count_as_selected field in the configuration structures, a PDU type can be "moved" to the 'selected' counter.

- struct vtss_oam_lb_counter_t

  LB counter.

- struct vtss_oam_tst_counter_t

  TST counter.

- struct vtss_oam_voe_counter_t

  Per-VOE counters.

- struct vtss_oam_ipt_conf_t

  OAM-related configuration for the IPT table.

Macros

- #define VTSS_OAM_PATH_SERVICE_VOE_CNT (64)
- #define VTSS_OAM_PORT_VOE_CNT (11)
- #define VTSS_OAM_PORT_VOE_BASE_IDX (64)
- #define VTSS_OAM_VOE_CNT ((VTSS_OAM_PATH_SERVICE_VOE_CNT) + (VTSS_OAM_PORT_VOE_CNT))
- #define VTSS_OAM_GENERIC_OPCODE_CFG_CNT (7)
- #define VTSS_OAM_VOE_IDX_NONE 0xFFFFFFFF
- #define VTSS_OAM_EVENT_MASK_ARRAY_SIZE ((VTSS_OAM_VOE_CNT+31)/32)
- #define VTSS_OAM_VOE_EVENT_CCM_PERIOD (1<<7)

  Event mask bits used by vtss_oam_voe_event_enable() and vtss_oam_voe_event_poll()
- #define VTSS_OAM_VOE_EVENT_CCM_PRIORITY (1<<6)
- #define VTSS_OAM_VOE_EVENT_CCM_ZERO_PERIOD (1<<5)
- #define VTSS_OAM_VOE_EVENT_CCM_RX_RDI (1<<4)
- #define VTSS_OAM_VOE_EVENT_CCM_LOC (1<<3)
- #define VTSS_OAM_VOE_EVENT_CCM_MEP_ID (1<<2)
- #define VTSS_OAM_VOE_EVENT_CCM_MEG_ID (1<<1)
- #define VTSS_OAM_VOE_EVENT_MEG_LEVEL (1<<0)
- #define VTSS_OAM_VOE_EVENT_MASK (VTSS_BITMASK(8))
- #define VTSS_OAM_CNT_CCM (1<<0)

  Mask values for clearing specific counters.
- #define VTSS_OAM_CNT_LM (1<<1)
- #define VTSS_OAM_CNT_LB (1<<2)
- #define VTSS_OAM_CNT_TST (1<<3)
- #define VTSS_OAM_CNT_SEL (1<<4)
- #define VTSS_OAM_CNT_ALL (VTSS_BITMASK(5))

Typedefs

- typedef u32 vtss_oam_voe_idx_t

  VOE index.
- typedef u32 vtss_oam_voe_event_mask_t

  Event generation flags. It is possible to generate events (interrupts) based on specific kinds of OAM PDU processing.

Enumerations

- enum vtss_oam_mep_type_t { **VTSS_OAM_DOWNMEP** = 0, **VTSS_OAM_UPMEP**, **VTSS_OAM_MIP** }

  MEP type – Up/Down MEP or MIP.
- enum vtss_oam_voe_type_t { **VTSS_OAM_VOE_SERVICE** = 0, **VTSS_OAM_VOE_PORT** }

  VOE type – Service/Path or Port.
- enum vtss_oam_opcode_value_t {
  VTSS_OAM_OPCODE_CCM = 1, **VTSS_OAM_OPCODE_LBM** = 3, **VTSS_OAM_OPCODE_LBR** = 2, **V↩TSS_OAM_OPCODE_LTM** = 5,
  VTSS_OAM_OPCODE_LTR = 4, **VTSS_OAM_OPCODE_AIS** = 33, **VTSS_OAM_OPCODE_LCK** = 35, **V↩TSS_OAM_OPCODE_TST** = 37,
  VTSS_OAM_OPCODE_LINEAR_APS = 39, **VTSS_OAM_OPCODE_RING_APS** = 40, **VTSS_OAM_OP↩CODE_MCC** = 41, **VTSS_OAM_OPCODE_LMM** = 43,
  VTSS_OAM_OPCODE_LMR = 42, **VTSS_OAM_OPCODE_IDM** = 45, **VTSS_OAM_OPCODE_DMM** = 47, **VTSS_OAM_OPCODE_DMR** = 46,
  VTSS_OAM_OPCODE_EXM = 49, **VTSS_OAM_OPCODE_EXR** = 48, **VTSS_OAM_OPCODE_VSM** = 51, **VTSS_OAM_OPCODE_VSR** = 50 }

  OAM PDU OpCode values, from Y.1731 table 9-1.

- enum vtss_oam_tlv_value_t { **VTSS_OAM_TLV_VALUE_END** = 0, **VTSS_OAM_TLV_VALUE_DATA** = 3, **VTSS_OAM_TLV_VALUE_REPLY_INGRESS** = 5, **VTSS_OAM_TLV_VALUE_REPLY_EGRESS** = 6, **VTSS_OAM_TLV_VALUE_LTM_EGRESS_ID** = 7, **VTSS_OAM_TLV_VALUE_LTR_EGRESS_ID** = 8, **VTSS_OAM_TLV_VALUE_TEST** = 32 }

  OAM PDU TLV type values, from Y.1731 table 9-2.
- enum vtss_oam_period_t { **VTSS_OAM_PERIOD_INV** = 0, **VTSS_OAM_PERIOD_3_3_MS** = 1, **VTSS_OAM_PERIOD_10_MS** = 2, **VTSS_OAM_PERIOD_100_MS** = 3, **VTSS_OAM_PERIOD_1_SEC** = 4 }

  Supported OAM periods for CCM/CCM-LM.
- enum vtss_oam_voe_auto_seq_no_conf_t { **VTSS_OAM_AUTOSEQ_DONT** = 0, **VTSS_OAM_AUTOSEQ_INCREMENT_AND_UPDATE**, **VTSS_OAM_AUTOSEQ_UPDATE** }

  Kind of automatic sequence number update to perform at CCM TX.
- enum vtss_oam_dmac_check_t { **VTSS_OAM_DMAC_CHECK_UNICAST** = 0, **VTSS_OAM_DMAC_CHECK_MULTICAST**, **VTSS_OAM_DMAC_CHECK_BOTH**, **VTSS_OAM_DMAC_CHECK_NONE** }

  Kind of DMAC check to perform on incoming OAM PDUs.

Functions

- vtss_rc vtss_oam_vop_conf_get (const vtss_inst_t inst, vtss_oam_vop_conf_t *const cfg)

  Get VOP configuration. At startup, the VOP is configured with default values and all VOE processing is disabled.
- vtss_rc vtss_oam_vop_conf_set (const vtss_inst_t inst, const vtss_oam_vop_conf_t *const cfg)

  Set VOP configuration.
- vtss_rc vtss_oam_event_poll (const vtss_inst_t inst, vtss_oam_event_mask_t *const mask)

  VOP polling function called by interrupt or periodically.
- vtss_rc vtss_oam_voe_alloc (const vtss_inst_t inst, const vtss_oam_voe_type_t type, const vtss_oam_voe_alloc_cfg_t *data, vtss_oam_voe_idx_t *voe_idx)

  Allocate a VOE.
- vtss_rc vtss_oam_voe_free (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx)

  Free a VOE. If it's still enabled, it will also be disabled.
- vtss_rc vtss_oam_voe_conf_get (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, vtss_oam_voe_conf_t *const cfg)

  Get VOE configuration. At startup, each VOE is configured with default values and is disabled.
- vtss_rc vtss_oam_voe_conf_set (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, const vtss_oam_voe_conf_t *const cfg)

  Set VOE configuration. Note that RDI and hitme configuration must be (re-)applied after a call to this function.
- vtss_rc vtss_oam_voe_event_enable (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, const vtss_oam_voe_event_mask_t mask, const BOOL enable)

  Enable/disable VOE event generation.
- vtss_rc vtss_oam_voe_event_poll (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, vtss_oam_voe_event_mask_t *const mask)

  VOE event polling.
- vtss_rc vtss_oam_voe_ccm_arm_hitme (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, BOOL enable)

  Arm "hitme" copy-to-CPU processing. The flags reset to zero after each occurrence, so software needs to re-arm after each event.
- vtss_rc vtss_oam_voe_ccm_set_rdi_flag (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, BOOL flag)

  Configure RDI flag for a CCM-enabled VOE.
- vtss_rc vtss_oam_ccm_status_get (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, vtss_oam_ccm_status_t *value)

Read CCM status from a VOE.

- vtss_rc vtss_oam_pdu_seen_status_get (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, vtss_oam_pdu_seen_status_t *status)

  Read PDU-seen status from a VOE.
- vtss_rc vtss_oam_proc_status_get (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, vtss_oam_proc_status_t *value)

  Read PDU processing status from a VOE.
- vtss_rc vtss_oam_voe_counter_update (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx)

  Update counters for a VOE. This function must be called at regular intervals in order to ensure that HW counters are accumulated before they get a chance to wrap around.
- vtss_rc vtss_oam_voe_counter_update_serval (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx)

  Update counters for a VOE. This is a utility function to quickly poll and update the "1-bit counters" accumulated in vtss_oam_lb_counter_t.rx_lbr_trans_id_err and vtss_oam_ccm_counter_t.rx_invalid_seq_no.
- vtss_rc vtss_oam_voe_counter_get (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, vtss_oam_voe_counter_t *value)

  Read counters from a VOE. Will call vtss_oam_voe_counter_update() first.
- vtss_rc vtss_oam_voe_counter_clear (const vtss_inst_t inst, const vtss_oam_voe_idx_t voe_idx, const u32 mask)

  Clear counters in a VOE.
- vtss_rc vtss_oam_ipt_conf_get (const vtss_inst_t inst, const u32 isdx, vtss_oam_ipt_conf_t *cfg)

  Get IPT OAM configuration for a service index.
- vtss_rc vtss_oam_ipt_conf_set (const vtss_inst_t inst, const u32 isdx, const vtss_oam_ipt_conf_t *const cfg)

  Set IPT OAM configuration for a service index.

10.21.1 Detailed Description

OAM API.

This header file describes Y.1731/IEEE802.1ag OAM functions.

10.21.2 Macro Definition Documentation

10.21.2.1 VTSS_OAM_PATH_SERVICE_VOE_CNT

```
#define VTSS_OAM_PATH_SERVICE_VOE_CNT (64)
```

Number of path/service VOEs in HW

Definition at line 59 of file vtss_oam_api.h.

10.21.2.2 VTSS_OAM_PORT_VOE_CNT

```
#define VTSS_OAM_PORT_VOE_CNT (11)
```

Number of port VOEs in HW

Definition at line 60 of file vtss_oam_api.h.

10.21.2.3 VTSS_OAM_PORT_VOE_BASE_IDX

```
#define VTSS_OAM_PORT_VOE_BASE_IDX (64)
```

Index of first port VOE

Definition at line 61 of file vtss_oam_api.h.

10.21.2.4 VTSS_OAM_VOE_CNT

```
#define VTSS_OAM_VOE_CNT ((VTSS_OAM_PATH_SERVICE_VOE_CNT) + (VTSS_OAM_PORT_VOE_CNT))
```

Total count of port + path/service VOEs

Definition at line 63 of file vtss_oam_api.h.

10.21.2.5 VTSS_OAM_GENERIC_OPCODE_CFG_CNT

```
#define VTSS_OAM_GENERIC_OPCODE_CFG_CNT (7)
```

Number of configurable generic opcodes

Definition at line 65 of file vtss_oam_api.h.

10.21.2.6 VTSS_OAM_VOE_IDX_NONE

```
#define VTSS_OAM_VOE_IDX_NONE 0xFFFFFFFF
```

Special value meaning no VOE

Definition at line 75 of file vtss_oam_api.h.

10.21.2.7 VTSS_OAM_EVENT_MASK_ARRAY_SIZE

```
#define VTSS_OAM_EVENT_MASK_ARRAY_SIZE ((VTSS_OAM_VOE_CNT+31)/32)
```

Size in u32's of event mask array used by vtss_oam_event_poll()

Definition at line 77 of file vtss_oam_api.h.

10.21.2.8 VTSS_OAM_VOE_EVENT_CCM_PERIOD

```
#define VTSS_OAM_VOE_EVENT_CCM_PERIOD (1<<7)
```

Event mask bits used by vtss_oam_voe_event_enable() and vtss_oam_voe_event_poll()

CCM: Period field value has changed

Definition at line 80 of file vtss_oam_api.h.

10.21.2.9 VTSS_OAM_VOE_EVENT_CCM_PRIORITY

```
#define VTSS_OAM_VOE_EVENT_CCM_PRIORITY (1<<6)
```

CCM: Priority field value has changed

Definition at line 81 of file vtss_oam_api.h.

10.21.2.10 VTSS_OAM_VOE_EVENT_CCM_ZERO_PERIOD

```
#define VTSS_OAM_VOE_EVENT_CCM_ZERO_PERIOD (1<<5)
```

CCM: Period zero-check result has changed

Definition at line 82 of file vtss_oam_api.h.

10.21.2.11 VTSS_OAM_VOE_EVENT_CCM_RX_RDI

```
#define VTSS_OAM_VOE_EVENT_CCM_RX_RDI (1<<4)
```

CCM: RDI flag value has changed

Definition at line 83 of file vtss_oam_api.h.

10.21.2.12 VTSS_OAM_VOE_EVENT_CCM_LOC

```
#define VTSS_OAM_VOE_EVENT_CCM_LOC (1<<3)
```

CCM: Loss Of Continuity state has changed

Definition at line 84 of file vtss_oam_api.h.

10.21.2.13 VTSS_OAM_VOE_EVENT_CCM_MEP_ID

```
#define VTSS_OAM_VOE_EVENT_CCM_MEP_ID (1<<2)
```

CCM: MEP ID check result has changed

Definition at line 85 of file vtss_oam_api.h.

10.21.2.14 VTSS_OAM_VOE_EVENT_CCM_MEG_ID

```
#define VTSS_OAM_VOE_EVENT_CCM_MEG_ID (1<<1)
```

CCM: MEG ID check result has changed

Definition at line 86 of file vtss_oam_api.h.

10.21.2.15 VTSS_OAM_VOE_EVENT_MEG_LEVEL

```
#define VTSS_OAM_VOE_EVENT_MEG_LEVEL (1<<0)
```

PDU with unexpected MEG Level was seen

Definition at line 87 of file vtss_oam_api.h.

10.21.2.16 VTSS_OAM_VOE_EVENT_MASK

```
#define VTSS_OAM_VOE_EVENT_MASK (VTSS_BITMASK(8))
```

Mask of configurable bits

Definition at line 88 of file vtss_oam_api.h.

10.21.2.17 VTSS_OAM_CNT_CCM

```
#define VTSS_OAM_CNT_CCM (1<<0)
```

Mask values for clearing specific counters.

CCM counters

Definition at line 918 of file vtss_oam_api.h.

10.21.2.18 VTSS_OAM_CNT_LM

```
#define VTSS_OAM_CNT_LM (1<<1)
```

LM counters

Definition at line 919 of file vtss_oam_api.h.

10.21.2.19 VTSS_OAM_CNT_LB

```
#define VTSS_OAM_CNT_LB (1<<2)
```

LB counters

Definition at line 920 of file vtss_oam_api.h.

10.21.2.20 VTSS_OAM_CNT_TST

```
#define VTSS_OAM_CNT_TST (1<<3)
```

TST counters

Definition at line 921 of file vtss_oam_api.h.

10.21.2.21 VTSS_OAM_CNT_SEL

```
#define VTSS_OAM_CNT_SEL (1<<4)
```

Selected/non-selected counters

Definition at line 922 of file vtss_oam_api.h.

10.21.2.22 VTSS_OAM_CNT_ALL

```
#define VTSS_OAM_CNT_ALL (VTSS_BITMASK(5))
```

All counters

Definition at line 923 of file vtss_oam_api.h.

10.21.3 Function Documentation

10.21.3.1 vtss_oam_vop_conf_get()

```
vtss_rc vtss_oam_vop_conf_get (
            const vtss_inst_t inst,
            vtss_oam_vop_conf_t *const cfg )
```

Get VOP configuration. At startup, the VOP is configured with default values and all VOE processing is disabled.

Parameters

inst	[IN] Target instance reference.
cfg	[OUT] Configuration for the VOP.

Returns

Return code.

10.21.3.2 vtss_oam_vop_conf_set()

```
vtss_rc vtss_oam_vop_conf_set (
            const vtss_inst_t inst,
            const vtss_oam_vop_conf_t *const cfg )
```

Set VOP configuration.

Parameters

inst	[IN] Target instance reference.
cfg	[OUT] New configuration for the VOP.

Returns

Return code.

10.21.3.3 vtss_oam_event_poll()

```
vtss_rc vtss_oam_event_poll (
            const vtss_inst_t inst,
            vtss_oam_event_mask_t *const mask )
```

VOP polling function called by interrupt or periodically.

Parameters

inst	[IN] Target instance reference.
mask	[OUT] Mask.

Returns

Return code.

10.21.3.4 vtss_oam_voe_alloc()

```
vtss_rc vtss_oam_voe_alloc (
            const vtss_inst_t inst,
            const vtss_oam_voe_type_t type,
            const vtss_oam_voe_alloc_cfg_t * data,
            vtss_oam_voe_idx_t * voe_idx )
```

Allocate a VOE.

Parameters

inst	[IN] Target instance reference.
type	[IN] VOE type: Service or port
data	[IN] Extra data for port VOE type. Can be NULL for service VOE type.
voe_idx	[OUT] Index of allocated VOE instance.

Returns

Return code. Will return an error code if no suitable VOE can be allocated.

10.21.3.5 vtss_oam_voe_free()

```
vtss_rc vtss_oam_voe_free (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx )
```

Free a VOE. If it's still enabled, it will also be disabled.

Parameters

inst	[IN] Target instance reference.
voe_idx	[OUT] Index of previously allocated VOE instance.

Returns

Return code.

10.21.3.6 vtss_oam_voe_conf_get()

```
vtss_rc vtss_oam_voe_conf_get (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            vtss_oam_voe_conf_t *const cfg )
```

Get VOE configuration. At startup, each VOE is configured with default values and is disabled.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE configuration to get.
cfg	[OUT] Configuration for the VOE.

Returns

Return code.

10.21.3.7 vtss_oam_voe_conf_set()

```
vtss_rc vtss_oam_voe_conf_set (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            const vtss_oam_voe_conf_t *const cfg )
```

Set VOE configuration. Note that RDI and hitme configuration must be (re-)applied after a call to this function.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE configuration to get.
cfg	[OUT] New configuration for the VOE.

Returns

Return code.

10.21.3.8 vtss_oam_voe_event_enable()

```
vtss_rc vtss_oam_voe_event_enable (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            const vtss_oam_voe_event_mask_t mask,
            const BOOL enable )
```

Enable/disable VOE event generation.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE to configure.
mask	[IN] Mask of events to either enable or disable.
enable	[IN] Whether to enable or disable the events specified in the mask.

Returns

Return code.

10.21.3.9 vtss_oam_voe_event_poll()

```
vtss_rc vtss_oam_voe_event_poll (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            vtss_oam_voe_event_mask_t *const mask )
```

VOE event polling.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE to configure.
mask	[OUT] Mask of occurred events.

Returns

Return code.

10.21.3.10 vtss_oam_voe_ccm_arm_hitme()

```
vtss_rc vtss_oam_voe_ccm_arm_hitme (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            BOOL enable )
```

Arm "hitme" copy-to-CPU processing. The flags reset to zero after each occurrence, so software needs to re-arm after each event.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE to configure.
enable	[IN] TRUE => arm

Returns

Return code.

10.21.3.11 vtss_oam_voe_ccm_set_rdi_flag()

```
vtss_rc vtss_oam_voe_ccm_set_rdi_flag (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            BOOL flag )
```

Configure RDI flag for a CCM-enabled VOE.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE to configure.
flag	[IN] RDI flag. TRUE => 1; FALSE => 0

Returns

Return code.

10.21.3.12 vtss_oam_ccm_status_get()

```
vtss_rc vtss_oam_ccm_status_get (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            vtss_oam_ccm_status_t * value )
```

Read CCM status from a VOE.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE to read from.
value	[OUT] Status values.

Returns

Return code.

10.21.3.13 vtss_oam_pdu_seen_status_get()

```
vtss_rc vtss_oam_pdu_seen_status_get (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            vtss_oam_pdu_seen_status_t * status )
```

Read PDU-seen status from a VOE.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE to read from.
status	[OUT] Status values.

Returns

Return code.

10.21.3.14 vtss_oam_proc_status_get()

```
vtss_rc vtss_oam_proc_status_get (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            vtss_oam_proc_status_t * value )
```

Read PDU processing status from a VOE.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE to read from.
value	[OUT] Status values.

Returns

Return code.

10.21.3.15 vtss_oam_voe_counter_update()

```
vtss_rc vtss_oam_voe_counter_update (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx )
```

Update counters for a VOE. This function must be called at regular intervals in order to ensure that HW counters are accumulated before they get a chance to wrap around.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE.

Returns

Return code

10.21.3.16 vtss_oam_voe_counter_update_serval()

```
vtss_rc vtss_oam_voe_counter_update_serval (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx )
```

Update counters for a VOE. This is a utility function to quickly poll and update the "1-bit counters" accumulated in vtss_oam_lb_counter_t.rx_lbr_trans_id_err and vtss_oam_ccm_counter_t.rx_invalid_seq_no.

This can improve (but not guarantee) the precision of the counters.

The function is ONLY provided for Serval-1.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE.

Returns

Return code

10.21.3.17 vtss_oam_voe_counter_get()

```
vtss_rc vtss_oam_voe_counter_get (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            vtss_oam_voe_counter_t * value )
```

Read counters from a VOE. Will call vtss_oam_voe_counter_update() first.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE.
value	[OUT] Counter values.

Returns

Return code.

10.21.3.18 vtss_oam_voe_counter_clear()

```
vtss_rc vtss_oam_voe_counter_clear (
            const vtss_inst_t inst,
            const vtss_oam_voe_idx_t voe_idx,
            const u32 mask )
```

Clear counters in a VOE.

Parameters

inst	[IN] Target instance reference.
voe_idx	[IN] Index of VOE.
mask	[IN] Mask of counters to clear, values taken from VTSS_OAM_CNT_...

Returns

Return code.

10.21.3.19 vtss_oam_ipt_conf_get()

```
vtss_rc vtss_oam_ipt_conf_get (
            const vtss_inst_t inst,
            const u32 isdx,
            vtss_oam_ipt_conf_t * cfg )
```

Get IPT OAM configuration for a service index.

Parameters

inst	[IN] Target instance reference.
isdx	[IN] Service index.
cfg	[OUT] IPT configuration.

Returns

Return code.

10.21.3.20 vtss_oam_ipt_conf_set()

```
vtss_rc vtss_oam_ipt_conf_set (
            const vtss_inst_t inst,
            const u32 isdx,
            const vtss_oam_ipt_conf_t *const cfg )
```

Set IPT OAM configuration for a service index.

Parameters

inst	[IN] Target instance reference.
isdx	[IN] Service index.
cfg	[IN] IPT configuration.

Returns

Return code.

10.22 vtss_api/include/vtss_oha_api.h File Reference

OHA API.

```
#include <vtss/api/types.h>
```

10.22.1 Detailed Description

OHA API.

10.23 vtss_api/include/vtss_os.h File Reference

OS Layer API.

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

10.23.1 Detailed Description

OS Layer API.

This header file includes the OS specific header file

10.24 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

10.24.1 Detailed Description

OS custom header file.

This file is a skeleton to be replaced by for customer specific OS.

10.24.2 Macro Definition Documentation

10.24.2.1 uint

```
#define uint unsigned int
```

Define API uint type - unsigned 16 bits

Definition at line 36 of file vtss_os_custom.h.

10.24.2.2 ulong

```
#define ulong unsigned long
```

Define API ulong - unsigned 32 bits

Definition at line 37 of file vtss_os_custom.h.

10.24.2.3 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.

10.24.2.4 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.

10.24.2.5 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.

10.24.2.6 VTSS_MTIMER_CANCEL

```
#define VTSS_MTIMER_CANCEL(
            pTimer ) <your impl>
```

Stop the timer

Definition at line 46 of file vtss_os_custom.h.

10.24.2.7 VTSS_DIV64

```
#define VTSS_DIV64(
            dividend,
            divisor ) <your impl>
```

support for 64 bit division

Definition at line 48 of file vtss_os_custom.h.

10.24.2.8 VTSS_MOD64

```
#define VTSS_MOD64(
            dividend,
            divisor ) <your impl>
```

support for 64 bit division

Definition at line 49 of file vtss_os_custom.h.

10.24.2.9 VTSS_LABS

```
#define VTSS_LABS(
            arg ) <your impl>
```

long to abs

Definition at line 50 of file vtss_os_custom.h.

10.24.2.10 VTSS_LLABS

```
#define VTSS_LLABS(
            arg ) <your impl>
```

long long to abs

Definition at line 51 of file vtss_os_custom.h.

10.24.2.11 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.

10.24.2.12 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.

10.24.2.13 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.

10.24.2.14 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.

10.24.2.15 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.

10.24.3 Typedef Documentation

10.24.3.1 vtss_mtimer_t

```
typedef int vtss_mtimer_t
```

Timer

Definition at line 43 of file vtss_os_custom.h.

10.25 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_DENOMINAT↩OR))
- #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.

10.25.1 Detailed Description

eCos OS API

This header file describes OS functions for eCos

10.25.2 Macro Definition Documentation

10.25.2.1 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.

10.25.2.2 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.

10.25.2.3 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.

10.25.2.4 VTSS_MTIMER_TIMEOUT

```
#define VTSS_MTIMER_TIMEOUT(
            pTimer ) (cyg_current_time() > *(pTimer))
```

Timer timeout

Definition at line 55 of file vtss_os_ecos.h.

10.25.2.5 VTSS_MTIMER_CANCEL

```
#define VTSS_MTIMER_CANCEL(
            pTimer )
```

No action in this implementation.

Definition at line 56 of file vtss_os_ecos.h.

10.25.2.6 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.

10.25.2.7 VTSS_DIV64

```
#define VTSS_DIV64(
              dividend,
              divisor ) ((dividend) / (divisor))
```

support for 64 bit division

Definition at line 73 of file vtss_os_ecos.h.

10.25.2.8 VTSS_MOD64

```
#define VTSS_MOD64(
              dividend,
              divisor ) ((dividend) % (divisor))
```

support for 64 bit division

Definition at line 74 of file vtss_os_ecos.h.

10.25.2.9 VTSS_LABS

```
#define VTSS_LABS(
              arg ) labs(arg)
```

long to abs

Definition at line 75 of file vtss_os_ecos.h.

10.25.2.10 VTSS_LLABS

```
#define VTSS_LLABS(
              arg ) llabs(arg)
```

long long to abs

Definition at line 76 of file vtss_os_ecos.h.

10.25.2.11 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.

10.25.2.12 VTSS_OS_CTZ64

```
#define VTSS_OS_CTZ64(
            val64 ) ((val64) == 0 ?  64 :  __builtin_ctzll(val64))
```

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).

Note: __builtin_ctzll() is undefined for zero input values.

Definition at line 102 of file vtss_os_ecos.h.

10.25.2.13 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.

10.25.2.14 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.

10.25.2.15 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.

10.25.2.16 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.

10.25.2.17 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.

10.25.2.18 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.

10.25.2.19 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.

10.25.2.20 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.

10.25.2.21 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.

10.25.2.22 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.

10.25.2.23 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.

10.25.2.24 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.

10.25.2.25 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.

10.25.2.26 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.

10.25.2.27 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_SCHED↩ULER_(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.

10.25.2.28 VTSS_OS_INTERRUPT_DISABLE

```
#define VTSS_OS_INTERRUPT_DISABLE(
              flags ) NOT_NEEDED
```

Disable interrupts.

Definition at line 276 of file vtss_os_ecos.h.

10.25.2.29 VTSS_OS_INTERRUPT_RESTORE

```
#define VTSS_OS_INTERRUPT_RESTORE(
              flags ) NOT_NEEDED
```

Enable interrupts.

Definition at line 280 of file vtss_os_ecos.h.

10.25.3 Typedef Documentation

10.25.3.1 vtss_mtimer_t

```
typedef cyg_tick_count_t vtss_mtimer_t
```

Timer

Definition at line 53 of file vtss_os_ecos.h.

10.25.4 Function Documentation

10.25.4.1 llabs()

```
long long int llabs (
            long long int val )
```

Obtain the absolute value of a long long integer.

Parameters

val	[IN] The value to convert to absolute value.

Returns

The absolute value of val.

10.25.4.2 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

size	[IN] Number of bytes to allocate.
flags	[IN] See vtss_mem_flags_t for details.

Returns

Pointer to allocated area.

10.25.4.3 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

ptr	[IN] Pointer previously obtained with call to vtss_callout_malloc().
flags	[IN] See vtss_mem_flags_t for details.

10.26 vtss_api/include/vtss_os_linux.h File Reference

Linux OS API.

```
#include <endian.h>
#include <asm/byteorder.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <errno.h>
#include <time.h>
#include <sys/time.h>
```

Data Structures

- struct vtss_mtimer_t

  Timer structure.
- struct vtss_timeofday_t

  Time of day structure.

Macros

- #define VTSS_OS_BIG_ENDIAN

  VTSS_OS_BIG_ENDIAN: If undefined, we're running little endian. If defined we're running big endian.
- #define VTSS_OS_NTOHL(x) __be32_to_cpu(x)
- #define VTSS_NSLEEP(nsec)
- #define VTSS_MSLEEP(msec)
- #define VTSS_MTIMER_START(timer, msec)
- #define VTSS_MTIMER_TIMEOUT(timer) (gettimeofday(&((timer)->now),NULL)==0 && timercmp(&((timer)->now),&((timer)->timeout),>))
- #define VTSS_MTIMER_CANCEL(timer)
- #define VTSS_TIME_OF_DAY(tod)

- #define VTSS_OS_SCHEDULER_FLAGS int
- #define VTSS_OS_SCHEDULER_LOCK(flags) do {flags = flags;} while (0);
- #define VTSS_OS_SCHEDULER_UNLOCK(flags) do {flags = flags;} while (0);
- #define VTSS_DIV64(dividend, divisor) ((dividend) / (divisor))
- #define VTSS_MOD64(dividend, divisor) ((dividend) % (divisor))
- #define VTSS_LABS(arg) labs(arg)
- #define VTSS_LLABS(arg) llabs(arg)
- #define VTSS_OS_CTZ(val32) ((val32) == 0 ? 32 : __builtin_ctzl((unsigned long)val32))
- #define VTSS_OS_CTZ64(val64)
- #define VTSS_OS_MALLOC(size, flags) malloc(size)
- #define VTSS_OS_FREE(ptr, flags) free(ptr)
- #define VTSS_OS_RAND() rand()

10.26.1 Detailed Description

Linux OS API.

This header file describes OS functions for Linux

10.26.2 Macro Definition Documentation

10.26.2.1 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.

10.26.2.2 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.

10.26.2.3 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

nsec	nanoseconds

Definition at line 69 of file vtss_os_linux.h.

10.26.2.4 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

msec	milliseconds

Definition at line 78 of file vtss_os_linux.h.

10.26.2.5 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.

10.26.2.6 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.

10.26.2.7 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.

10.26.2.8 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.

10.26.2.9 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.

10.26.2.10 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.

10.26.2.11 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.

10.26.2.12 VTSS_DIV64

```
#define VTSS_DIV64(
              dividend,
              divisor ) ((dividend) / (divisor))
```

VTSS_DIV64 - perform 64/32 bit division yelding 32 bit (at least) output

Definition at line 145 of file vtss_os_linux.h.

10.26.2.13 VTSS_MOD64

```
#define VTSS_MOD64(
              dividend,
              divisor ) ((dividend) % (divisor))
```

VTSS_MOD64 - perform 64/32 bit modulus yelding 32 bit (at least) output

Definition at line 149 of file vtss_os_linux.h.

10.26.2.14 VTSS_LABS

```
#define VTSS_LABS(
            arg ) labs(arg)
```

VTSS_LABS - perform abs() on long

Definition at line 153 of file vtss_os_linux.h.

10.26.2.15 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.

10.26.2.16 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

val32	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.

10.26.2.17 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.

10.26.2.18 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.

10.26.2.19 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.

10.26.2.20 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.

10.27 vtss_api/include/vtss_otn_api.h File Reference

OTN API.

```
#include <vtss/api/types.h>
```

10.27.1 Detailed Description

OTN API.

10.28 vtss_api/include/vtss_packet_api.h File Reference

Packet API.

```
#include <vtss/api/types.h>
#include <vtss_l2_api.h>
```

Data Structures

- struct vtss_npi_conf_t

  NPI configuration.
- struct vtss_packet_rx_queue_npi_conf_t

  CPU Rx queue NPI configuration.
- struct vtss_packet_rx_queue_conf_t

  CPU Rx queue configuration.
- struct vtss_packet_rx_reg_t

  CPU Rx packet registration.
- struct vtss_packet_rx_queue_map_t

  CPU Rx queue map.
- struct vtss_packet_rx_conf_t

  CPU Rx configuration.
- struct vtss_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_afi_frm_dscr_t

  AFI Frame description structure.
- 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_AFI_SLOT_CNT 1024
- #define VTSS_AFI_ID_NONE (0xFFFFFFFF)
- #define VTSS_AFI_FPS_MAX (3800000)
- #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_SVL_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 16

Typedefs

- typedef u32 vtss_packet_rx_queue_size_t

  CPU Rx queue buffer size in bytes.
- typedef u32 vtss_afi_id_t

  AFI identifier. Identifies the resources allocated for a periodically injected frame.

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_TW 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_TAGGE VTSS_TAG_TYPE_S_CUSTOM_TAGGED }
- enum vtss_packet_rx_hints_t { VTSS_PACKET_RX_HINTS_NONE = 0x00, VTSS_PACKET_RX_HINTS_VLAN_TAG_MISMA = 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 frmlen)

  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_afi_alloc (const vtss_inst_t inst, vtss_afi_frm_dscr_t *const dscr, vtss_afi_id_t *const id)

  Allocate resources for an AFI frame.
- vtss_rc vtss_afi_free (const vtss_inst_t inst, vtss_afi_id_t id)

  Free resources associated with an AFI frame.
- 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 extration/injection DMA.

10.28.1 Detailed Description

Packet API.

This header file describes CPU Rx/Tx packet functions.

10.28.2 Macro Definition Documentation

10.28.2.1 VTSS_PRIO_SUPER

```
#define VTSS_PRIO_SUPER VTSS_PRIO_END
```

Super priority

Definition at line 429 of file vtss_packet_api.h.

10.28.2.2 VTSS_AFI_SLOT_CNT

```
#define VTSS_AFI_SLOT_CNT 1024
```

Maximum number of AFI flows

Definition at line 562 of file vtss_packet_api.h.

10.28.2.3 VTSS_AFI_ID_NONE

```
#define VTSS_AFI_ID_NONE (0xFFFFFFFF)
```

Use this to encode Tx headers for non-AFI frame

Definition at line 658 of file vtss_packet_api.h.

10.28.2.4 VTSS_AFI_FPS_MAX

```
#define VTSS_AFI_FPS_MAX (3800000)
```

Maximum number of frames per second we support (corresponds to wirespeed 2.5Gbps 64 byte frames)

Definition at line 659 of file vtss_packet_api.h.

10.28.2.5 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.

10.28.2.6 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.

10.28.2.7 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.

10.28.2.8 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.

10.28.2.9 VTSS_PACKET_HDR_SIZE_BYTES

```
#define VTSS_PACKET_HDR_SIZE_BYTES VTSS_SVL_PACKET_HDR_SIZE_BYTES
```

Maximum header size. This define is only useful if you only compile for one target.

Definition at line 705 of file vtss_packet_api.h.

10.28.2.10 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.

10.28.2.11 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.

10.28.2.12 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.

10.28.2.13 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.

10.28.2.14 VTSS_PACKET_TX_IFH_MAX

```
#define VTSS_PACKET_TX_IFH_MAX 16
```

Tx IFH byte length (Constant)

Definition at line 2058 of file vtss_packet_api.h.

10.28.3 Enumeration Type Documentation

10.28.3.1 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.

10.28.3.2 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.

10.28.3.3 vtss_packet_ptp_action_t

```
enum vtss_packet_ptp_action_t
```

PTP actions used when encoding an injection header.

Enumerator

VTSS_PACKET_PTP_ACTION_NONE	No-op
VTSS_PACKET_PTP_ACTION_ONE_STEP	One-step PTP
VTSS_PACKET_PTP_ACTION_TWO_STEP	Two-step PTP
VTSS_PACKET_PTP_ACTION_ONE_AND_TWO↩_STEP	Both one- and two-step PTP
VTSS_PACKET_PTP_ACTION_ORIGIN_TIMEST↩AMP	Update time-of-day in PTP frame's originTimestamp field

Definition at line 543 of file vtss_packet_api.h.

10.28.3.4 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.

10.28.3.5 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_MISMA↩TCH	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. The same goes for frames received on S-ports or S-custom-ports with "foreign" tags. **Can be set by:** `Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y`

Enumerator

VTSS_PACKET_RX_HINTS_VLAN_FRAME_MIS↩ MATCH	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. **Can be set by:** `Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y`
VTSS_PACKET_RX_HINTS_VID_MISMATCH	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. **Can be set by:** `Luton26: Y Jaguar1: Y Serval: Y Jaguar2: Y Serval2: Y ServalT: Y`

Definition at line 915 of file vtss_packet_api.h.

10.28.3.6 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.

10.28.4 Function Documentation

10.28.4.1 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

inst	[IN] Target instance reference.
conf	[OUT] NPI port configuration.

Returns

Return code.

10.28.4.2 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

inst	[IN] Target instance reference.
conf	[IN] NPI port configuration.

Returns

Return code.

10.28.4.3 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

inst	[IN] Target instance reference.
conf	[IN] Packet Rx configuration.

Returns

Return code.

10.28.4.4 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

inst	[IN] Target instance reference.
conf	[IN] CPU Rx queue configuration.

Returns

Return code.

10.28.4.5 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] Packet port configuration structure.

Returns

Return code.

10.28.4.6 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] Packet port configuration structure.

Returns

Return code.

10.28.4.7 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

inst	[IN] Target instance reference.
queue_no	[IN] CPU queue number.
header	[OUT] Frame header.
frame	[OUT] Frame buffer.
length	[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.

10.28.4.8 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 preceeded 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

inst	[IN] Target instance reference.
data	[IN] Data buffer.
buflen	[IN] Length of data buffer.
ifhlen	[OUT] Length of IFH at the start of the buffer.
frmlen	[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.

10.28.4.9 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

inst	[IN] Target instance reference.
queue_no	[IN] CPU queue number.

Returns

Return code.

10.28.4.10 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
frame	[IN] Frame buffer excluding room for CRC.
length	[IN] Frame length excluding CRC.

Returns

Return code.

10.28.4.11 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
vid	[IN] VLAN ID inserted if the frame is tagged on egress.
frame	[IN] Frame buffer excluding room for CRC.
length	[IN] Frame length excluding CRC.

Returns

Return code.

10.28.4.12 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

inst	[IN] Target instance reference.
vid	[IN] VLAN ID inserted if the frame is tagged on egress.
frame	[IN] Frame buffer excluding room for CRC.
length	[IN] Frame length excluding CRC.

Returns

Return code.

10.28.4.13 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

info	[OUT] Frame information.

10.28.4.14 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

inst	[IN] Target instance reference.
info	[IN] Frame information.
filter	[OUT] Frame filter.

Returns

Return code.

10.28.4.15 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

info	[OUT] Frame information.

Returns

Return code.

10.28.4.16 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

inst	[IN] Target instance reference.
info	[IN] Frame information.
filter	[OUT] Frame filter.

Returns

Return code.

10.28.4.17 vtss_afi_alloc()

```
vtss_rc vtss_afi_alloc (
            const vtss_inst_t inst,
            vtss_afi_frm_dscr_t *const dscr,
            vtss_afi_id_t *const id )
```

Allocate resources for an AFI frame.

This function allocates resources required for a frame subject to automatic frame injection.

The function returns an ID that can be used in subsequent calls to vtss_packet_tx_hdr_encode() and vtss_afi_free(). The ID is only valid for one single frame.

Parameters

inst	[IN] Target instance reference.
dscr	[INOUT] Pointer to structure describing how the frame should be injected.
id	[OUT] Pointer receiving a unique ID that identifies the allocated resources for this frame.

Returns

VTSS_RC_OK on success. VTSS_RC_ERROR on error.

10.28.4.18 vtss_afi_free()

```
vtss_rc vtss_afi_free (
            const vtss_inst_t inst,
            vtss_afi_id_t id )
```

Free resources associated with an AFI frame.

This function cancels any ongoing transmissions of the frame associated with the ID passed to this function, and frees the allocated resources.

Parameters

inst	[IN] Target instance reference.
id	[IN] ID of AFI frame to cancel and free resources for.

Returns

VTSS_RC_OK on success. VTSS_RC_ERROR on error.

10.28.4.19 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

inst	[IN] Target instance reference.
meta	[IN] Meta info on received frame.
hdr	[IN] Packet header (IFH)
info	[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.

10.28.4.20 vtss_packet_tx_hdr_encode()

```
vtss_rc vtss_packet_tx_hdr_encode (
            const vtss_inst_t inst,
            const vtss_packet_tx_info_t *const info,
            u8 *const bin_hdr,
            u32 *const bin_hdr_len )
```

Compose binary injection/Tx header.

This function is mainly useful for external CPUs that inject frames into the switch core with an injection frame header.

In a few cases, it may also be useful for an application running on the internal CPU: If the frame is to be looped internally on a loop port, which is set-up to accept an injection header, the injection header may be encoded with this function.

The info structure is the input to the encoding, which results in a binary injection header and a length.

On many architectures, the resulting binary length is constant, but on some, it may vary with the contents of info properties. To overcome this, call this function twice. The first time, use a NULL pointer for bin_hdr. On return, the length parameter will contain the number of bytes required in bin_hdr. The second time, provide a non-NULL pointer to bin_hdr. On successful exit, bin_hdr_len will always be updated to contain the actual number of bytes required to hold the IFH. If you don't want to call this function twice, you may allocate an array of VTSS_PACKET_HDR↩_SIZE_BYTES (or VTSS_arch_PACKET_HDR_SIZE_BYTES) bytes, which will be long enough to accommodate all combinations of content of info. But notice that there is no guarantee that the actual length of the returned data matches VTSS_arch_PACKET_HDR_SIZE_BYTES.

For constellations that use the internal CPU in conjunction with the Frame DMA, this function will be called by the FDMA driver. Please refer to the FDMA API for details on how to specify injection properties for use with the FDMA.

Parameters

inst	[IN] Target instance reference.
info	[IN] Tx header properties.
bin_hdr	[OUT] NULL to get bin_hdr_len filled with required length in bytes. Non-NULL to get it filled in with the binary injection header.
bin_hdr_len	[INOUT] If bin_hdr is NULL, the [IN]-part is not used. Instead, it will be filled with resulting length of binary injection header in bytes. If bin_hdr is non-NULL, bin_hdr_len specifies the number of bytes that the function may write from the beginning of bin_hdr. On exit, it will contain the actual number of bytes.

Returns

Return code.

10.28.4.21 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

inst	[IN] Target instance reference.
info	[IN] Tx header properties.
ifh	[OUT] Compiled Tx header.

Returns

Return code.

10.28.4.22 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

inst	[IN] Target instance reference.
ifh	[IN] Compiled IFH
frame	[IN] Frame buffer excluding room for CRC.
length	[IN] Frame length excluding CRC.

Returns

Return code.

10.28.4.23 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

inst	[IN] Target instance reference.
info	[OUT] Pointer to structure that gets initialized to defaults.

Returns

VTSS_RC_OK. VTSS_RC_ERROR only if info == NULL.

10.28.4.24 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

inst	[IN] Target instance reference.
conf	[OUT] Packet DMA configuration structure.

Returns

VTSS_RC_OK.

10.28.4.25 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

inst	[IN] Target instance reference.
conf	[OUT] Packet DMA configuration structure.

Returns

VTSS_RC_OK.

10.28.4.26 vtss_packet_dma_offset()

```
vtss_rc vtss_packet_dma_offset (
            const vtss_inst_t inst,
            BOOL extraction,
            u32 * offset )
```

Retreive 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 approprate register offsets before this offset, such that the status offset if the last word written/read.

Parameters

inst	[IN] Target instance reference.
extraction	[IN]
offset	[OUT] Offset (32-bit word offset) for the DMA status register.

Returns

Return code.

10.29 vtss_api/include/vtss_pcs_10gbase_r_api.h File Reference

PCS_10BASE_R API.

```
#include <vtss/api/types.h>
```

10.29.1 Detailed Description

PCS_10BASE_R API.

10.30 vtss_api/include/vtss_phy_10g_api.h File Reference

10G PHY API

```
#include <vtss/api/types.h>
#include <vtss_misc_api.h>
```

10.30.1 Detailed Description

10G PHY API

This header file describes 10G PHY control functions

10.31 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_serd_pcs_cntl_t

  PHY MAC SerDes PCS Control, Reg16E3.
- struct vtss_phy_media_serd_pcs_cntl_t

  PHY MEDIA SerDes PCS Control, Reg23E3.
- struct vtss_phy_conf_t

  PHY configuration.
- struct vtss_phy_conf_1g_t

  PHY 1G configuration.
- struct vtss_phy_status_1g_t

  PHY 1G status.
- struct vtss_phy_power_conf_t

  PHY power configuration.
- struct vtss_phy_power_status_t

  PHY power status.
- struct vtss_phy_clock_conf_t

  PHY clock configuration.
- struct vtss_phy_veriphy_result_t

  VeriPHY result.
- struct vtss_phy_eee_conf_t

  EEE configuration.
- struct vtss_phy_enhanced_led_control_t

  enhanced LED control
- struct vtss_phy_statistic_t

  Phy statistic information.
- struct vtss_phy_loopback_t

  1G Phy loopbacks
- struct vtss_wol_mac_addr_t

  Structure for Wake-On-LAN MAC Address.
- struct vtss_secure_on_passwd_t

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

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

  Structure for Get PHY RC-PLL status.
- struct vtss_lcpll_status_t

  Structure for Get PHY LC-PLL status.

Macros

- #define MAX_CFG_BUF_SIZE 38
- #define MAX_STAT_BUF_SIZE 8
- #define VTSS_PHY_POWER_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, BASE1000_ACTIVITY,
  BASE100_FX_ACTIVITY, FORCE_LED_OFF, FORCE_LED_ON, FAST_LINK_FAIL }

  PHY LED modes.
- enum vtss_phy_led_number_t { **LED0**, **LED1**, **LED2**, **LED3** }

  List of LED pins per port.
- enum vtss_phy_media_interface_t {
  VTSS_PHY_MEDIA_IF_CU, VTSS_PHY_MEDIA_IF_SFP_PASSTHRU, VTSS_PHY_MEDIA_IF_FI_1000BX,
  VTSS_PHY_MEDIA_IF_FI_100FX,
  VTSS_PHY_MEDIA_IF_AMS_CU_PASSTHRU, VTSS_PHY_MEDIA_IF_AMS_FI_PASSTHRU, VTSS_PHY_MEDIA_IF_AMS
  VTSS_PHY_MEDIA_IF_AMS_FI_1000BX,
  VTSS_PHY_MEDIA_IF_AMS_CU_100FX, **VTSS_PHY_MEDIA_IF_AMS_FI_100FX** }

PHY media interface type.

- enum vtss_phy_mdi_t { VTSS_PHY_MDIX_AUTO, VTSS_PHY_MDI, VTSS_PHY_MDIX }

  PHY media interface type.

- enum rgmii_skew_delay_psec_t { rgmii_skew_delay_200_psec = 200, rgmii_skew_delay_800_psec = 800, rgmii_skew_delay_1100_psec = 1100, rgmii_skew_delay_1700_psec = 1700, rgmii_skew_delay_2000_psec = 2000, rgmii_skew_delay_2300_psec = 2300, rgmii_skew_delay_2600_psec = 2600, rgmii_skew_delay_3400_psec = 3400 }

  RGMII skew values.

- enum vtss_phy_forced_reset_t { VTSS_PHY_FORCE_RESET = 0, VTSS_PHY_NOFORCE_RESET = 1 }

  PHY forced reset interface type.

- enum vtss_phy_pkt_mode_t { VTSS_PHY_PKT_MODE_IEEE_1_5_KB, VTSS_PHY_PKT_MODE_JUMBO_9_KB, VTSS_PHY_PKT_MODE_JUMBO_12_KB }

  PHY packet mode configuration.

- enum vtss_phy_mode_t { VTSS_PHY_MODE_ANEG, VTSS_PHY_MODE_FORCED, VTSS_PHY_MODE_POWER_DOWN }

  PHY mode.

- enum vtss_phy_fast_link_fail_t { VTSS_PHY_FAST_LINK_FAIL_ENABLE = 1, VTSS_PHY_FAST_LINK_FAIL_DISABLE }

  PHY fast link failure pin enable/disable.

- enum vtss_phy_sigdet_polarity_t { VTSS_PHY_SIGDET_POLARITY_ACT_LOW = 1, VTSS_PHY_SIGDET_POLARITY_ACT_ }

  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_serd_pcs_sgmii_pre { VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_NONE = 0, VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_ONE = 1, VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_TWO = 2, VTSS_PHY_MAC_SERD_PCS_SGMII_IN_PRE_RSVD = 3 }

  PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

- enum vtss_phy_media_rem_fault_t { VTSS_PHY_MEDIA_SERD_PCS_REM_FAULT_NO_ERROR = 0, VTSS_PHY_MEDIA_SERD_PCS_REM_FAULT_OFFLINE = 1, VTSS_PHY_MEDIA_SERD_PCS_REM_FAULT_LINK_FAIL = 2, VTSS_PHY_MEDIA_SERD_PCS_REM_FAULT_ANEG_ERROR = 3 }

  PHY MEDIA SerDes PCS Remote Fault Indication, See Clause 37, Table 37-3.

- enum vtss_phy_media_force_ams_sel_t { VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_NORMAL = 0, VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_SERDES = 1, VTSS_PHY_MEDIA_FORCE_AMS_SELECTION_COPPER = 2 }

  PHY AMS Force configuration.

- enum vtss_phy_clk_source_t { VTSS_PHY_CLK_DISABLED, VTSS_PHY_SERDES_MEDIA, VTSS_PHY_COPPER_MEDIA, VTSS_PHY_TCLK_OUT, VTSS_PHY_LOCAL_XTAL }

  PHY clock sources.

- enum vtss_phy_freq_t { VTSS_PHY_FREQ_25M, VTSS_PHY_FREQ_125M, VTSS_PHY_FREQ_3125M }

  PHY clock frequencies.

- enum vtss_phy_clk_squelch { VTSS_PHY_CLK_SQUELCH_MAX = 0, VTSS_PHY_CLK_SQUELCH_MED = 1, VTSS_PHY_CLK_SQUELCH_MIN = 2, VTSS_PHY_CLK_SQUELCH_NONE = 3 }

  PHY clock squelch levels.

- enum vtss_phy_gpio_mode_t { VTSS_PHY_GPIO_ALT_0 = 0, VTSS_PHY_GPIO_ALT_1 = 1, VTSS_PHY_GPIO_ALT_2 = 2, VTSS_PHY_GPIO_OUT = 3, VTSS_PHY_GPIO_IN = 4 }

  GPIO pin operating mode.

- enum vtss_eee_mode_t { **EEE_DISABLE**, **EEE_ENABLE**, **EEE_REG_UPDATE** }

EEE mode.

- enum lb_type { VTSS_LB_1G_NONE, VTSS_LB_FAR_END, VTSS_LB_NEAR_END }

  Internal loop-back type.
- enum vtss_wol_passwd_len_type_t { VTSS_WOL_PASSWD_LEN_4 = 4, VTSS_WOL_PASSWD_LEN_6 = 6 }

  Structure for Wake-On-LAN Password Length configuration.
- enum vtss_fefi_mode_t { VTSS_100FX_FEFI_NORMAL = 0, VTSS_100FX_FEFI_FORCE_SUPPRESS = 2, VTSS_100FX_FEFI_FORCE_ENABLE = 3 }

  Far-End Failure Indication modes These settings map to PHY Reg23E3, bits 0 and 1.

Functions

- vtss_rc vtss_phy_pre_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Must be call previous to port PHY Reset (vtss_phy_reset).
- vtss_rc vtss_phy_post_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Must be call after port PHY Reset (vtss_phy_reset).
- vtss_rc vtss_phy_pre_system_reset (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Must be call before a system reset.
- vtss_rc vtss_phy_reset (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_reset_conf_t *const conf)

  Reset PHY.
- vtss_rc vtss_phy_reset_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_reset_conf_t *conf)

  Get reset configuration.
- vtss_rc vtss_phy_chip_temp_get (const vtss_inst_t inst, const vtss_port_no_t port_no, i16 *const temp)

  Get chip temperature.
- vtss_rc vtss_phy_chip_temp_init (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Init. chip temperature.
- vtss_rc vtss_phy_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_conf_t *const conf)

  Get PHY configuration.
- vtss_rc vtss_phy_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_conf_t *const conf)

  Set PHY configuration.
- vtss_rc vtss_phy_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)

  Get PHY status.
- vtss_rc vtss_phy_cl37_lp_abil_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)

  Get Clause37 Link pArtner's ability.
- vtss_rc vtss_phy_id_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_type_t *phy_id)

  Get the PHY type/id. The the Phy ID - can only be used after setting the operating mode.
- vtss_rc vtss_phy_conf_1g_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_conf_1g_t *const conf)

  Get PHY 1G configuration.
- vtss_rc vtss_phy_conf_1g_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_conf_1g_t *const conf)

  Set PHY 1G configuration.
- vtss_rc vtss_phy_status_1g_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_status_1g_t *const status)

  Get PHY 1G status.

- vtss_rc vtss_phy_power_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_power_conf_t *const conf)

  Get PHY power configuration.
- vtss_rc vtss_phy_power_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_power_conf_t *const conf)

  Set PHY power configuration.
- vtss_rc vtss_phy_power_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_power_status_t *const status)

  Get PHY power status.
- vtss_rc vtss_phy_clock_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_recov_clk_t clock_port, const vtss_phy_clock_conf_t *const conf)

  Set PHY clock configuration.
- vtss_rc vtss_phy_clock_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_recov_clk_t clock_port, vtss_phy_clock_conf_t *const conf, vtss_port_no_t *const clock_source)

  Get PHY clock configuration.
- vtss_rc vtss_phy_i2c_read (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 i2c_mux, const u8 i2c_reg_addr, const u8 i2c_device_addr, u8 *const value, u8 cnt, BOOL word_access)

  I2C read.
- vtss_rc vtss_phy_i2c_write (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 i2c_mux, const u8 i2c_reg_addr, const u8 i2c_device_addr, u8 *value, u8 cnt, BOOL word_access)

  I2C writes.
- vtss_rc vtss_phy_read (const vtss_inst_t inst, const vtss_port_no_t port_no, const u32 addr, u16 *const value)

  Read value from PHY register.
- vtss_rc vtss_phy_read_page (const vtss_inst_t inst, const vtss_port_no_t port_no, const u32 page, const u32 addr, u16 *const value)

  Read value from PHY register at a specific page. Page register is set to standard page when read is done.
- vtss_rc vtss_phy_mmd_read (const vtss_inst_t inst, const vtss_port_no_t port_no, const u16 devad, const u32 addr, u16 *const value)

  Read value from PHY mmd register.
- vtss_rc vtss_phy_mmd_write (const vtss_inst_t inst, const vtss_port_no_t port_no, const u16 devad, const u32 addr, u16 value)

  Write value to PHY mmd register.
- vtss_rc vtss_phy_write (const vtss_inst_t inst, const vtss_port_no_t port_no, const u32 addr, const u16 value)

  Write value to PHY register.
- vtss_rc vtss_phy_write_masked (const vtss_inst_t inst, const vtss_port_no_t port_no, const u32 addr, const u16 value, const u16 mask)

  Write masked value to PHY register.
- vtss_rc vtss_phy_write_masked_page (const vtss_inst_t inst, const vtss_port_no_t port_no, const u16 page, const u16 addr, const u16 value, const u16 mask)

  Write masked value to PHY register and setups the page register.
- vtss_rc vtss_phy_gpio_mode (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 gpio_no, const vtss_phy_gpio_mode_t mode)

  Configure GPIO mode.
- vtss_rc vtss_phy_gpio_get (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 gpio_no, BOOL *value)

  Get the value from a GPIO pin.
- vtss_rc vtss_phy_gpio_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 gpio_no, BOOL value)

  Set the value of a GPIO pin.
- vtss_rc vtss_phy_veriphy_start (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 mode)

  Start VeriPHY.

- vtss_rc vtss_phy_veriphy_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_veriphy_result_t *const result)

  Get VeriPHY result.
- vtss_rc vtss_phy_led_mode_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_led_mode_select_t led_mode_select)

  Setting the LEDs blink mode.
- vtss_rc vtss_phy_led_intensity_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_led_intensity intensity)

  Setting the LEDs intensity.
- vtss_rc vtss_phy_led_intensity_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_led_intensity *intensity)

  Getting the LEDs intensity.
- vtss_rc vtss_phy_enhanced_led_control_init (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_enhanced_led_control_t conf)

  Setting the enhanced LED control initial state (Should only be set once at startup)..
- vtss_rc vtss_phy_enhanced_led_control_init_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_enhanced_led_control_t *conf)

  Getting the enhanced LED control initial state (Should only be set once at startup)..
- vtss_rc vtss_phy_coma_mode_disable (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Pulling the coma mode pin low.
- vtss_rc vtss_phy_coma_mode_enable (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Pulling the coma mode pin high (Set in coma mode if no external hardware is controlling the pin)
- void vga_adc_debug (const vtss_inst_t inst, u8 vga_adc_pwr, vtss_port_no_t port_no)

  debug function for Atom family Rev. A. chips
- vtss_rc vtss_phy_port_eee_capable (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *eee_capable)

  Get information about if a port is EEE capable.
- vtss_rc vtss_phy_eee_ena (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL enable)

  Enabling / Disabling EEE (Energy Efficient Ethernet)
- vtss_rc vtss_phy_eee_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_eee_conf_t *conf)

  Getting the current EEE (Energy Efficient Ethernet) configuration.
- vtss_rc vtss_phy_eee_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_eee_conf_t conf)

  Setting the EEE (Energy Efficient Ethernet) configuration.
- vtss_rc vtss_phy_eee_power_save_state_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *rx_in_power_save_state, BOOL *tx_in_power_save_state)

  Getting the if phy is currently powered save mode due to EEE.
- vtss_rc vtss_phy_eee_link_partner_advertisements_get (const vtss_inst_t inst, const vtss_port_no_t port_no, u8 *advertisement)

  Getting the EEE advertisement.
- vtss_rc vtss_phy_event_enable_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_event_t ev_mask, const BOOL enable)

  Enabling / Disabling of events.
- vtss_rc vtss_phy_event_enable_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_event_t *ev_mask)

  Getting current interrupt event state.
- vtss_rc vtss_phy_event_poll (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_event_t *const ev_mask)

  Polling for active events.
- vtss_rc vtss_squelch_workaround (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL enable)

  Function for enabling/disabling squelch work around.

- vtss_rc vtss_phy_csr_wr (const vtss_inst_t inst, const u16 page, const vtss_port_no_t port_no, const u16 target, const u32 csr_reg_addr, const u32 value)

  Function for writing to CSR registers.
- vtss_rc vtss_phy_csr_rd (const vtss_inst_t inst, const u16 page, const vtss_port_no_t port_no, const u16 target, const u32 csr_reg_addr, u32 *value)

  Function for writing to CSR registers.
- vtss_rc vtss_phy_statistic_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_statistic_t *statistics)

  debug function for getting phy statistics.
- vtss_rc vtss_phy_do_page_chk_set (const vtss_inst_t inst, const BOOL enable)

  Debug function for enabling check of page register for all phy register accesses.
- vtss_rc vtss_phy_do_page_chk_get (const vtss_inst_t inst, BOOL *enable)

  Debug function for getting if check of page register is enabled.
- vtss_rc vtss_phy_loopback_set (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_loopback_t loopback)

  Debug function for setting phy internal loopback.
- vtss_rc vtss_phy_loopback_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_loopback_t *loopback)

  Debug function for getting the current phy internal loopback.
- vtss_rc vtss_phy_is_8051_crc_ok (const vtss_inst_t inst, const vtss_port_no_t port_no, u16 code_length, u16 expected_crc)

  Debug function for checking if the phy firmware is loaded correctly.
- vtss_rc vtss_phy_cfg_ob_post0 (const vtss_inst_t inst, const vtss_port_no_t port_no, const u16 value)

  Debug function for setting the ob post0 patch.
- vtss_rc vtss_phy_cfg_ib_cterm (const vtss_inst_t inst, const vtss_port_no_t port_no, const u8 ib_cterm_ena, const u8 ib_eq_mode)

  Debug function for setting the ib cterm patch.
- vtss_rc vtss_phy_atom12_patch_setttings_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_advertisment_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_advertisment_reg, const vtss_phy_conf_t phy_setup, vtss_port_status_t *const status)

  Function for finding flow control status based upon configuration and PHY registers.
- vtss_rc vtss_phy_debug_stat_print (const vtss_inst_t inst, const vtss_debug_printf_t pr, const vtss_port_no_t port_no, const BOOL print_hdr)

  debug function for printing PHY statistics
- vtss_rc vtss_phy_warm_start_failed_get (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Function for checking if any issue were seen during warm-start.
- vtss_rc vtss_phy_debug_phyinfo_print (const vtss_inst_t inst, const vtss_debug_printf_t pr, const vtss_port_no_t port_no, const BOOL print_hdr)

  debug function for printing some of the internal PHY state/configurations
- vtss_rc vtss_phy_debug_register_dump (const vtss_inst_t inst, const vtss_debug_printf_t pr, BOOL clear, const vtss_port_no_t port_no)

  debug function for printing some of the internal PHY state/configurations
- vtss_rc vtss_phy_detect_base_ports (const vtss_inst_t inst)

  Function for making getting the API updated with base port number for all ports. MUST not be called unless all ports are detected.
- vtss_rc vtss_phy_ext_connector_loopback (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL lpback)

Function for configuring External Connector Loopback.

- vtss_rc vtss_phy_serdes_sgmii_loopback (const vtss_inst_t inst, const vtss_port_no_t port_no, const u16 mode)

  Function for configuring MAC-SerDes(SGMII) Loopback.
- vtss_rc vtss_phy_serdes_fmedia_loopback (const vtss_inst_t inst, const vtss_port_no_t port_no, const u16 mode)

  Function for configuring Fibre-Media SerDes Loopback.
- vtss_rc vtss_phy_debug_regdump_print (const vtss_inst_t inst, const vtss_debug_printf_t pr, const vtss_port_no_t port_no, const vtss_port_no_t page_no, const BOOL print_hdr)

  debug function for printing some of the internal PHY Registers
- vtss_rc vtss_phy_wol_enable (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL enable)

  function to Enable or Disable WOL by enabling or disabling the interrupt
- vtss_rc vtss_phy_wol_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_wol_conf_t *const conf)

  function to Get Wake-On-LAN configuration
- vtss_rc vtss_phy_wol_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_wol_conf_t *const conf)

  function to Set Wake-On-LAN configuration
- vtss_rc vtss_phy_patch_settings_get (const vtss_inst_t inst, const vtss_port_no_t port_no, u8 *mcb_bus, u8 *cfg_buf, u8 *stat_buf)

  Debug function for getting PHY setting set by the micro patches.
- vtss_rc vtss_phy_serdes6g_rcpll_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_rcpll_status_t *rcpll_status)

  Debug function for getting PHY Serdes6G RC-PLL status.
- vtss_rc vtss_phy_serdes1g_rcpll_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_rcpll_status_t *rcpll_status)

  Debug function for getting PHY Serdes1G RC-PLL status.
- vtss_rc vtss_phy_lcpll_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_lcpll_status_t *lcpll_status)

  Debug function for getting PHY LC-PLL status.
- vtss_rc vtss_phy_reset_lcpll (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Debug function for Resetting the LCPLL for the PHY.
- vtss_rc vtss_phy_sd6g_ob_post_rd (vtss_inst_t inst, const vtss_port_no_t port_no, u8 *ob_post0, u8 *ob↩_post1)

  Debug function for reading the 6G SerDes ob_post0 and ob_post1 values.
- vtss_rc vtss_phy_sd6g_ob_post_wr (vtss_inst_t inst, const vtss_port_no_t port_no, const u8 ob_post0, const u8 ob_post1)

  Debug function for modifying the 6G SerDes ob_post0 and ob_post1 values.
- vtss_rc vtss_phy_sd6g_ob_lev_rd (vtss_inst_t inst, const vtss_port_no_t port_no, u8 *ob_level)

  Debug function for reading the 6G SerDes ob_level value.
- vtss_rc vtss_phy_sd6g_ob_lev_wr (vtss_inst_t inst, const vtss_port_no_t port_no, const u8 ob_level)

  Debug function for modifying the 6G SerDes ob_lev value.
- vtss_rc vtss_phy_mac_media_inhibit_odd_start (vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL mac_inhibit, const BOOL media_inhibit)

  Function to modify the values for the MAC and MEDIA I/F Inhibit ODD Start Delay.
- vtss_rc vtss_phy_fefi_get (vtss_inst_t inst, const vtss_port_no_t port_no, vtss_fefi_mode_t *fefi)

  Function to modify the values for the Far-End Fail Indication.
- vtss_rc vtss_phy_fefi_set (vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_fefi_mode_t fefi)

  Function to modify the values for the Far-End Fail Indication.
- vtss_rc vtss_phy_fefi_detect (vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *fefi_detect)

  Function to get the status for the Far-End Fail Indication.
- vtss_rc vtss_phy_mse_100m_get (vtss_inst_t inst, const vtss_port_no_t port_no, u32 *mse)

Function to get the Mean Squared Error (MSE), ie. Noise floor for 100m.

- vtss_rc vtss_phy_mse_1000m_get (vtss_inst_t inst, const vtss_port_no_t port_no, u32 *mseA, u32 *mseB, u32 *mseC, u32 *mseD)

  Function to get the Mean Squared Error (MSE), ie. Noise floor for 1000m.
- vtss_rc vtss_phy_read_tr_addr (vtss_inst_t inst, const vtss_port_no_t port_no, u16 tr_addr, u16 *tr_lower, u16 *tr_upper)

  Debug Function to retrieve the values from Token Ring.
- vtss_rc vtss_phy_is_viper_revB (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *is_viper_revB)

  Polling for to determine if the Chip Type and revision is Viper Rev_B.
- vtss_rc vtss_phy_ext_event_poll (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_event_t *const ev_mask)

  Polling for active EXT Interrupt events.
- vtss_rc vtss_phy_status_inst_poll (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)

  Get PHY status from the PHY Instance (Does not read PHY Registers).
- vtss_rc vtss_phy_macsec_csr_sd6g_rd (vtss_inst_t inst, const vtss_port_no_t port_no, const u16 target, const u32 csr_reg_addr, u32 *value)

  Function for Reading from 6G serdes registers (applicable only on VIPER,ELISE)
- vtss_rc vtss_phy_macsec_csr_sd6g_wr (vtss_inst_t inst, const vtss_port_no_t port_no, const u16 target, const u32 csr_reg_addr, u32 value)

  Function for Writing to 6G serdes registers (applicable only on VIPER,ELISE)
- 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)

10.31.1 Detailed Description

PHY API.

This header file describes PHY control functions

10.31.2 Macro Definition Documentation

10.31.2.1 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.

10.31.2.2 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.

10.31.2.3 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.

10.31.2.4 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.

10.31.2.5 VTSS_PHY_ACTIPHY_PWR

```
#define VTSS_PHY_ACTIPHY_PWR 100
```

ActiPHY power status

Definition at line 592 of file vtss_phy_api.h.

10.31.2.6 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.

10.31.2.7 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.

10.31.2.8 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.

10.31.2.9 VTSS_PHY_RECOV_CLK2

```
#define VTSS_PHY_RECOV_CLK2 1
```

RCVRD_CLK2

Definition at line 618 of file vtss_phy_api.h.

10.31.2.10 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.

10.31.2.11 VTSS_PHY_PAGE_STANDARD

```
#define VTSS_PHY_PAGE_STANDARD 0x0000
```

Standard registers

Definition at line 740 of file vtss_phy_api.h.

10.31.2.12 VTSS_PHY_PAGE_EXTENDED

```
#define VTSS_PHY_PAGE_EXTENDED 0x0001
```

Extended registers

Definition at line 741 of file vtss_phy_api.h.

10.31.2.13 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.

10.31.2.14 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.

10.31.2.15 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.

10.31.2.16 VTSS_PHY_PAGE_GPIO

```
#define VTSS_PHY_PAGE_GPIO 0x0010
```

GPIO registers

Definition at line 745 of file vtss_phy_api.h.

10.31.2.17 VTSS_PHY_PAGE_1588

```
#define VTSS_PHY_PAGE_1588 0x1588
```

1588 (PTP) registers

Definition at line 746 of file vtss_phy_api.h.

10.31.2.18 VTSS_PHY_PAGE_MACSEC

```
#define VTSS_PHY_PAGE_MACSEC 0x0004
```

MACSEC page

Definition at line 747 of file vtss_phy_api.h.

10.31.2.19 VTSS_PHY_PAGE_TEST

```
#define VTSS_PHY_PAGE_TEST 0x2A30
```

Test registers

Definition at line 748 of file vtss_phy_api.h.

10.31.2.20 VTSS_PHY_PAGE_TR

```
#define VTSS_PHY_PAGE_TR 0x52B5
```

Token ring registers

Definition at line 749 of file vtss_phy_api.h.

10.31.2.21 VTSS_PHY_PAGE_0x2DAF

```
#define VTSS_PHY_PAGE_0x2DAF 0x2DAF
```

0x2DAF registers

Definition at line 750 of file vtss_phy_api.h.

10.31.2.22 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.

10.31.2.23 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.

10.31.2.24 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.

10.31.2.25 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.

10.31.2.26 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.

10.31.2.27 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.

10.31.2.28 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.

10.31.2.29 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.

10.31.2.30 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.

10.31.2.31 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.

10.31.2.32 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.

10.31.2.33 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.

10.31.2.34 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.

10.31.2.35 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.

10.31.2.36 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.

10.31.2.37 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.

10.31.2.38 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.

10.31.2.39 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.

10.31.2.40 VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV

```
#define VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV (1 << 13)
```

PHY master/salve resolution error event

Definition at line 1224 of file vtss_phy_api.h.

10.31.2.41 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.

10.31.2.42 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.

10.31.2.43 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.

10.31.2.44 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.

10.31.2.45 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.

10.31.2.46 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.

10.31.2.47 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.

10.31.2.48 VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV

```
#define VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV (1 << 21)
```

PHY Ring Resiliancy Switchover complete interrupt event

Definition at line 1233 of file vtss_phy_api.h.

10.31.2.49 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.

10.31.2.50 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.

10.31.2.51 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.

10.31.2.52 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.

10.31.2.53 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.

10.31.2.54 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.

10.31.2.55 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.

10.31.2.56 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.

10.31.2.57 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.

10.31.3 Typedef Documentation

10.31.3.1 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.

10.31.3.2 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.

10.31.4 Enumeration Type Documentation

10.31.4.1 vtss_phy_led_mode_t

```
enum vtss_phy_led_mode_t
```

PHY LED modes.

Enumerator

LINK1000_ACTIVITY	No link in any speed on any media interface./Valid link at any speed on any media interface. Blink or pulse-stretch = Valid link at any speed on any media interface with activity present.
LINK100_ACTIVITY	No link in 1000BASE-T/Valid 1000BASE-T link. Blink or pulse-stretch = Valid 1000BASE-T link with activity present
LINK10_ACTIVITY	No link in 100BASE-T/Valid 100BASE-T link. Blink or pulse-stretch = Valid 100BASE-T link with activity present
LINK100_1000_ACTIVITY	No link in 10BASE-T/Valid 10BASE-T link. Blink or pulse-stretch = Valid 10BASE-T link with activity present
LINK10_1000_ACTIVITY	No link in 100BASE-T, 1000BASE-FX, or 1000BASE-TX/Valid 100BASE-T, 1000BASE-FX, or 1000BASE-TX link.Blink or pulse-stretch = Valid 100BASE-T, 1000BASE-FX, or 1000BASE-TX link with activity present.

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.
BASE1000_ACTIVITY	No link in 100BASE-FX.
BASE100_FX_ACTIVITY	No 1000BASE-X activity present.Blink or pulse-stretch = 1000BASE-X activity present.
FORCE_LED_OFF	No 100BASE-FX activity present, Blink or pulse-stretch = 100BASE-FX activity present.
FORCE_LED_ON	De-asserts the LED
FAST_LINK_FAIL	Asserts the LED

Definition at line 102 of file vtss_phy_api.h.

10.31.4.2 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.

10.31.4.3 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.

10.31.4.4 rgmii_skew_delay_psec_t

`enum rgmii_skew_delay_psec_t`

RGMII skew values.

Enumerator

rgmii_skew_delay_200_psec	RGMII 200 Peco-Second Skew
rgmii_skew_delay_800_psec	RGMII 800 Peco-Second Skew
rgmii_skew_delay_1100_psec	RGMII 1100 Peco-Second Skew
rgmii_skew_delay_1700_psec	RGMII 1700 Peco-Second Skew
rgmii_skew_delay_2000_psec	RGMII 2000 Peco-Second Skew
rgmii_skew_delay_2300_psec	RGMII 2300 Peco-Second Skew
rgmii_skew_delay_2600_psec	RGMII 2600 Peco-Second Skew
rgmii_skew_delay_3400_psec	RGMII 3400 Peco-Second Skew

Definition at line 182 of file vtss_phy_api.h.

10.31.4.5 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.

10.31.4.6 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.

10.31.4.7 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.

10.31.4.8 vtss_phy_fast_link_fail_t

enum vtss_phy_fast_link_fail_t

PHY fast link failure pin enable/disable.

Enumerator

VTSS_PHY_FAST_LINK_FAIL_ENABLE	Enable fast link failure pin
VTSS_PHY_FAST_LINK_FAIL_DISABLE	Disble fast link failure pin

Definition at line 325 of file vtss_phy_api.h.

10.31.4.9 vtss_phy_sigdet_polarity_t

enum vtss_phy_sigdet_polarity_t

PHY Sigdet pin polarity configuration.

Enumerator

VTSS_PHY_SIGDET_POLARITY_ACT_LOW	Set Sigdet polarity Active low
VTSS_PHY_SIGDET_POLARITY_ACT_HIGH	Set Sigdet polarity Active High

Definition at line 331 of file vtss_phy_api.h.

10.31.4.10 vtss_phy_unidirectional_t

enum vtss_phy_unidirectional_t

PHY Unidirectional enable/disable.

Enumerator

VTSS_PHY_UNIDIRECTIONAL_DISABLE	Disable Unidirectional (Default)
VTSS_PHY_UNIDIRECTIONAL_ENABLE	Enable Unidirectional

Definition at line 337 of file vtss_phy_api.h.

10.31.4.11 vtss_phy_mac_serd_pcs_sgmii_pre

enum vtss_phy_mac_serd_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.

10.31.4.12 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.

10.31.4.13 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.

10.31.4.14 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.

10.31.4.15 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.

10.31.4.16 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.

10.31.4.17 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.

10.31.4.18 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.

10.31.4.19 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.

10.31.4.20 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 specifed by Reg23E3.0
VTSS_100FX_FEFI_FORCE_ENABLE	Force FEFI, as specifed by Reg23E3.0

Definition at line 1900 of file vtss_phy_api.h.

10.31.5 Function Documentation

10.31.5.1 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

inst	[IN] Target instance reference.
port_no	[IN] Port number (MUST be the first port for the chip).

Returns

Return code.

10.31.5.2 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

inst	[IN] Target instance reference.
port_no	[IN] Port number (Any port with the chip can be used).

Returns

Return code.

10.31.5.3 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

inst	[IN] Target instance reference.
port_no	[IN] Port number (Any port with the chip can be used).

Returns

Return code.

10.31.5.4 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] Reset configuration.

Returns

Return code.

10.31.5.5 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

inst	[IN] Target instance reference
port_no	[IN] Port number (Any port within the chip can be used)
conf	[OUT] Reset configuration

Returns

Return code.

10.31.5.6 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

inst	[IN] Target instance reference
port_no	[IN] Port number (Any port within the chip can be used).
temp	[OUT] Chip temperature

Returns

Return code.

10.31.5.7 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

inst	[IN] Target instance reference
port_no	[IN] Port number (Any port within the chip can be used).

Returns

Return code.

10.31.5.8 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] PHY configuration.

Returns

Return code.

10.31.5.9 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] PHY configuration.

Returns

Return code.

10.31.5.10 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
status	[OUT] PHY status.

Returns

Return code.

10.31.5.11 vtss_phy_cl37_lp_abil_get()

```
vtss_rc vtss_phy_cl37_lp_abil_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_port_status_t *const status )
```

Get Clause37 Link pArtner's ability.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
status	[OUT] PHY status.

Returns

Return code.

10.31.5.12 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
phy_id	[OUT] PHY Type/id.

Returns

Return code.

10.31.5.13 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] PHY 1G configuration.

Returns

Return code.

10.31.5.14 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] PHY 1G configuration.

Returns

Return code.

10.31.5.15 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
status	[OUT] PHY 1G status.

Returns

Return code.

10.31.5.16 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] PHY power configuration.

Returns

Return code.

10.31.5.17 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] PHY power configuration.

Returns

Return code.

10.31.5.18 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
status	[OUT] PHY power configuration.

Returns

Return code.

10.31.5.19 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

inst	[IN] Target instance reference.
port_no	[IN] Port number to become clock source.
clock_port	[IN] Set configuration for this clock port.
conf	[IN] PHY clock configuration.

Returns

Return code.

10.31.5.20 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

inst	[IN] Target instance reference.
port_no	[IN] Port number of the first port at this PHY instance.
clock_port	[IN] Get configuration for this clock port.
conf	[OUT] PHY clock configuration.
clock_source	[OUT] Port number that is clock source for this clock_port.

Returns

Return code.

10.31.5.21 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
i2c_mux	[IN] The i2c clock mux
i2c_reg_addr	[IN] The i2c register address to access.
i2c_device_addr	[IN] The i2c address of the device to access.
value	[OUT] Pointer to where array which in going to contain the values read.
cnt	[IN] The number of registers to read. Note: The reg_addr is incremented by 1 for each of the read counts. If you want to read 16 bites registers (2 times 8 bits from the same register address), you need to do that by calling the vtss_phy_i2c_read twice, and not use the cnt (set cnt to 1).
word_access	[IN] Set to TRUE if the register data width is 16bit. FALSE = 8 bits data width

Returns

Return code.

10.31.5.22 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
i2c_mux	[IN] The i2c clock mux
i2c_reg_addr	[IN] The i2c register address to access.
i2c_device_addr	[IN] The i2c address of the device to access.
value	[IN] Pointer to where array containing the values to write.
cnt	[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).
word_access	[IN] Set to TRUE if the register data width is 16bit. FALSE = 8 bits data width

Returns

Return code.

10.31.5.23 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
addr	[IN] Register address. The page number is encoded in the 16 MSB.
value	[OUT] Register value.

Returns

Return code.

10.31.5.24 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
page	[IN] Page - Page do to the read at.
addr	[IN] Register address. The page number is encoded in the 16 MSB.
value	[OUT] Register value.

Returns

Return code.

10.31.5.25 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
devad	[IN] Devad register address.
addr	[IN] Register address.
value	[OUT] Register value.

Returns

Return code.

10.31.5.26 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
devad	[IN] Devad register address.
addr	[IN] Register address.
value	[OUT] Register value.

Returns

Return code.

10.31.5.27 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
addr	[IN] Register address. The page number is encoded in the 16 MSB.
value	[IN] Register value.

Returns

Return code.

10.31.5.28 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
addr	[IN] Register address. The page number is encoded in the 16 MSB.
value	[IN] Register value.
mask	[IN] Register mask.

Returns

Return code.

10.31.5.29 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
page	[IN] Page number.
addr	[IN] Register address. The page number is encoded in the 16 MSB.
value	[IN] Register value.
mask	[IN] Register mask.

Returns

Return code.

10.31.5.30 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

inst	[IN] Target instance reference.
port_no	[IN] Any port number within the chip containing the GPIO -
gpio_no	[IN] The GPIO number.
mode	[IN] The mode the GPIO pin should operate in.

Returns

VTSS_RC_OK when configuration was done correctly else error code.

10.31.5.31 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

inst	[IN] Target instance reference.
port_no	[IN] Any port number within the chip containing the GPIO.
gpio_no	[IN] The GPIO number.
value	[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.

10.31.5.32 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

inst	[IN] Target instance reference.
port_no	[IN] Any port number within the chip containing the GPIO.
gpio_no	[IN] The GPIO number.
value	[IN] The pin value. (TRUE = set pin high, FALSE = set pin low)

Returns

VTSS_RC_OK when setting was done correctly else error code.

10.31.5.33 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
mode	[IN] VeriPHY mode.

Returns

Return code.

10.31.5.34 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
result	[OUT VeriPHY result.

Returns

Return code.

10.31.5.35 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

inst	[IN] Target instance reference.
port_no	[IN] Port number the port in question.
led_mode_select	[IN] The LEDs mode

Returns

Return code.

10.31.5.36 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

inst	[IN] Target instance reference.
port_no	[IN] Port number - Don't care for time being (Just set to a valid port), since all LEDs intensities are set to the same.
intensity	[IN] The LEDs intensities in % (0-100)

Returns

Return code.

10.31.5.37 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

inst	[IN] Target instance reference.
port_no	[IN] Port number - Don't care for time being (Just set to a valid port), since all LEDs intensities are set to the same.
intensity	[IN] The LEDs intensities in % (0-100)

Returns

Return code.

10.31.5.38 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] Enhanced LED control configuration

Returns

Return code.

10.31.5.39 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] Enhanced LED control configuration

Returns

Return code.

10.31.5.40 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

inst	[IN] Target instance reference.
port_no	[IN] Port number (Any port number for the PHY which shall pull the coma mode pin low).

Returns

Return code.

10.31.5.41 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

inst	[IN] Target instance reference.
port_no	[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.

10.31.5.42 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

inst	[IN] Target instance reference.
vga_adc_pwr	[IN] allows VGA and/or ADC to power down for EEE
port_no	[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).

10.31.5.43 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
eee_capable	[OUT] True if port is EEE capable else FALSE

Returns

Return code.

10.31.5.44 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
enable	[IN] Enable EEE

Returns

Return code.

10.31.5.45 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] EEE configuration.

Returns

Return code.

10.31.5.46 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] EEE configuration.

Returns

Return code.

10.31.5.47 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
rx_in_power_save_state	[OUT] TRUE is phy rx part is in power save mode
tx_in_power_save_state	[OUT] TRUE is phy tx part is in power save mode

Returns

Return code.

10.31.5.48 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
advertisement	[OUT] Advertisement bit mask. Bit 0 = Link partner advertises 100BASE-T capability. Bit 1 = Link partner advertises 1000BASE-T capability.

Returns

Return code.

10.31.5.49 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
ev_mask	[IN] Mask containing events that are enabled/disabled
enable	[IN] Enable/disable of event

Returns

Return code.

10.31.5.50 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
ev_mask	[IN] Mask containing events that are enabled/disabled

Returns

Return code.

10.31.5.51 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
ev_mask	[OUT] Mask containing events that are active

Returns

Return code.

10.31.5.52 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

inst	[IN] Target instance reference.
port_no	[IN] Any phy port with the chip
enable	[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

10.31.5.53 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

inst	[IN] Target instance reference.
port_no	[IN] The port in question
page	[IN] The page for the CSR register access e.g. VTSS_PHY_PAGE_1588 or VTSS_PHY_PAGE_MACSEC
target	[IN] The CSR target
csr_reg_addr	[IN] The CSR register to write
value	[IN] The value to write

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something when wrong and access was never granted.

10.31.5.54 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

inst	[IN] Target instance reference.
port_no	[IN] Any phy port with the chip
page	[IN] The page for the CSR register access e.g. VTSS_PHY_PAGE_1588 or VTSS_PHY_PAGE_MACSEC
target	[IN] The CSR target
csr_reg_addr	[IN] The CSR register to read.
value	[IN] The value read

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something when wrong and access was never granted.

10.31.5.55 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

inst	[IN] Target instance reference.
port_no	[IN] Any phy port with the chip
statistics	[OUT] Pointer to where to put the statistics.

Returns

VTSS_RC_OK - Statistics is valid else statistics is invalid

10.31.5.56 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

inst	[IN] Target instance reference.
enable	[IN] TRUE to enable phy page register check. FALSE to disable

Returns

Return code. VTSS_RC_OK if phy page check were set.

10.31.5.57 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

inst	[IN] Target instance reference.
enable	[OUT] TRUE if phy page register check is enabled else FALSE

Returns

Return code. VTSS_RC_OK when enable is valid.

10.31.5.58 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

inst	[IN] Target instance reference.
port_no	[IN] Phy port that should have the internal loopback
loopback	[IN] Loopback type

Returns

Return code. VTSS_RC_OK if firmware is loaded correctly else VTSS_RC_ERROR

10.31.5.59 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

inst	[IN] Target instance reference.
port_no	[IN] Phy port that with the internal loopback
loopback	[IN] Current loopback type

Returns

Return code. VTSS_RC_OK if firmware is loaded correctly else VTSS_RC_ERROR

10.31.5.60 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

inst	[IN] Target instance reference.
port_no	[IN] Must the first PHY port within the chip.
code_length	[IN] The length of the microcode patch
expected_crc	[IN] The expected CRC.

Returns

Return code. VTSS_RC_OK if firmware is loaded correctly else VTSS_RC_ERROR

10.31.5.61 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
value	[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

10.31.5.62 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
ib_cterm_ena	[IN] The value of ib_cterm_ena to call the ib cterm patch with.
ib_eq_mode	[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

10.31.5.63 vtss_phy_atom12_patch_setttings_get()

```
vtss_rc vtss_phy_atom12_patch_setttings_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

inst	[IN] Target instance reference.

Parameters

port_no	[IN] PHY port within the chip.
mcb_bus	[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 .
cfg_buf	[IN] Pointer to array where to put the current settings.
stat_buf	[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

10.31.5.64 vtss_phy_reg_decode_status()

```
void vtss_phy_reg_decode_status (
            vtss_port_no_t port_no,
            u16 lp_auto_neg_advertisment_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

port_no	[IN] The port in question.
lp_auto_neg_advertisment_reg	[IN] The value from the register containing the Link partners auto negotiation advertisement (Standard page 5)
lp_1000base_t_status_reg	[IN] The value from the register containing the Link partners 1000BASE-T Status (Standard page 10)
mii_status_reg	[IN] The value from the register containing mii status (Standard page 1)
phy_setup	[IN] The phy configuration setup
status	[INOUT] Pointer to where to put the result

10.31.5.65 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_advertisment_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

inst	[IN] Target instance reference.
port_no	[IN] The port in question.
lp_auto_neg_advertisment_reg	[IN] The value from the register containing the Link partners auto negotiation advertisement (Standard page 5)
phy_setup	[IN] The phy configuration setup
status	[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.

10.31.5.66 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

inst	[IN] Target instance reference.
pr	[IN] Function pointer to print function e.g. CPRINTF
port_no	[IN] Port in question
print_hdr	[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.

10.31.5.67 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

inst	[IN] Target instance reference.
port_no	[IN] The port in question.

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

10.31.5.68 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

inst	[IN] Target instance reference.
pr	[IN] Function pointer to print function e.g. CPRINTF
port_no	[IN] Port in question
print_hdr	[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.

10.31.5.69 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

inst	[IN] Target instance reference.
pr	[IN] Function pointer to print function e.g. CPRINTF
clear	[IN] Set to TRUE to clear the counters & Stickt bits if any
port_no	[IN] Port in question

Returns

Return code. VTSS_RC_OK if not errors were seen during warm-start else VTSS_RC_ERROR.

10.31.5.70 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

inst	[IN] Target instance reference.

Returns

Return code. VTSS_RC_OK if all base ports were updated correctly else error code.

10.31.5.71 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

inst	[IN] Target instance reference.
port_no	[IN] The port in question.
lpback	[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.

10.31.5.72 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

inst	[IN] Target instance reference.
port_no	[IN] The port in question.
mode	[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.

10.31.5.73 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

inst	[IN] Target instance reference.
port_no	[IN] The port in question.
mode	[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.

10.31.5.74 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

inst	[IN] Target instance reference.
pr	[IN] Function pointer to print function e.g. CPRINTF
port_no	[IN] Port in question
page_no	[IN] Page No in question, 0, 1, 2, 3, 1588, MACSEC, TEST, TR
print_hdr	[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.

10.31.5.75 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

inst	[IN] Target instance reference.
port_no	[IN] Port in question
enable	[IN] Boolean, Enable=TRUE or 1, Disable=False or 0

Returns

Return code. VTSS_RC_OK if no errors.

10.31.5.76 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

inst	[IN] Target instance reference.
port_no	[IN] Port in question
conf	[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.

10.31.5.77 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

inst	[IN] Target instance reference.
port_no	[IN] Port in question
conf	[IN] Ptr to WoL Structure vtss_phy_wol_conf_t filled out by User

Returns

Return code. VTSS_RC_OK if no errors.

10.31.5.78 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.

Parameters

mcb_bus	[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 .
cfg_buf	[IN] Pointer to array where to put the current settings.
stat_buf	[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

10.31.5.79 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
rcpll_status	[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

10.31.5.80 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
rcpll_status	[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

10.31.5.81 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
lcpll_status	[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

10.31.5.82 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

inst	[IN] Target instance reference.
port_no	[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.

10.31.5.83 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
ob_post0	[OUT] ob_post0 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 6 bits, default:0, Customizeable.
ob_post1	[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.

10.31.5.84 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
ob_post0	[IN] Modify ob_post0 settings in serdes6g_ob_cfg for 6G SerDes Macro (See TN1052), 6 bits, default:0, Customizeable.
ob_post1	[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.

10.31.5.85 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
ob_level	[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.

10.31.5.86 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
ob_level	[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.

10.31.5.87 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
mac_inhibit	[IN] Modify Inhibit MAC Odd-start Delay settings, ie. Reg 16E3.2.
media_inhibit	[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.

10.31.5.88 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
fefi	[OUT] PHY port Far End Failure Indicator Config.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

10.31.5.89 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
fefi	[IN] PHY port Far End Failure Indicator Config.

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

10.31.5.90 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
fefi_detect	[OUT] PHY port Far End Failure Indicator

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

10.31.5.91 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
mse	[OUT] PHY port MSE Value In order to convert the returned mse value to dB, do the following This computation would be required for each channel pair mse_dbl = mse / (1024 ∗ 2048); Note: Convert to Double and Scale by 2^21 = 1024 ∗ 2048 mse_dbl = 20 ∗ log10(mse_dbl); Note: Convert to dB Nominal Computed Values for mse_dbl are: -22dB to -31dB

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

10.31.5.92 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
mseA	[OUT] PHY port MSE Chan A Value
mseB	[OUT] PHY port MSE Chan B Value
mseC	[OUT] PHY port MSE Chan C Value
mseD	[OUT] PHY port MSE Chan D Value In order to convert the returned mse value to dB, do the following This computation would be required for each channel pair mse_dbl = mse / (1024 ∗ 2048); Note: Convert to Double and Scale by 2^21 = 1024 ∗ 2048 mse_dbl = 20 ∗ log10(mse_dbl); Note: Convert to dB Nominal Computed Values for mse_dbl are: -22dB to -31dB

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

10.31.5.93 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

inst	[IN] Target instance reference.
port_no	[IN] PHY port within the chip.
tr_addr	[IN] Token Ring ADDR (TR16) to Read
tr_lower	[OUT] Token Ring Lower 16bits of Value (TR17)
tr_upper	[OUT] Token Ring Upper 16bits of Value (TR18)

Returns

Return code. VTSS_RC_OK if all Ok VTSS_RC_ERROR if and error occurred.

10.31.5.94 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
is_viper_revB	[OUT] Boolean to indicate that the Chip/Rev is Viper RevB

Returns

Return code.

10.31.5.95 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

inst	[IN] Target instance reference.
port_no	[IN] Port number
ev_mask	[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.

10.31.5.96 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
status	[OUT] PHY status.

Returns

Return code.

10.31.5.97 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

inst	[IN] Target instance reference.
port_no	[IN] Any phy port with the chip
target	[IN] The CSR target
csr_reg_addr	[IN] The CSR register to read,
value	[OUT] The value read

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something when wrong and access was never granted.

10.31.5.98 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

inst	[IN] Target instance reference.
port_no	[IN] Any phy port with the chip
target	[IN] The CSR target
csr_reg_addr	[IN] The CSR register to read,
value	[IN] The value read

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something when wrong and access was never granted.

10.31.5.99 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

inst	[IN] Target instance reference.
port_no	[IN] Any phy port with the chip

Returns

VTSS_RC_OK when new access can be done - VTSS_RC_ERROR if something when wrong and access was never granted.

10.32 vtss_api/include/vtss_phy_ts_api.h File Reference

PHY TimeStamping API.

```
#include <vtss/api/types.h>
#include <vtss/api/port.h>
#include "vtss_misc_api.h"
```

Data Structures

- struct vtss_phy_ts_alt_clock_mode_s

  parameter for setting the alternative clock mode.
- struct vtss_phy_ts_pps_config_s

  PPS Configuration.
- struct vtss_phy_timestamp_t

  PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)
- struct vtss_phy_ts_sertod_conf_t

  PHY timestamp in seconds and nanoseconds (10 bytes Timestamp)
- struct vtss_phy_ltc_freq_synth_s

  Frequency systhesis pulse configuration.
- struct vtss_phy_daisy_chain_conf_t

  SPI daisy chain configuration.
- struct vtss_phy_ts_fifo_sig_t

  Tx TSFIFO entry signature.
- struct vtss_phy_ts_eng_init_conf_t

  Defines the basic engine parameters passed to the engine_init_conf_get() function.
- struct vtss_phy_ts_eth_conf_t

  Analyzer Ethernet comparator configuration options.
- struct vtss_phy_ts_ip_conf_t

  Analyzer IP comparator configuration options.
- struct vtss_phy_ts_mpls_lvl_rng_t

  MPLS level range.
- struct vtss_phy_ts_mpls_conf_t

  Analyzer MPLS comparator configuration options.
- struct vtss_phy_ts_ach_conf_t

  Analyzer ACH comparator configuration options.
- struct vtss_phy_ts_gen_conf_t

  Analyzer Generic data configuration options using IP comparator.
- struct vtss_phy_ts_ptp_engine_flow_conf_t

  PTP engine flow configuration options.
- struct vtss_phy_ts_oam_engine_flow_conf_t

  OAM engine flow configuration options.
- struct vtss_phy_ts_generic_flow_conf_t

  Generic engine flow configuration options.
- struct vtss_phy_ts_engine_flow_conf_t

  Analyzer flow configuration options.
- struct vtss_phy_ts_ptp_conf_t

  Analyzer PTP comparator configuration options.
- struct vtss_phy_ts_ptp_engine_action_t

  Analyzer PTP action configuration options.
- struct vtss_phy_ts_y1731_oam_conf_t

  Analyzer OAM comparator, Y.1731 OAM Packet format configuration options.
- struct vtss_phy_ts_ietf_mpls_ach_oam_conf_t

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

- struct vtss_phy_ts_oam_engine_action_t

  OAM Action configuration options.
- struct vtss_phy_ts_generic_action_t

  Generic Action configuration option.
- struct vtss_phy_ts_engine_action_t

  Engine Action configuration options.
- struct vtss_phy_ts_stats_t

  Timestamping Statistics.
- struct vtss_phy_ts_init_conf_t

  Defines the initial parameters to be passed to init function.
- struct vtss_phy_ts_nphase_status_t

  n-phase status
- struct vtss_phy_10g_fifo_sync_t

  10G OOS workaround options
- struct vtss_phy_ts_fifo_conf_t

  Defines the params for FIFO SYNC function.

Macros

- #define VTSS_PHY_TS_FIFO_SIG_SRC_IP 0x01

  Defines Tx TSFIFO signature mask.
- #define VTSS_PHY_TS_FIFO_SIG_DEST_IP 0x02
- #define VTSS_PHY_TS_FIFO_SIG_MSG_TYPE 0x04
- #define VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM 0x08
- #define VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID 0x10
- #define VTSS_PHY_TS_FIFO_SIG_SEQ_ID 0x20
- #define VTSS_PHY_TS_FIFO_SIG_DEST_MAC 0x40
- #define VTSS_PHY_TS_FIFO_SIG_SH_SPORT_ID 0x80
- #define VTSS_PHY_TS_SIG_LEN 16
- #define VTSS_PHY_TS_SIG_TIME_STAMP_LEN 10
- #define VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN 1
- #define VTSS_PHY_TS_SIG_SEQ_ID_LEN 1
- #define VTSS_PHY_TS_SIG_MSG_TYPE_LEN 1
- #define VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN 10
- #define VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN 2
- #define VTSS_PHY_TS_SIG_DEST_IP_LEN 4
- #define VTSS_PHY_TS_SIG_SRC_IP_LEN 4
- #define VTSS_PHY_TS_SIG_DEST_MAC_LEN 6
- #define VTSS_PHY_TS_SIG_SH_SPORT_ID_LEN 8
- #define VTSS_PTP_IP_1588_VERSION_2_1 0x21

  Defines 1588 version code for Gen-2.1.
- #define VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT ((u8)0x01)
- #define VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST ((u8)0x02)
- #define VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST ((u8)0x04)
- #define VTSS_PHY_TS_ETH_MATCH_DEST_ADDR ((u8)0x00)
- #define VTSS_PHY_TS_ETH_MATCH_SRC_ADDR ((u8)0x01)
- #define VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST ((u8)0x02)
- #define VTSS_PHY_TS_TAG_TYPE_C ((u8)0x01)
- #define VTSS_PHY_TS_TAG_TYPE_S ((u8)0x02)
- #define VTSS_PHY_TS_TAG_TYPE_I ((u8)0x03)
- #define VTSS_PHY_TS_TAG_TYPE_B ((u8)0x04)

- #define VTSS_PHY_TS_TAG_RANGE_NONE ((u8)0x00)
- #define VTSS_PHY_TS_TAG_RANGE_OUTER ((u8)0x01)
- #define VTSS_PHY_TS_TAG_RANGE_INNER ((u8)0x02)
- #define VTSS_PHY_TS_IP_VER_4 0x01
- #define VTSS_PHY_TS_IP_VER_6 0x02
- #define VTSS_PHY_TS_IP_MATCH_SRC 0x00
- #define VTSS_PHY_TS_IP_MATCH_DEST 0x01
- #define VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST 0x02
- #define VTSS_PHY_TS_MPLS_STACK_DEPTH_1 ((u8)0x01)
- #define VTSS_PHY_TS_MPLS_STACK_DEPTH_2 ((u8)0x02)
- #define VTSS_PHY_TS_MPLS_STACK_DEPTH_3 ((u8)0x04)
- #define VTSS_PHY_TS_MPLS_STACK_DEPTH_4 ((u8)0x08)
- #define VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP 0x00
- #define VTSS_PHY_TS_MPLS_STACK_REF_POINT_END 0x01
- #define VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0 0x01

  Port to flow mapping within analyzer engine.
- #define VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1 0x02
- #define VTSS_PHY_TS_INGR_ENGINE_ERR 0x01

  Timestamp interrupt events.
- #define VTSS_PHY_TS_INGR_RW_PREAM_ERR 0x02
- #define VTSS_PHY_TS_INGR_RW_FCS_ERR 0x04
- #define VTSS_PHY_TS_EGR_ENGINE_ERR 0x08
- #define VTSS_PHY_TS_EGR_RW_FCS_ERR 0x10
- #define VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED 0x20
- #define VTSS_PHY_TS_EGR_FIFO_OVERFLOW 0x40
- #define VTSS_PHY_TS_DATA_IN_RSRVD_FIELD 0x80
- #define VTSS_PHY_TS_LTC_NEW_PPS_INTRPT 0x100
- #define VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD 0x200
- #define VTSS_PHY_TS_8487_XAUI_SEL_0 0x01

  Define the Channel selection for 8487.
- #define VTSS_PHY_TS_8487_XAUI_SEL_1 0x02
- #define VTSS_PHY_TS_INGR_DATAPATH_RESET 0x01

  1588 block reset
- #define VTSS_PHY_TS_EGR_DATAPATH_RESET 0x02
- #define VTSS_PHY_TS_INGR_LTC1_RESET 0x04
- #define VTSS_PHY_TS_EGR_LTC2_RESET 0x08
- #define VTSS_PHY_TS_EGR_FIFO_RESET 0x10

Typedefs

- typedef struct vtss_phy_ts_alt_clock_mode_s vtss_phy_ts_alt_clock_mode_t

  parameter for setting the alternative clock mode.
- typedef struct vtss_phy_ts_pps_config_s vtss_phy_ts_pps_conf_t

  PPS Configuration.
- typedef i64 vtss_phy_ts_scaled_ppb_t

  Data type defines the clock frequency ratio in scaled ppb.
- typedef struct vtss_phy_ltc_freq_synth_s vtss_phy_ltc_freq_synth_t

  Frequency systhesis pulse configuration.
- typedef u32 vtss_phy_ts_fifo_sig_mask_t
- typedef void(∗ vtss_phy_ts_fifo_read) (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_timestamp_t ∗const ts, const vtss_phy_ts_fifo_sig_t ∗const sig, void ∗cntxt, const vtss_phy_ts_fifo_status_t status)

  Tx TSFIFO read callback function prototype.
- typedef u8 vtss_phy_ts_engine_channel_map_t
- typedef u32 vtss_phy_ts_event_t
- typedef u32 vtss_phy_ts_8487_xaui_sel_t
- typedef u32 vtss_phy_ts_soft_reset_t

Enumerations

- enum vtss_phy_ts_todadj_status_t { VTSS_PHY_TS_TODADJ_INPROGRESS, VTSS_PHY_TS_TODADJ_DONE, VTSS_PHY_TS_TODADJ_FAIL }

  parameter describing various Tx TSFIFO status.
- enum vtss_phy_ts_fifo_status_t { VTSS_PHY_TS_FIFO_SUCCESS, VTSS_PHY_TS_FIFO_OVERFLOW }

  parameter describing various Tx TSFIFO status.
- enum vtss_phy_ts_encap_t { **VTSS_PHY_TS_ENCAP_ETH_PTP**, **VTSS_PHY_TS_ENCAP_ETH_IP_PTP**, **VTSS_PHY_TS_ENCAP_ETH_IP_IP_PTP**, **VTSS_PHY_TS_ENCAP_ETH_ETH_PTP**, **VTSS_PHY_TS_ENCAP_ETH_ETH_IP_PTP**, **VTSS_PHY_TS_ENCAP_ETH_MPLS_IP_PTP**, **VTSS_PHY_TS_ENCAP_ETH_MPLS_ETH_PTP**, **VTSS_PHY_TS_ENCAP_ETH_MPLS_ETH_IP_PTP**, **VTSS_PHY_TS_ENCAP_ETH_MPLS_ACH_PTP**, **VTSS_PHY_TS_ENCAP_ETH_OAM**, **VTSS_PHY_TS_ENCAP_ETH_ETH_OAM**, **VTSS_PHY_TS_ENCAP_ETH_MPLS_ETH_OAM**, **VTSS_PHY_TS_ENCAP_ETH_MPLS_ACH_OAM**, **VTSS_PHY_TS_ENCAP_ANY**, **VTSS_PHY_TS_ENCAP_ETH_GEN**, **VTSS_PHY_TS_ENCAP_NONE** }

  Analyzer supported frame encapsulation type.
- enum vtss_phy_ts_engine_t { VTSS_PHY_TS_PTP_ENGINE_ID_0, VTSS_PHY_TS_PTP_ENGINE_ID_1, VTSS_PHY_TS_OAM_ENGINE_ID_2A, VTSS_PHY_TS_OAM_ENGINE_ID_2B, VTSS_PHY_TS_ENGINE_ID_INVALID }

  Defines Analyzer engine ID.
- enum vtss_phy_ts_engine_flow_match_t { VTSS_PHY_TS_ENG_FLOW_MATCH_ANY, VTSS_PHY_TS_ENG_FLOW_MATC }

  Flow matching within an analyzer engine.
- enum vtss_phy_ts_ptp_clock_mode_t { VTSS_PHY_TS_PTP_CLOCK_MODE_BC1STEP, VTSS_PHY_TS_PTP_CLOCK_MODE_BC2STEP, VTSS_PHY_TS_PTP_CLOCK_MODE_TC1STEP, VTSS_PHY_TS_PTP_CLOCK_MODE_TC2STEP, VTSS_PHY_TS_PTP_DELAY_COMP_ENGINE }

  PTP Timestamp Engine operational modes.
- enum vtss_phy_ts_ptp_delaym_type_t { VTSS_PHY_TS_PTP_DELAYM_P2P, VTSS_PHY_TS_PTP_DELAYM_E2E }

  PTP delay measurement method.
- enum vtss_phy_ts_y1731_oam_delaym_type_t { VTSS_PHY_TS_Y1731_OAM_DELAYM_1DM, VTSS_PHY_TS_Y1731_OAM }

  Y.1731 OAM delay messurement method.
- enum vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t { VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAYM_DMM, VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAYM_LDM }

  IETF MPLS ACH, OAM delay messurement method.
- enum vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t { VTSS_PHY_TS_IETF_MPLS_ACH_OAM_TS_FORMAT_PTP = 0x3 }

  Analyzer OAM comparator, IETF MPLS ACH OAM Supported Timestamp Format.
- enum vtss_phy_ts_action_format { VTSS_PHY_TS_4_BYTE, VTSS_PHY_TS_10_BYTE }

  Timestamp format to be configured in action configuration.
- enum vtss_phy_ts_clockfreq_t { VTSS_PHY_TS_CLOCK_FREQ_125M, VTSS_PHY_TS_CLOCK_FREQ_15625M, VTSS_PHY_TS_CLOCK_FREQ_200M, VTSS_PHY_TS_CLOCK_FREQ_250M, VTSS_PHY_TS_CLOCK_FREQ_500M, VTSS_PHY_TS_CLOCK_FREQ_MAX }

  Timestamp block clock frequencies.
- enum vtss_phy_ts_clock_src_t { VTSS_PHY_TS_CLOCK_SRC_EXTERNAL, VTSS_PHY_TS_CLOCK_SRC_CLIENT_RX, VTSS_PHY_TS_CLOCK_SRC_CL VTSS_PHY_TS_CLOCK_SRC_LINE_RX, VTSS_PHY_TS_CLOCK_SRC_LINE_TX, VTSS_PHY_TS_CLOCK_SRC_INTERNAL }

  Clock input source.

- enum vtss_phy_ts_rxtimestamp_pos_t { VTSS_PHY_TS_RX_TIMESTAMP_POS_IN_PTP, VTSS_PHY_TS_RX_TIMESTAMP
  }

  defines Rx Timestamp position inside PTP frame.
- enum vtss_phy_ts_rxtimestamp_len_t { VTSS_PHY_TS_RX_TIMESTAMP_LEN_30BIT, VTSS_PHY_TS_RX_TIMESTAMP_L
  }

  Defines RX Timestamp format i.e. 30bit or 32bit Rx timestamp.
- enum vtss_phy_ts_fifo_mode_t { VTSS_PHY_TS_FIFO_MODE_NORMAL, VTSS_PHY_TS_FIFO_MODE_SPI
  }

  Defines Tx TSFIFO access mode.
- enum vtss_phy_ts_fifo_timestamp_len_t { VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_4BYTE, VTSS_PHY_TS_FIFO_TIMEST
  }

  Defines 4 bytes vs. the default 10 bytes Timestamp stored in Tx TSFIFO.
- enum vtss_phy_ts_tc_op_mode_t { VTSS_PHY_TS_TC_OP_MODE_A = 1, VTSS_PHY_TS_TC_OP_MODE_B = 0, VTSS_PHY_TS_TC_OP_MODE_C = 2 }

  defines the Transparent Clock Operating Mode.
- enum vtss_phy_ts_nphase_sampler_t { VTSS_PHY_TS_NPHASE_PPS_O, VTSS_PHY_TS_NPHASE_PPS_RI, VTSS_PHY_TS_NPHASE_EGR_SOF, VTSS_PHY_TS_NPHASE_ING_SOF, VTSS_PHY_TS_NPHASE_LS, VTSS_PHY_TS_NPHASE_MAX }

  enum for n-phase samplers
- enum vtss_phy_ts_ptp_message_type_t { **PTP_SYNC_MSG**, **PTP_DELAY_REQ_MSG**, **PTP_PDELAY_REQ_MSG**, **PTP_PDELAY_RESP_MSG** }

  PTP Event Message TYPES.

Functions

- vtss_rc vtss_phy_ts_alt_clock_saved_get (const vtss_inst_t inst, const vtss_port_no_t port_no, u32 *const saved)

  Get the latest saved nanosec counter from the alternative clock.
- vtss_rc vtss_phy_ts_alt_clock_mode_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_alt_clock_mode_t *const phy_alt_clock_mode)

  Get the alternative external clock mode.
- vtss_rc vtss_phy_ts_alt_clock_mode_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_alt_clock_mode_t *const phy_alt_clock_mode)

  Set the alternative clock mode. This function configures the loopbacks.
- vtss_rc vtss_phy_ts_pps_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_pps_conf_t *const phy_pps_conf)

  Set offset for the PPS generation.
- vtss_rc vtss_phy_ts_pps_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_pps_conf_t *const phy_pps_conf)

  Get offset for the PPS generation.
- vtss_rc vtss_phy_ts_ingress_latency_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const latency)

  Set the ingress latency.
- vtss_rc vtss_phy_ts_ingress_latency_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const latency)

  Get the ingress latency.
- vtss_rc vtss_phy_ts_egress_latency_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const latency)

  Set the egress latency.
- vtss_rc vtss_phy_ts_egress_latency_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const latency)

Get the egress latency.

- vtss_rc vtss_phy_ts_path_delay_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const path_delay)

  Set the path delay.

- vtss_rc vtss_phy_ts_path_delay_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const path_delay)

  Get the path delay.

- vtss_rc vtss_phy_ts_delay_asymmetry_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const delay_asym)

  Set the delay asymmetry.

- vtss_rc vtss_phy_ts_delay_asymmetry_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const delay_asym)

  Get the delay asymmetry.

- vtss_rc vtss_phy_ts_ptptime_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_timestamp_t *const ts)

  Set the current PTP time into the PHY.

- vtss_rc vtss_phy_ts_ptptime_set_done (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Setting of the current PTP time into the PHY is completed.

- vtss_rc vtss_phy_ts_ptptime_arm (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Arm the local time of the PHY so that in next pps it can be read.

- vtss_rc vtss_phy_ts_ptptime_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_timestamp_t *const ts)

  Get the armed PTP time from the PHY.

- vtss_rc vtss_phy_ts_load_ptptime_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_timestamp_t *const ts)

  Get the PTP time from the PHY load registers.

- vtss_rc vtss_phy_ts_sertod_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_sertod_conf_t *const sertod_conf)

  Set Enable/Disable Serial ToD.

- vtss_rc vtss_phy_ts_sertod_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_sertod_conf_t *const sertod_conf)

  Get Enable/Disable Serial ToD.

- vtss_rc vtss_phy_ts_loadpulse_delay_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const u16 *const delay)

  Set load pulse delay.

- vtss_rc vtss_phy_ts_loadpulse_delay_get (const vtss_inst_t inst, const vtss_port_no_t port_no, u16 *const delay)

  Get load pulse delay.

- vtss_rc vtss_phy_ts_clock_rateadj_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_scaled_ppb_t *const adj)

  Adjust the local clock rate.

- vtss_rc vtss_phy_ts_clock_rateadj_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_scaled_ppb_t *const adj)

  Get the clock rate adjustment value.

- vtss_rc vtss_phy_ts_clock_rateadj_ppm_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_scaled_ppb_t *const adj)

  Adjust ppm of the local clock rate .

- vtss_rc vtss_phy_ts_clock_rateadj_ppm_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_scaled_ppb_t *const adj)

  Get the clock rate ppm adjustment value.

- vtss_rc vtss_phy_ts_ptptime_adj1ns (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL incr)

  Increment/decrement the LTC clock value by 1 ns.

- vtss_rc vtss_phy_ts_timeofday_offset_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const i32 offset)

  Subtract offset from the current time.
- vtss_rc vtss_phy_ts_ongoing_adjustment (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_todadj_status_t *const ongoing_adjustment)

  Return the status of the LTC time adjustment.
- vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ltc_freq_synth_t *const ltc_freq_synthesis)

  Return the status of the LTC time adjustment.
- vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ltc_freq_synth_t *const ltc_freq_synthesis)

  Return the status of the LTC time adjustment.
- vtss_rc vtss_phy_daisy_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_daisy_chain_conf_t *const daisy_chain)

  configure the daisy chain for TS FIFO
- vtss_rc vtss_phy_daisy_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_daisy_chain_conf_t *const daisy_chain)

  getting the daisy chain for TS FIFO
- vtss_rc vtss_phy_ts_fifo_sig_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_fifo_sig_mask_t sig_mask)

  Set Tx TSFIFO i.e. frame signature in Tx TSFIFO.
- vtss_rc vtss_phy_ts_fifo_sig_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_fifo_sig_mask_t *const sig_mask)

  Get frame signature mask in Tx TSFIFO.
- vtss_rc vtss_phy_ts_fifo_empty (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Read timestamp from Tx TSFIFO.
- vtss_rc vtss_phy_ts_fifo_read_install (const vtss_inst_t inst, vtss_phy_ts_fifo_read rd_cb, void *cntxt)

  Install callback to read data (signature + timestamp) from Tx TSFIFO.
- vtss_rc vtss_phy_ts_fifo_read_cb_get (const vtss_inst_t inst, vtss_phy_ts_fifo_read *rd_cb, void **cntxt)

  Get the fifo read callback function installed.
- vtss_rc vtss_phy_ts_ingress_engine_init (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, const vtss_phy_ts_encap_t encap_type, const u8 flow_st_index, const u8 flow_end_index, const vtss_phy_ts_engine_flow_match_t flow_match_mode)

  Initialize an analyzer ingress engine for an encapsulation type.
- vtss_rc vtss_phy_ts_ingress_engine_init_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, vtss_phy_ts_eng_init_conf_t *const init_conf)

  Get the configuration parameters passed in engine_init of an analyzer ingress engine for a specific engine ID.
- vtss_rc vtss_phy_ts_ingress_engine_clear (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id)

  Clear/release an analyzer ingress engine already initialized.
- vtss_rc vtss_phy_ts_egress_engine_init (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, const vtss_phy_ts_encap_t encap_type, const u8 flow_st_index, const u8 flow_end_index, const vtss_phy_ts_engine_flow_match_t flow_match_mode)

  Initialize an analyzer egress engine for an encapsulation type.
- vtss_rc vtss_phy_ts_egress_engine_init_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, vtss_phy_ts_eng_init_conf_t *const init_conf)

  Get the configuration parameters passed in engine_init of an analyzer egress engine for a specific engine ID.
- vtss_rc vtss_phy_ts_egress_engine_clear (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id)

  Clear/release an analyzer egress engine already initialized.
- vtss_rc vtss_phy_ts_ingress_engine_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, const vtss_phy_ts_engine_flow_conf_t *const flow_conf)

  Configure ingress analyzer flow.

- vtss_rc vtss_phy_ts_ingress_engine_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, vtss_phy_ts_engine_flow_conf_t *const flow_conf)

  Get ingress analyzer flow.
- vtss_rc vtss_phy_ts_egress_engine_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, const vtss_phy_ts_engine_flow_conf_t *const flow_conf)

  Configure egress analyzer flow.
- vtss_rc vtss_phy_ts_egress_engine_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, vtss_phy_ts_engine_flow_conf_t *const flow_conf)

  Get egress analyzer flow.
- vtss_rc vtss_phy_ts_ingress_engine_action_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, const vtss_phy_ts_engine_action_t *const action_conf)

  Configure ingress analyzer engine action.
- vtss_rc vtss_phy_ts_ingress_engine_action_get (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, vtss_phy_ts_engine_action_t *const action_conf)

  Get ingress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the ingress engine.
- vtss_rc vtss_phy_ts_egress_engine_action_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, const vtss_phy_ts_engine_action_t *const action_conf)

  Configure egress analyzer engine action.
- vtss_rc vtss_phy_ts_egress_engine_action_get (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_engine_t eng_id, vtss_phy_ts_engine_action_t *const action_conf)

  Get egress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the ingress engine.
- vtss_rc vtss_phy_ts_event_enable_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL enable, const vtss_phy_ts_event_t ev_mask)

  Enabling / Disabling of events.
- vtss_rc vtss_phy_ts_event_enable_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_event_t *const ev_mask)

  Get Enabling of events.
- vtss_rc vtss_phy_ts_event_poll (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_event_t *const status)

  Polling function called at by interrupt or periodically.
- vtss_rc vtss_phy_ts_stats_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_stats_t *const statistics)

  Get Timestamp statistics.
- vtss_rc vtss_phy_ts_correction_overflow_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *const ingr_overflow, BOOL *const egr_overflow)

  Get the correction field overflow status in ingress and egress.
- vtss_rc vtss_phy_ts_mode_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL enable)

  Enable/disable timestamp block.
- vtss_rc vtss_phy_ts_mode_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *const enable)

  Get timestamp block status i.e. enable/disable.
- vtss_rc vtss_phy_ts_init (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_init_conf_t *const conf)

  Init timestamp block.
- vtss_rc vtss_phy_ts_init_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *const init_done, vtss_phy_ts_init_conf_t *const conf)

  Get the timestamp init config parameters.
- vtss_rc vtss_phy_ts_nphase_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_nphase_sampler_t sampler, vtss_phy_ts_nphase_status_t *const status)

  Get N-Phase sampler status.
- vtss_rc vtss_phy_ts_hiacc_set (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_nphase_sampler_t sampler, const BOOL enable)

Enable N-Phase sampler.

- vtss_rc vtss_phy_ts_hiacc_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_phy_ts_nphase_sampler_t sampler, BOOL const *enable)

  N-Phase sampler status get.

- vtss_rc vtss_phy_ts_block_soft_reset (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_soft_reset_t ts_reset)

  reset 1588 block.

- vtss_rc vtss_phy_ts_new_spi_mode_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const BOOL enable)

  Enable/disable New SPI mode for 8574-15 (Rev A & Rev B) that uses PPS0 pin as the new SPI_CLK.

- vtss_rc vtss_phy_ts_new_spi_mode_get (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL *const mode)

  Get New SPI mode for 8574-15 (Rev A & Rev B) described above.

- vtss_rc vtss_phy_1588_csr_reg_write (const vtss_inst_t inst, const vtss_port_no_t port_no, const u32 blk_id, const u16 csr_address, const u32 *const value)

  Set the the 1588 block CSR registers.

- vtss_rc vtss_phy_1588_csr_reg_read (const vtss_inst_t inst, const vtss_port_no_t port_no, const u32 blk_id, const u16 csr_address, u32 *const value)

  get the the 1588 block CSR registers.

- vtss_rc vtss_phy_ts_status_check (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL wait, const vtss_debug_printf_t pr)

  TS status check function supported for 10G Phys like 8488 & 8492.

- vtss_rc vtss_phy_ts_10g_extended_fifo_sync (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_10g_fifo_sync_t *conf)

  Reset 1588 PHY Fifo oos. As of now, supported for 10G Phys like 8488.

- vtss_rc vtss_phy_ts_viper_fifo_reset (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_phy_ts_fifo_conf_t *fifo_conf)

  Viper 1588 FIFO reset.

- vtss_rc vtss_phy_ts_tesla_tsp_fifo_sync (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_debug_printf_t pr, const vtss_phy_ts_fifo_conf_t *fifo_conf, BOOL *OOS)

  API to detect and correct Timestamp FIFO OOS.

- vtss_rc vtss_phy_1g_ts_fifo_sync (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_debug_printf_t pr, const vtss_phy_ts_fifo_conf_t *fifo_conf, BOOL *OOS)

  API to detect and correct Timestamp FIFO OOS for 1G PHY's (Viper and Tesla)

- vtss_rc vtss_phy_1588_debug_reg_read (const vtss_inst_t inst, const vtss_port_no_t port_no, const u32 blk_id, const vtss_debug_printf_t p_routine)

  API to dump PHY timestamp registers (for Debugging)

- vtss_rc vtss_phy_ts_flow_clear_cf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, BOOL ingress, const vtss_phy_ts_engine_t eng_id, u8 act_id, vtss_phy_ts_ptp_message_type_t msgtype)

  Clear Correction field for specified PTP message type.

10.32.1 Detailed Description

PHY TimeStamping API.

This header file describes PTP/OAM TimeStamping API functions

10.32.2 Macro Definition Documentation

10.32.2.1 VTSS_PHY_TS_FIFO_SIG_SRC_IP

```
#define VTSS_PHY_TS_FIFO_SIG_SRC_IP 0x01
```

Defines Tx TSFIFO signature mask.

Src IP address: inner IP for IP-over-IP

Definition at line 570 of file vtss_phy_ts_api.h.

10.32.2.2 VTSS_PHY_TS_FIFO_SIG_DEST_IP

```
#define VTSS_PHY_TS_FIFO_SIG_DEST_IP 0x02
```

Dest IP address

Definition at line 571 of file vtss_phy_ts_api.h.

10.32.2.3 VTSS_PHY_TS_FIFO_SIG_MSG_TYPE

```
#define VTSS_PHY_TS_FIFO_SIG_MSG_TYPE 0x04
```

Message type

Definition at line 573 of file vtss_phy_ts_api.h.

10.32.2.4 VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM

```
#define VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM 0x08
```

Domain number

Definition at line 574 of file vtss_phy_ts_api.h.

10.32.2.5 VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID

```
#define VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID 0x10
```

Source port identity

Definition at line 575 of file vtss_phy_ts_api.h.

10.32.2.6 VTSS_PHY_TS_FIFO_SIG_SEQ_ID

```
#define VTSS_PHY_TS_FIFO_SIG_SEQ_ID 0x20
```

PTP frame Sequence ID

Definition at line 576 of file vtss_phy_ts_api.h.

10.32.2.7 VTSS_PHY_TS_FIFO_SIG_DEST_MAC

```
#define VTSS_PHY_TS_FIFO_SIG_DEST_MAC 0x40
```

Dest MAC address

Definition at line 578 of file vtss_phy_ts_api.h.

10.32.2.8 VTSS_PHY_TS_FIFO_SIG_SH_SPORT_ID

```
#define VTSS_PHY_TS_FIFO_SIG_SH_SPORT_ID 0x80
```

PTP:Short Source port identity

Definition at line 581 of file vtss_phy_ts_api.h.

10.32.2.9 VTSS_PHY_TS_SIG_LEN

```
#define VTSS_PHY_TS_SIG_LEN 16
```

TS Signature length

Definition at line 586 of file vtss_phy_ts_api.h.

10.32.2.10 VTSS_PHY_TS_SIG_TIME_STAMP_LEN

```
#define VTSS_PHY_TS_SIG_TIME_STAMP_LEN 10
```

Timestamp Bytes in TS Signature

Definition at line 587 of file vtss_phy_ts_api.h.

10.32.2.11 VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN

```
#define VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN 1
```

Domain number length in TS Signature

Definition at line 589 of file vtss_phy_ts_api.h.

10.32.2.12 VTSS_PHY_TS_SIG_SEQ_ID_LEN

```
#define VTSS_PHY_TS_SIG_SEQ_ID_LEN 1
```

Seq ID length in TS Signature

Definition at line 590 of file vtss_phy_ts_api.h.

10.32.2.13 VTSS_PHY_TS_SIG_MSG_TYPE_LEN

```
#define VTSS_PHY_TS_SIG_MSG_TYPE_LEN 1
```

MSG Type length in TS Signature

Definition at line 591 of file vtss_phy_ts_api.h.

10.32.2.14 VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN

```
#define VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN 10
```

Source Port length in TS Signature

Definition at line 592 of file vtss_phy_ts_api.h.

10.32.2.15 VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN

```
#define VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN 2
```

Sequence ID length in TS Signature

Definition at line 593 of file vtss_phy_ts_api.h.

10.32.2.16 VTSS_PHY_TS_SIG_DEST_IP_LEN

```
#define VTSS_PHY_TS_SIG_DEST_IP_LEN 4
```

Dest IP length in TS Signature

Definition at line 594 of file vtss_phy_ts_api.h.

10.32.2.17 VTSS_PHY_TS_SIG_SRC_IP_LEN

```
#define VTSS_PHY_TS_SIG_SRC_IP_LEN 4
```

SRC IP length in TS Signature

Definition at line 595 of file vtss_phy_ts_api.h.

10.32.2.18 VTSS_PHY_TS_SIG_DEST_MAC_LEN

```
#define VTSS_PHY_TS_SIG_DEST_MAC_LEN 6
```

Dest MAC length in TS Signature

Definition at line 596 of file vtss_phy_ts_api.h.

10.32.2.19 VTSS_PHY_TS_SIG_SH_SPORT_ID_LEN

```
#define VTSS_PHY_TS_SIG_SH_SPORT_ID_LEN 8
```

Source port ID length in TS Signature

Definition at line 599 of file vtss_phy_ts_api.h.

10.32.2.20 VTSS_PTP_IP_1588_VERSION_2_1

```
#define VTSS_PTP_IP_1588_VERSION_2_1 0x21
```

Defines 1588 version code for Gen-2.1.

1588 block version for Gen2.1

Definition at line 605 of file vtss_phy_ts_api.h.

10.32.2.21 VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT

```
#define VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT ((u8)0x01)
```

Full 48-bit address match

Definition at line 965 of file vtss_phy_ts_api.h.

10.32.2.22 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST

```
#define VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST ((u8)0x02)
```

Match any unicast MAC addres

Definition at line 966 of file vtss_phy_ts_api.h.

10.32.2.23 VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST

```
#define VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTICAST ((u8)0x04)
```

Match any multicast MAC addres

Definition at line 967 of file vtss_phy_ts_api.h.

10.32.2.24 VTSS_PHY_TS_ETH_MATCH_DEST_ADDR

```
#define VTSS_PHY_TS_ETH_MATCH_DEST_ADDR ((u8)0x00)
```

Match destination MAC address

Definition at line 969 of file vtss_phy_ts_api.h.

10.32.2.25 VTSS_PHY_TS_ETH_MATCH_SRC_ADDR

```
#define VTSS_PHY_TS_ETH_MATCH_SRC_ADDR ((u8)0x01)
```

Match source MAC address

Definition at line 970 of file vtss_phy_ts_api.h.

10.32.2.26 VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST

```
#define VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST ((u8)0x02)
```

Match source or destination MAC address

Definition at line 971 of file vtss_phy_ts_api.h.

10.32.2.27 VTSS_PHY_TS_TAG_TYPE_C

```
#define VTSS_PHY_TS_TAG_TYPE_C ((u8)0x01)
```

Tag type: C

Definition at line 977 of file vtss_phy_ts_api.h.

10.32.2.28 VTSS_PHY_TS_TAG_TYPE_S

```
#define VTSS_PHY_TS_TAG_TYPE_S ((u8)0x02)
```

Tag type: S

Definition at line 978 of file vtss_phy_ts_api.h.

10.32.2.29 VTSS_PHY_TS_TAG_TYPE_I

```
#define VTSS_PHY_TS_TAG_TYPE_I ((u8)0x03)
```

Tag type: I

Definition at line 979 of file vtss_phy_ts_api.h.

10.32.2.30 VTSS_PHY_TS_TAG_TYPE_B

```
#define VTSS_PHY_TS_TAG_TYPE_B ((u8)0x04)
```

Tag type: B

Definition at line 980 of file vtss_phy_ts_api.h.

10.32.2.31 VTSS_PHY_TS_TAG_RANGE_NONE

```
#define VTSS_PHY_TS_TAG_RANGE_NONE ((u8)0x00)
```

Neither inner nor outer tag allows range config

Definition at line 983 of file vtss_phy_ts_api.h.

10.32.2.32 VTSS_PHY_TS_TAG_RANGE_OUTER

```
#define VTSS_PHY_TS_TAG_RANGE_OUTER ((u8)0x01)
```

Outer tag allows range config

Definition at line 984 of file vtss_phy_ts_api.h.

10.32.2.33 VTSS_PHY_TS_TAG_RANGE_INNER

```
#define VTSS_PHY_TS_TAG_RANGE_INNER ((u8)0x02)
```

Inner tag allows range config

Definition at line 985 of file vtss_phy_ts_api.h.

10.32.2.34 VTSS_PHY_TS_IP_VER_4

```
#define VTSS_PHY_TS_IP_VER_4 0x01
```

Version: IPv4

Definition at line 1021 of file vtss_phy_ts_api.h.

10.32.2.35 VTSS_PHY_TS_IP_VER_6

```
#define VTSS_PHY_TS_IP_VER_6 0x02
```

Version: IPv6

Definition at line 1022 of file vtss_phy_ts_api.h.

10.32.2.36 VTSS_PHY_TS_IP_MATCH_SRC

```
#define VTSS_PHY_TS_IP_MATCH_SRC 0x00
```

Match source IP address

Definition at line 1033 of file vtss_phy_ts_api.h.

10.32.2.37 VTSS_PHY_TS_IP_MATCH_DEST

```
#define VTSS_PHY_TS_IP_MATCH_DEST 0x01
```

Match destination IP address

Definition at line 1034 of file vtss_phy_ts_api.h.

10.32.2.38 VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST

```
#define VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST 0x02
```

Match source or destination IP address

Definition at line 1035 of file vtss_phy_ts_api.h.

10.32.2.39 VTSS_PHY_TS_MPLS_STACK_DEPTH_1

```
#define VTSS_PHY_TS_MPLS_STACK_DEPTH_1 ((u8)0x01)
```

MPLS stack of depth 1 only allows

Definition at line 1068 of file vtss_phy_ts_api.h.

10.32.2.40 VTSS_PHY_TS_MPLS_STACK_DEPTH_2

```
#define VTSS_PHY_TS_MPLS_STACK_DEPTH_2 ((u8)0x02)
```

MPLS stack of depth 2 only allows

Definition at line 1069 of file vtss_phy_ts_api.h.

10.32.2.41 VTSS_PHY_TS_MPLS_STACK_DEPTH_3

```
#define VTSS_PHY_TS_MPLS_STACK_DEPTH_3 ((u8)0x04)
```

MPLS stack of depth 3 only allows

Definition at line 1070 of file vtss_phy_ts_api.h.

10.32.2.42 VTSS_PHY_TS_MPLS_STACK_DEPTH_4

```
#define VTSS_PHY_TS_MPLS_STACK_DEPTH_4 ((u8)0x08)
```

MPLS stack of depth 4 only allows

Definition at line 1071 of file vtss_phy_ts_api.h.

10.32.2.43 VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP

```
#define VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP 0x00
```

Search starts from the top of the stack

Definition at line 1074 of file vtss_phy_ts_api.h.

10.32.2.44 VTSS_PHY_TS_MPLS_STACK_REF_POINT_END

```
#define VTSS_PHY_TS_MPLS_STACK_REF_POINT_END 0x01
```

Search starts from the end of the stack

Definition at line 1075 of file vtss_phy_ts_api.h.

10.32.2.45 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0

```
#define VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0 0x01
```

Port to flow mapping within analyzer engine.

Note

This is applicable for multi-channel timestamp block.Channel-0 mapped

Definition at line 1139 of file vtss_phy_ts_api.h.

10.32.2.46 VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1

```
#define VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1 0x02
```

Channel-1 mapped

Definition at line 1140 of file vtss_phy_ts_api.h.

10.32.2.47 VTSS_PHY_TS_INGR_ENGINE_ERR

```
#define VTSS_PHY_TS_INGR_ENGINE_ERR 0x01
```

Timestamp interrupt events.

More than one engine find match

Definition at line 1508 of file vtss_phy_ts_api.h.

10.32.2.48 VTSS_PHY_TS_INGR_RW_PREAM_ERR

```
#define VTSS_PHY_TS_INGR_RW_PREAM_ERR 0x02
```

Preamble too short to append timestamp

Definition at line 1509 of file vtss_phy_ts_api.h.

10.32.2.49 VTSS_PHY_TS_INGR_RW_FCS_ERR

```
#define VTSS_PHY_TS_INGR_RW_FCS_ERR 0x04
```

FCS error in ingress

Definition at line 1510 of file vtss_phy_ts_api.h.

10.32.2.50 VTSS_PHY_TS_EGR_ENGINE_ERR

```
#define VTSS_PHY_TS_EGR_ENGINE_ERR 0x08
```

More than one engine find match

Definition at line 1511 of file vtss_phy_ts_api.h.

10.32.2.51 VTSS_PHY_TS_EGR_RW_FCS_ERR

```
#define VTSS_PHY_TS_EGR_RW_FCS_ERR 0x10
```

FCS error in egress

Definition at line 1512 of file vtss_phy_ts_api.h.

10.32.2.52 VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED

```
#define VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED 0x20
```

Timestamp captured in Tx TSFIFO

Definition at line 1513 of file vtss_phy_ts_api.h.

10.32.2.53 VTSS_PHY_TS_EGR_FIFO_OVERFLOW

```
#define VTSS_PHY_TS_EGR_FIFO_OVERFLOW 0x40
```

Tx TSFIFO overflow

Definition at line 1514 of file vtss_phy_ts_api.h.

10.32.2.54 VTSS_PHY_TS_DATA_IN_RSRVD_FIELD

```
#define VTSS_PHY_TS_DATA_IN_RSRVD_FIELD 0x80
```

Data in reserved Field

Definition at line 1515 of file vtss_phy_ts_api.h.

10.32.2.55 VTSS_PHY_TS_LTC_NEW_PPS_INTRPT

```
#define VTSS_PHY_TS_LTC_NEW_PPS_INTRPT 0x100
```

New PPS pushed onto external PPS pin

Definition at line 1516 of file vtss_phy_ts_api.h.

10.32.2.56 VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD

```
#define VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD 0x200
```

New LTC value either loaded in to HW or saved into registers

Definition at line 1517 of file vtss_phy_ts_api.h.

10.32.2.57 VTSS_PHY_TS_8487_XAUI_SEL_0

```
#define VTSS_PHY_TS_8487_XAUI_SEL_0 0x01
```

Define the Channel selection for 8487.

Select XAUI Lane - 0

Definition at line 1736 of file vtss_phy_ts_api.h.

10.32.2.58 VTSS_PHY_TS_8487_XAUI_SEL_1

```
#define VTSS_PHY_TS_8487_XAUI_SEL_1 0x02
```

Select XAUI Lane - 1

Definition at line 1737 of file vtss_phy_ts_api.h.

10.32.2.59 VTSS_PHY_TS_INGR_DATAPATH_RESET

```
#define VTSS_PHY_TS_INGR_DATAPATH_RESET 0x01
```

1588 block reset

chip's ingress data path in the 1588 processing block

Definition at line 1938 of file vtss_phy_ts_api.h.

10.32.2.60 VTSS_PHY_TS_EGR_DATAPATH_RESET

```
#define VTSS_PHY_TS_EGR_DATAPATH_RESET 0x02
```

chip's egress data path in the 1588 processing block

Definition at line 1939 of file vtss_phy_ts_api.h.

10.32.2.61 VTSS_PHY_TS_INGR_LTC1_RESET

```
#define VTSS_PHY_TS_INGR_LTC1_RESET 0x04
```

Ingress LTC clock domain logic for this channel

Definition at line 1940 of file vtss_phy_ts_api.h.

10.32.2.62 VTSS_PHY_TS_EGR_LTC2_RESET

```
#define VTSS_PHY_TS_EGR_LTC2_RESET 0x08
```

Egress LTC clock domain logic for this channel

Definition at line 1941 of file vtss_phy_ts_api.h.

10.32.2.63 VTSS_PHY_TS_EGR_FIFO_RESET

```
#define VTSS_PHY_TS_EGR_FIFO_RESET 0x10
```

Egress FIFO reset

Definition at line 1942 of file vtss_phy_ts_api.h.

10.32.3 Typedef Documentation

10.32.3.1 vtss_phy_ts_alt_clock_mode_t

```
typedef struct vtss_phy_ts_alt_clock_mode_s vtss_phy_ts_alt_clock_mode_t
```

parameter for setting the alternative clock mode.

external clock output configuration.

10.32.3.2 vtss_phy_ts_scaled_ppb_t

```
typedef i64 vtss_phy_ts_scaled_ppb_t
```

Data type defines the clock frequency ratio in scaled ppb.

Note

The frequency of the internal clock can be adjusted in units of scaledPartsPerBillion, which is defined as the rate in units of ppb and multiplied by 2^16 and contained in a signed 64 bit value. For example, 2.5 ppb is expressed as 0000 0000 0002 8000

Definition at line 393 of file vtss_phy_ts_api.h.

10.32.3.3 vtss_phy_ts_fifo_sig_mask_t

```
typedef u32 vtss_phy_ts_fifo_sig_mask_t
```

Signature mask which can be OR of multiple fields above

Definition at line 584 of file vtss_phy_ts_api.h.

10.32.3.4 vtss_phy_ts_fifo_read

```
typedef void(* vtss_phy_ts_fifo_read) (const vtss_inst_t inst, const vtss_port_no_t port_↩
no, const vtss_phy_timestamp_t *const ts, const vtss_phy_ts_fifo_sig_t *const sig, void *cntxt,
const vtss_phy_ts_fifo_status_t status)
```

Tx TSFIFO read callback function prototype.

Tx TSFIFO API to access the HW TXFIFO. Application has to install the callback function which is called to push timestamp from the HW TXFIFO to the application. inst handle to an API instance port_no port number ts captured timestamp sig timestamp signature cntxt context to be returned in callback status FIFO read status

Definition at line 696 of file vtss_phy_ts_api.h.

10.32.3.5 vtss_phy_ts_engine_channel_map_t

```
typedef u8 vtss_phy_ts_engine_channel_map_t
```

Channel-0 or channel-1 or both the channels

Definition at line 1141 of file vtss_phy_ts_api.h.

10.32.3.6 vtss_phy_ts_event_t

```
typedef u32 vtss_phy_ts_event_t
```

Int events: Single event or 'OR' multiple events above

Definition at line 1519 of file vtss_phy_ts_api.h.

10.32.3.7 vtss_phy_ts_8487_xaui_sel_t

```
typedef u32 vtss_phy_ts_8487_xaui_sel_t
```

XAUI Lane-0 or Lane-1 or both

Definition at line 1738 of file vtss_phy_ts_api.h.

10.32.3.8 vtss_phy_ts_soft_reset_t

```
typedef u32 vtss_phy_ts_soft_reset_t
```

Reset blocks: Single or 'OR' multiple above

Definition at line 1944 of file vtss_phy_ts_api.h.

10.32.4 Enumeration Type Documentation

10.32.4.1 vtss_phy_ts_todadj_status_t

```
enum vtss_phy_ts_todadj_status_t
```

parameter describing various Tx TSFIFO status.

Enumerator

VTSS_PHY_TS_TODADJ_INPROGRESS	ToD Adjustment is in progress
VTSS_PHY_TS_TODADJ_DONE	ToD Adjustment is completed
VTSS_PHY_TS_TODADJ_FAIL	ToD Adjustment Failed

Definition at line 477 of file vtss_phy_ts_api.h.

10.32.4.2 vtss_phy_ts_fifo_status_t

```
enum vtss_phy_ts_fifo_status_t
```

parameter describing various Tx TSFIFO status.

Following Tx TSFIFO related API are used if FIFO access mode is set as PHY_TS_FIFO_MODE_NORMAL. In SPI mode, timestamps are pushed into SPI interface as soon as they are available.

Enumerator

VTSS_PHY_TS_FIFO_SUCCESS	FIFO read success
VTSS_PHY_TS_FIFO_OVERFLOW	FIFO overflow

Definition at line 648 of file vtss_phy_ts_api.h.

10.32.4.3 vtss_phy_ts_encap_t

```
enum vtss_phy_ts_encap_t
```

Analyzer supported frame encapsulation type.

Analyzer API

Definition at line 738 of file vtss_phy_ts_api.h.

10.32.4.4 vtss_phy_ts_engine_t

```
enum vtss_phy_ts_engine_t
```

Defines Analyzer engine ID.

Note

Timestamp block has 2 PTP engines and 1 OAM engine. OAM engine has two sub-engines (each supports different frame encapsulation) which share OAM comparator to config time stamp functionality. API will expose these 2 sub-engines to the application as 2 independent engines which can have common/shared time stamping functionality (we call it as action). So API will provide 2 PTP and 2 OAM engines to the application to use with following properties/restriction which application must remember while programming an engine. (1) Multi-port timestamp block can share the same engine for both the ports where for each flow in the engine aplication has to mention flow belong to either one of the ports or both the ports; we call it as channel map. (2) Each PTP engine supports 8 flows in ETH, IP and MPLS comparators and 6 actions in PTP/OAM comparator. OAM engines (2A and 2B) have total of 8 flows and 6 actions. Application has to associate flows to OAM engines. But OAM actions can be shared between the 2 OAM engines. (3) There is one HW limitation for flow match mode (strict/non-strict). For same engine ID in ingress and egress direction flow match mode must be same i.e. if engine VTSS_PHY_TS_PTP_ENGINE_ID_0 in ingress is configured as strict flow match then engine VTSS_PHY_TS_PTP_ENGINE_ID_0 in egress has to be in strict flow match. Also OAM engine 2A and 2B can not have different flow match mode i.e engine 2A and 2B for ingress and egress must have same flow match mode. (4) OAM engine does not support TSFIFO and it can not be used for PTP application. But OAM application can use PTP engine. (5) OAM engine 2B only support OAM application with single ethernet encap i.e. OAM-over-ETH

Enumerator

VTSS_PHY_TS_PTP_ENGINE_ID_0	PTP engine 0
VTSS_PHY_TS_PTP_ENGINE_ID_1	PTP engine 1
VTSS_PHY_TS_OAM_ENGINE_ID_2A	OAM engine 2A, no PTP support
VTSS_PHY_TS_OAM_ENGINE_ID_2B	OAM engine 2B, no PTP; only OAM-over-ETH support
VTSS_PHY_TS_ENGINE_ID_INVALID	Invalid Engine ID

Definition at line 791 of file vtss_phy_ts_api.h.

10.32.4.5 vtss_phy_ts_engine_flow_match_t

```
enum vtss_phy_ts_engine_flow_match_t
```

Flow matching within an analyzer engine.

Note

There are two types of flow match possible: (1) Strict flow matching: A valid frame must use the same flow IDs in all comparators in the engine except the PTP and MPLS comparators. (2) A valid frame may match any enabled flow within each comparator. There is one HW restriction mentioned above for flow match mode i.e. ingress and egress for same engine ID must have same flow match. In other words there is no provision to configure strict flow match in ingress, but non-strict flow match for egress. Same restriction for OAM engine 2A and 2B and also for ingress and egress i.e. engine 2A and 2B both ingress and egress must have same flow match mode.

Enumerator

VTSS_PHY_TS_ENG_FLOW_MATCH_ANY	match any flow in comparators
VTSS_PHY_TS_ENG_FLOW_MATCH_STRICT	strict flow match

Definition at line 813 of file vtss_phy_ts_api.h.

10.32.4.6 vtss_phy_ts_ptp_clock_mode_t

`enum vtss_phy_ts_ptp_clock_mode_t`

PTP Timestamp Engine operational modes.

Note

From the operational mode (vtss_phy_ts_ptp_clock_mode_t) and delay measurement method (vtss_phy_ts↩_ptp_delaym_type_t) the API sets up flows in the PTP comparator.

Enumerator

VTSS_PHY_TS_PTP_CLOCK_MODE_BC1STEP	Ordinary/Boundary clock, 1 step
VTSS_PHY_TS_PTP_CLOCK_MODE_BC2STEP	Ordinary/Boundary clock, 2 step
VTSS_PHY_TS_PTP_CLOCK_MODE_TC1STEP	Transparent clock, 1 step
VTSS_PHY_TS_PTP_CLOCK_MODE_TC2STEP	Transparent clock, 2 step
VTSS_PHY_TS_PTP_DELAY_COMP_ENGINE	Delay Compenstaion

Definition at line 1288 of file vtss_phy_ts_api.h.

10.32.4.7 vtss_phy_ts_ptp_delaym_type_t

`enum vtss_phy_ts_ptp_delaym_type_t`

PTP delay measurement method.

Note

As described above, using clock mode and delay measurement method, API sets up flows in PTP comparator.

Enumerator

VTSS_PHY_TS_PTP_DELAYM_P2P	Peer-to-Peer delay measurement method
VTSS_PHY_TS_PTP_DELAYM_E2E	End-to-End delay measurement method

Definition at line 1301 of file vtss_phy_ts_api.h.

10.32.4.8 vtss_phy_ts_y1731_oam_delaym_type_t

```
enum vtss_phy_ts_y1731_oam_delaym_type_t
```

Y.1731 OAM delay messurement method.

Note

Using delay messurement method, API sets up OAM flows in OAM comparator.

Enumerator

VTSS_PHY_TS_Y1731_OAM_DELAYM_1DM	One-Way delay messurement method
VTSS_PHY_TS_Y1731_OAM_DELAYM_DMM	Two-Way delay messurement method

Definition at line 1324 of file vtss_phy_ts_api.h.

10.32.4.9 vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t

```
enum vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t
```

IETF MPLS ACH, OAM delay messurement method.

Note

Using delay messurement method, API sets up OAM flows in OAM comparator.

Enumerator

VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAY↩ M_DMM	Two-way delay measurement method
VTSS_PHY_TS_IETF_MPLS_ACH_OAM_DELAY↩ M_LDM	Loss/Delay Message combined Measurement Message

Definition at line 1334 of file vtss_phy_ts_api.h.

10.32.4.10 vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t

```
enum vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t
```

Analyzer OAM comparator, IETF MPLS ACH OAM Supported Timestamp Format.

Note

PTP Timestamp Format is Supported.

Enumerator

VTSS_PHY_TS_IETF_MPLS_ACH_OAM_TS_FORMAT_PTP	PTP - TimeStamp Format

Definition at line 1361 of file vtss_phy_ts_api.h.

10.32.4.11 vtss_phy_ts_action_format

```
enum vtss_phy_ts_action_format
```

Timestamp format to be configured in action configuration.

Enumerator

VTSS_PHY_TS_4_BYTE	Nano second timestamp
VTSS_PHY_TS_10_BYTE	10 byte timestamp

Definition at line 1392 of file vtss_phy_ts_api.h.

10.32.4.12 vtss_phy_ts_clockfreq_t

```
enum vtss_phy_ts_clockfreq_t
```

Timestamp block clock frequencies.

Enumerator

VTSS_PHY_TS_CLOCK_FREQ_125M	125 MHz
VTSS_PHY_TS_CLOCK_FREQ_15625M	156.25 MHz
VTSS_PHY_TS_CLOCK_FREQ_200M	200 MHz
VTSS_PHY_TS_CLOCK_FREQ_250M	250 MHz
VTSS_PHY_TS_CLOCK_FREQ_500M	500 MHz
VTSS_PHY_TS_CLOCK_FREQ_MAX	MAX Freq

Definition at line 1644 of file vtss_phy_ts_api.h.

10.32.4.13 vtss_phy_ts_clock_src_t

enum vtss_phy_ts_clock_src_t

Clock input source.

Enumerator

VTSS_PHY_TS_CLOCK_SRC_EXTERNAL	External source
VTSS_PHY_TS_CLOCK_SRC_CLIENT_RX	10G: XAUI lane 0 recovered clock, 1G: MAC RX clock (note: direction is opposite to 10G, i.e. PHY->MAC)
VTSS_PHY_TS_CLOCK_SRC_CLIENT_TX	10G: XAUI lane 0 recovered clock, 1G: MAC TX clock (note: direction is opposite to 10G, i.e. MAC->PHY)
VTSS_PHY_TS_CLOCK_SRC_LINE_RX	Received line clock
VTSS_PHY_TS_CLOCK_SRC_LINE_TX	transmitted line clock
VTSS_PHY_TS_CLOCK_SRC_INTERNAL	10G: Invalid, 1G: Internal 250 MHz Clock

Definition at line 1656 of file vtss_phy_ts_api.h.

10.32.4.14 vtss_phy_ts_rxtimestamp_pos_t

enum vtss_phy_ts_rxtimestamp_pos_t

defines Rx Timestamp position inside PTP frame.

Note

There are two options to put Rx timestamp in PTP frame: (a) Rx timestamp in Reserved 4 bytes of PTP header. (b) Shrink Preamble by 4 bytes and append 4 bytes at the end of frame. In this case Ethernet C↩RC will be overwritten by Rx timestamp and a new CRC will be appended after timestamp Also note that Rx Timestamp position must be same for all the ports in the system; otherwsie in ingress timestamp will be put in one position based on that port config whereas egress extract the time from different position as par that port config.

Enumerator

VTSS_PHY_TS_RX_TIMESTAMP_POS_IN_PTP	4 reserved bytes in PTP header
VTSS_PHY_TS_RX_TIMESTAMP_POS_AT_END	4 bytes appended at the end

Definition at line 1680 of file vtss_phy_ts_api.h.

10.32.4.15 vtss_phy_ts_rxtimestamp_len_t

```
enum vtss_phy_ts_rxtimestamp_len_t
```

Defines RX Timestamp format i.e. 30bit or 32bit Rx timestamp.

Note

30bit mode: The value in the reserved field is simply the nanosecCounter i.e. [0..999999999] 32bit mode: The value in the reserved field is a 32 bit value and equals: (nanosecCounter + secCounter∗10^9) mod 2^32

Enumerator

VTSS_PHY_TS_RX_TIMESTAMP_LEN_30BIT	30 bit Rx timestamp
VTSS_PHY_TS_RX_TIMESTAMP_LEN_32BIT	32 bit Rx timestamp

Definition at line 1692 of file vtss_phy_ts_api.h.

10.32.4.16 vtss_phy_ts_fifo_mode_t

```
enum vtss_phy_ts_fifo_mode_t
```

Defines Tx TSFIFO access mode.

Enumerator

VTSS_PHY_TS_FIFO_MODE_NORMAL	in this mode, timestamp can be read from normal CPU interface
VTSS_PHY_TS_FIFO_MODE_SPI	Timestamps are pushed out on the SPI interface

Definition at line 1700 of file vtss_phy_ts_api.h.

10.32.4.17 vtss_phy_ts_fifo_timestamp_len_t

```
enum vtss_phy_ts_fifo_timestamp_len_t
```

Defines 4 bytes vs. the default 10 bytes Timestamp stored in Tx TSFIFO.

Enumerator

VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_4BYTE	4 byte Tx timestamp
VTSS_PHY_TS_FIFO_TIMESTAMP_LEN_10BYTE	10 byte Tx timestamp

Definition at line 1708 of file vtss_phy_ts_api.h.

10.32.4.18 vtss_phy_ts_tc_op_mode_t

`enum vtss_phy_ts_tc_op_mode_t`

defines the Transparent Clock Operating Mode.

Note

There are two Modes TC can work: (a) Mode A: called SUB and ADD Mode where the local time is subtracted from the correction field at ingress and added at egress port. (b) Mode B: also called SUB_ADD Mode which uses reserve bytes (or append at the end of frame by replacing CRC) in PTP header to write RX_timestamp. (c) Mode C: also called 48-bit Mode. This mode uses 48-bits of the CF. This mode is similar to Mode A. At ingress local time is subtracted from CF and local time is added at egress. Also note that TC operating Mode must be same for all the ports in the system.

Enumerator

VTSS_PHY_TS_TC_OP_MODE_A	Sub local time at ingress and add at egress from CF
VTSS_PHY_TS_TC_OP_MODE_B	RX_timestamp using reserved bytes or append at the end as defined in vtss_phy_ts_rxtimestamp_pos_t
VTSS_PHY_TS_TC_OP_MODE_C	Sub local time at ingress and add at egress from CF and use 48 bits in CF

Definition at line 1727 of file vtss_phy_ts_api.h.

10.32.4.19 vtss_phy_ts_nphase_sampler_t

`enum vtss_phy_ts_nphase_sampler_t`

enum for n-phase samplers

Enumerator

VTSS_PHY_TS_NPHASE_PPS_O	N-Phase sampler for PPS_O
VTSS_PHY_TS_NPHASE_PPS_RI	N-Phase sampler for PPS_RI
VTSS_PHY_TS_NPHASE_EGR_SOF	N-Phase sampler for egress SOF
VTSS_PHY_TS_NPHASE_ING_SOF	N-Phase sampler for ingress SOF
VTSS_PHY_TS_NPHASE_LS	N-Phase sampler for Load/Save
VTSS_PHY_TS_NPHASE_MAX	Max N-Phase samplers

Definition at line 1873 of file vtss_phy_ts_api.h.

10.32.4.20 vtss_phy_ts_ptp_message_type_t

```
enum vtss_phy_ts_ptp_message_type_t
```

PTP Event Message TYPES.

Note

4 Types of Event messages.

Definition at line 2229 of file vtss_phy_ts_api.h.

10.32.5 Function Documentation

10.32.5.1 vtss_phy_ts_alt_clock_saved_get()

```
vtss_rc vtss_phy_ts_alt_clock_saved_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            u32 *const saved )
```

Get the latest saved nanosec counter from the alternative clock.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
saved	[OUT] latest saved value.

Returns

Return code.

10.32.5.2 vtss_phy_ts_alt_clock_mode_get()

```
vtss_rc vtss_phy_ts_alt_clock_mode_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_alt_clock_mode_t *const phy_alt_clock_mode )
```

Get the alternative external clock mode.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
phy_alt_clock_mode	[OUT] alternative clock mode.

Returns

Return code.

10.32.5.3 vtss_phy_ts_alt_clock_mode_set()

```
vtss_rc vtss_phy_ts_alt_clock_mode_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_alt_clock_mode_t *const phy_alt_clock_mode )
```

Set the alternative clock mode. This function configures the loopbacks.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
phy_alt_clock_mode	[IN] alternative clock mode.

Returns

Return code.

10.32.5.4 vtss_phy_ts_pps_conf_set()

```
vtss_rc vtss_phy_ts_pps_conf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_pps_conf_t *const phy_pps_conf )
```

Set offset for the PPS generation.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
phy_pps_conf	[IN] pps configuration

Returns

Return code.

10.32.5.5 vtss_phy_ts_pps_conf_get()

```
vtss_rc vtss_phy_ts_pps_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_pps_conf_t *const phy_pps_conf )
```

Get offset for the PPS generation.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
phy_pps_conf	[OUT] pps configuration

Returns

Return code.

10.32.5.6 vtss_phy_ts_ingress_latency_set()

```
vtss_rc vtss_phy_ts_ingress_latency_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_timeinterval_t *const latency )
```

Set the ingress latency.

Note

Latency is taken as TimeInterval which is 48bit nanosec + 16bit sub-nanosec though HW supports nanosec in the range 0 - 2^16.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
latency	[IN] ingress latency

Returns

Return code.

10.32.5.7 vtss_phy_ts_ingress_latency_get()

```
vtss_rc vtss_phy_ts_ingress_latency_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_timeinterval_t *const latency )
```

Get the ingress latency.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
latency	[OUT] ingress latency

Returns

Return code.

10.32.5.8 vtss_phy_ts_egress_latency_set()

```
vtss_rc vtss_phy_ts_egress_latency_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_timeinterval_t *const latency )
```

Set the egress latency.

Note

Latency is taken as TimeInterval which is 48bit nanosec + 16bit sub-nanosec though HW supports nanosec in the range 0 - 2^16.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
latency	[IN] egress latency

Returns

Return code.

10.32.5.9 vtss_phy_ts_egress_latency_get()

```
vtss_rc vtss_phy_ts_egress_latency_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_timeinterval_t *const latency )
```

Get the egress latency.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
latency	[OUT] egress latency

Returns

Return code.

10.32.5.10 vtss_phy_ts_path_delay_set()

```
vtss_rc vtss_phy_ts_path_delay_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_timeinterval_t *const path_delay )
```

Set the path delay.

Note

Path delay is taken as TimeInterval which is 48bit nanosec + 16bit sub-nanosec though HW supports nanosec in the range 0 - 2^32.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
path_delay	[IN] path delay (measured)

Returns

Return code.

10.32.5.11 vtss_phy_ts_path_delay_get()

```
vtss_rc vtss_phy_ts_path_delay_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_timeinterval_t *const path_delay )
```

Get the path delay.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
path_delay	[OUT] path delay

Returns

Return code.

10.32.5.12 vtss_phy_ts_delay_asymmetry_set()

```
vtss_rc vtss_phy_ts_delay_asymmetry_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_timeinterval_t *const delay_asym )
```

Set the delay asymmetry.

Note

Asymmetry is taken as TimeInterval which is 48bit nanosec + 16bit sub-nanosec though HW supports scaled nanosec which is 16 bit nanosec + 16 bit sub-nanosec, i.e. the range is -2^15 - (+2^15-2^-16).

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
delay_asym	[IN] link delay asymmetry

Returns

Return code.

10.32.5.13 vtss_phy_ts_delay_asymmetry_get()

```
vtss_rc vtss_phy_ts_delay_asymmetry_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_timeinterval_t *const delay_asym )
```

Get the delay asymmetry.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
delay_asym	[OUT] link delay asymmetry

Returns

Return code.

10.32.5.14 vtss_phy_ts_ptptime_set()

```
vtss_rc vtss_phy_ts_ptptime_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_timestamp_t *const ts )
```

Set the current PTP time into the PHY.

Note

Time to be set must be next pps time.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ts	[IN] current PTP time

Returns

Return code.

10.32.5.15 vtss_phy_ts_ptptime_set_done()

```
vtss_rc vtss_phy_ts_ptptime_set_done (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no )
```

Setting of the current PTP time into the PHY is completed.

Note

This function should be called after the next pps after setting the next pps time.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number

Returns

Return code.

10.32.5.16 vtss_phy_ts_ptptime_arm()

```
vtss_rc vtss_phy_ts_ptptime_arm (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no )
```

Arm the local time of the PHY so that in next pps it can be read.

Note

Once armed, in next pps it will load the local time and can be read using vtss_phy_ts_ptptime_get. Must call before get the local time of the PHY.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number

Returns

Return code.

10.32.5.17 vtss_phy_ts_ptptime_get()

```
vtss_rc vtss_phy_ts_ptptime_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_timestamp_t *const ts )
```

Get the armed PTP time from the PHY.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ts	[OUT] local time in PHY

Returns

Return code. If the time has not been updated after the vtss_phy_ts_ptptime_arm function is called, it returns error.

10.32.5.18 vtss_phy_ts_load_ptptime_get()

```
vtss_rc vtss_phy_ts_load_ptptime_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_timestamp_t *const ts )
```

Get the PTP time from the PHY load registers.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ts	[OUT] local time in PHY

Returns

Return code. If the time has not been updated after the vtss_phy_ts_ptptime_arm function is called, it returns error.

10.32.5.19 vtss_phy_ts_sertod_set()

```
vtss_rc vtss_phy_ts_sertod_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_sertod_conf_t *const sertod_conf )
```

Set Enable/Disable Serial ToD.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
sertod_conf	[IN] configure Serial ToD input

Returns

Return code.

10.32.5.20 vtss_phy_ts_sertod_get()

```
vtss_rc vtss_phy_ts_sertod_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_sertod_conf_t *const sertod_conf )
```

Get Enable/Disable Serial ToD.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
sertod_conf	[OUT] Serial ToD configuration

Returns

Return code.

10.32.5.21 vtss_phy_ts_loadpulse_delay_set()

```
vtss_rc vtss_phy_ts_loadpulse_delay_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const u16 *const delay )
```

Set load pulse delay.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
delay	[IN] delay value in nano seconds

Returns

Return code.

10.32.5.22 vtss_phy_ts_loadpulse_delay_get()

```
vtss_rc vtss_phy_ts_loadpulse_delay_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            u16 *const delay )
```

Get load pulse delay.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
delay	[OUT] delay value in nano seconds

Returns

Return code.

10.32.5.23 vtss_phy_ts_clock_rateadj_set()

```
vtss_rc vtss_phy_ts_clock_rateadj_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_scaled_ppb_t *const adj )
```

Adjust the local clock rate.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
adj	[IN] Clock ratio frequency offset in units of scaled ppb (parts pr billion). ratio > 0 => clock runs faster.

Returns

Return code.

10.32.5.24 vtss_phy_ts_clock_rateadj_get()

```
vtss_rc vtss_phy_ts_clock_rateadj_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_scaled_ppb_t *const adj )
```

Get the clock rate adjustment value.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
adj	[OUT] Clock ratio frequency offset in scaled ppb. ratio > 0 => clock runs faster

Returns

Return code.

10.32.5.25 vtss_phy_ts_clock_rateadj_ppm_set()

```
vtss_rc vtss_phy_ts_clock_rateadj_ppm_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_scaled_ppb_t *const adj )
```

Adjust ppm of the local clock rate .

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
adj	[IN] Clock ratio frequency offset in units of scaled ppb (parts per billion). ratio > 0 => clock runs faster.

Returns

Return code.

10.32.5.26 vtss_phy_ts_clock_rateadj_ppm_get()

```
vtss_rc vtss_phy_ts_clock_rateadj_ppm_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_scaled_ppb_t *const adj )
```

Get the clock rate ppm adjustment value.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
adj	[OUT] Clock ratio frequency offset in scaled ppb. ratio > 0 => clock runs faster

Returns

Return code.

10.32.5.27 vtss_phy_ts_ptptime_adj1ns()

```
vtss_rc vtss_phy_ts_ptptime_adj1ns (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const BOOL incr )
```

Increment/decrement the LTC clock value by 1 ns.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
incr	[IN] increment/decrement: TRUE = incr, FALSE = decr

Returns

Return code.

10.32.5.28 vtss_phy_ts_timeofday_offset_set()

```
vtss_rc vtss_phy_ts_timeofday_offset_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const i32 offset )
```

Subtract offset from the current time.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
offset	[IN] offset in nano seconds

Returns

Return code.

10.32.5.29 vtss_phy_ts_ongoing_adjustment()

```
vtss_rc vtss_phy_ts_ongoing_adjustment (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_todadj_status_t *const ongoing_adjustment )
```

Return the status of the LTC time adjustment.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
ongoing_adjustment	[OUT] LTC offset operation status

Returns

Return code.

10.32.5.30 vtss_phy_ts_ltc_freq_synth_pulse_set()

```
vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ltc_freq_synth_t *const ltc_freq_synthesis )
```

Return the status of the LTC time adjustment.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
ltc_freq_synthesis	[IN] Frequency synthesis pulse configuration

Returns

Return code.

10.32.5.31 vtss_phy_ts_ltc_freq_synth_pulse_get()

```
vtss_rc vtss_phy_ts_ltc_freq_synth_pulse_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ltc_freq_synth_t *const ltc_freq_synthesis )
```

Return the status of the LTC time adjustment.

Parameters

inst	[IN] handle to an API instance.
port_no	[IN] port number
ltc_freq_synthesis	[OUT] Frequency synthesis pulse configuration

Returns

Return code.

10.32.5.32 vtss_phy_daisy_conf_set()

```
vtss_rc vtss_phy_daisy_conf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_daisy_chain_conf_t *const daisy_chain )
```

configure the daisy chain for TS FIFO

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
daisy_chain	[IN] daisy-chaining configuration

Returns

Return code.

10.32.5.33 vtss_phy_daisy_conf_get()

```
vtss_rc vtss_phy_daisy_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_daisy_chain_conf_t *const daisy_chain )
```

getting the daisy chain for TS FIFO

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
daisy_chain	[IN] daisy-chaining configuration

Returns

Return code.

10.32.5.34 vtss_phy_ts_fifo_sig_set()

```
vtss_rc vtss_phy_ts_fifo_sig_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_fifo_sig_mask_t sig_mask )
```

Set Tx TSFIFO i.e. frame signature in Tx TSFIFO.

Note

Ports sharing timestamp IP block use common register in analyzer to configure the signature i.e. all the ports within the timestamp IP block will have same signature in TSFIFO. In other words, to configure the signature in the FIFO, any of the ports within timestamp block can be used.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
sig_mask	[IN] signature mask

Returns

Return code.

10.32.5.35 vtss_phy_ts_fifo_sig_get()

```
vtss_rc vtss_phy_ts_fifo_sig_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_fifo_sig_mask_t *const sig_mask )
```

Get frame signature mask in Tx TSFIFO.

Note

As described in vtss_phy_ts_fifo_sig_set, any of the ports within IP timestamp block can be used to get the signature.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
sig_mask	[OUT] signature mask

Returns

Return code.

10.32.5.36 vtss_phy_ts_fifo_empty()

```
vtss_rc vtss_phy_ts_fifo_empty (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no )
```

Read timestamp from Tx TSFIFO.

Note

Application will call this function upon receipt of a signal for timestamp in FIFO.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number

Returns

Return code.

10.32.5.37 vtss_phy_ts_fifo_read_install()

```
vtss_rc vtss_phy_ts_fifo_read_install (
            const vtss_inst_t inst,
            vtss_phy_ts_fifo_read rd_cb,
            void * cntxt )
```

Install callback to read data (signature + timestamp) from Tx TSFIFO.

Note

Registered callback will be called for each entry in TSFIFO from vtss_phy_ts_fifo_empty function.

Parameters

inst	[IN] handle to an API instance
rd_cb	[IN] read callback
cntxt	[IN] context to be returned in callback

Returns

Return code.

10.32.5.38 vtss_phy_ts_fifo_read_cb_get()

```
vtss_rc vtss_phy_ts_fifo_read_cb_get (
            const vtss_inst_t inst,
            vtss_phy_ts_fifo_read * rd_cb,
            void ** cntxt )
```

Get the fifo read callback function installed.

Parameters

inst	[IN] handle to an API instance
rd_cb	[OUT] read callback
cntxt	[OUT] context

Returns

Return code.

10.32.5.39 vtss_phy_ts_ingress_engine_init()

```
vtss_rc vtss_phy_ts_ingress_engine_init (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            const vtss_phy_ts_encap_t encap_type,
            const u8 flow_st_index,
            const u8 flow_end_index,
            const vtss_phy_ts_engine_flow_match_t flow_match_mode )
```

Initialize an analyzer ingress engine for an encapsulation type.

Note

Depending on chip type, an analyzer engine may be shared by two ports, and therefore no need to initialize separate engine for each port. If the same encapsulation is used by the two ports, both can use same engine. To initialize an engine shared by both ports, either of the ports no can be passed to this API and use flow_map within the engine to map flows to the ports.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
encap_type	[IN] frame encapsulation
flow_st_index	[IN] flow start index
flow_end_index	[IN] flow end index
flow_match_mode	[IN] flow match mode: strict/non-strict flow match

Returns

Return code.

10.32.5.40 vtss_phy_ts_ingress_engine_init_conf_get()

```
vtss_rc vtss_phy_ts_ingress_engine_init_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            vtss_phy_ts_eng_init_conf_t *const init_conf )
```

Get the configuration parameters passed in engine_init of an analyzer ingress engine for a specific engine ID.

Note

Depending on chip type, an analyzer engine may be shared by two ports, and therefore the configuration is same for both the ports.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
init_conf	[OUT] config parameters in engine_init

Returns

Return code.

10.32.5.41 vtss_phy_ts_ingress_engine_clear()

```
vtss_rc vtss_phy_ts_ingress_engine_clear (
            const vtss_inst_t inst,
```

```
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id )
```

Clear/release an analyzer ingress engine already initialized.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID

Returns

Return code.

10.32.5.42 vtss_phy_ts_egress_engine_init()

```
vtss_rc vtss_phy_ts_egress_engine_init (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            const vtss_phy_ts_encap_t encap_type,
            const u8 flow_st_index,
            const u8 flow_end_index,
            const vtss_phy_ts_engine_flow_match_t flow_match_mode )
```

Initialize an analyzer egress engine for an encapsulation type.

Note

Depending on chip type, an analyzer engine may be shared by two ports, and therefore no need to initialize separate engine for each port. If the same encapsulation is used by the two ports, both can use same engine. To initialize an engine shared by both ports, either of the ports no can be passed to this API and use flow_map within the engine to map flows to the ports.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
encap_type	[IN] frame encapsulation
flow_st_index	[IN] flow start index
flow_end_index	[IN] flow end index
flow_match_mode	[IN] flow match mode: strict/non-strict flow match

Returns

Return code.

10.32.5.43 vtss_phy_ts_egress_engine_init_conf_get()

```
vtss_rc vtss_phy_ts_egress_engine_init_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            vtss_phy_ts_eng_init_conf_t *const init_conf )
```

Get the configuration parameters passed in engine_init of an analyzer egress engine for a specific engine ID.

Note

Depending on chip type, an analyzer engine may be shared by two ports, and therefore the configuration is same for both the ports.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
init_conf	[OUT] config parameters in engine_init

Returns

Return code.

10.32.5.44 vtss_phy_ts_egress_engine_clear()

```
vtss_rc vtss_phy_ts_egress_engine_clear (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id )
```

Clear/release an analyzer egress engine already initialized.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID

Returns

Return code.

10.32.5.45 vtss_phy_ts_ingress_engine_conf_set()

```
vtss_rc vtss_phy_ts_ingress_engine_conf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            const vtss_phy_ts_engine_flow_conf_t *const flow_conf )
```

Configure ingress analyzer flow.

Note

For multi-port timestamp block, either of the port can be used to configure the ingress engine, port to which frame will be matched is specified in flow-map parameter. Note that during the process of updating an engine configuration, the engine mode will be disabled and it will start processing PTP frame after config finished. Also if the local time counter is out of sync, and has to be set, then the received ptp packets must be ignored, and no ptp packets should be transmitted, but this should be controlled by the application.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
flow_conf	[IN] pointer to engine configuration

Returns

Return code.

10.32.5.46 vtss_phy_ts_ingress_engine_conf_get()

```
vtss_rc vtss_phy_ts_ingress_engine_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            vtss_phy_ts_engine_flow_conf_t *const flow_conf )
```

Get ingress analyzer flow.

Note

For multi-port timestamp block, either of the port can used to get the ingress engine configuration.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
flow_conf	[OUT] pointer to engine configuration

Returns

Return code.

10.32.5.47 vtss_phy_ts_egress_engine_conf_set()

```
vtss_rc vtss_phy_ts_egress_engine_conf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            const vtss_phy_ts_engine_flow_conf_t *const flow_conf )
```

Configure egress analyzer flow.

Note

For multi-port timestamp block, either of the port can be used to configure the ingress engine, port to which frame will be matched is specified in flow_map parameter.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
flow_conf	[IN] pointer to engine configuration

Returns

Return code.

10.32.5.48 vtss_phy_ts_egress_engine_conf_get()

```
vtss_rc vtss_phy_ts_egress_engine_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            vtss_phy_ts_engine_flow_conf_t *const flow_conf )
```

Get egress analyzer flow.

Note

For multi-port timestamp block, either of the port can used to get the ingress engine configuration.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
flow_conf	[OUT] pointer to engine configuration

Returns

Return code.

10.32.5.49 vtss_phy_ts_ingress_engine_action_set()

```
vtss_rc vtss_phy_ts_ingress_engine_action_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            const vtss_phy_ts_engine_action_t *const action_conf )
```

Configure ingress analyzer engine action.

Note

For multi-port timestamp block, either of the port can be used to configure the ingress engine. Note that during the process of updating an engine configuration, the engine mode will be disabled and it will start processing frame after config finished.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
action_conf	[IN] action associated with the engine

Returns

Return code.

10.32.5.50 vtss_phy_ts_ingress_engine_action_get()

```
vtss_rc vtss_phy_ts_ingress_engine_action_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
```

```
const vtss_phy_ts_engine_t eng_id,
vtss_phy_ts_engine_action_t *const action_conf )
```

Get ingress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the ingress engine.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
action_conf	[OUT] action associated with the engine

Returns

Return code.

10.32.5.51 vtss_phy_ts_egress_engine_action_set()

```
vtss_rc vtss_phy_ts_egress_engine_action_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_engine_t eng_id,
            const vtss_phy_ts_engine_action_t *const action_conf )
```

Configure egress analyzer engine action.

Note

For multi-port timestamp block, either of the port can be used to configure the ingress engine. Note that during the process of updating an engine configuration, the engine mode will be disabled and it will start processing frame after config finished.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
action_conf	[IN] action associated with the engine

Returns

Return code.

10.32.5.52 vtss_phy_ts_egress_engine_action_get()

```
vtss_rc vtss_phy_ts_egress_engine_action_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
```

```
const vtss_phy_ts_engine_t eng_id,
vtss_phy_ts_engine_action_t *const action_conf )
```

Get egress analyzer engine action. For multi-port timestamp block, either of the port can be used to get the ingress engine.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
eng_id	[IN] engine ID
action_conf	[OUT] action associated with the engine

Returns

Return code.

10.32.5.53 vtss_phy_ts_event_enable_set()

```
vtss_rc vtss_phy_ts_event_enable_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const BOOL enable,
            const vtss_phy_ts_event_t ev_mask )
```

Enabling / Disabling of events.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number
enable	[IN] Enable or disable events
ev_mask	[IN] Event type(s) to control (mask)

Returns

Return code.

10.32.5.54 vtss_phy_ts_event_enable_get()

```
vtss_rc vtss_phy_ts_event_enable_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_event_t *const ev_mask )
```

Get Enabling of events.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number
ev_mask	[OUT] Mask containing events that are enabled

Returns

Return code.

10.32.5.55 vtss_phy_ts_event_poll()

```
vtss_rc vtss_phy_ts_event_poll (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_event_t *const status )
```

Polling function called at by interrupt or periodically.

Note

Interrupt status will be cleared on read

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number
status	[OUT] Event status, bit set indicates corresponding event/interrupt has detected

Returns

Return code.

10.32.5.56 vtss_phy_ts_stats_get()

```
vtss_rc vtss_phy_ts_stats_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_stats_t *const statistics )
```

Get Timestamp statistics.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
statistics	[OUT] pointer to statistics structure

Returns

Return code.

10.32.5.57 vtss_phy_ts_correction_overflow_get()

```
vtss_rc vtss_phy_ts_correction_overflow_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            BOOL *const ingr_overflow,
            BOOL *const egr_overflow )
```

Get the correction field overflow status in ingress and egress.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ingr_overflow	[OUT] ingress overflow status (enable/disable)
egr_overflow	[OUT] egress overflow status (enable/disable)

Returns

Return code.

10.32.5.58 vtss_phy_ts_mode_set()

```
vtss_rc vtss_phy_ts_mode_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const BOOL enable )
```

Enable/disable timestamp block.

Note

Disabling the timestamp block will 'BYPASS' the block.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
enable	[IN] enable/disable parameter

Returns

Return code.

10.32.5.59 vtss_phy_ts_mode_get()

```
vtss_rc vtss_phy_ts_mode_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            BOOL *const enable )
```

Get timestamp block status i.e. enable/disable.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
enable	[OUT] enable/disable parameter

Returns

Return code.

10.32.5.60 vtss_phy_ts_init()

```
vtss_rc vtss_phy_ts_init (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_init_conf_t *const conf )
```

Init timestamp block.

Note

Init has to be called for each port in the time stamp block.

Parameters

inst	[IN] Target instance reference
port_no	[IN] Port number
conf	[IN] Init config parameters

Returns

Return code.

10.32.5.61 vtss_phy_ts_init_conf_get()

```
vtss_rc vtss_phy_ts_init_conf_get (
            const vtss_inst_t inst,
```

```
            const vtss_port_no_t port_no,
            BOOL *const init_done,
            vtss_phy_ts_init_conf_t *const conf )
```

Get the timestamp init config parameters.

Parameters

inst	[IN] Target instance reference
port_no	[IN] Port number
init_done	[OUT] Timestamp init done or not for the port
conf	[OUT] Init config parameters

Returns

Return code.

10.32.5.62 vtss_phy_ts_nphase_status_get()

```
vtss_rc vtss_phy_ts_nphase_status_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_nphase_sampler_t sampler,
            vtss_phy_ts_nphase_status_t *const status )
```

Get N-Phase sampler status.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
sampler	[IN] N-Phase sampler type
status	[OUT] status

Returns

Return code.

10.32.5.63 vtss_phy_ts_hiacc_set()

```
vtss_rc vtss_phy_ts_hiacc_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_nphase_sampler_t sampler,
            const BOOL enable )
```

Enable N-Phase sampler.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
sampler	[IN] N-Phase sampler type
enable	[IN] enable/disable N-Phase sampler

Returns

Return code.

10.32.5.64 vtss_phy_ts_hiacc_get()

```
vtss_rc vtss_phy_ts_hiacc_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_phy_ts_nphase_sampler_t sampler,
            BOOL const * enable )
```

N-Phase sampler status get.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
sampler	[IN] N-Phase sampler type
enable	[OUT] enable/disable N-Phase sampler

Returns

Return code.

10.32.5.65 vtss_phy_ts_block_soft_reset()

```
vtss_rc vtss_phy_ts_block_soft_reset (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_soft_reset_t ts_reset )
```

reset 1588 block.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ts_reset	[IN] 1588 block reset

Returns

Return code.

10.32.5.66 vtss_phy_ts_new_spi_mode_set()

```
vtss_rc vtss_phy_ts_new_spi_mode_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const BOOL enable )
```

Enable/disable New SPI mode for 8574-15 (Rev A & Rev B) that uses PPS0 pin as the new SPI_CLK.

Note

This will activate the New SPI bus while enabled. User must make note of the following: (1) SPI mode to access TS_FIFO has to be mentioned in vtss_phy_ts_init. (2) Both the ports of a block must have same configuration in terms of FIFO access mode i.e. MDIO, Old SPI Mode or New SPI mode. Also Old and New SPI cannot be mixed between blocks of a chip and both the blocks must be in same SPI mode i.e. both must be either Old SPI or New SPI. (3) This must be the last step after all the config done for both the ports of the block. (4) This is required/supported for 8574-15 RevA and RevB, not for 8487/8488-15.

Parameters

inst	[IN] Target instance reference
port_no	[IN] Port number
enable	[IN] enable/disable of New SPI. If tx_fifo_mode is VTSS_PHY_TS_FIFO_MODE_SPI: disable => SPI is in Old SPI Mode enable => SPI in New SPI Mode

Returns

Return code.

10.32.5.67 vtss_phy_ts_new_spi_mode_get()

```
vtss_rc vtss_phy_ts_new_spi_mode_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            BOOL *const mode )
```

Get New SPI mode for 8574-15 (Rev A & Rev B) described above.

Parameters

inst	[IN] Target instance reference
port_no	[IN] Port number
mode	[OUT] Status of new SPI mode disable => SPI is in Old SPI Mode enable => SPI in New SPI Mode

Returns

Return code.

10.32.5.68 vtss_phy_1588_csr_reg_write()

```
vtss_rc vtss_phy_1588_csr_reg_write (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const u32 blk_id,
            const u16 csr_address,
            const u32 *const value )
```

Set the the 1588 block CSR registers.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
blk_id	[IN] 1588 CSR block Index [0-7]
csr_address	[IN] 1588 CSR block Register Offset [0x00 - 0x7f]
value	[IN] 32 bit value

Returns

Return code.

10.32.5.69 vtss_phy_1588_csr_reg_read()

```
vtss_rc vtss_phy_1588_csr_reg_read (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const u32 blk_id,
            const u16 csr_address,
            u32 *const value )
```

get the the 1588 block CSR registers.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
blk_id	[IN] 1588 CSR block Index [0-7]
csr_address	[IN] 1588 CSR block Register Offset [0x00 - 0x7f]
value	[OUT] 32 bit value

Returns

Return code.

10.32.5.70 vtss_phy_ts_status_check()

```
vtss_rc vtss_phy_ts_status_check (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            BOOL wait,
            const vtss_debug_printf_t pr )
```

TS status check function supported for 10G Phys like 8488 & 8492.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number
wait	[IN] wait if needed
pr	[IN] print function to be passed for default logging

Returns

Return code.

10.32.5.71 vtss_phy_ts_10g_extended_fifo_sync()

```
vtss_rc vtss_phy_ts_10g_extended_fifo_sync (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_10g_fifo_sync_t * conf )
```

Reset 1588 PHY Fifo oos. As of now, supported for 10G Phys like 8488.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number
conf	[IN] Select modes for running the API.

Returns

Return code.

10.32.5.72 vtss_phy_ts_viper_fifo_reset()

```
vtss_rc vtss_phy_ts_viper_fifo_reset (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_phy_ts_fifo_conf_t * fifo_conf )
```

Viper 1588 FIFO reset.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
fifo_conf	[IN] FIFO algorithm Operation mode.

Returns

Return Code.

10.32.5.73 vtss_phy_ts_tesla_tsp_fifo_sync()

```
vtss_rc vtss_phy_ts_tesla_tsp_fifo_sync (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_debug_printf_t pr,
            const vtss_phy_ts_fifo_conf_t * fifo_conf,
            BOOL * OOS )
```

API to detect and correct Timestamp FIFO OOS.

Note

Compile time flag TESLA_ING_TS_ERRFIX is needed for this API to work else this API will have no effect

Parameters

inst	[IN] Target instance reference
port_no	[IN] Port number
pr	[IN] print function needed for default logging
fifo_conf	[IN] fifo config struct
OOS	[OUT] True/False of Out-of-Sync for both Ingress and Egress combined

Returns

Return code.

10.32.5.74 vtss_phy_1g_ts_fifo_sync()

```
vtss_rc vtss_phy_1g_ts_fifo_sync (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_debug_printf_t pr,
            const vtss_phy_ts_fifo_conf_t * fifo_conf,
            BOOL * OOS )
```

API to detect and correct Timestamp FIFO OOS for 1G PHY's (Viper and Tesla)

Note

: In Viper OOS API there is no detection mechanism.

Parameters

inst	[IN] Target instance reference
port_no	[IN] Port number
pr	[IN] print function needed for default logging
fifo_conf	[IN] fifo config struct
OOS	[OUT] True/False of Out-of-Sync for both Ingress and Egress combined(Only for Tesla)

Returns

Return code.

10.32.5.75 vtss_phy_1588_debug_reg_read()

```
vtss_rc vtss_phy_1588_debug_reg_read (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const u32 blk_id,
            const vtss_debug_printf_t p_routine )
```

API to dump PHY timestamp registers (for Debugging)

Parameters

inst	[IN] Target instance reference
port_no	[IN] Port number
blk_id	[IN] Register block id
p_routine	[IN] print function needed for default logging

Returns

Return code.

10.32.5.76 vtss_phy_ts_flow_clear_cf_set()

```
vtss_rc vtss_phy_ts_flow_clear_cf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            BOOL ingress,
            const vtss_phy_ts_engine_t eng_id,
            u8 act_id,
            vtss_phy_ts_ptp_message_type_t msgtype )
```

Clear Correction field for specified PTP message type.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ingress	[IN] TRUE if Ingress elses Egress
eng_id	[IN] 1588 Engine ID
act_id	[IN] 1588 Action ID
msgtype	[IN] PTP Message Type

Returns

Return code.

10.33 vtss_api/include/vtss_port_api.h File Reference

Port API.

```
#include <vtss/api/port.h>
```

Data Structures

- struct vtss_port_map_t
  Port map structure.
- struct vtss_port_clause_37_adv_t
  Advertisement control data for Clause 37 aneg.
- struct vtss_port_sgmii_aneg_t
  Advertisement control data for SGMII aneg.
- struct vtss_port_clause_37_control_t
  Auto-negotiation control parameter struct.
- struct vtss_port_flow_control_conf_t
  Flow control setup.
- struct vtss_port_frame_gaps_t
  Inter frame gap structure.
- struct vtss_port_serdes_conf_t
  SFI Serdes configuration.
- struct vtss_port_conf_t
  Port configuration structure.
- struct vtss_basic_counters_t
  Basic counters structure.
- struct vtss_port_ifh_t
  Port Internal Frame Header structure.

Macros

- #define CHIP_PORT_UNUSED -1
- #define VTSS_FRAME_GAP_DEFAULT 0
- #define VTSS_MAX_FRAME_LENGTH_STANDARD 1518
- #define VTSS_MAX_FRAME_LENGTH_MAX 10240
- #define VTSS_MAX_FRAME_LENGTH_MAX 10240

Enumerations

- enum vtss_miim_controller_t { VTSS_MIIM_CONTROLLER_0 = 0, VTSS_MIIM_CONTROLLER_1 = 1, VTSS_MIIM_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 configured with the same configuration. This is needed in order to achieve correct comma alignment at the QSGMII interface. Which ports that are mapped together can be found in the chip data-sheet.
- vtss_rc vtss_port_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_conf_t *const conf)

  Get port setup.

- vtss_rc vtss_port_status_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_status_t *const status)

  Get port status.
- vtss_rc vtss_port_counters_update (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Update counters for port.
- vtss_rc vtss_port_counters_clear (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Clear counters for port.
- vtss_rc vtss_port_counters_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_counters_t *const counters)

  Get counters for port.
- vtss_rc vtss_port_basic_counters_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_basic_counters_t *const counters)

  Get basic counters for port.
- vtss_rc vtss_port_forward_state_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_forward_t *const forward)

  Get port forwarding state.
- vtss_rc vtss_port_forward_state_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_port_forward_t forward)

  Set port forwarding state.
- vtss_rc vtss_port_ifh_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_port_ifh_t *const conf)

  Set port Internal Frame Header settings.
- vtss_rc vtss_port_ifh_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_port_ifh_t *const conf)

  Get port Internal Frame Header settings.
- vtss_rc vtss_miim_read (const vtss_inst_t inst, const vtss_chip_no_t chip_no, const vtss_miim_controller_t miim_controller, const u8 miim_addr, const u8 addr, u16 *const value)

  Direct MIIM read (bypassing port map)
- vtss_rc vtss_miim_write (const vtss_inst_t inst, const vtss_chip_no_t chip_no, const vtss_miim_controller_t miim_controller, const u8 miim_addr, const u8 addr, const u16 value)

  Direct MIIM write (bypassing port map)

10.33.1 Detailed Description

Port API.

10.33.2 Macro Definition Documentation

10.33.2.1 CHIP_PORT_UNUSED

```
#define CHIP_PORT_UNUSED -1
```

Signifies an unused chip port

Definition at line 69 of file vtss_port_api.h.

10.33.2.2 VTSS_FRAME_GAP_DEFAULT

```
#define VTSS_FRAME_GAP_DEFAULT 0
```

Default frame gap used

Definition at line 203 of file vtss_port_api.h.

10.33.2.3 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.

10.33.2.4 VTSS_MAX_FRAME_LENGTH_MAX [1/2]

```
#define VTSS_MAX_FRAME_LENGTH_MAX 10240
```

Maximum frame length supported

Definition at line 224 of file vtss_port_api.h.

10.33.2.5 VTSS_MAX_FRAME_LENGTH_MAX [2/2]

```
#define VTSS_MAX_FRAME_LENGTH_MAX 10240
```

Maximum frame length supported

Definition at line 224 of file vtss_port_api.h.

10.33.3 Enumeration Type Documentation

10.33.3.1 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.

10.33.3.2 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.

10.33.3.3 vtss_port_clause_37_remote_fault_t

`enum vtss_port_clause_37_remote_fault_t`

Auto-negotiation remote fault type.

Advertisement Word (Refer to IEEE 802.3 Clause 37): MSB LSB D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 +—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+ | NP | Ack| RF2| R↩ F1|rsvd|rsvd|rsvd| PS2| PS1| HD | FD |rsvd|rsvd|rsvd|rsvd|rsvd| +—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+—+

Enumerator

VTSS_PORT_CLAUSE_37_RF_LINK_OK	Link OK
VTSS_PORT_CLAUSE_37_RF_OFFLINE	Off line
VTSS_PORT_CLAUSE_37_RF_LINK_FAILURE	Link failure
VTSS_PORT_CLAUSE_37_RF_AUTONEG_ERROR	Autoneg error

Definition at line 118 of file vtss_port_api.h.

10.33.3.4 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.

10.33.3.5 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.

10.33.3.6 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.

10.33.4 Function Documentation

10.33.4.1 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

inst	[IN] Target instance reference.
port_map	[IN] Port map array.

Returns

Return code.

10.33.4.2 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

inst	[IN] Target instance reference.
port_map	[OUT] Port map.

Returns

Return code.

10.33.4.3 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
control	[OUT] Control structure.

Returns

Return code.

10.33.4.4 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
control	[IN] Control structure.

Returns

Return code.

10.33.4.5 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 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] Port setup structure.

Returns

Return code.

10.33.4.6 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] Port configuration.

Returns

Return code.

10.33.4.7 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
status	[OUT] Status structure.

Returns

Return code.

10.33.4.8 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.

Returns

Return code.

10.33.4.9 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

inst	[IN] Target instance reference.
port_no	[IN] Port/aggregation number.

Returns

Return code.

10.33.4.10 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

inst	[IN] Target instance reference.
port_no	[IN] Port/aggregation number.
counters	[OUT] Counter structure.

Returns

Return code.

10.33.4.11 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

inst	[IN] Target instance reference.
port_no	[IN] Port/aggregation number.
counters	[OUT] Counter structure.

Returns

Return code.

10.33.4.12 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
forward	[OUT] Forwarding state.

Returns

Return code.

10.33.4.13 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
forward	[IN] Forwarding state.

Returns

Return code.

10.33.4.14 vtss_port_ifh_conf_set()

```
vtss_rc vtss_port_ifh_conf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_port_ifh_t *const conf )
```

Set port Internal Frame Header settings.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] Port IFH structure.

Returns

Return code.

10.33.4.15 vtss_port_ifh_conf_get()

```
vtss_rc vtss_port_ifh_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_port_ifh_t *const conf )
```

Get port Internal Frame Header settings.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] Port IFH configuration.

Returns

Return code.

10.33.4.16 vtss_miim_read()

```
vtss_rc vtss_miim_read (
            const vtss_inst_t inst,
            const vtss_chip_no_t chip_no,
            const vtss_miim_controller_t miim_controller,
            const u8 miim_addr,
            const u8 addr,
            u16 *const value )
```

Direct MIIM read (bypassing port map)

Parameters

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
miim_controller	[IN] MIIM Controller Instance
miim_addr	[IN] MIIM Device Address
addr	[IN] MIIM Register Address
value	[OUT] Register value read

Returns

Return code.

10.33.4.17 vtss_miim_write()

```
vtss_rc vtss_miim_write (
            const vtss_inst_t inst,
            const vtss_chip_no_t chip_no,
            const vtss_miim_controller_t miim_controller,
            const u8 miim_addr,
            const u8 addr,
            const u16 value )
```

Direct MIIM write (bypassing port map)

Parameters

inst	[IN] Target instance reference.
chip_no	[IN] Chip number (if multi-chip instance).
miim_controller	[IN] MIIM Controller Instance
miim_addr	[IN] MIIM Device Address
addr	[IN] MIIM Register Address
value	[IN] Register value to write

Returns

Return code.

10.34 vtss_api/include/vtss_qos_api.h File Reference

QoS API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct vtss_red_v2_t

  Random Early Detection configuration struct version 2 (per queue, per dpl - switch global)
- struct vtss_qos_conf_t

  All parameters below are defined per chip.
- struct vtss_policer_t

  Policer.
- struct vtss_policer_ext_t

  Policer Extensions.
- struct vtss_dlb_policer_conf_t

  Dual leaky buckets policer configuration.
- struct vtss_shaper_t

  Shaper.
- struct vtss_qos_port_conf_t

  QoS setup per port.
- struct vtss_qce_mac_t

QCE MAC information.

- struct vtss_qce_tag_t

  QCE tag information.
- struct vtss_qce_frame_etype_t

  Frame data for VTSS_QCE_TYPE_ETYPE.
- struct vtss_qce_frame_llc_t

  Frame data for VTSS_QCE_TYPE_LLC.
- struct vtss_qce_frame_snap_t

  Frame data for VTSS_QCE_TYPE_SNAP.
- struct vtss_qce_frame_ipv4_t

  Frame data for VTSS_QCE_TYPE_IPV4.
- struct vtss_qce_frame_ipv6_t

  Frame data for VTSS_QCE_TYPE_IPV6.
- struct vtss_qce_key_t

  QCE key.
- struct vtss_qce_action_t

  QCE action.
- struct vtss_qce_t

  QoS Control Entry.

Macros

- #define VTSS_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_wred_v2_max_t { VTSS_WRED_V2_MAX_DP, VTSS_WRED_V2_MAX_FL }

  Random Early Detection version 2. Select if max means max drop probability or max fill level.
- enum vtss_shaper_mode_t { VTSS_SHAPER_MODE_LINE, VTSS_SHAPER_MODE_DATA }

  Shaper Accounting Mode.
- enum vtss_tag_remark_mode_t { VTSS_TAG_REMARK_MODE_CLASSIFIED = 0, VTSS_TAG_REMARK_MODE_DEFAULT = 2, VTSS_TAG_REMARK_MODE_MAPPED = 3 }

  Tag Remark Mode.
- enum vtss_dscp_mode_t { VTSS_DSCP_MODE_NONE, VTSS_DSCP_MODE_ZERO, VTSS_DSCP_MODE_SEL, VTSS_DSCP_MODE_ALL }

  DSCP mode for ingress port.
- enum vtss_dscp_emode_t { VTSS_DSCP_EMODE_DISABLE, VTSS_DSCP_EMODE_REMARK, VTSS_DSCP_EMODE_REMAP, VTSS_DSCP_EMODE_REMAP_DPA }

  DSCP mode for egress port.
- enum vtss_qce_type_t { VTSS_QCE_TYPE_ANY, VTSS_QCE_TYPE_ETYPE, VTSS_QCE_TYPE_LLC, VTSS_QCE_TYPE_SNAP, VTSS_QCE_TYPE_IPV4, VTSS_QCE_TYPE_IPV6 }

  QoS Control Entry type.

Functions

- vtss_rc vtss_qos_conf_get (const vtss_inst_t inst, vtss_qos_conf_t *const conf)

  Get QoS setup for switch.
- vtss_rc vtss_qos_conf_set (const vtss_inst_t inst, const vtss_qos_conf_t *const conf)

  Set QoS setup for switch.
- vtss_rc vtss_qos_port_conf_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_qos_port_conf_t *const conf)

  Get QoS setup for port.
- vtss_rc vtss_qos_port_conf_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_qos_port_conf_t *const conf)

  Set QoS setup for port.
- vtss_rc vtss_qce_init (const vtss_inst_t inst, const vtss_qce_type_t type, vtss_qce_t *const qce)

  Initialize QCE to default values.
- vtss_rc vtss_qce_add (const vtss_inst_t inst, const vtss_qcl_id_t qcl_id, const vtss_qce_id_t qce_id, const vtss_qce_t *const qce)

  Add QCE to QCL.
- vtss_rc vtss_qce_del (const vtss_inst_t inst, const vtss_qcl_id_t qcl_id, const vtss_qce_id_t qce_id)

  Delete QCE from QCL.
- vtss_rc vtss_qos_shaper_calibrate (const vtss_inst_t inst)

  Calibrate egress shaper rates.

10.34.1 Detailed Description

QoS API.

This header file describes Quality of Service functions

10.34.2 Macro Definition Documentation

10.34.2.1 VTSS_PORT_POLICERS

```
#define VTSS_PORT_POLICERS 1
```

Number of port policers

Definition at line 179 of file vtss_qos_api.h.

10.34.2.2 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.

10.34.2.3 VTSS_QCL_IDS

```
#define VTSS_QCL_IDS 1
```

Number of QCLs

Definition at line 452 of file vtss_qos_api.h.

10.34.2.4 VTSS_QCL_ID_START

```
#define VTSS_QCL_ID_START 0
```

QCL ID start number

Definition at line 453 of file vtss_qos_api.h.

10.34.2.5 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.

10.34.2.6 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.

10.34.2.7 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.

10.34.3 Enumeration Type Documentation

10.34.3.1 vtss_wred_v2_max_t

```
enum vtss_wred_v2_max_t
```

Random Early Detection version 2. Select if max means max drop probability or max fill level.

Enumerator

VTSS_WRED_V2_MAX_DP	Unit for max is drop probability
VTSS_WRED_V2_MAX_FL	Unit for max is fill level

Definition at line 65 of file vtss_qos_api.h.

10.34.3.2 vtss_shaper_mode_t

`enum vtss_shaper_mode_t`

Shaper Accounting Mode.

Enumerator

VTSS_SHAPER_MODE_LINE	Use line-rate for the shaper
VTSS_SHAPER_MODE_DATA	Use data-rate for the shaper

Definition at line 288 of file vtss_qos_api.h.

10.34.3.3 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.

10.34.3.4 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.

10.34.3.5 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.

10.34.3.6 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.

10.34.4 Function Documentation

10.34.4.1 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

inst	[IN] Target instance reference.
conf	[OUT] QoS setup structure.

Returns

Return code.

10.34.4.2 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

inst	[IN] Target instance reference.
conf	[IN] QoS setup structure.

Returns

Return code.

10.34.4.3 vtss_qos_port_conf_get()

```
vtss_rc vtss_qos_port_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_qos_port_conf_t *const conf )
```

Get QoS setup for port.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] QoS setup structure.

Returns

Return code.

10.34.4.4 vtss_qos_port_conf_set()

```
vtss_rc vtss_qos_port_conf_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_qos_port_conf_t *const conf )
```

Set QoS setup for port.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] QoS setup structure.

Returns

Return code.

10.34.4.5 vtss_qce_init()

```
vtss_rc vtss_qce_init (
            const vtss_inst_t inst,
            const vtss_qce_type_t type,
            vtss_qce_t *const qce )
```

Initialize QCE to default values.

Parameters

inst	[IN] Target instance reference.
type	[IN] QCE type.
qce	[OUT] QCE structure.

Returns

Return code.

10.34.4.6 vtss_qce_add()

```
vtss_rc vtss_qce_add (
            const vtss_inst_t inst,
            const vtss_qcl_id_t qcl_id,
            const vtss_qce_id_t qce_id,
            const vtss_qce_t *const qce )
```

Add QCE to QCL.

Parameters

inst	[IN] Target instance reference.
qcl_id	[IN] QCL ID.
qce↩ _id	[IN] QCE ID. The QCE will be added before the entry with this ID. VTSS_QCE_ID_LAST is reserved for inserting last.
qce	[IN] QCE structure.

Returns

Return code.

10.34.4.7 vtss_qce_del()

```
vtss_rc vtss_qce_del (
            const vtss_inst_t inst,
            const vtss_qcl_id_t qcl_id,
            const vtss_qce_id_t qce_id )
```

Delete QCE from QCL.

Parameters

inst	[IN] Target instance reference.
qcl_id	[IN] QCL ID.
qce↩ _id	[IN] QCE ID.

Returns

Return code.

10.34.4.8 vtss_qos_shaper_calibrate()

```
vtss_rc vtss_qos_shaper_calibrate (
            const vtss_inst_t inst )
```

Calibrate egress shaper rates.

This function calibrates the egress port and queue shapers on VTSS_ARCH_SERVAL.

Calling this function periodically will enhance the accuracy of the egress port and queue shapers.

Recommended rate is between 1 to 50 times per second.

Parameters

inst	[IN] Target instance reference.

Returns

Return code.

10.35 vtss_api/include/vtss_rab_api.h File Reference

RAB API.

```
#include <vtss/api/types.h>
```

10.35.1 Detailed Description

RAB API.

10.36 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_LLC.
- struct vtss_ace_frame_snap_t

  Frame data for VTSS_ACE_TYPE_SNAP.
- struct vtss_ace_frame_arp_t

  Frame data for VTSS_ACE_TYPE_ARP.
- struct vtss_ace_frame_ipv4_t

  Frame data for VTSS_ACE_TYPE_IPV4.
- struct vtss_ace_frame_ipv6_t

  Frame data for VTSS_ACE_TYPE_IPV6.
- struct vtss_ace_t

  Access Control Entry.

Macros

- #define VTSS_ACE_ID_LAST 0

Typedefs

- typedef u32 vtss_acl_port_counter_t

  ACL port counter.
- typedef u32 vtss_ace_id_t

  ACE ID type.
- typedef vtss_vcap_u8_t vtss_ace_u8_t

  ACE 8 bit value and mask.
- typedef vtss_vcap_u16_t vtss_ace_u16_t

  ACE 16 bit value and mask.
- typedef vtss_vcap_u32_t vtss_ace_u32_t

  ACE 32 bit value and mask.
- typedef vtss_vcap_u40_t vtss_ace_u40_t

  ACE 40 bit value and mask.
- typedef vtss_vcap_u48_t vtss_ace_u48_t

  ACE 48 bit value and mask.
- typedef vtss_vcap_u128_t vtss_ace_u128_t

ACE 128 bit value and mask.

- typedef vtss_vcap_vid_t vtss_ace_vid_t

  ACE VLAN ID value and mask.
- typedef vtss_vcap_ip_t vtss_ace_ip_t

  ACE IP address value and mask.
- typedef vtss_vcap_udp_tcp_t vtss_ace_udp_tcp_t

  ACE UDP/TCP port range.
- typedef u32 vtss_ace_counter_t

  ACE hit counter.

Enumerations

- enum vtss_auth_state_t { VTSS_AUTH_STATE_NONE, VTSS_AUTH_STATE_EGRESS, VTSS_AUTH_STATE_BOTH }

  Authentication state.
- enum vtss_acl_port_action_t { VTSS_ACL_PORT_ACTION_NONE, VTSS_ACL_PORT_ACTION_FILTER, VTSS_ACL_PORT_ACTION_REDIR }

  ACL port action.
- enum vtss_acl_ptp_action_t { VTSS_ACL_PTP_ACTION_NONE, VTSS_ACL_PTP_ACTION_ONE_STEP, VTSS_ACL_PTP_ACTION_ONE_STEP_ADD_D VTSS_ACL_PTP_ACTION_ONE_STEP_SUB_DELAY_1, VTSS_ACL_PTP_ACTION_ONE_STEP_SUB_DELAY_2, VTSS_ACL_PTP_ACTION_TWO_STEP }

  ACL PTP action.
- enum vtss_ace_type_t { VTSS_ACE_TYPE_ANY, VTSS_ACE_TYPE_ETYPE, 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.

10.36.1 Detailed Description

Security API.

This header file describes security functions

10.36.2 Macro Definition Documentation

10.36.2.1 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.

10.36.3 Enumeration Type Documentation

10.36.3.1 vtss_auth_state_t

```
enum vtss_auth_state_t
```

Authentication state.

Enumerator

VTSS_AUTH_STATE_NONE	Not authenticated
VTSS_AUTH_STATE_EGRESS	Authenticated in egress direction
VTSS_AUTH_STATE_BOTH	Authenticated in both directions

Definition at line 59 of file vtss_security_api.h.

10.36.3.2 vtss_acl_port_action_t

```
enum vtss_acl_port_action_t
```

ACL port action.

Enumerator

VTSS_ACL_PORT_ACTION_NONE	No action from port list
VTSS_ACL_PORT_ACTION_FILTER	Port list filter is used
VTSS_ACL_PORT_ACTION_REDIR	Port list redirect is used

Definition at line 194 of file vtss_security_api.h.

10.36.3.3 vtss_acl_ptp_action_t

```
enum vtss_acl_ptp_action_t
```

ACL PTP action.

Enumerator

VTSS_ACL_PTP_ACTION_NONE	No PTP action
VTSS_ACL_PTP_ACTION_ONE_STEP	PTP one-step time-stamping
VTSS_ACL_PTP_ACTION_ONE_STEP_ADD_DE↩LAY	PTP one-step time-stamping, Serval: add delay, Jr2: Add EDLY
VTSS_ACL_PTP_ACTION_ONE_STEP_SUB_DE↩LAY_1	PTP one-step time-stamping, Serval: subtract delay 1, Jr2: Add IDLY1
VTSS_ACL_PTP_ACTION_ONE_STEP_SUB_DE↩LAY_2	PTP one-step time-stamping, Serval: subtract delay 2, Jr2: Add IDLY2
VTSS_ACL_PTP_ACTION_TWO_STEP	PTP two-step time-stamping

Definition at line 202 of file vtss_security_api.h.

10.36.3.4 vtss_ace_type_t

```
enum vtss_ace_type_t
```

ACE frame type.

Enumerator

VTSS_ACE_TYPE_ANY	Any frame type
VTSS_ACE_TYPE_ETYPE	Ethernet Type
VTSS_ACE_TYPE_LLC	LLC
VTSS_ACE_TYPE_SNAP	SNAP
VTSS_ACE_TYPE_ARP	ARP/RARP
VTSS_ACE_TYPE_IPV4	IPv4
VTSS_ACE_TYPE_IPV6	IPv6

Definition at line 338 of file vtss_security_api.h.

10.36.3.5 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.

10.36.4 Function Documentation

10.36.4.1 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
state	[OUT] Authentication state.

Returns

Return code.

10.36.4.2 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
state	[IN] Authentication state.

Returns

Return code.

10.36.4.3 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

inst	[IN] Target instance reference.
policer_no	[IN] ACL policer number.
conf	[OUT] ACL policer configuration.

Returns

Return code.

10.36.4.4 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

inst	[IN] Target instance reference.
policer_no	[IN] ACL policer number.
conf	[IN] ACL policer configuration.

Returns

Return code.

10.36.4.5 vtss_acl_port_conf_get()

```
vtss_rc vtss_acl_port_conf_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_acl_port_conf_t *const conf )
```

Get ACL configuration for port.

Parameters

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[OUT] Port configuration.

Returns

Return code.

10.36.4.6 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
conf	[IN] Port configuration.

Returns

Return code.

10.36.4.7 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.
counter	[OUT] Default action counter for port.

Returns

Return code.

10.36.4.8 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

inst	[IN] Target instance reference.
port_no	[IN] Port number.

Returns

Return code.

10.36.4.9 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

inst	[IN] Target instance reference.
type	[IN] ACE type.
ace	[OUT] ACE structure.

Returns

Return code.

10.36.4.10 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

inst	[IN] Target instance reference.
ace_id_next	[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.
ace	[IN] ACE structure.

Returns

Return code.

10.36.4.11 vtss_ace_del()

```
vtss_rc vtss_ace_del (
            const vtss_inst_t inst,
            const vtss_ace_id_t ace_id )
```

Delete ACE.

Parameters

inst	[IN] Target instance reference.
ace↩ _id	[IN] ACE ID.

Returns

Return code.

10.36.4.12 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

inst	[IN] Target instance reference.
ace_id	[IN] ACE ID.
counter	[OUT] ACE counter.

Returns

Return code.

10.36.4.13 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

inst	[IN] Target instance reference.
ace↩ _id	[IN] ACE ID.

Returns

Return code.

10.37 vtss_api/include/vtss_sfi4_api.h File Reference

SFI4 API.

```
#include <vtss/api/types.h>
```

10.37.1 Detailed Description

SFI4 API.

10.38 vtss_api/include/vtss_sync_api.h File Reference

Synchronization API.

```
#include "vtss/api/types.h"
```

Data Structures

- struct vtss_synce_clock_out_t

  Struct containing configuration for a recovered clock output port.
- struct vtss_synce_clock_in_t

  Struct containing configuration selecting the recovered input clock port, to be delivered to a selected output clock port.

Macros

- #define VTSS_SYNCE_CLK_A 0
- #define VTSS_SYNCE_CLK_B 1
- #define VTSS_SYNCE_CLK_MAX 2

Typedefs

- typedef u32 vtss_synce_clk_port_t

  Identification of a output clock port.

Enumerations

- enum vtss_synce_divider_t { VTSS_SYNCE_DIVIDER_1, VTSS_SYNCE_DIVIDER_4, VTSS_SYNCE_DIVIDER_5 }

  Identification of a Clock dividing value used when selected input clock goes to output.

Functions

- vtss_rc vtss_synce_clock_out_set (const vtss_inst_t inst, const vtss_synce_clk_port_t clk_port, const vtss_synce_clock_out_t *const conf)

  Set the configuration of a selected output clock port - against external clock controller.
- vtss_rc vtss_synce_clock_out_get (const vtss_inst_t inst, const vtss_synce_clk_port_t clk_port, vtss_synce_clock_out_t *const conf)

  Get the configuration of a selected output clock port - against external clock controller.
- vtss_rc vtss_synce_clock_in_set (const vtss_inst_t inst, const vtss_synce_clk_port_t clk_port, const vtss_synce_clock_in_t *const conf)

  Set the configuration of input port for a selected output clock port.
- vtss_rc vtss_synce_clock_in_get (const vtss_inst_t inst, const vtss_synce_clk_port_t clk_port, vtss_synce_clock_in_t *const conf)

  Get the configuration of input port for a selected output clock port.

10.38.1 Detailed Description

Synchronization API.

This header file describes synchronization functions

10.38.2 Macro Definition Documentation

10.38.2.1 VTSS_SYNCE_CLK_A

```
#define VTSS_SYNCE_CLK_A 0
```

Clock A output port

Definition at line 56 of file vtss_sync_api.h.

10.38.2.2 VTSS_SYNCE_CLK_B

```
#define VTSS_SYNCE_CLK_B 1
```

Clock B output port

Definition at line 57 of file vtss_sync_api.h.

10.38.2.3 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.

10.38.3 Enumeration Type Documentation

10.38.3.1 vtss_synce_divider_t

```
enum vtss_synce_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.

10.38.4 Function Documentation

10.38.4.1 vtss_synce_clock_out_set()

```
vtss_rc vtss_synce_clock_out_set (
            const vtss_inst_t inst,
```

```
            const vtss_synce_clk_port_t clk_port,
            const vtss_synce_clock_out_t *const conf )
```

Set the configuration of a selected output clock port - against external clock controller.

Parameters

inst	[IN] handle to an API instance.
clk_port	[IN] Selection of the output clock port (VTSS_SYNCE_CLK_A or VTSS_SYNCE_CLK_B)
conf	[IN] pointer to a vtss_synce_clock_out_t configuration structure.

Returns

Return code.

10.38.4.2 vtss_synce_clock_out_get()

```
vtss_rc vtss_synce_clock_out_get (
            const vtss_inst_t inst,
            const vtss_synce_clk_port_t clk_port,
            vtss_synce_clock_out_t *const conf )
```

Get the configuration of a selected output clock port - against external clock controller.

Parameters

inst	[IN] handle to an API instance.
clk_port	[IN] Selection of the output clock port (VTSS_SYNCE_CLK_A or VTSS_SYNCE_CLK_B)
conf	[IN] pointer to a vtss_synce_clock_out_t configuration structure.

Returns

Return code.

10.38.4.3 vtss_synce_clock_in_set()

```
vtss_rc vtss_synce_clock_in_set (
            const vtss_inst_t inst,
            const vtss_synce_clk_port_t clk_port,
            const vtss_synce_clock_in_t *const conf )
```

Set the configuration of input port for a selected output clock port.

Parameters

inst	[IN] handle to an API instance.
clk_port	[IN] Selection of the output clock port (VTSS_SYNCE_CLK_A or VTSS_SYNCE_CLK_B)
conf	[IN] pointer to a vtss_synce_clock_in_t configuration structure.

Returns

Return code.

10.38.4.4 vtss_synce_clock_in_get()

```
vtss_rc vtss_synce_clock_in_get (
            const vtss_inst_t inst,
            const vtss_synce_clk_port_t clk_port,
            vtss_synce_clock_in_t *const conf )
```

Get the configuration of input port for a selected output clock port.

Parameters

inst	[IN] handle to an API instance.
clk_port	[IN] Selection of the output clock port (VTSS_SYNCE_CLK_A or VTSS_SYNCE_CLK_B)
conf	[IN] pointer to a vtss_synce_clock_in_t configuration structure.

Returns

Return code.

10.39 vtss_api/include/vtss_tfi5_api.h File Reference

TFI5 API.

```
#include <vtss/api/types.h>
```

10.39.1 Detailed Description

TFI5 API.

10.40 vtss_api/include/vtss_ts_api.h File Reference

TimeStamping API.

```
#include <vtss/api/types.h>
```

Data Structures

- struct vtss_ts_alt_clock_mode_t

  parameter for setting the alternative clock mode.
- struct vtss_ts_ext_clock_mode_t

  external clock output configuration.
- struct vtss_ts_operation_mode_t

  Timestamp operation.
- struct vtss_ts_internal_mode_t

  Hardware timestamping format mode for internal ports.
- struct vtss_ts_id_t

  Timestamp identifier.
- struct vtss_ts_timestamp_t

  Timestamp structure.
- struct vtss_oam_ts_id_s

  parameter for requesting an oam timestamp
- struct vtss_oam_ts_timestamp_s

  parameter for returning an oam timestamp
- struct vtss_ts_timestamp_alloc_t

  Timestamp allocation.

Macros

- #define VTSS_HW_TIME_CNT_PR_SEC 1000000000 /∗ Serval counts ns instead of clock cycles ∗/

  Number of clock cycle counts pr sec.
- #define VTSS_HW_TIME_NSEC_PR_CNT 1

  Number of nanoseconds pr clock count.
- #define VTSS_HW_TIME_WRAP_LIMIT 0 /∗ time counter wrap around limit+1 (=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 10 /∗ 1 ppb ∗/

  Serval minimum adjustment rate in units of 0,1 ppb.
- #define VTSS_HW_TIME_MAX_FINE_ADJ 25

  This is the max time offset adjustment that os done without setting ports in disabled state.

Typedefs

- typedef struct vtss_ts_alt_clock_mode_t vtss_ts_alt_clock_mode_t

  parameter for setting the alternative clock mode.
- typedef struct vtss_ts_ext_clock_mode_t vtss_ts_ext_clock_mode_t

  external clock output configuration.
- typedef struct vtss_ts_operation_mode_t vtss_ts_operation_mode_t

  Timestamp operation.
- typedef struct vtss_ts_internal_mode_t vtss_ts_internal_mode_t

  Hardware timestamping format mode for internal ports.
- typedef struct vtss_ts_id_t vtss_ts_id_t

  Timestamp identifier.
- typedef struct vtss_ts_timestamp_t vtss_ts_timestamp_t

  Timestamp structure.
- typedef struct vtss_oam_ts_id_s vtss_oam_ts_id_t

  parameter for requesting an oam timestamp
- typedef struct vtss_oam_ts_timestamp_s vtss_oam_ts_timestamp_t

  parameter for returning an oam timestamp
- typedef struct vtss_ts_timestamp_alloc_t vtss_ts_timestamp_alloc_t

  Timestamp allocation.

Enumerations

- enum vtss_ts_ext_clock_one_pps_mode_t { **TS_EXT_CLOCK_MODE_ONE_PPS_DISABLE**, **TS_EXT_CLOCK_MODE_ONE_PPS_OUTPUT**, **TS_EXT_CLOCK_MODE_ONE_PPS_INPUT**, **TS_EXT_CLOCK_MODE_ONE_PPS_OUTPUT_INPUT**, **TS_EXT_CLOCK_MODE_MAX** }

  parameter for setting the external clock mode.
- enum vtss_ts_mode_t { **TS_MODE_NONE**, **TS_MODE_EXTERNAL**, **TS_MODE_INTERNAL**, **TX_MODE_MAX** }

  parameter for setting the timestamp operating mode
- enum vtss_ts_internal_fmt_t { **TS_INTERNAL_FMT_NONE**, **TS_INTERNAL_FMT_RESERVED_LEN_30BIT**, **TS_INTERNAL_FMT_RESERVED_LEN_32BIT**, **TS_INTERNAL_FMT_SUB_ADD_LEN_44BIT_CF62**, **TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_3_0**, **TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF_0**, **TS_INTERNAL_FMT_RESERVED_LEN_48BIT_CF**, **TX_INTERNAL_FMT_MAX** }

  parameter for setting the internal timestamp format

Functions

- vtss_rc vtss_ts_timeofday_set (const vtss_inst_t inst, const vtss_timestamp_t *const ts)

  Set the current time in a Timestamp format.
- vtss_rc vtss_ts_timeofday_set_delta (const vtss_inst_t inst, const vtss_timestamp_t *ts, BOOL negative)

  Set delta the current time in a Timestamp format.
- vtss_rc vtss_ts_timeofday_offset_set (const vtss_inst_t inst, const i32 offset)

  Subtract offset from the current time.
- vtss_rc vtss_ts_adjtimer_one_sec (const vtss_inst_t inst, BOOL *const ongoing_adjustment)

  Do the one sec administration in the Timestamp function.
- vtss_rc vtss_ts_ongoing_adjustment (const vtss_inst_t inst, BOOL *const ongoing_adjustment)

  Check if the clock adjustment is ongoing.
- vtss_rc vtss_ts_timeofday_get (const vtss_inst_t inst, vtss_timestamp_t *const ts, u32 *const tc)

  Get the current time in a Timestamp format, and the corresponding time counter.
- vtss_rc vtss_ts_timeofday_next_pps_get (const vtss_inst_t inst, vtss_timestamp_t *const ts)

  Get the time at the next 1PPS pulse edge in a Timestamp format.
- vtss_rc vtss_ts_adjtimer_set (const vtss_inst_t inst, const i32 adj)

  Adjust the clock timer ratio.
- vtss_rc vtss_ts_adjtimer_get (const vtss_inst_t inst, i32 *const adj)

  get the clock timer ratio.
- vtss_rc vtss_ts_freq_offset_get (const vtss_inst_t inst, i32 *const adj)

  get the clock internal timer frequency offset, compared to external clock input.
- vtss_rc vtss_ts_alt_clock_saved_get (const vtss_inst_t inst, u32 *const saved)

  Get the latest saved nanosec counter from the alternative clock.
- vtss_rc vtss_ts_alt_clock_mode_get (const vtss_inst_t inst, vtss_ts_alt_clock_mode_t *const alt_clock_mode)

  Get the alternative external clock mode.
- vtss_rc vtss_ts_alt_clock_mode_set (const vtss_inst_t inst, const vtss_ts_alt_clock_mode_t *const alt_clock_mode)

  Set the alternative external clock mode. This function configures the 1PPS, L/S pin usage for pin set no 0 in Serval.
- vtss_rc vtss_ts_timeofday_next_pps_set (const vtss_inst_t inst, const vtss_timestamp_t *const ts)

  Set the time at the next 1PPS pulse edge in a Timestamp format.
- vtss_rc vtss_ts_external_clock_mode_get (const vtss_inst_t inst, vtss_ts_ext_clock_mode_t *const ext_clock_mode)

Get the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.

- vtss_rc vtss_ts_external_clock_mode_set (const vtss_inst_t inst, const vtss_ts_ext_clock_mode_t *const ext_clock_mode)

  Set the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.
- vtss_rc vtss_ts_external_clock_saved_get (const vtss_inst_t inst, u32 *const saved)

  Get the latest saved time counter in nanosec.
- vtss_rc vtss_ts_ingress_latency_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const ingress_latency)

  Set the ingress latency.
- vtss_rc vtss_ts_ingress_latency_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const ingress_latency)

  Get the ingress latency.
- vtss_rc vtss_ts_p2p_delay_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const p2p_delay)

  Set the P2P delay.
- vtss_rc vtss_ts_p2p_delay_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const p2p_delay)

  Get the P2P delay.
- vtss_rc vtss_ts_egress_latency_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const egress_latency)

  Set the egress latency.
- vtss_rc vtss_ts_egress_latency_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const egress_latency)

  Get the egress latency.
- vtss_rc vtss_ts_delay_asymmetry_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_timeinterval_t *const delay_asymmetry)

  Set the delay asymmetry.
- vtss_rc vtss_ts_delay_asymmetry_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_timeinterval_t *const delay_asymmetry)

  Get the delay asymmetry.
- vtss_rc vtss_ts_operation_mode_set (const vtss_inst_t inst, const vtss_port_no_t port_no, const vtss_ts_operation_mode_t *const mode)

  Set the timestamping operation mode for a port.
- vtss_rc vtss_ts_operation_mode_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_ts_operation_mode_t *const mode)

  Get the timestamping operation mode for a port.
- vtss_rc vtss_ts_internal_mode_set (const vtss_inst_t inst, const vtss_ts_internal_mode_t *const mode)

  Set the internal timestamping mode.
- vtss_rc vtss_ts_internal_mode_get (const vtss_inst_t inst, vtss_ts_internal_mode_t *const mode)

  Get the internal timestamping mode.
- vtss_rc vtss_tx_timestamp_update (const vtss_inst_t inst)

  Update the internal timestamp table, from HW.
- vtss_rc vtss_rx_timestamp_get (const vtss_inst_t inst, const vtss_ts_id_t *const ts_id, vtss_ts_timestamp_t *const ts)

  Get the rx FIFO timestamp for a {timestampId}.
- vtss_rc vtss_rx_timestamp_id_release (const vtss_inst_t inst, const vtss_ts_id_t *const ts_id)

  Release the FIFO rx timestamp id.
- vtss_rc vtss_rx_master_timestamp_get (const vtss_inst_t inst, const vtss_port_no_t port_no, vtss_ts_timestamp_t *const ts)

  Get rx timestamp from a port (convert from slave time to the master time)

- vtss_rc vtss_oam_timestamp_get (const vtss_inst_t inst, const vtss_oam_ts_id_t *const id, vtss_oam_ts_timestamp_t *const ts)

  Get oam timestamp.
- vtss_rc vtss_tx_timestamp_idx_alloc (const vtss_inst_t inst, const vtss_ts_timestamp_alloc_t *const alloc↩_parm, vtss_ts_id_t *const ts_id)

  Allocate a timestamp id for a two step transmission.
- vtss_rc vtss_timestamp_age (const vtss_inst_t inst)

  Age the FIFO timestamps.
- vtss_rc vtss_ts_status_change (const vtss_inst_t inst, const vtss_port_no_t port_no)

  Signal port status change (used to detect and compensate for the internal ingress and egress latencies)

10.40.1 Detailed Description

TimeStamping API.

This header file describes PTP/OAM TimeStamping API functions and associated types.

10.40.2 Typedef Documentation

10.40.2.1 vtss_ts_alt_clock_mode_t

```
typedef struct vtss_ts_alt_clock_mode_t vtss_ts_alt_clock_mode_t
```

parameter for setting the alternative clock mode.

external clock output configuration.

10.40.3 Function Documentation

10.40.3.1 vtss_ts_timeofday_set()

```
vtss_rc vtss_ts_timeofday_set (
            const vtss_inst_t inst,
            const vtss_timestamp_t *const ts )
```

Set the current time in a Timestamp format.

Parameters

inst	[IN] handle to an API instance.
ts	[IN] pointer to a TimeStamp structure.

Returns

Return code.

10.40.3.2 vtss_ts_timeofday_set_delta()

```
vtss_rc vtss_ts_timeofday_set_delta (
            const vtss_inst_t inst,
            const vtss_timestamp_t * ts,
            BOOL negative )
```

Set delta the current time in a Timestamp format.

Parameters

inst	[IN] handle to an API instance.
ts	[IN] pointer to a TimeStamp structure.
negative	[IN] True if ts is subtracted from current time, else ts is added.

Returns

Return code.

10.40.3.3 vtss_ts_timeofday_offset_set()

```
vtss_rc vtss_ts_timeofday_offset_set (
            const vtss_inst_t inst,
            const i32 offset )
```

Subtract offset from the current time.

Parameters

inst	[IN] handle to an API instance.
offset	[IN] offset in ns.

Returns

Return code.

10.40.3.4 vtss_ts_adjtimer_one_sec()

```
vtss_rc vtss_ts_adjtimer_one_sec (
            const vtss_inst_t inst,
            BOOL *const ongoing_adjustment )
```

Do the one sec administration in the Timestamp function.

Parameters

inst	[IN] handle to an API instance.
ongoing_adjustment	[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

10.40.3.5 vtss_ts_ongoing_adjustment()

```
vtss_rc vtss_ts_ongoing_adjustment (
            const vtss_inst_t inst,
            BOOL *const ongoing_adjustment )
```

Check if the clock adjustment is ongoing.

Parameters

inst	[IN] handle to an API instance.
ongoing_adjustment	[OUT] True if clock adjustment is ongoing

Returns

Return code.

10.40.3.6 vtss_ts_timeofday_get()

```
vtss_rc vtss_ts_timeofday_get (
            const vtss_inst_t inst,
            vtss_timestamp_t *const ts,
            u32 *const tc )
```

Get the current time in a Timestamp format, and the corresponding time counter.

Parameters

inst	[IN] handle to an API instance
ts	[OUT] pointer to a TimeStamp structure
tc	[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.

10.40.3.7 vtss_ts_timeofday_next_pps_get()

```
vtss_rc vtss_ts_timeofday_next_pps_get (
            const vtss_inst_t inst,
            vtss_timestamp_t *const ts )
```

Get the time at the next 1PPS pulse edge in a Timestamp format.

Parameters

inst	[IN] handle to an API instance
ts	[OUT] pointer to a TimeStamp structure

Returns

Return code.

10.40.3.8 vtss_ts_adjtimer_set()

```
vtss_rc vtss_ts_adjtimer_set (
            const vtss_inst_t inst,
            const i32 adj )
```

Adjust the clock timer ratio.

Parameters

inst	[IN] handle to an API instance.
adj	[IN] Clock ratio frequency offset in units of 0,1 ppb (parts pr billion). ratio > 0 => clock runs faster

Returns

Return code.

10.40.3.9 vtss_ts_adjtimer_get()

```
vtss_rc vtss_ts_adjtimer_get (
            const vtss_inst_t inst,
            i32 *const adj )
```

get the clock timer ratio.

Parameters

inst	[IN] handle to an API instance.
adj	[OUT] Clock ratio frequency offset in ppb (parts pr billion). ratio > 0 => clock runs faster

Returns

Return code.

10.40.3.10 vtss_ts_freq_offset_get()

```
vtss_rc vtss_ts_freq_offset_get (
            const vtss_inst_t inst,
            i32 *const adj )
```

get the clock internal timer frequency offset, compared to external clock input.

Parameters

inst	[IN] handle to an API instance.
adj	[OUT] Clock ratio frequency offset in ppb (parts pr billion). ratio > 0 => internal clock runs faster than external clock

Returns

Return code.

10.40.3.11 vtss_ts_alt_clock_saved_get()

```
vtss_rc vtss_ts_alt_clock_saved_get (
            const vtss_inst_t inst,
            u32 *const saved )
```

Get the latest saved nanosec counter from the alternative clock.

Parameters

inst	[IN] handle to an API instance
saved	[OUT] latest saved value.

Returns

Return code.

10.40.3.12 vtss_ts_alt_clock_mode_get()

```
vtss_rc vtss_ts_alt_clock_mode_get (
            const vtss_inst_t inst,
            vtss_ts_alt_clock_mode_t *const alt_clock_mode )
```

Get the alternative external clock mode.

Parameters

inst	[IN] handle to an API instance
alt_clock_mode	[OUT] alternative clock mode.

Returns

Return code.

10.40.3.13 vtss_ts_alt_clock_mode_set()

```
vtss_rc vtss_ts_alt_clock_mode_set (
            const vtss_inst_t inst,
            const vtss_ts_alt_clock_mode_t *const alt_clock_mode )
```

Set the alternative external clock mode. This function configures the 1PPS, L/S pin usage for pin set no 0 in Serval.

Parameters

inst	[IN] handle to an API instance
alt_clock_mode	[IN] alternative clock mode.

Returns

Return code.

10.40.3.14 vtss_ts_timeofday_next_pps_set()

```
vtss_rc vtss_ts_timeofday_next_pps_set (
            const vtss_inst_t inst,
            const vtss_timestamp_t *const ts )
```

Set the time at the next 1PPS pulse edge in a Timestamp format.

Parameters

inst	[IN] handle to an API instance
ts	[OUT] pointer to a TimeStamp structure

Returns

Return code.

10.40.3.15 vtss_ts_external_clock_mode_get()

```
vtss_rc vtss_ts_external_clock_mode_get (
            const vtss_inst_t inst,
            vtss_ts_ext_clock_mode_t *const ext_clock_mode )
```

Get the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.

Parameters

inst	[IN] handle to an API instance
ext_clock_mode	[OUT] external clock mode.

Returns

Return code.

10.40.3.16 vtss_ts_external_clock_mode_set()

```
vtss_rc vtss_ts_external_clock_mode_set (
            const vtss_inst_t inst,
            const vtss_ts_ext_clock_mode_t *const ext_clock_mode )
```

Set the external clock mode. The mode depends on the hardware capability, it may be: Enable/disable external synch pulse Set clock output frequency.

Parameters

inst	[IN] handle to an API instance
ext_clock_mode	[IN] external clock mode.

Returns

Return code.

10.40.3.17 vtss_ts_external_clock_saved_get()

```
vtss_rc vtss_ts_external_clock_saved_get (
            const vtss_inst_t inst,
            u32 *const saved )
```

Get the latest saved time counter in nanosec.

Parameters

inst	[IN] handle to an API instance
saved	[OUT] latest saved value. [0..999.999.999]

Returns

Return code.

10.40.3.18 vtss_ts_ingress_latency_set()

```
vtss_rc vtss_ts_ingress_latency_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_timeinterval_t *const ingress_latency )
```

Set the ingress latency.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ingress_latency	[IN] pointer to ingress latency

Returns

Return code.

10.40.3.19 vtss_ts_ingress_latency_get()

```
vtss_rc vtss_ts_ingress_latency_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_timeinterval_t *const ingress_latency )
```

Get the ingress latency.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ingress_latency	[OUT] pointer to ingress_latency

Returns

Return code.

10.40.3.20 vtss_ts_p2p_delay_set()

```
vtss_rc vtss_ts_p2p_delay_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_timeinterval_t *const p2p_delay )
```

Set the P2P delay.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
p2p_delay	[IN] peer-2-peer delay (measured)

Returns

Return code.

10.40.3.21 vtss_ts_p2p_delay_get()

```
vtss_rc vtss_ts_p2p_delay_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_timeinterval_t *const p2p_delay )
```

Get the P2P delay.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
p2p_delay	[OUT] pointer to peer-2-peer delay

Returns

Return code.

10.40.3.22 vtss_ts_egress_latency_set()

```
vtss_rc vtss_ts_egress_latency_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_timeinterval_t *const egress_latency )
```

Set the egress latency.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
egress_latency	[IN] egress latency

Returns

Return code.

10.40.3.23 vtss_ts_egress_latency_get()

```
vtss_rc vtss_ts_egress_latency_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_timeinterval_t *const egress_latency )
```

Get the egress latency.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
egress_latency	[OUT] pointer to egress latency

Returns

Return code.

10.40.3.24 vtss_ts_delay_asymmetry_set()

```
vtss_rc vtss_ts_delay_asymmetry_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_timeinterval_t *const delay_asymmetry )
```

Set the delay asymmetry.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
delay_asymmetry	[IN] delay asymmetry

Returns

Return code.

10.40.3.25 vtss_ts_delay_asymmetry_get()

```
vtss_rc vtss_ts_delay_asymmetry_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_timeinterval_t *const delay_asymmetry )
```

Get the delay asymmetry.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
delay_asymmetry	[OUT] pointer to delay asymmetry

Returns

Return code.

10.40.3.26 vtss_ts_operation_mode_set()

```
vtss_rc vtss_ts_operation_mode_set (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            const vtss_ts_operation_mode_t *const mode )
```

Set the timestamping operation mode for a port.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
mode	[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

10.40.3.27 vtss_ts_operation_mode_get()

```
vtss_rc vtss_ts_operation_mode_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_ts_operation_mode_t *const mode )
```

Get the timestamping operation mode for a port.

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
mode	[OUT] pointer to a struct holding the operation mode

Returns

Return code.

10.40.3.28 vtss_ts_internal_mode_set()

```
vtss_rc vtss_ts_internal_mode_set (
            const vtss_inst_t inst,
            const vtss_ts_internal_mode_t *const mode )
```

Set the internal timestamping mode.

Parameters

inst	[IN] handle to an API instance
mode	[IN] pointer to a struct holding the operation mode

Returns

Return code.

Serval: Used to set INTERNAL mode timestamping format Other : Not used

10.40.3.29 vtss_ts_internal_mode_get()

```
vtss_rc vtss_ts_internal_mode_get (
            const vtss_inst_t inst,
            vtss_ts_internal_mode_t *const mode )
```

Get the internal timestamping mode.

Parameters

inst	[IN] handle to an API instance
mode	[OUT] pointer to a struct holding the operation mode

Returns

Return code.

10.40.3.30 vtss_tx_timestamp_update()

```
vtss_rc vtss_tx_timestamp_update (
            const vtss_inst_t inst )
```

Update the internal timestamp table, from HW.

Parameters

inst	[IN] handle to an API instance

Returns

Return code.

10.40.3.31 vtss_rx_timestamp_get()

```
vtss_rc vtss_rx_timestamp_get (
            const vtss_inst_t inst,
            const vtss_ts_id_t *const ts_id,
            vtss_ts_timestamp_t *const ts )
```

Get the rx FIFO timestamp for a {timestampId}.

Parameters

inst	[IN] handle to an API instance
ts_id	[IN] timestamp id
ts	[OUT] pointer to a struct holding the fifo timestamp

Returns

Return code.

10.40.3.32 vtss_rx_timestamp_id_release()

```
vtss_rc vtss_rx_timestamp_id_release (
            const vtss_inst_t inst,
            const vtss_ts_id_t *const ts_id )
```

Release the FIFO rx timestamp id.

Parameters

inst	[IN] handle to an API instance
ts↩ _id	[IN] timestamp id

Returns

Return code.

10.40.3.33 vtss_rx_master_timestamp_get()

```
vtss_rc vtss_rx_master_timestamp_get (
            const vtss_inst_t inst,
            const vtss_port_no_t port_no,
            vtss_ts_timestamp_t *const ts )
```

Get rx timestamp from a port (convert from slave time to the master time)

Parameters

inst	[IN] handle to an API instance
port_no	[IN] port number
ts	[IN/OUT] pointer to a struct holding the timestamp

Returns

Return code.

10.40.3.34 vtss_oam_timestamp_get()

```
vtss_rc vtss_oam_timestamp_get (
            const vtss_inst_t inst,
            const vtss_oam_ts_id_t *const id,
            vtss_oam_ts_timestamp_t *const ts )
```

Get oam timestamp.

Parameters

inst	[IN] handle to an API instance
id	[IN] identifies the requested timestamp id
ts	[OUT] pointer to a struct holding the timestamp

Returns

Return code.

10.40.3.35 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

inst	[IN] handle to an API instance
alloc_parm	[IN] pointer allocation parameters
ts_id	[OUT] timestamp id

Returns

Return code.

10.40.3.36 vtss_timestamp_age()

```
vtss_rc vtss_timestamp_age (
            const vtss_inst_t inst )
```

Age the FIFO timestamps.

Parameters

inst	[IN] handle to an API instance

Returns

Return code.

10.40.3.37 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

inst	[IN] handle to an API instance
port_no	[IN] port number

Returns

Return code.

10.41 vtss_api/include/vtss_upi_api.h File Reference

Define UPI API interface.

```
#include <vtss/api/types.h>
```

10.41.1 Detailed Description

Define UPI API interface.

10.42 vtss_api/include/vtss_wis_api.h File Reference

eWIS layer API

```
#include <vtss/api/types.h>
```

10.42.1 Detailed Description

eWIS layer API

10.43 vtss_api/include/vtss_xaui_api.h File Reference

XAUI API.

```
#include <vtss/api/types.h>
```

10.43.1 Detailed Description

XAUI API.

10.44 vtss_api/include/vtss_xfi_api.h File Reference

XFI API.

```
#include <vtss/api/types.h>
```

10.44.1 Detailed Description

XFI API.

Index

ach_opt
- vtss_phy_ts_oam_engine_flow_conf_t, 368
- vtss_phy_ts_ptp_engine_flow_conf_t, 375

acknowledge
- vtss_port_clause_37_adv_t, 390

acl_hit
- vtss_packet_rx_info_t, 268

acl_idx
- vtss_packet_rx_info_t, 268

act_len
- tag_vtss_fdma_list, 42

action
- vtss_ace_t, 64
- vtss_acl_port_conf_t, 73
- vtss_ece_t, 106
- vtss_mce_t, 173
- vtss_phy_ts_engine_action_t, 334
- vtss_qce_t, 435
- vtss_vce_t, 499

action_gen
- vtss_phy_ts_engine_action_t, 333

action_ptp
- vtss_phy_ts_engine_action_t, 333

active
- vtss_irq_status_t, 153

actual_fps
- vtss_afi_frm_dscr_t, 74

addr
- vtss_ip_addr_t, 146
- vtss_ipv6_t, 150
- vtss_mac_t, 159
- vtss_phy_ts_ip_conf_t, 358
- vtss_wol_mac_addr_t, 510

addr_match_mode
- vtss_phy_ts_eth_conf_t, 338

addr_match_select
- vtss_phy_ts_eth_conf_t, 339

address
- vtss_ip_network_t, 146
- vtss_ipv4_network_t, 147
- vtss_ipv6_network_t, 149

adv_dis
- port_custom_conf_t, 39

advertisement
- vtss_port_clause_37_control_t, 391

afi_buf_cnt
- vtss_fdma_cfg_t, 128

afi_done_cb
- vtss_fdma_cfg_t, 129

afi_fps
- vtss_fdma_tx_info_t, 136

afi_id
- vtss_packet_tx_info_t, 294

afi_type
- vtss_fdma_tx_info_t, 136

aged
- vtss_mac_table_entry_t, 161
- vtss_mac_table_status_t, 163

aggr_code
- vtss_packet_tx_info_t, 288

aggr_rx_disable
- vtss_packet_frame_info_t, 257
- vtss_packet_port_info_t, 259

aggr_tx_disable
- vtss_packet_frame_info_t, 257
- vtss_packet_port_info_t, 259

alloc_ptr
- tag_vtss_fdma_list, 43

aneg
- vtss_phy_conf_t, 302
- vtss_port_status_t, 419

aneg_complete
- vtss_port_sgmii_aneg_t, 417
- vtss_port_status_t, 419

aneg_enable
- vtss_phy_tbi_conf_t, 326

aneg_pd_detect
- vtss_phy_media_serd_pcs_cntl_t, 316

aneg_restart
- vtss_phy_mac_serd_pcs_cntl_t, 314

arp
- vtss_ace_frame_arp_t, 45
- vtss_ace_t, 66

arrived_tagged
- vtss_packet_rx_header_t, 263

asymmetric_pause
- vtss_phy_aneg_t, 298
- vtss_port_clause_37_adv_t, 390

auto_clear_ls
- vtss_phy_ts_init_conf_t, 355

automatic
- vtss_learn_mode_t, 158

autoneg
- port_custom_conf_t, 38

BOOL
- types.h, 578

base_port_no
- vtss_phy_type_t, 382

bit_count
vtss_sgpio_conf_t, 457
bit_rate
vtss_acl_policer_conf_t, 72
bit_rate_enable
vtss_acl_policer_conf_t, 72
bmode
vtss_sgpio_conf_t, 457
bottom_up
vtss_phy_ts_mpls_conf_t, 362
bpdu_cpu_only
vtss_packet_rx_reg_t, 283
bpdu_queue
vtss_packet_rx_queue_map_t, 280
bpdu_reg
vtss_packet_rx_port_conf_t, 278
bridge
vtss_port_counters_t, 397
bypass_in_api
vtss_phy_10g_fifo_sync_t, 295
byte_limit_per_tick
vtss_fdma_throttle_cfg_t, 134
bytes
vtss_counter_pair_t, 81

CALIB_DONE
vtss_phy_ts_nphase_status_t, 365
CALIB_ERR
vtss_phy_ts_nphase_status_t, 364
CHIP_PORT_UNUSED
vtss_port_api.h, 963
cal_done
vtss_lcpll_status_t, 157
cal_error
vtss_lcpll_status_t, 157
vtss_rcpll_status_t, 448
cal_not_done
vtss_rcpll_status_t, 449
cb
vtss_ts_timestamp_alloc_t, 473
cbs
vtss_dlb_policer_conf_t, 87
ccm
vtss_oam_voe_conf_t, 227
vtss_oam_voe_counter_t, 230
vtss_oam_vop_pdu_type_conf_t, 252
ccm_check_only
vtss_oam_proc_conf_t, 209
ccm_lm
vtss_oam_lm_counter_t, 202
vtss_oam_voe_conf_t, 228
vtss_oam_vop_pdu_type_conf_t, 252
ccm_lm_enable_rx_fcf_in_reserved_field
vtss_oam_vop_conf_t, 248
ccm_lm_sample_lost
vtss_oam_lm_counter_t, 203
ccm_lm_seen
vtss_oam_pdu_seen_status_t, 208
ccm_lm_valid
vtss_oam_lm_counter_t, 203
ccm_seen
vtss_oam_pdu_seen_status_t, 208
cf
vtss_dlb_policer_conf_t, 86
cf_update
vtss_phy_ts_ptp_engine_action_t, 373
cfg
vtss_phy_conf_1g_t, 300
cfi
vtss_ace_vlan_t, 68
vtss_tci_t, 464
channel_id
vtss_phy_type_t, 382
channel_map
vtss_phy_ts_engine_flow_conf_t, 335
vtss_phy_ts_generic_action_t, 349
vtss_phy_ts_oam_engine_action_t, 366
vtss_phy_ts_ptp_engine_action_t, 372
channel_type
vtss_phy_ts_ach_conf_t, 329
check_dmac
vtss_oam_vop_generic_opcode_conf_t, 251
chip_no
vtss_debug_info_t, 83
vtss_debug_lock_t, 84
vtss_fdma_ch_cfg_t, 132
vtss_packet_rx_meta_t, 274
vtss_port_map_t, 406
chip_port
vtss_port_map_t, 406
chk_ing_modified
vtss_phy_ts_init_conf_t, 355
cir
vtss_dlb_policer_conf_t, 86
clear
vtss_debug_info_t, 83
clk_freq
vtss_phy_ts_init_conf_t, 353
clk_mode
vtss_phy_ts_ptp_engine_action_t, 373
clk_src
vtss_phy_ts_init_conf_t, 353
comm_opt
vtss_phy_ts_ach_conf_t, 329
vtss_phy_ts_eth_conf_t, 338
vtss_phy_ts_gen_conf_t, 347
vtss_phy_ts_ip_conf_t, 357
vtss_phy_ts_mpls_conf_t, 360
common_multicast_dmac
vtss_oam_vop_conf_t, 248
connector_enable
vtss_phy_loopback_t, 310
context
vtss_ts_timestamp_alloc_t, 472
vtss_ts_timestamp_t, 474
copper
vtss_port_status_t, 420

copy_1dm_to_cpu
vtss_oam_voe_dm_conf_t, 232
copy_dmm_to_cpu
vtss_oam_voe_dm_conf_t, 232
copy_dmr_to_cpu
vtss_oam_voe_dm_conf_t, 232
copy_lbm_to_cpu
vtss_oam_voe_lb_conf_t, 236
copy_lbr_to_cpu
vtss_oam_voe_lb_conf_t, 236
copy_lmm_to_cpu
vtss_oam_voe_se_lm_conf_t, 241
copy_lmr_to_cpu
vtss_oam_voe_se_lm_conf_t, 241
copy_ltm_to_cpu
vtss_oam_voe_lt_conf_t, 239
copy_ltr_to_cpu
vtss_oam_voe_lt_conf_t, 239
copy_next_only
vtss_oam_proc_conf_t, 209
copy_on_ccm_err
vtss_oam_proc_conf_t, 209
copy_on_ccm_more_than_one_tlv
vtss_oam_proc_conf_t, 210
copy_on_dmac_err
vtss_oam_proc_conf_t, 210
copy_on_mel_too_low_err
vtss_oam_proc_conf_t, 210
copy_to_cpu
vtss_mac_table_entry_t, 161
vtss_oam_voe_ccm_conf_t, 220
vtss_oam_voe_ccm_lm_conf_t, 223
vtss_oam_voe_generic_conf_t, 234
vtss_oam_voe_tst_conf_t, 243
vtss_oam_voe_unknown_conf_t, 245
cos
vtss_packet_rx_info_t, 268
vtss_packet_tx_info_t, 289
count
vtss_oam_voe_ccm_lm_conf_t, 223
vtss_oam_voe_se_lm_conf_t, 242
count_as_data
vtss_oam_voe_ccm_conf_t, 220
vtss_oam_voe_dm_conf_t, 233
vtss_oam_voe_generic_conf_t, 235
vtss_oam_voe_lb_conf_t, 237
vtss_oam_voe_lt_conf_t, 240
vtss_oam_voe_se_lm_conf_t, 242
vtss_oam_voe_tst_conf_t, 244
vtss_oam_voe_unknown_conf_t, 246
count_as_selected
vtss_oam_voe_ccm_conf_t, 220
vtss_oam_voe_ccm_lm_conf_t, 223
vtss_oam_voe_dm_conf_t, 233
vtss_oam_voe_generic_conf_t, 234
vtss_oam_voe_lb_conf_t, 236
vtss_oam_voe_lt_conf_t, 240
vtss_oam_voe_se_lm_conf_t, 242
vtss_oam_voe_tst_conf_t, 244
vtss_oam_voe_unknown_conf_t, 246
cpu
vtss_acl_action_t, 69
vtss_learn_mode_t, 158
cpu_once
vtss_acl_action_t, 69
cpu_queue
vtss_acl_action_t, 69
vtss_mac_table_entry_t, 161
cs_wait_ns
vtss_pi_conf_t, 386
cu_bad
vtss_phy_statistic_t, 323
cu_good
vtss_phy_statistic_t, 323
cur_version
vtss_restart_status_t, 451
cw
vtss_mpls_pw_conf_t, 181
cw_en
vtss_phy_ts_mpls_conf_t, 360

data
vtss_ace_frame_etype_t, 48
vtss_ace_frame_ipv4_t, 51
vtss_ace_frame_ipv6_t, 55
vtss_ece_frame_etype_t, 91
vtss_ece_frame_llc_t, 96
vtss_ece_frame_snap_t, 97
vtss_oam_meg_id_t, 205
vtss_phy_ts_gen_conf_t, 347
vtss_phy_ts_generic_action_t, 349
vtss_qce_frame_etype_t, 424
vtss_qce_frame_llc_t, 428
vtss_qce_frame_snap_t, 429
vtss_vce_frame_etype_t, 489
vtss_vce_frame_llc_t, 493
vtss_vce_frame_snap_t, 494
default_dei
vtss_qos_port_conf_t, 444
default_dpl
vtss_qos_port_conf_t, 444
default_prio
vtss_qos_port_conf_t, 443
dei
vtss_ece_inner_tag_t, 98
vtss_ece_outer_tag_t, 105
vtss_ece_tag_t, 107
vtss_mce_outer_tag_t, 171
vtss_mce_tag_t, 175
vtss_mpls_l2_t, 178
vtss_qce_action_t, 422
vtss_qce_tag_t, 436
vtss_vce_tag_t, 500
vtss_vlan_tag_t, 506
dei_mode
vtss_ece_inner_tag_t, 98
vtss_ece_outer_tag_t, 105

vtss_mce_outer_tag_t, 171
delaym_type
vtss_phy_ts_ietf_mpls_ach_oam_conf_t, 351
vtss_phy_ts_ptp_engine_action_t, 373
vtss_phy_ts_y1731_oam_conf_t, 379
dest_ip
vtss_phy_ts_fifo_sig_t, 345
dest_mac
vtss_phy_ts_fifo_sig_t, 346
destination
vtss_ipv4_uc_t, 148
vtss_ipv6_uc_t, 151
vtss_irq_conf_t, 152
vtss_mac_table_entry_t, 160
detect_only
vtss_phy_ts_fifo_conf_t, 343
dgroup_no
vtss_dgroup_port_conf_t, 85
dip
vtss_ace_frame_arp_t, 47
vtss_ace_frame_ipv4_t, 50
vtss_ece_frame_ipv4_t, 93
vtss_ece_frame_ipv6_t, 95
vtss_qce_frame_ipv4_t, 425
vtss_qce_frame_ipv6_t, 427
dir
vtss_ece_action_t, 88
disable
vtss_phy_mac_serd_pcs_cntl_t, 313
discard
vtss_learn_mode_t, 159
discard_rx
vtss_oam_voe_up_mep_conf_t, 247
divider
vtss_synce_clock_out_t, 462
dm
vtss_oam_voe_conf_t, 228
dma_enable
vtss_packet_dma_conf_t, 256
dmac
vtss_ace_frame_etype_t, 47
vtss_ace_frame_llc_t, 58
vtss_ace_frame_snap_t, 59
vtss_ece_mac_t, 102
vtss_mce_key_t, 169
vtss_qce_mac_t, 433
dmac_bc
vtss_ace_t, 65
vtss_ece_mac_t, 103
vtss_qce_mac_t, 433
vtss_vce_mac_t, 497
dmac_check_type
vtss_oam_proc_conf_t, 209
dmac_dip
vtss_evc_port_conf_t, 122
vtss_qos_port_conf_t, 447
vtss_vcl_port_conf_t, 501
dmac_dip_1
vtss_vcap_port_conf_t, 476
dmac_enable
vtss_aggr_mode_t, 76
dmac_match
vtss_ace_frame_arp_t, 46
dmac_mc
vtss_ace_t, 65
vtss_ece_mac_t, 103
vtss_mce_key_t, 169
vtss_qce_mac_t, 433
vtss_vce_mac_t, 497
dmm
vtss_oam_vop_pdu_type_conf_t, 253
dmm_seen
vtss_oam_pdu_seen_status_t, 207
dmr
vtss_oam_vop_pdu_type_conf_t, 253
dmr_seen
vtss_oam_pdu_seen_status_t, 207
domain
vtss_phy_ts_ptp_conf_t, 371
domain_num
vtss_phy_ts_fifo_sig_t, 345
dot1dTpPortInDiscards
vtss_port_bridge_counters_t, 388
dot3InPauseFrames
vtss_port_ethernet_like_counters_t, 398
dot3OutPauseFrames
vtss_port_ethernet_like_counters_t, 398
down_mep
vtss_oam_lm_counter_t, 201
down_mep_lmr_proprietary_fcf_use
vtss_oam_vop_conf_t, 248
dp
vtss_ece_action_t, 90
vtss_mpls_tc_to_qos_map_entry_t, 188
vtss_packet_tx_info_t, 293
vtss_qce_action_t, 421
dp0_tc
vtss_mpls_qos_to_tc_map_entry_t, 182
dp1_tc
vtss_mpls_qos_to_tc_map_entry_t, 182
dp_enable
vtss_ece_action_t, 90
vtss_qce_action_t, 421
dp_level_map
vtss_qos_port_conf_t, 445
dport
vtss_ace_frame_ipv4_t, 51
vtss_ace_frame_ipv6_t, 55
vtss_ece_frame_ipv4_t, 93
vtss_ece_frame_ipv6_t, 95
vtss_qce_frame_ipv4_t, 426
vtss_qce_frame_ipv6_t, 427
vtss_vce_frame_ipv4_t, 491
vtss_vce_frame_ipv6_t, 492
dport_mask
vtss_phy_ts_ip_conf_t, 357

dport_val
vtss_phy_ts_ip_conf_t, 357
ds
vtss_ace_frame_ipv4_t, 50
vtss_ace_frame_ipv6_t, 55
vtss_phy_ts_ietf_mpls_ach_oam_conf_t, 352
dscp
vtss_ece_frame_ipv4_t, 92
vtss_ece_frame_ipv6_t, 94
vtss_qce_action_t, 422
vtss_qce_frame_ipv4_t, 425
vtss_qce_frame_ipv6_t, 426
vtss_vce_frame_ipv4_t, 490
vtss_vce_frame_ipv6_t, 492
dscp_class_enable
vtss_qos_port_conf_t, 445
dscp_dp_level_map
vtss_qos_conf_t, 438
dscp_emode
vtss_qos_port_conf_t, 445
dscp_enable
vtss_qce_action_t, 422
dscp_mode
vtss_qos_port_conf_t, 445
dscp_qos_class_map
vtss_qos_conf_t, 438
dscp_qos_map
vtss_qos_conf_t, 438
dscp_qos_map_dp1
vtss_qos_conf_t, 438
dscp_remap
vtss_qos_conf_t, 439
dscp_remap_dp1
vtss_qos_conf_t, 439
dscp_remark
vtss_qos_conf_t, 438
dscp_translate
vtss_qos_port_conf_t, 445
dscp_translate_map
vtss_qos_conf_t, 439
dscp_trust
vtss_qos_conf_t, 437
dst_port_mask
vtss_packet_tx_info_t, 286
dual_media_fiber_speed
port_custom_conf_t, 39
dwrr_cnt
vtss_hqos_conf_t, 138
vtss_qos_port_conf_t, 447
dwrr_enable
vtss_qos_port_conf_t, 447

e_lsp
vtss_mpls_segment_t, 184
ebs
vtss_dlb_policer_conf_t, 87
vtss_shaper_t, 460
eee_ena
vtss_eee_port_conf_t, 108
eee_ena_phy
vtss_phy_eee_conf_t, 305
eee_fast_queues
vtss_eee_port_conf_t, 108
eee_mode
vtss_phy_eee_conf_t, 305
egr_fcs_err
vtss_phy_ts_stats_t, 377
egr_frm_mod_cnt
vtss_phy_ts_stats_t, 378
egr_pream_shrink_err
vtss_phy_ts_stats_t, 377
egress_xc
vtss_evc_mpls_pw_info_t, 117
eir
vtss_dlb_policer_conf_t, 87
vtss_shaper_t, 460
ena_inj_header
vtss_port_ifh_t, 405
ena_xtr_header
vtss_port_ifh_t, 405
enable
port_custom_conf_t, 37
vtss_ace_ptp_t, 61
vtss_ace_sip_smac_t, 62
vtss_dlb_policer_conf_t, 86
vtss_ece_outer_tag_t, 104
vtss_fdma_cfg_t, 125
vtss_mce_outer_tag_t, 170
vtss_npi_conf_t, 193
vtss_oam_ipt_conf_t, 198
vtss_oam_voe_ccm_conf_t, 220
vtss_oam_voe_ccm_lm_conf_t, 223
vtss_oam_voe_generic_conf_t, 234
vtss_oam_voe_lb_conf_t, 236
vtss_oam_voe_lt_conf_t, 239
vtss_oam_voe_se_lm_conf_t, 241
vtss_oam_voe_tst_conf_t, 243
vtss_oam_voe_unknown_conf_t, 245
vtss_packet_rx_queue_npi_conf_t, 282
vtss_phy_ltc_freq_synth_s, 312
vtss_phy_ts_generic_action_t, 348
vtss_phy_ts_nphase_status_t, 364
vtss_phy_ts_oam_engine_action_t, 366
vtss_phy_ts_ptp_engine_action_t, 372
vtss_port_clause_37_control_t, 391
vtss_red_v2_t, 449
vtss_synce_clock_in_t, 462
vtss_synce_clock_out_t, 463
vtss_ts_ext_clock_mode_t, 469
enable_1dm
vtss_oam_voe_dm_conf_t, 231
enable_all_voe
vtss_oam_vop_conf_t, 248
enable_dmm
vtss_oam_voe_dm_conf_t, 231
enabled
vtss_sgpio_port_conf_t, 458

encap_type
vtss_phy_ts_eng_init_conf_t, 331
end
vtss_phy_ts_mpls_conf_t, 362
eng_minE
vtss_phy_ts_fifo_conf_t, 343
eng_mode
vtss_phy_ts_engine_flow_conf_t, 335
eng_recov
vtss_phy_ts_fifo_conf_t, 343
eng_used
vtss_phy_ts_eng_init_conf_t, 331
err
vtss_oam_vop_pdu_type_conf_t, 254
eth1_opt
vtss_phy_ts_generic_flow_conf_t, 350
vtss_phy_ts_oam_engine_flow_conf_t, 367
vtss_phy_ts_ptp_engine_flow_conf_t, 374
eth2_opt
vtss_phy_ts_oam_engine_flow_conf_t, 368
vtss_phy_ts_ptp_engine_flow_conf_t, 374
ethernet
vtss_ace_frame_arp_t, 46
ethernet_like
vtss_port_counters_t, 397
etype
vtss_ace_frame_etype_t, 48
vtss_ace_t, 65
vtss_ece_frame_etype_t, 91
vtss_ece_key_t, 101
vtss_packet_rx_meta_t, 275
vtss_phy_ts_eth_conf_t, 338
vtss_qce_frame_etype_t, 424
vtss_qce_key_t, 431
vtss_vce_frame_etype_t, 489
vtss_vce_key_t, 495
evc_counting
vtss_mce_action_t, 164
evc_etree
vtss_mce_action_t, 164
evc_id
vtss_ece_action_t, 89
vtss_mce_action_t, 164
evc_police
vtss_acl_action_t, 70
evc_policer_id
vtss_acl_action_t, 70
exc_col_cont
port_custom_conf_t, 39
vtss_port_conf_t, 395
excess_enable
vtss_qos_port_conf_t, 443
external
vtss_irq_conf_t, 152
external_cpu_portmask
vtss_oam_vop_conf_t, 248
extract
vtss_oam_vop_generic_opcode_conf_t, 251
FALSE
types.h, 559
fan_low_pol
vtss_fan_conf_t, 123
fan_open_col
vtss_fan_conf_t, 123
fan_pwm_freq
vtss_fan_conf_t, 123
far_end_enable
vtss_phy_loopback_t, 309
fast_link_stat_ena
vtss_phy_mac_serd_pcs_cntl_t, 315
fcs
vtss_packet_rx_meta_t, 275
fdx
port_custom_conf_t, 38
vtss_phy_forced_t, 307
vtss_port_clause_37_adv_t, 389
vtss_port_conf_t, 394
vtss_port_sgmii_aneg_t, 416
vtss_port_status_t, 419
fdx_gap
vtss_port_frame_gaps_t, 401
fiber
vtss_port_status_t, 420
fid
vtss_vlan_vid_conf_t, 509
file
vtss_api_lock_t, 78
fill_level
vtss_eee_port_counter_t, 110
fill_level_get
vtss_eee_port_counter_t, 110
fill_level_thres
vtss_eee_port_counter_t, 110
filter
vtss_packet_port_filter_t, 258
flf
vtss_phy_conf_t, 302
flow_conf
vtss_phy_ts_engine_flow_conf_t, 336
flow_control
port_custom_conf_t, 38
vtss_policer_ext_t, 387
vtss_port_conf_t, 394
flow_en
vtss_phy_ts_eth_conf_t, 338
vtss_phy_ts_gen_conf_t, 347
vtss_phy_ts_ip_conf_t, 357
vtss_phy_ts_mpls_conf_t, 360
flow_end_index
vtss_phy_ts_eng_init_conf_t, 332
flow_handle
vtss_mpls_xc_t, 191
flow_id
vtss_phy_ts_generic_action_t, 349
flow_match_mode
vtss_phy_ts_eng_init_conf_t, 332

flow_offset
vtss_phy_ts_gen_conf_t, 347
flow_opt
vtss_phy_ts_eth_conf_t, 342
vtss_phy_ts_gen_conf_t, 348
vtss_phy_ts_ip_conf_t, 359
vtss_phy_ts_mpls_conf_t, 363
flow_st_index
vtss_phy_ts_eng_init_conf_t, 332
force
vtss_phy_reset_conf_t, 321
force_adv_ability
vtss_phy_mac_serd_pcs_cntl_t, 314
vtss_phy_media_serd_pcs_cntl_t, 316
force_ams_sel
vtss_phy_conf_t, 303
force_fefi
vtss_phy_media_serd_pcs_cntl_t, 317
force_fefi_value
vtss_phy_media_serd_pcs_cntl_t, 318
force_hls
vtss_phy_media_serd_pcs_cntl_t, 317
forced
vtss_phy_conf_t, 301
forward
vtss_oam_voe_ccm_conf_t, 220
vtss_oam_voe_ccm_lm_conf_t, 223
vtss_oam_voe_generic_conf_t, 234
vtss_oam_voe_tst_conf_t, 243
forward_1dm
vtss_oam_voe_dm_conf_t, 232
forward_dmm
vtss_oam_voe_dm_conf_t, 232
forward_dmr
vtss_oam_voe_dm_conf_t, 233
forward_lbm
vtss_oam_voe_lb_conf_t, 236
forward_lbr
vtss_oam_voe_lb_conf_t, 236
forward_lmm
vtss_oam_voe_se_lm_conf_t, 241
forward_lmr
vtss_oam_voe_se_lm_conf_t, 242
forward_ltm
vtss_oam_voe_lt_conf_t, 239
forward_ltr
vtss_oam_voe_lt_conf_t, 240
fps
vtss_afi_frm_dscr_t, 74
fragment
vtss_ace_frame_ipv4_t, 49
vtss_ece_frame_ipv4_t, 92
vtss_qce_frame_ipv4_t, 425
vtss_vce_frame_ipv4_t, 490
frame
vtss_ace_t, 66
vtss_ece_key_t, 102
vtss_qce_key_t, 432
vtss_vce_key_t, 496
frame_gaps
vtss_port_conf_t, 393
frame_length_chk
port_custom_conf_t, 40
vtss_port_conf_t, 394
frame_rate
vtss_policer_ext_t, 387
frame_type
vtss_vlan_port_conf_t, 504
frames
vtss_counter_pair_t, 81
freq
vtss_phy_clock_conf_t, 299
vtss_ts_ext_clock_mode_t, 469
frm_len
vtss_packet_tx_info_t, 287
frm_limit_per_tick
vtss_fdma_throttle_cfg_t, 134
frm_ptr
tag_vtss_fdma_list, 42
frst_lvl_after_top
vtss_phy_ts_mpls_conf_t, 361
frst_lvl_before_end
vtss_phy_ts_mpls_conf_t, 362
fsm_lock
vtss_lcpll_status_t, 157
fsm_stat
vtss_lcpll_status_t, 157
full
vtss_debug_info_t, 83
function
vtss_api_lock_t, 78
fwd_enable
vtss_mirror_conf_t, 176

gain_stat
vtss_lcpll_status_t, 157
garp_cpu_only
vtss_packet_rx_reg_t, 283
garp_queue
vtss_packet_rx_queue_map_t, 280
garp_reg
vtss_packet_rx_port_conf_t, 278
gen
vtss_phy_ts_engine_flow_conf_t, 336
gen_conf
vtss_phy_ts_engine_action_t, 334
gen_opt
vtss_phy_ts_generic_flow_conf_t, 351
generate
vtss_port_flow_control_conf_t, 399
generate_pause
vtss_aneg_t, 77
generic
vtss_oam_voe_conf_t, 227
vtss_oam_vop_pdu_type_conf_t, 252
generic_seen
vtss_oam_pdu_seen_status_t, 206

glag_no
vtss_packet_rx_info_t, 265
group
vtss_debug_info_t, 82
group_id
vtss_vlan_trans_grp2vlan_conf_t, 507
vtss_vlan_trans_port2grp_conf_t, 508
grp_map
vtss_packet_rx_conf_t, 261

hdx
vtss_port_clause_37_adv_t, 389
vtss_port_sgmii_aneg_t, 416
hdx_gap_1
vtss_port_frame_gaps_t, 400
hdx_gap_2
vtss_port_frame_gaps_t, 400
header
vtss_ace_ptp_t, 61
high
vtss_phy_timestamp_t, 327
vtss_vcap_udp_tcp_t, 484
vtss_vcap_vr_t, 487
high_duty_cycle
vtss_phy_ltc_freq_synth_s, 312
hints
vtss_packet_rx_info_t, 264
hqos_id
vtss_mpls_segment_t, 185
hw_cnt
vtss_os_timestamp_t, 255
hw_tstamp
vtss_packet_rx_info_t, 270
hw_tstamp_decoded
vtss_packet_rx_info_t, 270

i16
types.h, 576
i32
types.h, 577
i64
types.h, 577
i8
types.h, 576
i_cpu_en
vtss_phy_reset_conf_t, 321
i_tag
vtss_phy_ts_eth_conf_t, 342
ib_cterm_ena
vtss_serdes_macro_conf_t, 455
id
vtss_ace_t, 63
vtss_ece_t, 106
vtss_mce_t, 173
vtss_qce_t, 434
vtss_ts_timestamp_t, 474
vtss_vce_t, 498
ietf_oam_conf
vtss_phy_ts_oam_engine_action_t, 366
if_group
vtss_port_counters_t, 397
if_type
vtss_port_conf_t, 392
ifInBroadcastPkts
vtss_port_if_group_counters_t, 402
ifInDiscards
vtss_port_if_group_counters_t, 402
ifInErrors
vtss_port_if_group_counters_t, 403
ifInMulticastPkts
vtss_port_if_group_counters_t, 402
ifInNUcastPkts
vtss_port_if_group_counters_t, 402
ifInOctets
vtss_port_if_group_counters_t, 402
ifInUcastPkts
vtss_port_if_group_counters_t, 402
ifOutBroadcastPkts
vtss_port_if_group_counters_t, 403
ifOutDiscards
vtss_port_if_group_counters_t, 404
ifOutErrors
vtss_port_if_group_counters_t, 404
ifOutMulticastPkts
vtss_port_if_group_counters_t, 403
ifOutNUcastPkts
vtss_port_if_group_counters_t, 404
ifOutOctets
vtss_port_if_group_counters_t, 403
ifOutUcastPkts
vtss_port_if_group_counters_t, 403
ifh
vtss_packet_tx_ifh_t, 285
igmp_cpu_only
vtss_packet_rx_reg_t, 284
igmp_queue
vtss_packet_rx_queue_map_t, 280
igmp_reg
vtss_packet_rx_port_conf_t, 277
in_range
vtss_vcap_udp_tcp_t, 484
in_seg_idx
vtss_mpls_xc_t, 191
ingr_fcs_err
vtss_phy_ts_stats_t, 377
ingr_frm_mod_cnt
vtss_phy_ts_stats_t, 378
ingr_pream_shrink_err
vtss_phy_ts_stats_t, 377
ingress_filter
vtss_vlan_port_conf_t, 504
ingress_xc
vtss_evc_mpls_pw_info_t, 117
inhibit_odd_start
vtss_phy_mac_serd_pcs_cntl_t, 315
vtss_phy_media_serd_pcs_cntl_t, 317
inj_grp_mask

vtss_fdma_ch_cfg_t, 130
injected
vtss_mce_key_t, 168
inner_tag
vtss_ece_action_t, 89
vtss_ece_key_t, 100
vtss_mce_key_t, 168
vtss_phy_ts_eth_conf_t, 342
vtss_qce_key_t, 431
inner_tag_type
vtss_phy_ts_eth_conf_t, 340
inst
vtss_api_lock_t, 78
int_fmt
vtss_ts_internal_mode_t, 471
int_pol_high
vtss_sgpio_port_conf_t, 458
ip
vtss_ace_frame_arp_t, 46
ip1_opt
vtss_phy_ts_ptp_engine_flow_conf_t, 374
ip2_opt
vtss_phy_ts_ptp_engine_flow_conf_t, 374
ip_addr
vtss_phy_ts_ip_conf_t, 359
ip_enable
vtss_phy_ts_sertod_conf_t, 376
ip_mode
vtss_phy_ts_ip_conf_t, 356
ipmc_ctrl_cpu_copy
vtss_packet_rx_reg_t, 283
ipmc_ctrl_queue
vtss_packet_rx_queue_map_t, 281
ipmc_ctrl_reg
vtss_packet_rx_port_conf_t, 277
ipv4
vtss_ace_t, 66
vtss_ece_key_t, 101
vtss_ip_addr_t, 145
vtss_phy_ts_ip_conf_t, 358
vtss_qce_key_t, 432
vtss_vce_key_t, 496
ipv4_uc
vtss_routing_entry_t, 452
ipv4uc_received_frames
vtss_l3_counters_t, 154
ipv4uc_received_octets
vtss_l3_counters_t, 154
ipv4uc_transmitted_frames
vtss_l3_counters_t, 155
ipv4uc_transmitted_octets
vtss_l3_counters_t, 155
ipv6
vtss_ace_t, 66
vtss_ece_key_t, 101
vtss_ip_addr_t, 145
vtss_phy_ts_ip_conf_t, 358
vtss_qce_key_t, 432
vtss_vce_key_t, 496
ipv6_uc
vtss_routing_entry_t, 452
ipv6uc_received_frames
vtss_l3_counters_t, 155
ipv6uc_received_octets
vtss_l3_counters_t, 155
ipv6uc_transmitted_frames
vtss_l3_counters_t, 156
ipv6uc_transmitted_octets
vtss_l3_counters_t, 155
is_in
vtss_mpls_segment_t, 184
is_pw
vtss_mpls_pw_conf_t, 180
isdx
vtss_mce_action_t, 165
vtss_mce_port_info_t, 172
vtss_packet_rx_info_t, 272
vtss_packet_tx_info_t, 292
isdx_disable
vtss_ace_t, 64
isdx_dont_use
vtss_packet_tx_info_t, 292
isdx_enable
vtss_ace_t, 63
ivid
vtss_evc_pb_conf_t, 120

key
vtss_ece_t, 106
vtss_mce_t, 173
vtss_qce_t, 434
vtss_vce_t, 498
key_type
vtss_evc_port_conf_t, 122
vtss_qos_port_conf_t, 447
key_type_is1_1
vtss_vcap_port_conf_t, 476

l2_idx
vtss_mpls_segment_t, 184
l2_types.h
vtss_sflow_type_t, 511
l_lsp_cos
vtss_mpls_segment_t, 184
label
vtss_mpls_segment_t, 185
latch_timestamp
vtss_packet_tx_info_t, 291
layer
vtss_debug_info_t, 82
lb
vtss_oam_voe_conf_t, 228
vtss_oam_voe_counter_t, 230
lb_type
vtss_phy_api.h, 844
lbm
vtss_oam_vop_pdu_type_conf_t, 254

lbm_seen
vtss_oam_pdu_seen_status_t, 207
lbr
vtss_oam_vop_pdu_type_conf_t, 254
lbr_seen
vtss_oam_pdu_seen_status_t, 207
leaf
vtss_evc_pb_conf_t, 121
leaf_ivid
vtss_evc_pb_conf_t, 121
leaf_vid
vtss_evc_pb_conf_t, 121
learn
vtss_acl_action_t, 70
vtss_packet_rx_header_t, 262
learn_queue
vtss_packet_rx_queue_map_t, 280
learned
vtss_mac_table_status_t, 162
learning
vtss_evc_conf_t, 114
vtss_vlan_vid_conf_t, 509
length
vtss_ace_frame_arp_t, 46
vtss_packet_rx_header_t, 262
vtss_packet_rx_info_t, 265
vtss_packet_rx_meta_t, 276
vtss_packet_tx_ifh_t, 285
vtss_phy_veriphy_result_t, 383
level
vtss_phy_power_status_t, 319
vtss_policer_t, 387
vtss_shaper_t, 460
vtss_trace_conf_t, 467
line
vtss_api_lock_t, 78
line_rate
vtss_dlb_policer_conf_t, 86
link
vtss_phy_veriphy_result_t, 383
vtss_port_sgmii_aneg_t, 416
vtss_port_status_t, 418
link_change
vtss_intr_t, 144
link_down
vtss_port_status_t, 418
list
vtss_fdma_ch_cfg_t, 131
llabs
vtss_os_ecos.h, 784
llc
vtss_ace_frame_llc_t, 59
vtss_ace_t, 65
vtss_ece_key_t, 101
vtss_qce_key_t, 431
vtss_vce_key_t, 495
lm
vtss_oam_voe_counter_t, 230
lm_cnt_disable
vtss_acl_action_t, 71
lm_count
vtss_oam_lm_counter_t, 204
lmm
vtss_oam_lm_counter_t, 202
vtss_oam_vop_pdu_type_conf_t, 253
lmm_sample_lost
vtss_oam_lm_counter_t, 203
lmm_seen
vtss_oam_pdu_seen_status_t, 206
lmm_valid
vtss_oam_lm_counter_t, 202
lmr
vtss_oam_lm_counter_t, 202
vtss_oam_vop_pdu_type_conf_t, 253
lmr_sample_lost
vtss_oam_lm_counter_t, 203
lmr_seen
vtss_oam_pdu_seen_status_t, 206
lmr_valid
vtss_oam_lm_counter_t, 203
load
vtss_ts_alt_clock_mode_t, 468
loc
vtss_oam_ccm_status_t, 196
lock_status
vtss_lcpll_status_t, 156
locked
vtss_mac_table_entry_t, 161
lookup
vtss_ace_t, 63
vtss_ece_key_t, 100
vtss_mce_key_t, 169
loop
vtss_port_conf_t, 395
loop_isdx_p
vtss_oam_voe_conf_t, 226
loop_isdx_w
vtss_oam_voe_conf_t, 226
loop_portidx_p
vtss_oam_voe_conf_t, 226
loopback
vtss_oam_voe_up_mep_conf_t, 247
low
vtss_phy_timestamp_t, 327
vtss_vcap_udp_tcp_t, 484
vtss_vcap_vr_t, 487
low_duty_cycle
vtss_phy_ltc_freq_synth_s, 312
lower
vtss_phy_ts_eth_conf_t, 340
vtss_phy_ts_mpls_lvl_rng_t, 363
vtss_phy_ts_ptp_conf_t, 371
vtss_phy_ts_y1731_oam_conf_t, 380
lp_advertisement
vtss_eee_port_conf_t, 109
lrn_all_queue

vtss_packet_rx_queue_map_t, 281
ls_inv
vtss_phy_ts_sertod_conf_t, 376
ls_lpbk
vtss_phy_ts_alt_clock_mode_s, 330
ls_pps_lpbk
vtss_phy_ts_alt_clock_mode_s, 330
lt
vtss_oam_voe_conf_t, 229
vtss_oam_vop_pdu_type_conf_t, 253
ltm_seen
vtss_oam_pdu_seen_status_t, 206
ltr_seen
vtss_oam_pdu_seen_status_t, 206

MAC_ADDR_BROADCAST
types.h, 567
MAX_CFG_BUF_SIZE
vtss_phy_api.h, 824
MAX_STAT_BUF_SIZE
vtss_phy_api.h, 824
MAX_WOL_MAC_ADDR_SIZE
vtss_phy_api.h, 835
MAX_WOL_PASSWD_SIZE
vtss_phy_api.h, 835
MIN_WOL_PASSWD_SIZE
vtss_phy_api.h, 835
mac
vtss_ece_key_t, 100
vtss_qce_key_t, 430
vtss_vce_key_t, 495
vtss_vid_mac_t, 502
mac_addr
vtss_phy_ts_eth_conf_t, 339
mac_if
vtss_phy_reset_conf_t, 320
mac_if_pcs
vtss_phy_conf_t, 303
mac_serdes_equipment_enable
vtss_phy_loopback_t, 310
mac_serdes_facility_enable
vtss_phy_loopback_t, 310
mac_serdes_input_enable
vtss_phy_loopback_t, 310
mac_swap
vtss_acl_action_t, 71
mac_vid_queue
vtss_packet_rx_queue_map_t, 281
macsec_ena
vtss_phy_ts_init_conf_t, 355
magic_pkt_cnt
vtss_phy_wol_conf_t, 385
map
vtss_packet_rx_conf_t, 261
mask
vtss_phy_ts_ach_conf_t, 328, 329
vtss_phy_ts_eth_conf_t, 341, 342
vtss_phy_ts_gen_conf_t, 347
vtss_phy_ts_generic_action_t, 349
vtss_phy_ts_ip_conf_t, 358
vtss_phy_ts_ptp_conf_t, 370
vtss_phy_ts_y1731_oam_conf_t, 380
vtss_vcap_ip_t, 475
vtss_vcap_u128_t, 477
vtss_vcap_u16_t, 478
vtss_vcap_u24_t, 479
vtss_vcap_u32_t, 480
vtss_vcap_u40_t, 481
vtss_vcap_u48_t, 482
vtss_vcap_u8_t, 483
vtss_vcap_vid_t, 485
vtss_vcap_vr_t, 486
masquerade_port
vtss_packet_tx_info_t, 293
master
vtss_phy_conf_1g_t, 300
vtss_phy_status_1g_t, 325
master_cfg_fault
vtss_phy_status_1g_t, 325
match_mode
vtss_phy_ts_ip_conf_t, 358
max
vtss_red_v2_t, 450
max_frame_length
vtss_port_conf_t, 394
max_length
port_custom_conf_t, 39
max_tags
port_custom_conf_t, 40
vtss_port_conf_t, 395
max_unit
vtss_red_v2_t, 450
mdi
vtss_phy_conf_t, 302
mdi_cross
vtss_port_status_t, 420
media_if
vtss_phy_reset_conf_t, 320
media_if_pcs
vtss_phy_conf_t, 303
media_mac_serdes_crc
vtss_phy_statistic_t, 324
media_mac_serdes_good
vtss_phy_statistic_t, 324
media_serdes_equipment_enable
vtss_phy_loopback_t, 311
media_serdes_facility_enable
vtss_phy_loopback_t, 311
media_serdes_input_enable
vtss_phy_loopback_t, 310
meg_id_err
vtss_oam_ccm_status_t, 197
meg_level
vtss_oam_proc_conf_t, 209
vtss_phy_ts_y1731_oam_conf_t, 380
megid
vtss_oam_voe_ccm_conf_t, 221

mel
vtss_mce_key_t, 168
mep_id_err
vtss_oam_ccm_status_t, 196
mep_type
vtss_oam_voe_conf_t, 225
mepid
vtss_oam_voe_ccm_conf_t, 220
miim_addr
vtss_port_map_t, 406
miim_chip_no
vtss_port_map_t, 407
miim_controller
vtss_port_map_t, 406
miim_read
vtss_init_conf_t, 140
miim_write
vtss_init_conf_t, 141
min_fl
vtss_red_v2_t, 450
min_rate
vtss_hqos_conf_t, 138
mirror
vtss_acl_action_t, 71
vtss_vlan_vid_conf_t, 509
mld_cpu_only
vtss_packet_rx_reg_t, 284
mld_reg
vtss_packet_rx_port_conf_t, 278
mmd_read
vtss_init_conf_t, 141
mmd_read_inc
vtss_init_conf_t, 141
mmd_write
vtss_init_conf_t, 141
mode
vtss_phy_conf_t, 301
vtss_phy_led_mode_select_t, 308
vtss_phy_power_conf_t, 318
vtss_sgpio_port_conf_t, 458
vtss_shaper_t, 460
vtss_ts_operation_mode_t, 471
moved
vtss_mac_table_status_t, 162
mpls_opt
vtss_phy_ts_oam_engine_flow_conf_t, 368
vtss_phy_ts_ptp_engine_flow_conf_t, 375
mpls_tp
vtss_evc_conf_t, 114
msg_type
vtss_phy_ts_fifo_sig_t, 344

nanoseconds
vtss_phy_timestamp_t, 327
vtss_timestamp_t, 466
near_end_enable
vtss_phy_loopback_t, 309
network
vtss_evc_conf_t, 114
vtss_ipv4_uc_t, 148
vtss_ipv6_uc_t, 151
next
tag_vtss_fdma_list, 44
next_page
vtss_port_clause_37_adv_t, 390
next_prot_offset
vtss_phy_ts_gen_conf_t, 347
nni
vtss_evc_pb_conf_t, 120
no_wait
vtss_packet_rx_meta_t, 273
nonselected_frames
vtss_oam_sel_counter_t, 215
now
vtss_mtimer_t, 192
npi
vtss_packet_rx_queue_conf_t, 279
num_tag
vtss_phy_ts_eth_conf_t, 339
number
vtss_phy_led_mode_select_t, 308

oam
vtss_phy_ts_engine_flow_conf_t, 336
oam_active
vtss_mpls_segment_status_t, 183
oam_conf
vtss_mpls_segment_t, 185
vtss_phy_ts_engine_action_t, 334
vtss_phy_ts_oam_engine_action_t, 367
oam_detect
vtss_mce_action_t, 165
oam_info
vtss_packet_rx_info_t, 271
oam_info_decoded
vtss_packet_rx_info_t, 272
oam_type
vtss_packet_tx_info_t, 291
ob_post0
serdes_fields_t, 41
ob_sr
serdes_fields_t, 41
obey
vtss_port_flow_control_conf_t, 399
obey_pause
vtss_aneg_t, 77
one_dm_seen
vtss_oam_pdu_seen_status_t, 207
one_pps_in
vtss_ts_alt_clock_mode_t, 468
one_pps_mode
vtss_ts_ext_clock_mode_t, 469
one_pps_out
vtss_ts_alt_clock_mode_t, 467
one_step_txfifo
vtss_phy_ts_init_conf_t, 355
op_enable
vtss_phy_ts_sertod_conf_t, 376

opcode
vtss_oam_vop_generic_opcode_conf_t, 251
oper_up
port_custom_conf_t, 40
optimized_for_power
vtss_eee_port_conf_t, 109
options
vtss_ace_frame_ipv4_t, 50
vtss_vce_frame_ipv4_t, 490
options.h
TESLA_ING_TS_ERRFIX, 528
VIPER_B_FIFO_RESET, 528
VTSS_AFI_V1, 533
VTSS_ARCH_SERVAL_CPU, 514
VTSS_ARCH_SERVAL_CE, 514
VTSS_ARCH_SERVAL, 514
VTSS_CHIP_CU_PHY, 533
VTSS_FEATURE_ACL_V2, 526
VTSS_FEATURE_ACL, 526
VTSS_FEATURE_AFI_SWC, 532
VTSS_FEATURE_CLAUSE_37, 515
VTSS_FEATURE_E_TREE, 532
VTSS_FEATURE_EEE, 525
VTSS_FEATURE_EVC, 532
VTSS_FEATURE_EXC_COL_CONT, 516
VTSS_FEATURE_FAN, 530
VTSS_FEATURE_FDMA, 532
VTSS_FEATURE_HQOS, 533
VTSS_FEATURE_INTERRUPTS, 529
VTSS_FEATURE_IRQ_CONTROL, 527
VTSS_FEATURE_LAYER2, 524
VTSS_FEATURE_LED_POW_REDUC, 527
VTSS_FEATURE_MAC_AGE_AUTO, 525
VTSS_FEATURE_MAC_CPU_QUEUE, 525
VTSS_FEATURE_MIRROR_CPU, 527
VTSS_FEATURE_MISC, 515
VTSS_FEATURE_MPLS, 533
VTSS_FEATURE_NPI, 526
VTSS_FEATURE_OAM, 532
VTSS_FEATURE_PACKET_GROUPING, 524
VTSS_FEATURE_PACKET_PORT_REG, 524
VTSS_FEATURE_PACKET_RX, 523
VTSS_FEATURE_PACKET_TX, 523
VTSS_FEATURE_PACKET, 523
VTSS_FEATURE_PFC, 516
VTSS_FEATURE_PHY_TIMESTAMP, 530
VTSS_FEATURE_PORT_CNT_BRIDGE, 516
VTSS_FEATURE_PORT_CONTROL, 515
VTSS_FEATURE_PORT_IFH, 515
VTSS_FEATURE_PTP_RS422, 531
VTSS_FEATURE_PVLAN, 524
VTSS_FEATURE_QCL_DMAC_DIP, 517
VTSS_FEATURE_QCL_KEY_DIP, 518
VTSS_FEATURE_QCL_KEY_DMAC, 518
VTSS_FEATURE_QCL_KEY_INNER_TAG, 517
VTSS_FEATURE_QCL_KEY_S_TAG, 517
VTSS_FEATURE_QCL_KEY_TYPE, 517
VTSS_FEATURE_QCL_PCP_DEI_ACTION, 518
VTSS_FEATURE_QCL_POLICY_ACTION, 518
VTSS_FEATURE_QCL_V2, 517
VTSS_FEATURE_QCL, 516
VTSS_FEATURE_QOS_CLASSIFICATION_V2, 521
VTSS_FEATURE_QOS_CPU_PORT_SHAPER, 523
VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE, 522
VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWARE, 522
VTSS_FEATURE_QOS_DSCP_REMARK_V2, 522
VTSS_FEATURE_QOS_DSCP_REMARK, 522
VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS_EB, 521
VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPERS, 521
VTSS_FEATURE_QOS_EGRESS_SHAPERS_DLB, 521
VTSS_FEATURE_QOS_EGRESS_SHAPERS_RT, 521
VTSS_FEATURE_QOS_POLICER_BC_SWITCH, 519
VTSS_FEATURE_QOS_POLICER_DLB, 523
VTSS_FEATURE_QOS_POLICER_MC_SWITCH, 519
VTSS_FEATURE_QOS_POLICER_UC_SWITCH, 518
VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS, 519
VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC, 519
VTSS_FEATURE_QOS_PORT_POLICER_EXT, 519
VTSS_FEATURE_QOS_QUEUE_POLICER, 520
VTSS_FEATURE_QOS_QUEUE_TX, 520
VTSS_FEATURE_QOS_SCHEDULER_DWRR_CNT, 520
VTSS_FEATURE_QOS_SCHEDULER_V2, 520
VTSS_FEATURE_QOS_TAG_REMARK_V2, 520
VTSS_FEATURE_QOS_WRED_V2, 522
VTSS_FEATURE_QOS, 516
VTSS_FEATURE_SERDES_MACRO_SETTINGS, 528, 529
VTSS_FEATURE_SERIAL_GPIO, 529
VTSS_FEATURE_SFLOW, 527
VTSS_FEATURE_SYNCE, 529
VTSS_FEATURE_TIMESTAMP_ASYMMETRY_COMP, 531
VTSS_FEATURE_TIMESTAMP_LATENCY_COMP, 530
VTSS_FEATURE_TIMESTAMP_ONE_STEP, 530
VTSS_FEATURE_TIMESTAMP_ORG_TIME, 531
VTSS_FEATURE_TIMESTAMP_P2P_DELAY_COMP, 531
VTSS_FEATURE_TIMESTAMP, 530
VTSS_FEATURE_VCAP, 526, 535

VTSS_FEATURE_VCL, 526
VTSS_FEATURE_VLAN_PORT_V2, 524
VTSS_FEATURE_VLAN_SVL, 525
VTSS_FEATURE_VLAN_TRANSLATION, 527
VTSS_FEATURE_VLAN_TX_TAG, 525
VTSS_FEATURE_WARM_START, 515, 535
VTSS_OPT_FDMA_DEBUG, 534
VTSS_OPT_FDMA_IRQ_CONTEXT, 534
VTSS_OPT_PORT_COUNT, 534
VTSS_OPT_TRACE, 533
VTSS_OPT_VAUI_EQ_CTRL, 534
VTSS_OPT_VCORE_III, 531
VTSS_PHY_OPT_VERIPHY, 534
VTSS_PHY_TS_SPI_CLK_THRU_PPS0, 529
VTSS_TS_FIFO_MEDIA_SWAP_SYNC, 528
VTSS_TS_FIFO_SYNC_LOOPBACK, 528
other
vtss_oam_vop_pdu_type_conf_t, 254
out_of_range
vtss_rcpll_status_t, 448
out_seg_idx
vtss_mpls_xc_t, 191
outer_tag
vtss_ece_action_t, 89
vtss_mce_action_t, 165
vtss_phy_ts_eth_conf_t, 341
outer_tag_type
vtss_phy_ts_eth_conf_t, 339

PORT_CAP_100M_FDX
port.h, 540
PORT_CAP_100M_HDX
port.h, 540
PORT_CAP_10G_FDX
port.h, 541
PORT_CAP_10M_FDX
port.h, 540
PORT_CAP_10M_HDX
port.h, 540
PORT_CAP_1G_FDX
port.h, 540
PORT_CAP_1G_PHY
port.h, 545
PORT_CAP_2_5G_FDX
port.h, 541
PORT_CAP_2_5G_TRI_SPEED_COPPER
port.h, 548
PORT_CAP_2_5G_TRI_SPEED_FDX
port.h, 548
PORT_CAP_2_5G_TRI_SPEED
port.h, 548
PORT_CAP_5G_FDX
port.h, 541
PORT_CAP_ANY_FIBER
port.h, 546
PORT_CAP_AUTONEG
port.h, 539
PORT_CAP_COPPER
port.h, 541
PORT_CAP_DUAL_COPPER_100FX
port.h, 544
PORT_CAP_DUAL_COPPER
port.h, 542
PORT_CAP_DUAL_FIBER_1000X
port.h, 547
PORT_CAP_DUAL_FIBER_100FX
port.h, 543
PORT_CAP_DUAL_FIBER
port.h, 542
PORT_CAP_DUAL_SFP_DETECT
port.h, 544
PORT_CAP_FIBER
port.h, 542
PORT_CAP_FLOW_CTRL
port.h, 541
PORT_CAP_HDX
port.h, 544
PORT_CAP_NONE
port.h, 539
PORT_CAP_SD_ENABLE
port.h, 542
PORT_CAP_SD_HIGH
port.h, 542
PORT_CAP_SD_INTERNAL
port.h, 543
PORT_CAP_SFP_1G
port.h, 547
PORT_CAP_SFP_2_5G
port.h, 548
PORT_CAP_SFP_DETECT
port.h, 543
PORT_CAP_SFP_ONLY
port.h, 544
PORT_CAP_SFP_SD_HIGH
port.h, 548
PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXED_S↩
PEED
port.h, 546
PORT_CAP_SPEED_DUAL_ANY_FIBER
port.h, 547
PORT_CAP_STACKING
port.h, 544
PORT_CAP_TRI_SPEED_COPPER
port.h, 545
PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_FIXE↩
D_SFP_SPEED
port.h, 547
PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER
port.h, 547
PORT_CAP_TRI_SPEED_DUAL_COPPER
port.h, 546
PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX
port.h, 546
PORT_CAP_TRI_SPEED_DUAL_FIBER
port.h, 546
PORT_CAP_TRI_SPEED_FDX
port.h, 545

PORT_CAP_TRI_SPEED_FIBER
port.h, 545
PORT_CAP_TRI_SPEED
port.h, 545
PORT_CAP_VTSS_10G_PHY
port.h, 543
PORT_CAP_XAUI_LANE_FLIP
port.h, 543
PRIi64
types.h, 558
PRIu64
types.h, 557
PRIx64
types.h, 558
part_number
vtss_chip_id_t, 80
vtss_phy_type_t, 381
passwd
vtss_secure_on_passwd_t, 454
pb
vtss_evc_conf_t, 114
pbb_en
vtss_phy_ts_eth_conf_t, 338
pcp
vtss_ece_inner_tag_t, 98
vtss_ece_outer_tag_t, 104
vtss_ece_tag_t, 107
vtss_mce_outer_tag_t, 171
vtss_mce_tag_t, 175
vtss_mpls_l2_t, 177
vtss_qce_action_t, 422
vtss_qce_tag_t, 436
vtss_vce_tag_t, 500
vtss_vlan_tag_t, 505
pcp_dei_enable
vtss_qce_action_t, 422
pcp_mode
vtss_ece_inner_tag_t, 98
vtss_ece_outer_tag_t, 104
vtss_mce_outer_tag_t, 170
pd_enable
vtss_phy_mac_serd_pcs_cntl_t, 313
pdu_offset
vtss_packet_tx_info_t, 294
pdu_type
vtss_oam_vop_conf_t, 249
peer_mac
vtss_mpls_l2_t, 177
period
vtss_oam_voe_ccm_lm_conf_t, 224
period_err
vtss_oam_ccm_status_t, 195
pfc
port_custom_conf_t, 38
vtss_port_flow_control_conf_t, 399
phy.h
VTSS_PHY_POWER_ACTIPHY_BIT, 536
VTSS_PHY_POWER_DYNAMIC_BIT, 536
vtss_phy_power_mode_t, 536
vtss_phy_veriphy_status_t, 537
phy_api_base_no
vtss_phy_type_t, 382
phys_port
vtss_oam_voe_alloc_cfg_t, 219
pi
vtss_init_conf_t, 143
pkt_mode
vtss_phy_reset_conf_t, 321
police
vtss_acl_action_t, 69
policer_bc
vtss_qos_conf_t, 440
policer_bc_frame_rate
vtss_qos_conf_t, 441
policer_bc_mode
vtss_qos_conf_t, 441
policer_ext_port
vtss_qos_port_conf_t, 442
policer_id
vtss_ece_action_t, 89
vtss_evc_conf_t, 113
vtss_mce_action_t, 166
policer_mc
vtss_qos_conf_t, 440
policer_mc_frame_rate
vtss_qos_conf_t, 440
policer_mc_mode
vtss_qos_conf_t, 440
policer_no
vtss_acl_action_t, 69
policer_port
vtss_qos_port_conf_t, 442
policer_queue
vtss_qos_port_conf_t, 443
policer_uc
vtss_qos_conf_t, 439
policer_uc_frame_rate
vtss_qos_conf_t, 439
policer_uc_mode
vtss_qos_conf_t, 440
policy
vtss_ace_t, 64
policy_no
vtss_acl_port_conf_t, 73
vtss_ece_action_t, 90
vtss_mce_action_t, 166
vtss_qce_action_t, 423
vtss_vce_action_t, 488
policy_no_enable
vtss_qce_action_t, 423
pop_cnt
vtss_mce_action_t, 167
pop_tag
vtss_ece_action_t, 89
port
vtss_mpls_l2_t, 177

vtss_oam_voe_up_mep_conf_t, 247
port.h
PORT_CAP_100M_FDX, 540
PORT_CAP_100M_HDX, 540
PORT_CAP_10G_FDX, 541
PORT_CAP_10M_FDX, 540
PORT_CAP_10M_HDX, 540
PORT_CAP_1G_FDX, 540
PORT_CAP_1G_PHY, 545
PORT_CAP_2_5G_FDX, 541
PORT_CAP_2_5G_TRI_SPEED_COPPER, 548
PORT_CAP_2_5G_TRI_SPEED_FDX, 548
PORT_CAP_2_5G_TRI_SPEED, 548
PORT_CAP_5G_FDX, 541
PORT_CAP_ANY_FIBER, 546
PORT_CAP_AUTONEG, 539
PORT_CAP_COPPER, 541
PORT_CAP_DUAL_COPPER_100FX, 544
PORT_CAP_DUAL_COPPER, 542
PORT_CAP_DUAL_FIBER_1000X, 547
PORT_CAP_DUAL_FIBER_100FX, 543
PORT_CAP_DUAL_FIBER, 542
PORT_CAP_DUAL_SFP_DETECT, 544
PORT_CAP_FIBER, 542
PORT_CAP_FLOW_CTRL, 541
PORT_CAP_HDX, 544
PORT_CAP_NONE, 539
PORT_CAP_SD_ENABLE, 542
PORT_CAP_SD_HIGH, 542
PORT_CAP_SD_INTERNAL, 543
PORT_CAP_SFP_1G, 547
PORT_CAP_SFP_2_5G, 548
PORT_CAP_SFP_DETECT, 543
PORT_CAP_SFP_ONLY, 544
PORT_CAP_SFP_SD_HIGH, 548
PORT_CAP_SPEED_DUAL_ANY_FIBER_FIXE↩ D_SPEED, 546
PORT_CAP_SPEED_DUAL_ANY_FIBER, 547
PORT_CAP_STACKING, 544
PORT_CAP_TRI_SPEED_COPPER, 545
PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER_↩ FIXED_SFP_SPEED, 547
PORT_CAP_TRI_SPEED_DUAL_ANY_FIBER, 547
PORT_CAP_TRI_SPEED_DUAL_COPPER, 546
PORT_CAP_TRI_SPEED_DUAL_FIBER_100FX, 546
PORT_CAP_TRI_SPEED_DUAL_FIBER, 546
PORT_CAP_TRI_SPEED_FDX, 545
PORT_CAP_TRI_SPEED_FIBER, 545
PORT_CAP_TRI_SPEED, 545
PORT_CAP_VTSS_10G_PHY, 543
PORT_CAP_XAUI_LANE_FLIP, 543
port_cap_t, 549
vtss_fiber_port_speed_t, 549
vtss_port_speed_t, 549
port_action
vtss_acl_action_t, 70
port_cap_t
port.h, 549
port_cnt
vtss_phy_type_t, 382
port_conf
vtss_sgpio_conf_t, 457
port_cpu
vtss_mce_key_t, 168
port_custom_conf_t, 37
adv_dis, 39
autoneg, 38
dual_media_fiber_speed, 39
enable, 37
exc_col_cont, 39
fdx, 38
flow_control, 38
frame_length_chk, 40
max_length, 39
max_tags, 40
oper_up, 40
pfc, 38
power_mode, 39
speed, 38
port_list
vtss_ace_t, 64
vtss_acl_action_t, 70
vtss_debug_info_t, 83
vtss_ece_key_t, 99
vtss_mce_action_t, 164
vtss_mce_key_t, 168
vtss_qce_key_t, 430
vtss_vce_key_t, 495
port_mask
vtss_ts_timestamp_alloc_t, 472
port_no
vtss_eps_port_conf_t, 113
vtss_hqos_conf_t, 137
vtss_mirror_conf_t, 176
vtss_npi_conf_t, 193
vtss_oam_ts_timestamp_s, 217
vtss_packet_frame_info_t, 256
vtss_packet_port_info_t, 259
vtss_packet_rx_header_t, 262
vtss_packet_rx_info_t, 265
vtss_synce_clock_in_t, 461
port_tx
vtss_packet_frame_info_t, 257
port_type
vtss_vlan_port_conf_t, 503
ports
vtss_vlan_trans_port2grp_conf_t, 508
power_down
vtss_port_conf_t, 393
power_mode
port_custom_conf_t, 39
ppr
vtss_fan_conf_t, 124
pps_ls_lpbk

vtss_phy_ts_alt_clock_mode_s, 330
pps_offset
vtss_phy_ts_pps_config_s, 369
pps_output_enable
vtss_phy_ts_pps_config_s, 369
pps_width_adj
vtss_phy_ts_pps_config_s, 369
pr
vtss_phy_10g_fifo_sync_t, 296
pre_cb
vtss_fdma_tx_info_t, 135
pre_cb_ctxt1
vtss_fdma_tx_info_t, 135
pre_cb_ctxt2
vtss_fdma_tx_info_t, 135
prefix_size
vtss_ip_network_t, 146
vtss_ipv4_network_t, 147
vtss_ipv6_network_t, 149
prev_version
vtss_restart_status_t, 451
prio
vtss_ece_action_t, 90
vtss_fdma_ch_cfg_t, 132
vtss_mce_action_t, 166
vtss_qce_action_t, 421
prio_enable
vtss_ece_action_t, 90
vtss_mce_action_t, 166
vtss_qce_action_t, 421
priority_err
vtss_oam_ccm_status_t, 196
priority_mask
vtss_oam_voe_lm_counter_conf_t, 238
prios
vtss_qos_conf_t, 437
proc
vtss_oam_voe_conf_t, 227
process_cw
vtss_mpls_pw_conf_t, 181
prop
vtss_port_counters_t, 397
proto
vtss_ace_frame_ipv4_t, 50
vtss_ace_frame_ipv6_t, 54
vtss_ece_frame_ipv4_t, 93
vtss_ece_frame_ipv6_t, 94
vtss_qce_frame_ipv4_t, 425
vtss_qce_frame_ipv6_t, 427
vtss_vce_frame_ipv4_t, 490
vtss_vce_frame_ipv6_t, 492
proto_id
vtss_phy_ts_ach_conf_t, 329
ptp
vtss_ace_frame_etype_t, 48
vtss_ace_frame_ipv4_t, 53
vtss_ace_frame_ipv6_t, 57
vtss_phy_ts_engine_flow_conf_t, 335
ptp_action
vtss_acl_action_t, 71
vtss_packet_tx_info_t, 289
ptp_conf
vtss_phy_ts_engine_action_t, 333
vtss_phy_ts_ptp_engine_action_t, 372
ptp_id
vtss_packet_tx_info_t, 290
ptp_timestamp
vtss_packet_tx_info_t, 290
pvid
vtss_vlan_port_conf_t, 504
pw
vtss_evc_mpls_tp_conf_t, 118
pw_conf
vtss_mpls_segment_t, 185
pw_num
vtss_evc_mpls_pw_info_t, 118

qos
vtss_mpls_tc_to_qos_map_entry_t, 187
qos_class_map
vtss_qos_port_conf_t, 444
qos_to_tc_map
vtss_mpls_tc_conf_t, 187
qsgmii
vtss_serdes_macro_conf_t, 455
queue
vtss_packet_rx_conf_t, 260
queue_mask
vtss_packet_rx_header_t, 262
queue_pct
vtss_hqos_conf_t, 138
vtss_qos_port_conf_t, 447

r
vtss_vcap_vr_t, 487
range
vtss_phy_ts_eth_conf_t, 340, 341
vtss_phy_ts_ptp_conf_t, 371
vtss_phy_ts_y1731_oam_conf_t, 380
range_en
vtss_phy_ts_ptp_conf_t, 370
vtss_phy_ts_y1731_oam_conf_t, 379
rate
vtss_acl_policer_conf_t, 72
vtss_policer_t, 388
vtss_shaper_t, 460
raw_ident
vtss_irq_status_t, 153
raw_mask
vtss_irq_status_t, 153
raw_status
vtss_irq_status_t, 153
red_v2
vtss_qos_conf_t, 441
reg
vtss_packet_rx_conf_t, 260
reg_read

vtss_init_conf_t, 140
reg_write
vtss_init_conf_t, 140
remote_fault
vtss_phy_media_serd_pcs_cntl_t, 316
vtss_port_clause_37_adv_t, 390
vtss_port_status_t, 419
replaced
vtss_mac_table_status_t, 162
req
vtss_ace_frame_arp_t, 45
restart
vtss_phy_mac_serd_pcs_cntl_t, 313
vtss_restart_status_t, 451
restart_info_port
vtss_init_conf_t, 142
restart_info_src
vtss_init_conf_t, 142
revision
vtss_chip_id_t, 80
vtss_phy_type_t, 381
rgmii
vtss_phy_reset_conf_t, 320
rgmii_skew_delay_psec_t
vtss_phy_api.h, 838
rmon
vtss_port_counters_t, 396
route
vtss_routing_entry_t, 453
rule
vtss_ece_action_t, 88
vtss_mce_action_t, 165
rx
vtss_oam_rx_tx_counter_t, 213
rx_FCl
vtss_oam_lm_counter_t, 201
rx_alloc_cb
vtss_fdma_cfg_t, 125
rx_allow_multiple_dcbs
vtss_fdma_cfg_t, 127
rx_allow_vlan_tag_mismatch
vtss_fdma_cfg_t, 127
rx_buf_cnt
vtss_fdma_cfg_t, 125
rx_cb
vtss_fdma_cfg_t, 127
rx_ccm_lm_sample_seq_no
vtss_oam_lm_counter_t, 202
rx_ccm_seq_no
vtss_oam_proc_status_t, 211
rx_clk_skew_ps
vtss_phy_rgmii_conf_t, 322
rx_discard
vtss_evc_counters_t, 116
vtss_mpls_xc_counters_t, 189
rx_dmac_err_seen
vtss_oam_proc_status_t, 212
rx_dont_reinsert_vlan_tag
vtss_fdma_cfg_t, 127
rx_dont_strip_vlan_tag
vtss_fdma_cfg_t, 126
rx_err_cnt_base_tx
vtss_phy_statistic_t, 324
rx_etherStatsBroadcastPkts
vtss_port_rmon_counters_t, 409
rx_etherStatsCRCAlignErrors
vtss_port_rmon_counters_t, 410
rx_etherStatsDropEvents
vtss_port_rmon_counters_t, 409
rx_etherStatsFragments
vtss_port_rmon_counters_t, 410
rx_etherStatsJabbers
vtss_port_rmon_counters_t, 410
rx_etherStatsMulticastPkts
vtss_port_rmon_counters_t, 409
rx_etherStatsOctets
vtss_port_rmon_counters_t, 409
rx_etherStatsOversizePkts
vtss_port_rmon_counters_t, 410
rx_etherStatsPkts
vtss_port_rmon_counters_t, 409
rx_etherStatsPkts1024to1518Octets
vtss_port_rmon_counters_t, 412
rx_etherStatsPkts128to255Octets
vtss_port_rmon_counters_t, 411
rx_etherStatsPkts1519toMaxOctets
vtss_port_rmon_counters_t, 412
rx_etherStatsPkts256to511Octets
vtss_port_rmon_counters_t, 411
rx_etherStatsPkts512to1023Octets
vtss_port_rmon_counters_t, 411
rx_etherStatsPkts64Octets
vtss_port_rmon_counters_t, 411
rx_etherStatsPkts65to127Octets
vtss_port_rmon_counters_t, 411
rx_etherStatsUndersizePkts
vtss_port_rmon_counters_t, 410
rx_frames
vtss_basic_counters_t, 79
rx_green
vtss_evc_counters_t, 115
vtss_mpls_xc_counters_t, 188
rx_info
tag_vtss_fdma_list, 43
rx_invalid_count
vtss_oam_ccm_counter_t, 194
rx_invalid_seq_no
vtss_oam_ccm_counter_t, 195
rx_lbr
vtss_oam_lb_counter_t, 199
rx_lbr_trans_id_err
vtss_oam_lb_counter_t, 200
rx_lbr_transaction_id
vtss_oam_proc_status_t, 211
rx_lmm_sample_seq_no
vtss_oam_lm_counter_t, 201

rx_lmr_sample_seq_no
vtss_oam_lm_counter_t, 201
rx_meg_level_err
vtss_oam_proc_status_t, 212
rx_meg_level_err_seen
vtss_oam_proc_status_t, 212
rx_mtu
vtss_fdma_cfg_t, 125
rx_period
vtss_oam_voe_ccm_conf_t, 222
rx_prio
vtss_port_proprietary_counters_t, 407
rx_priority
vtss_oam_voe_ccm_conf_t, 222
rx_queue
vtss_oam_vop_extract_conf_t, 250
rx_rdi
vtss_oam_ccm_status_t, 196
rx_red
vtss_evc_counters_t, 115
vtss_mpls_xc_counters_t, 189
rx_seq_no
vtss_oam_voe_ccm_conf_t, 221
vtss_oam_voe_tst_conf_t, 244
rx_seq_no_check
vtss_oam_voe_ccm_conf_t, 221
rx_transaction_id
vtss_oam_voe_lb_conf_t, 237
rx_ts_len
vtss_phy_ts_init_conf_t, 353
rx_ts_pos
vtss_phy_ts_init_conf_t, 353
rx_tst
vtss_oam_tst_counter_t, 218
rx_tst_seq_no
vtss_oam_proc_status_t, 212
rx_tst_trans_id_err
vtss_oam_tst_counter_t, 218
rx_valid_count
vtss_oam_ccm_counter_t, 194
rx_yellow
vtss_evc_counters_t, 115
vtss_mpls_xc_counters_t, 189

s_etype
vtss_vlan_conf_t, 503
vtss_vlan_port_conf_t, 504
s_tag
vtss_qce_tag_t, 436
vtss_vce_tag_t, 500
s_tagged
vtss_ece_tag_t, 107
vtss_mce_tag_t, 174
sampling_rate
vtss_sflow_port_conf_t, 456
save
vtss_ts_alt_clock_mode_t, 468
sch_mode
vtss_hqos_port_conf_t, 139
sd_active_high
vtss_port_conf_t, 393
sd_enable
vtss_port_conf_t, 393
sd_internal
vtss_port_conf_t, 393
sdlb_cpy_copy_idx
vtss_oam_vop_conf_t, 249
sdlb_enable
vtss_oam_voe_conf_t, 226
sdlb_idx
vtss_oam_voe_conf_t, 227
sec
vtss_timeofday_t, 465
sec_msb
vtss_timestamp_t, 466
seconds
vtss_phy_timestamp_t, 327
vtss_timestamp_t, 466
section_oam
vtss_mpls_l2_t, 178
secure_on_enable
vtss_phy_wol_conf_t, 384
sel
vtss_oam_voe_counter_t, 230
select
vtss_eee_port_state_t, 111
selected_frames
vtss_oam_sel_counter_t, 214
self_mac
vtss_mpls_l2_t, 177
seq_zero
vtss_ace_frame_ipv4_t, 53
vtss_ace_frame_ipv6_t, 57
sequence_id
vtss_phy_ts_fifo_sig_t, 345
ser_led_frame_rate
vtss_phy_enhanced_led_control_t, 306
ser_led_output_1
vtss_phy_enhanced_led_control_t, 306
ser_led_output_2
vtss_phy_enhanced_led_control_t, 306
ser_led_select
vtss_phy_enhanced_led_control_t, 306
serdes
vtss_init_conf_t, 143
vtss_port_conf_t, 396
serdes1g_vdd
vtss_serdes_macro_conf_t, 454
serdes6g_vdd
vtss_serdes_macro_conf_t, 454
serdes_aneg_ena
vtss_phy_mac_serd_pcs_cntl_t, 314
serdes_fields_t, 40
ob_post0, 41
ob_sr, 41
serdes_pol_inv_in
vtss_phy_mac_serd_pcs_cntl_t, 315

vtss_phy_media_serd_pcs_cntl_t, 317
serdes_pol_inv_out
vtss_phy_mac_serd_pcs_cntl_t, 315
vtss_phy_media_serd_pcs_cntl_t, 317
serdes_tx_bad
vtss_phy_statistic_t, 323
serdes_tx_good
vtss_phy_statistic_t, 323
server_idx
vtss_mpls_segment_t, 186
service_detect
vtss_mce_key_t, 169
service_voe_idx
vtss_oam_ipt_conf_t, 198
sflow_port_no
vtss_packet_rx_info_t, 271
sflow_queue
vtss_packet_rx_queue_map_t, 281
sflow_type
vtss_packet_rx_info_t, 270
sfp_dac
vtss_port_serdes_conf_t, 415
sgmii_in_pre
vtss_phy_mac_serd_pcs_cntl_t, 314
sgmii_out_pre
vtss_phy_mac_serd_pcs_cntl_t, 314
shaper
vtss_hqos_conf_t, 137
shaper_port
vtss_qos_port_conf_t, 443
shaper_queue
vtss_hqos_conf_t, 138
vtss_qos_port_conf_t, 443
shaper_rate
vtss_packet_rx_conf_t, 261
sig_mask
vtss_phy_ts_fifo_sig_t, 344
sigdet
vtss_phy_conf_t, 302
single_ended_lm
vtss_oam_voe_conf_t, 228
sip
vtss_ace_frame_arp_t, 46
vtss_ace_frame_ipv4_t, 50
vtss_ace_frame_ipv6_t, 54
vtss_ace_sip_smac_t, 62
vtss_ece_frame_ipv4_t, 93
vtss_ece_frame_ipv6_t, 95
vtss_qce_frame_ipv4_t, 425
vtss_qce_frame_ipv6_t, 427
vtss_vce_frame_ipv4_t, 491
vtss_vce_frame_ipv6_t, 492
sip_dip_enable
vtss_aggr_mode_t, 76
sip_eq_dip
vtss_ace_frame_ipv4_t, 52
vtss_ace_frame_ipv6_t, 57
sip_smac
vtss_ace_frame_ipv4_t, 53
size
vtss_packet_rx_queue_conf_t, 279
skip_coma
vtss_phy_conf_t, 303
skip_rev_check
vtss_phy_10g_fifo_sync_t, 295
vtss_phy_ts_fifo_conf_t, 343
smac
vtss_ace_frame_arp_t, 45
vtss_ace_frame_etype_t, 48
vtss_ace_frame_llc_t, 58
vtss_ace_frame_snap_t, 60
vtss_ace_sip_smac_t, 62
vtss_ece_mac_t, 103
vtss_port_flow_control_conf_t, 399
vtss_qce_mac_t, 433
vtss_vce_mac_t, 497
smac_enable
vtss_aggr_mode_t, 75
smac_match
vtss_ace_frame_arp_t, 45
snap
vtss_ace_frame_snap_t, 60
vtss_ace_t, 66
vtss_ece_key_t, 101
vtss_qce_key_t, 431
vtss_vce_key_t, 496
snd_lvl_after_top
vtss_phy_ts_mpls_conf_t, 361
snd_lvl_before_end
vtss_phy_ts_mpls_conf_t, 362
speed
port_custom_conf_t, 38
vtss_phy_forced_t, 307
vtss_port_conf_t, 394
vtss_port_status_t, 418
speed_100M
vtss_port_sgmii_aneg_t, 417
speed_100m_fdx
vtss_phy_aneg_t, 297
speed_100m_hdx
vtss_phy_aneg_t, 297
speed_10M
vtss_port_sgmii_aneg_t, 417
speed_10m_fdx
vtss_phy_aneg_t, 297
speed_10m_hdx
vtss_phy_aneg_t, 297
speed_1G
vtss_port_sgmii_aneg_t, 417
speed_1g_fdx
vtss_phy_aneg_t, 297
speed_1g_hdx
vtss_phy_aneg_t, 297
spi_32bit_read_write
vtss_init_conf_t, 142
spi_64bit_read_write

vtss_init_conf_t, 142
spi_daisy_input
vtss_phy_daisy_chain_conf_t, 304
spi_daisy_output
vtss_phy_daisy_chain_conf_t, 304
spi_read_write
vtss_init_conf_t, 141
split_horizon
vtss_evc_mpls_pw_info_t, 117
sport
vtss_ace_frame_ipv4_t, 51
vtss_ace_frame_ipv6_t, 55
vtss_ece_frame_ipv4_t, 93
vtss_ece_frame_ipv6_t, 95
vtss_qce_frame_ipv4_t, 425
vtss_qce_frame_ipv6_t, 427
sport_dport_enable
vtss_aggr_mode_t, 76
sport_eq_dport
vtss_ace_frame_ipv4_t, 53
vtss_ace_frame_ipv6_t, 57
sport_mask
vtss_phy_ts_ip_conf_t, 357
sport_val
vtss_phy_ts_ip_conf_t, 356
squelch
vtss_phy_clock_conf_t, 299
squelsh
vtss_synce_clock_in_t, 461
src
vtss_phy_clock_conf_t, 299
src_ip
vtss_phy_ts_fifo_sig_t, 345
src_port_identity
vtss_phy_ts_fifo_sig_t, 345
stack_depth
vtss_phy_ts_mpls_conf_t, 360
stack_level
vtss_phy_ts_mpls_conf_t, 362
stack_queue
vtss_packet_rx_queue_map_t, 281
stack_ref_point
vtss_phy_ts_mpls_conf_t, 360
state
vtss_mpls_segment_status_t, 182
status
vtss_phy_veriphy_result_t, 383
stripped_tag
vtss_packet_rx_info_t, 267
suspend_tick_cnt
vtss_fdma_throttle_cfg_t, 134
svc_to_path
vtss_oam_voe_conf_t, 225
svc_to_path_idx_p
vtss_oam_voe_conf_t, 226
svc_to_path_idx_w
vtss_oam_voe_conf_t, 225
sw_tstamp
tag_vtss_fdma_list, 43
vtss_packet_rx_info_t, 269
vtss_packet_rx_meta_t, 276
switch_frm
vtss_packet_tx_info_t, 286
symmetric_pause
vtss_phy_aneg_t, 298
vtss_port_clause_37_adv_t, 390

TESLA_ING_TS_ERRFIX
options.h, 528
TRUE
types.h, 559
tag
vtss_ece_key_t, 100
vtss_mce_key_t, 168
vtss_packet_rx_header_t, 263
vtss_packet_rx_info_t, 266
vtss_packet_tx_info_t, 287
vtss_qce_key_t, 430
vtss_vce_key_t, 495
tag_class_enable
vtss_qos_port_conf_t, 444
tag_default_dei
vtss_qos_port_conf_t, 446
tag_default_pcp
vtss_qos_port_conf_t, 446
tag_dei_map
vtss_qos_port_conf_t, 446
tag_pcp_map
vtss_qos_port_conf_t, 446
tag_range_mode
vtss_phy_ts_eth_conf_t, 340
tag_remark_mode
vtss_qos_port_conf_t, 446
tag_type
vtss_mpls_l2_t, 177
vtss_packet_rx_info_t, 266
tag_vtss_fdma_list, 41
act_len, 42
alloc_ptr, 43
frm_ptr, 42
next, 44
rx_info, 43
sw_tstamp, 43
user, 43
tagged
vtss_ace_vlan_t, 68
vtss_ece_tag_t, 107
vtss_mce_tag_t, 174
vtss_qce_tag_t, 436
vtss_vce_tag_t, 500
tagprio
vtss_tci_t, 464
target
vtss_inst_create_t, 144
tbi
vtss_phy_reset_conf_t, 321
tc

vtss_mpls_label_t, 179
tc_op_mode
vtss_phy_ts_init_conf_t, 354
tc_qos_map_idx
vtss_mpls_segment_t, 184
tc_to_qos_map
vtss_mpls_tc_conf_t, 187
tc_tunnel_mode
vtss_mpls_xc_t, 191
tcp_ack
vtss_ace_frame_ipv4_t, 52
vtss_ace_frame_ipv6_t, 56
tcp_fin
vtss_ace_frame_ipv4_t, 51
vtss_ace_frame_ipv6_t, 56
tcp_psh
vtss_ace_frame_ipv4_t, 52
vtss_ace_frame_ipv6_t, 56
tcp_rst
vtss_ace_frame_ipv4_t, 52
vtss_ace_frame_ipv6_t, 56
tcp_syn
vtss_ace_frame_ipv4_t, 51
vtss_ace_frame_ipv6_t, 56
tcp_urg
vtss_ace_frame_ipv4_t, 52
vtss_ace_frame_ipv6_t, 57
thrd_lvl_after_top
vtss_phy_ts_mpls_conf_t, 361
thrd_lvl_before_end
vtss_phy_ts_mpls_conf_t, 362
timeout
vtss_mtimer_t, 192
to_front
vtss_oam_vop_extract_conf_t, 250
tod_get_ns_cnt_cb_t
vtss_misc_api.h, 706
top
vtss_phy_ts_mpls_conf_t, 361
top_down
vtss_phy_ts_mpls_conf_t, 361
tpid
vtss_packet_port_filter_t, 258
vtss_phy_ts_eth_conf_t, 338
vtss_vlan_tag_t, 505
trans_vid
vtss_vlan_trans_grp2vlan_conf_t, 507
ts
vtss_oam_ts_timestamp_s, 216
vtss_ts_timestamp_t, 473
ts_fifo_drop_cnt
vtss_phy_ts_stats_t, 378
ts_fifo_tx_cnt
vtss_phy_ts_stats_t, 378
ts_format
vtss_phy_ts_ietf_mpls_ach_oam_conf_t, 352
ts_id
vtss_ts_id_t, 470
ts_offset
vtss_phy_ts_generic_action_t, 350
ts_type
vtss_phy_ts_generic_action_t, 349
ts_valid
vtss_oam_ts_timestamp_s, 217
vtss_ts_timestamp_t, 474
tst
vtss_oam_voe_conf_t, 228
vtss_oam_voe_counter_t, 230
tstamp_id
vtss_packet_rx_info_t, 269
tstamp_id_decoded
vtss_packet_rx_info_t, 269
tt_loop
vtss_mce_t, 173
ttl
vtss_ace_frame_ipv4_t, 49
vtss_ace_frame_ipv6_t, 55
vtss_mpls_label_t, 179
ttl_tunnel_mode
vtss_mpls_xc_t, 191
tx
vtss_oam_rx_tx_counter_t, 214
tx_FCf
vtss_oam_lm_counter_t, 201
tx_buf_cnt
vtss_fdma_cfg_t, 128
tx_clk_skew_ps
vtss_phy_rgmii_conf_t, 322
tx_discard
vtss_evc_counters_t, 116
vtss_mpls_xc_counters_t, 189
tx_done_cb
vtss_fdma_cfg_t, 128
tx_etherStatsBroadcastPkts
vtss_port_rmon_counters_t, 413
tx_etherStatsCollisions
vtss_port_rmon_counters_t, 413
tx_etherStatsDropEvents
vtss_port_rmon_counters_t, 412
tx_etherStatsMulticastPkts
vtss_port_rmon_counters_t, 413
tx_etherStatsOctets
vtss_port_rmon_counters_t, 412
tx_etherStatsPkts
vtss_port_rmon_counters_t, 412
tx_etherStatsPkts1024to1518Octets
vtss_port_rmon_counters_t, 414
tx_etherStatsPkts128to255Octets
vtss_port_rmon_counters_t, 414
tx_etherStatsPkts1519toMaxOctets
vtss_port_rmon_counters_t, 414
tx_etherStatsPkts256to511Octets
vtss_port_rmon_counters_t, 414
tx_etherStatsPkts512to1023Octets
vtss_port_rmon_counters_t, 414
tx_etherStatsPkts64Octets

vtss_port_rmon_counters_t, 413
tx_etherStatsPkts65to127Octets
vtss_port_rmon_counters_t, 413
tx_fifo_hi_clk_cycs
vtss_phy_ts_init_conf_t, 354
tx_fifo_lo_clk_cycs
vtss_phy_ts_init_conf_t, 354
tx_fifo_mode
vtss_phy_ts_init_conf_t, 353
tx_fifo_spi_conf
vtss_phy_ts_init_conf_t, 354
tx_frames
vtss_basic_counters_t, 79
tx_green
vtss_evc_counters_t, 116
vtss_mpls_xc_counters_t, 189
tx_lbm
vtss_oam_lb_counter_t, 199
tx_lookup
vtss_ece_action_t, 88
vtss_mce_action_t, 165
tx_meg_level_err_seen
vtss_oam_proc_status_t, 213
tx_next_ccm_seq_no
vtss_oam_proc_status_t, 211
tx_next_lbm_transaction_id
vtss_oam_proc_status_t, 212
tx_out_bytes
vtss_eee_port_counter_t, 111
tx_out_bytes_get
vtss_eee_port_counter_t, 110
tx_prio
vtss_port_proprietary_counters_t, 408
tx_remote_fault
vtss_phy_aneg_t, 298
tx_seq_no
vtss_oam_voe_ccm_conf_t, 221
vtss_oam_voe_tst_conf_t, 244
tx_seq_no_auto_upd_op
vtss_oam_voe_ccm_conf_t, 221
tx_seq_no_auto_update
vtss_oam_voe_tst_conf_t, 244
tx_transaction_id
vtss_oam_voe_lb_conf_t, 237
tx_ts_len
vtss_phy_ts_init_conf_t, 354
tx_tst
vtss_oam_tst_counter_t, 218
tx_tw
vtss_eee_port_conf_t, 109
tx_update_transaction_id
vtss_oam_voe_lb_conf_t, 237
tx_yellow
vtss_evc_counters_t, 116
vtss_mpls_xc_counters_t, 190
type
vtss_ace_t, 64
vtss_dlb_policer_conf_t, 86
vtss_ece_inner_tag_t, 97
vtss_ece_key_t, 100
vtss_eps_port_conf_t, 112
vtss_fan_conf_t, 123
vtss_ip_addr_t, 145
vtss_mpls_oam_conf_t, 180
vtss_mpls_xc_t, 191
vtss_qce_key_t, 431
vtss_routing_entry_t, 452
vtss_sflow_port_conf_t, 456
vtss_vcap_vr_t, 486
vtss_vce_key_t, 495
types.h
BOOL, 578
FALSE, 559
i16, 576
i32, 577
i64, 577
i8, 576
MAC_ADDR_BROADCAST, 567
PRIi64, 558
PRIu64, 557
PRIx64, 558
TRUE, 559
u16, 577
u32, 577
u64, 578
u8, 577
uintptr_t, 578
VTSS_ACL_POLICER_NO_END, 572
VTSS_ACL_POLICER_NO_START, 572
VTSS_ACL_POLICERS, 572
VTSS_ACL_POLICIES, 573
VTSS_ACL_POLICY_NO_END, 573
VTSS_ACL_POLICY_NO_MAX, 573
VTSS_ACL_POLICY_NO_MIN, 572
VTSS_ACL_POLICY_NO_NONE, 572
VTSS_ACL_POLICY_NO_START, 573
VTSS_AGGR_NO_END, 569
VTSS_AGGR_NO_NONE, 568
VTSS_AGGR_NO_START, 568
VTSS_AGGRS, 568
VTSS_BIT64, 558
VTSS_BITMASK64, 558
VTSS_BITRATE_DISABLED, 566
VTSS_CLOCK_IDENTITY_LENGTH, 576
VTSS_DEI_ARRAY_SIZE, 565
VTSS_DEI_END, 565
VTSS_DEI_START, 564
VTSS_DEIS, 564
VTSS_DPL_ARRAY_SIZE, 566
VTSS_DPL_END, 565
VTSS_DPL_START, 565
VTSS_DPLS, 565
VTSS_ENCODE_BITFIELD64, 559
VTSS_ENCODE_BITMASK64, 559
VTSS_ETYPE_VTSS, 567
VTSS_EVCS, 568

VTSS_EXTRACT_BITFIELD64, 558
VTSS_GLAG_NO_END, 569
VTSS_GLAG_NO_NONE, 569
VTSS_GLAG_NO_START, 569
VTSS_GLAG_PORT_ARRAY_SIZE, 570
VTSS_GLAG_PORT_END, 570
VTSS_GLAG_PORT_START, 570
VTSS_GLAG_PORTS, 570
VTSS_GLAGS, 569
VTSS_HQOS_COUNT, 573
VTSS_HQOS_ID_NONE, 574
VTSS_INTERVAL_MS, 575
VTSS_INTERVAL_NS, 575
VTSS_INTERVAL_PS, 575
VTSS_INTERVAL_SEC, 574
VTSS_INTERVAL_US, 575
VTSS_ISDX_NONE, 568
VTSS_MAC_ADDR_SZ_BYTES, 567
VTSS_MAX_TIMEINTERVAL, 574
VTSS_ONE_MILL, 574
VTSS_ONE_MIA, 574
VTSS_PACKET_RATE_DISABLED, 560
VTSS_PACKET_RX_GRP_CNT, 571
VTSS_PACKET_RX_QUEUE_CNT, 570
VTSS_PACKET_RX_QUEUE_END, 571
VTSS_PACKET_RX_QUEUE_NONE, 571
VTSS_PACKET_RX_QUEUE_START, 571
VTSS_PACKET_TX_GRP_CNT, 571
VTSS_PCP_ARRAY_SIZE, 564
VTSS_PCP_END, 564
VTSS_PCP_START, 564
VTSS_PCPS, 563
VTSS_PORT_ARRAY_SIZE, 561
VTSS_PORT_COUNT, 560
VTSS_PORT_IS_PORT, 561
VTSS_PORT_NO_CPU, 561
VTSS_PORT_NO_END, 561
VTSS_PORT_NO_NONE, 560
VTSS_PORT_NO_START, 561
VTSS_PORTS, 560
VTSS_PRIO_ARRAY_SIZE, 562
VTSS_PRIO_END, 562
VTSS_PRIO_NO_NONE, 562
VTSS_PRIO_START, 562
VTSS_PRIOS, 562
VTSS_QUEUE_ARRAY_SIZE, 563
VTSS_QUEUE_END, 563
VTSS_QUEUE_START, 563
VTSS_QUEUES, 563
VTSS_SEC_NS_INTERVAL, 575
VTSS_SYNCE_CLK_PORT_ARRAY_SIZE, 576
VTSS_VID_ALL, 567
VTSS_VID_DEFAULT, 566
VTSS_VID_NULL, 566
VTSS_VID_RESERVED, 566
VTSS_VIDS, 567
vtss_ece_dei_mode_t, 589
vtss_ece_dir_t, 588
vtss_ece_inner_tag_type_t, 590
vtss_ece_pcp_mode_t, 589
vtss_ece_pop_tag_t, 588
vtss_ece_rule_t, 588
vtss_ece_tx_lookup_t, 589
vtss_hqos_sch_mode_t, 590
vtss_ip_type_t, 586
vtss_isdx_t, 578
vtss_mac_addr_t, 578
vtss_mem_flags_t, 582
vtss_packet_reg_type_t, 586
vtss_packet_rx_grp_t, 579
vtss_packet_tx_grp_t, 579
vtss_policer_type_t, 584
vtss_port_interface_t, 583
vtss_serdes_mode_t, 583
vtss_storm_policer_mode_t, 584
vtss_vcap_bit_t, 587
vtss_vcap_key_type_t, 587
vtss_vcap_vr_type_t, 587
vtss_vdd_t, 586
vtss_vlan_frame_t, 584

u16
types.h, 577
u32
types.h, 577
u64
types.h, 578
u8
types.h, 577
uint
vtss_os_custom.h, 771
uintptr_t
types.h, 578
ulong
vtss_os_custom.h, 771
unicast_mac
vtss_oam_voe_conf_t, 225
unidir
vtss_phy_conf_t, 302
unidirectional_ability
vtss_port_status_t, 419
unknown
vtss_ace_frame_arp_t, 45
vtss_oam_voe_conf_t, 227
unknown_seen
vtss_oam_pdu_seen_status_t, 206
untagged_vid
vtss_vlan_port_conf_t, 504
up_mep
vtss_oam_lm_counter_t, 204
upmep
vtss_oam_voe_conf_t, 229
upper
vtss_phy_ts_eth_conf_t, 340
vtss_phy_ts_mpls_lvl_rng_t, 363
vtss_phy_ts_ptp_conf_t, 371
vtss_phy_ts_y1731_oam_conf_t, 380

upstream
vtss_mpls_segment_t, 185
usage
vtss_fdma_ch_cfg_t, 130
user
tag_vtss_fdma_list, 43
usr_prio
vtss_ace_vlan_t, 67
vtss_qos_port_conf_t, 444

v
vtss_vcap_vr_t, 487
VIPER_B_FIFO_RESET
options.h, 528
VTSS_ACE_ID_LAST
vtss_security_api.h, 986
VTSS_ACL_POLICER_NO_END
types.h, 572
VTSS_ACL_POLICER_NO_START
types.h, 572
VTSS_ACL_POLICERS
types.h, 572
VTSS_ACL_POLICIES
types.h, 573
VTSS_ACL_POLICY_NO_END
types.h, 573
VTSS_ACL_POLICY_NO_MAX
types.h, 573
VTSS_ACL_POLICY_NO_MIN
types.h, 572
VTSS_ACL_POLICY_NO_NONE
types.h, 572
VTSS_ACL_POLICY_NO_START
types.h, 573
VTSS_AFI_FPS_MAX
vtss_fdma_api.h, 615
vtss_packet_api.h, 796
VTSS_AFI_ID_NONE
vtss_packet_api.h, 796
VTSS_AFI_SLOT_CNT
vtss_packet_api.h, 796
VTSS_AFI_V1
options.h, 533
VTSS_AGGR_NO_END
types.h, 569
VTSS_AGGR_NO_NONE
types.h, 568
VTSS_AGGR_NO_START
types.h, 568
VTSS_AGGRS
types.h, 568
VTSS_ARCH_SERVAL_CPU
options.h, 514
VTSS_ARCH_SERVAL_CE
options.h, 514
VTSS_ARCH_SERVAL
options.h, 514
VTSS_BIT64
types.h, 558
VTSS_BITMASK64
types.h, 558
VTSS_BITRATE_DISABLED
types.h, 566
VTSS_CHIP_CU_PHY
options.h, 533
VTSS_CLOCK_IDENTITY_LENGTH
types.h, 576
VTSS_DEI_ARRAY_SIZE
types.h, 565
VTSS_DEI_END
types.h, 565
VTSS_DEI_START
types.h, 564
VTSS_DEIS
types.h, 564
VTSS_DIV64
vtss_os_custom.h, 772
vtss_os_ecos.h, 777
vtss_os_linux.h, 789
VTSS_DPL_ARRAY_SIZE
types.h, 566
VTSS_DPL_END
types.h, 565
VTSS_DPL_START
types.h, 565
VTSS_DPLS
types.h, 565
VTSS_ECE_ID_LAST
vtss_evc_api.h, 597
VTSS_ENCODE_BITFIELD64
types.h, 559
VTSS_ENCODE_BITMASK64
types.h, 559
VTSS_ERPI_ARRAY_SIZE
vtss_l2_api.h, 657
VTSS_ERPI_END
vtss_l2_api.h, 657
VTSS_ERPI_START
vtss_l2_api.h, 657
VTSS_ERPIS
vtss_l2_api.h, 657
VTSS_ETYPE_VTSS
types.h, 567
VTSS_EVC_ID_NONE
vtss_evc_api.h, 596
VTSS_EVC_MPLS_PW_CNT
vtss_evc_api.h, 596
VTSS_EVC_POLICER_ID_DISCARD
vtss_evc_api.h, 596
VTSS_EVC_POLICER_ID_EVC
vtss_evc_api.h, 596
VTSS_EVC_POLICER_ID_NONE
vtss_evc_api.h, 596
VTSS_EVC_POLICERS
vtss_evc_api.h, 596
VTSS_EVCS
types.h, 568

VTSS_EXTRACT_BITFIELD64
types.h, 558
VTSS_FDMA_CH_CNT
vtss_fdma_api.h, 612
VTSS_FDMA_DCB_SIZE_BYTES
vtss_fdma_api.h, 613
VTSS_FDMA_HDR_SIZE_BYTES
vtss_fdma_api.h, 613
VTSS_FDMA_MAX_DATA_PER_DCB_BYTES
vtss_fdma_api.h, 613
VTSS_FDMA_MAX_FRAME_SIZE_BYTES
vtss_fdma_api.h, 614
VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BY↩
TES
vtss_fdma_api.h, 613
VTSS_FDMA_MIN_DATA_PER_NON_SOF_DCB_B↩
YTES
vtss_fdma_api.h, 614
VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_B↩
YTES
vtss_fdma_api.h, 614
VTSS_FDMA_MIN_FRAME_SIZE_BYTES
vtss_fdma_api.h, 613
VTSS_FEATURE_ACL_V2
options.h, 526
VTSS_FEATURE_ACL
options.h, 526
VTSS_FEATURE_AFI_SWC
options.h, 532
VTSS_FEATURE_CLAUSE_37
options.h, 515
VTSS_FEATURE_E_TREE
options.h, 532
VTSS_FEATURE_EEE
options.h, 525
VTSS_FEATURE_EVC
options.h, 532
VTSS_FEATURE_EXC_COL_CONT
options.h, 516
VTSS_FEATURE_FAN
options.h, 530
VTSS_FEATURE_FDMA
options.h, 532
VTSS_FEATURE_HQOS
options.h, 533
VTSS_FEATURE_INTERRUPTS
options.h, 529
VTSS_FEATURE_IRQ_CONTROL
options.h, 527
VTSS_FEATURE_LAYER2
options.h, 524
VTSS_FEATURE_LED_POW_REDUC
options.h, 527
VTSS_FEATURE_MAC_AGE_AUTO
options.h, 525
VTSS_FEATURE_MAC_CPU_QUEUE
options.h, 525
VTSS_FEATURE_MIRROR_CPU
options.h, 527
VTSS_FEATURE_MISC
options.h, 515
VTSS_FEATURE_MPLS
options.h, 533
VTSS_FEATURE_NPI
options.h, 526
VTSS_FEATURE_OAM
options.h, 532
VTSS_FEATURE_PACKET_GROUPING
options.h, 524
VTSS_FEATURE_PACKET_PORT_REG
options.h, 524
VTSS_FEATURE_PACKET_RX
options.h, 523
VTSS_FEATURE_PACKET_TX
options.h, 523
VTSS_FEATURE_PACKET
options.h, 523
VTSS_FEATURE_PFC
options.h, 516
VTSS_FEATURE_PHY_TIMESTAMP
options.h, 530
VTSS_FEATURE_PORT_CNT_BRIDGE
options.h, 516
VTSS_FEATURE_PORT_CONTROL
options.h, 515
VTSS_FEATURE_PORT_IFH
options.h, 515
VTSS_FEATURE_PTP_RS422
options.h, 531
VTSS_FEATURE_PVLAN
options.h, 524
VTSS_FEATURE_QCL_DMAC_DIP
options.h, 517
VTSS_FEATURE_QCL_KEY_DIP
options.h, 518
VTSS_FEATURE_QCL_KEY_DMAC
options.h, 518
VTSS_FEATURE_QCL_KEY_INNER_TAG
options.h, 517
VTSS_FEATURE_QCL_KEY_S_TAG
options.h, 517
VTSS_FEATURE_QCL_KEY_TYPE
options.h, 517
VTSS_FEATURE_QCL_PCP_DEI_ACTION
options.h, 518
VTSS_FEATURE_QCL_POLICY_ACTION
options.h, 518
VTSS_FEATURE_QCL_V2
options.h, 517
VTSS_FEATURE_QCL
options.h, 516
VTSS_FEATURE_QOS_CLASSIFICATION_V2
options.h, 521
VTSS_FEATURE_QOS_CPU_PORT_SHAPER
options.h, 523
VTSS_FEATURE_QOS_DSCP_CLASS_DP_AWARE

options.h, 522
VTSS_FEATURE_QOS_DSCP_REMARK_DP_AWA↩RE
options.h, 522
VTSS_FEATURE_QOS_DSCP_REMARK_V2
options.h, 522
VTSS_FEATURE_QOS_DSCP_REMARK
options.h, 522
VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPE↩RS_EB
options.h, 521
VTSS_FEATURE_QOS_EGRESS_QUEUE_SHAPE↩RS
options.h, 521
VTSS_FEATURE_QOS_EGRESS_SHAPERS_DLB
options.h, 521
VTSS_FEATURE_QOS_EGRESS_SHAPERS_RT
options.h, 521
VTSS_FEATURE_QOS_POLICER_BC_SWITCH
options.h, 519
VTSS_FEATURE_QOS_POLICER_DLB
options.h, 523
VTSS_FEATURE_QOS_POLICER_MC_SWITCH
options.h, 519
VTSS_FEATURE_QOS_POLICER_UC_SWITCH
options.h, 518
VTSS_FEATURE_QOS_PORT_POLICER_EXT_FPS
options.h, 519
VTSS_FEATURE_QOS_PORT_POLICER_EXT_FC
options.h, 519
VTSS_FEATURE_QOS_PORT_POLICER_EXT
options.h, 519
VTSS_FEATURE_QOS_QUEUE_POLICER
options.h, 520
VTSS_FEATURE_QOS_QUEUE_TX
options.h, 520
VTSS_FEATURE_QOS_SCHEDULER_DWRR_CNT
options.h, 520
VTSS_FEATURE_QOS_SCHEDULER_V2
options.h, 520
VTSS_FEATURE_QOS_TAG_REMARK_V2
options.h, 520
VTSS_FEATURE_QOS_WRED_V2
options.h, 522
VTSS_FEATURE_QOS
options.h, 516
VTSS_FEATURE_SERDES_MACRO_SETTINGS
options.h, 528, 529
VTSS_FEATURE_SERIAL_GPIO
options.h, 529
VTSS_FEATURE_SFLOW
options.h, 527
VTSS_FEATURE_SYNCE
options.h, 529
VTSS_FEATURE_TIMESTAMP_ASYMMETRY_COMP
options.h, 531
VTSS_FEATURE_TIMESTAMP_LATENCY_COMP
options.h, 530
VTSS_FEATURE_TIMESTAMP_ONE_STEP
options.h, 530
VTSS_FEATURE_TIMESTAMP_ORG_TIME
options.h, 531
VTSS_FEATURE_TIMESTAMP_P2P_DELAY_COMP
options.h, 531
VTSS_FEATURE_TIMESTAMP
options.h, 530
VTSS_FEATURE_VCAP
options.h, 526, 535
VTSS_FEATURE_VCL
options.h, 526
VTSS_FEATURE_VLAN_PORT_V2
options.h, 524
VTSS_FEATURE_VLAN_SVL
options.h, 525
VTSS_FEATURE_VLAN_TRANSLATION
options.h, 527
VTSS_FEATURE_VLAN_TX_TAG
options.h, 525
VTSS_FEATURE_WARM_START
options.h, 515, 535
VTSS_FRAME_GAP_DEFAULT
vtss_port_api.h, 963
VTSS_GLAG_NO_END
types.h, 569
VTSS_GLAG_NO_NONE
types.h, 569
VTSS_GLAG_NO_START
types.h, 569
VTSS_GLAG_PORT_ARRAY_SIZE
types.h, 570
VTSS_GLAG_PORT_END
types.h, 570
VTSS_GLAG_PORT_START
types.h, 570
VTSS_GLAG_PORTS
types.h, 570
VTSS_GLAGS
types.h, 569
VTSS_HQOS_COUNT
types.h, 573
VTSS_HQOS_ID_NONE
types.h, 574
VTSS_I2C_NO_MULTIPLEXER
vtss_init_api.h, 633
VTSS_INTERVAL_MS
types.h, 575
VTSS_INTERVAL_NS
types.h, 575
VTSS_INTERVAL_PS
types.h, 575
VTSS_INTERVAL_SEC
types.h, 574
VTSS_INTERVAL_US
types.h, 575
VTSS_ISDX_CPU_TX
vtss_evc_api.h, 597

VTSS_ISDX_NONE
types.h, 568
VTSS_JR1_PACKET_HDR_SIZE_BYTES
vtss_packet_api.h, 796
VTSS_JR1_RX_IFH_SIZE
vtss_packet_api.h, 797
VTSS_JR2_PACKET_HDR_SIZE_BYTES
vtss_packet_api.h, 796
VTSS_JR2_RX_IFH_SIZE
vtss_packet_api.h, 798
VTSS_L26_PACKET_HDR_SIZE_BYTES
vtss_packet_api.h, 797
VTSS_L26_RX_IFH_SIZE
vtss_packet_api.h, 798
VTSS_LABS
vtss_os_custom.h, 773
vtss_os_ecos.h, 778
vtss_os_linux.h, 789
VTSS_LLABS
vtss_os_custom.h, 773
vtss_os_ecos.h, 778
vtss_os_linux.h, 790
VTSS_MAC_ADDR_SZ_BYTES
types.h, 567
VTSS_MAC_ADDRS
vtss_l2_api.h, 652
VTSS_MAX_FRAME_LENGTH_MAX
vtss_port_api.h, 964
VTSS_MAX_FRAME_LENGTH_STANDARD
vtss_port_api.h, 964
VTSS_MAX_TIMEINTERVAL
types.h, 574
VTSS_MCE_ID_LAST
vtss_evc_api.h, 597
VTSS_MCE_ISDX_NEW
vtss_evc_api.h, 597
VTSS_MCE_ISDX_NONE
vtss_evc_api.h, 597
VTSS_MCE_POP_NONE
vtss_evc_api.h, 598
VTSS_MOD64
vtss_os_custom.h, 772
vtss_os_ecos.h, 778
vtss_os_linux.h, 789
VTSS_MPLS_IDX_IS_DEF
vtss_mpls_api.h, 736
VTSS_MPLS_IDX_IS_UNDEF
vtss_mpls_api.h, 736
VTSS_MPLS_IDX_UNDEFINED
vtss_mpls_api.h, 736
VTSS_MPLS_IDX_UNDEF
vtss_mpls_api.h, 737
VTSS_MPLS_LABEL_VALUE_DONTCARE
vtss_mpls_api.h, 736
VTSS_MPLS_QOS_TO_TC_ENTRY_CNT
vtss_mpls_api.h, 737
VTSS_MPLS_QOS_TO_TC_MAP_CNT
vtss_mpls_api.h, 737
VTSS_MPLS_TC_TO_QOS_ENTRY_CNT
vtss_mpls_api.h, 737
VTSS_MPLS_TC_TO_QOS_MAP_CNT
vtss_mpls_api.h, 737
VTSS_MPLS_TC_VALUE_DONTCARE
vtss_mpls_api.h, 736
VTSS_MSLEEP
vtss_os_custom.h, 771
vtss_os_ecos.h, 776
vtss_os_linux.h, 787
VTSS_MSTI_ARRAY_SIZE
vtss_l2_api.h, 652
VTSS_MSTI_END
vtss_l2_api.h, 652
VTSS_MSTI_START
vtss_l2_api.h, 652
VTSS_MSTIS
vtss_l2_api.h, 652
VTSS_MTIMER_CANCEL
vtss_os_custom.h, 772
vtss_os_ecos.h, 777
vtss_os_linux.h, 788
VTSS_MTIMER_START
vtss_os_custom.h, 772
vtss_os_ecos.h, 777
vtss_os_linux.h, 787
VTSS_MTIMER_TIMEOUT
vtss_os_custom.h, 772
vtss_os_ecos.h, 777
vtss_os_linux.h, 787
VTSS_NSLEEP
vtss_os_ecos.h, 776
vtss_os_linux.h, 786
VTSS_OAM_CNT_ALL
vtss_oam_api.h, 761
VTSS_OAM_CNT_CCM
vtss_oam_api.h, 760
VTSS_OAM_CNT_LB
vtss_oam_api.h, 761
VTSS_OAM_CNT_LM
vtss_oam_api.h, 760
VTSS_OAM_CNT_SEL
vtss_oam_api.h, 761
VTSS_OAM_CNT_TST
vtss_oam_api.h, 761
VTSS_OAM_EVENT_MASK_ARRAY_SIZE
vtss_oam_api.h, 758
VTSS_OAM_GENERIC_OPCODE_CFG_CNT
vtss_oam_api.h, 758
VTSS_OAM_PATH_SERVICE_VOE_CNT
vtss_oam_api.h, 757
VTSS_OAM_PORT_VOE_BASE_IDX
vtss_oam_api.h, 757
VTSS_OAM_PORT_VOE_CNT
vtss_oam_api.h, 757
VTSS_OAM_VOE_CNT
vtss_oam_api.h, 758
VTSS_OAM_VOE_EVENT_CCM_LOC

vtss_oam_api.h, 759
VTSS_OAM_VOE_EVENT_CCM_MEG_ID
vtss_oam_api.h, 760
VTSS_OAM_VOE_EVENT_CCM_MEP_ID
vtss_oam_api.h, 759
VTSS_OAM_VOE_EVENT_CCM_PERIOD
vtss_oam_api.h, 758
VTSS_OAM_VOE_EVENT_CCM_PRIORITY
vtss_oam_api.h, 759
VTSS_OAM_VOE_EVENT_CCM_RX_RDI
vtss_oam_api.h, 759
VTSS_OAM_VOE_EVENT_CCM_ZERO_PERIOD
vtss_oam_api.h, 759
VTSS_OAM_VOE_EVENT_MASK
vtss_oam_api.h, 760
VTSS_OAM_VOE_EVENT_MEG_LEVEL
vtss_oam_api.h, 760
VTSS_OAM_VOE_IDX_NONE
vtss_oam_api.h, 758
VTSS_ONE_MILL
types.h, 574
VTSS_ONE_MIA
types.h, 574
VTSS_OPT_FDMA_DEBUG
options.h, 534
VTSS_OPT_FDMA_IRQ_CONTEXT
options.h, 534
VTSS_OPT_PORT_COUNT
options.h, 534
VTSS_OPT_TRACE
options.h, 533
VTSS_OPT_VAUI_EQ_CTRL
options.h, 534
VTSS_OPT_VCORE_III
options.h, 531
VTSS_OS_BIG_ENDIAN
vtss_os_ecos.h, 781
vtss_os_linux.h, 786
VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED
vtss_fdma_api.h, 614
vtss_os_ecos.h, 780
VTSS_OS_CTZ64
vtss_os_custom.h, 773
vtss_os_ecos.h, 779
vtss_os_linux.h, 790
VTSS_OS_CTZ
vtss_os_custom.h, 773
vtss_os_ecos.h, 778
vtss_os_linux.h, 790
VTSS_OS_DCACHE_FLUSH
vtss_os_ecos.h, 781
VTSS_OS_DCACHE_INVALIDATE
vtss_os_ecos.h, 781
VTSS_OS_DCACHE_LINE_SIZE_BYTES
vtss_fdma_api.h, 614
vtss_os_ecos.h, 780
VTSS_OS_FREE
vtss_os_custom.h, 774
vtss_os_ecos.h, 779
vtss_os_linux.h, 791
VTSS_OS_INTERRUPT_DISABLE
vtss_os_ecos.h, 783
VTSS_OS_INTERRUPT_FLAGS
vtss_os_ecos.h, 783
VTSS_OS_INTERRUPT_RESTORE
vtss_os_ecos.h, 783
VTSS_OS_MALLOC
vtss_os_custom.h, 774
vtss_os_ecos.h, 779
vtss_os_linux.h, 791
VTSS_OS_NTOHL
vtss_os_ecos.h, 782
vtss_os_linux.h, 786
VTSS_OS_RAND
vtss_os_custom.h, 774
vtss_os_ecos.h, 780
vtss_os_linux.h, 791
VTSS_OS_REORDER_BARRIER
vtss_os_ecos.h, 780
VTSS_OS_SCHEDULER_FLAGS
vtss_os_ecos.h, 782
vtss_os_linux.h, 788
VTSS_OS_SCHEDULER_LOCK
vtss_os_ecos.h, 782
vtss_os_linux.h, 788
VTSS_OS_SCHEDULER_UNLOCK
vtss_os_ecos.h, 782
vtss_os_linux.h, 789
VTSS_OS_TIMESTAMP_TYPE
vtss_misc_api.h, 705
VTSS_OS_TIMESTAMP
vtss_misc_api.h, 706
VTSS_OS_VIRT_TO_PHYS
vtss_os_ecos.h, 781
VTSS_PACKET_HDR_SIZE_BYTES
vtss_packet_api.h, 797
VTSS_PACKET_RATE_DISABLED
types.h, 560
VTSS_PACKET_RX_GRP_CNT
types.h, 571
VTSS_PACKET_RX_QUEUE_CNT
types.h, 570
VTSS_PACKET_RX_QUEUE_END
types.h, 571
VTSS_PACKET_RX_QUEUE_NONE
types.h, 571
VTSS_PACKET_RX_QUEUE_START
types.h, 571
VTSS_PACKET_TX_GRP_CNT
types.h, 571
VTSS_PACKET_TX_IFH_MAX
vtss_packet_api.h, 798
VTSS_PCP_ARRAY_SIZE
types.h, 564
VTSS_PCP_END
types.h, 564

VTSS_PCP_START
types.h, 564
VTSS_PCPS
types.h, 563
VTSS_PHY_ACTIPHY_PWR
vtss_phy_api.h, 825
VTSS_PHY_LINK_AMS_EV
vtss_phy_api.h, 830
VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV
vtss_phy_api.h, 830
VTSS_PHY_LINK_AUTO_NEG_ERROR_EV
vtss_phy_api.h, 830
VTSS_PHY_LINK_DOWN_PWR
vtss_phy_api.h, 825
VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV
vtss_phy_api.h, 833
VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV
vtss_phy_api.h, 833
VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV
vtss_phy_api.h, 833
VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV
vtss_phy_api.h, 833
VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV
vtss_phy_api.h, 834
VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_EV
vtss_phy_api.h, 834
VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC_EV
vtss_phy_api.h, 834
VTSS_PHY_LINK_EXT_MACSEC_INGRESS_EV
vtss_phy_api.h, 834
VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_EV
vtss_phy_api.h, 834
VTSS_PHY_LINK_EXT_MEM_INT_RING_EV
vtss_phy_api.h, 835
VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV
vtss_phy_api.h, 833
VTSS_PHY_LINK_EXTENDED_REG_INT_EV
vtss_phy_api.h, 832
VTSS_PHY_LINK_FALSE_CARRIER_INT_EV
vtss_phy_api.h, 831
VTSS_PHY_LINK_FDX_STATE_CHANGE_EV
vtss_phy_api.h, 830
VTSS_PHY_LINK_FFAIL_EV
vtss_phy_api.h, 829
VTSS_PHY_LINK_INLINE_POW_DEV_DETECT_EV
vtss_phy_api.h, 831
VTSS_PHY_LINK_LINK_SPEED_DS_DETECT_EV
vtss_phy_api.h, 832
VTSS_PHY_LINK_LOS_EV
vtss_phy_api.h, 829
VTSS_PHY_LINK_MASTER_SLAVE_RES_ERR_EV
vtss_phy_api.h, 832
VTSS_PHY_LINK_RX_ER_INT_EV
vtss_phy_api.h, 832
VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT_EV
vtss_phy_api.h, 831
VTSS_PHY_LINK_SPEED_STATE_CHANGE_EV
vtss_phy_api.h, 830
VTSS_PHY_LINK_SYMBOL_ERR_INT_EV
vtss_phy_api.h, 831
VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT_EV
vtss_phy_api.h, 831
VTSS_PHY_LINK_UP_FULL_PWR
vtss_phy_api.h, 825
VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV
vtss_phy_api.h, 832
VTSS_PHY_OPT_VERIPHY
options.h, 534
VTSS_PHY_PAGE_0x2DAF
vtss_phy_api.h, 828
VTSS_PHY_PAGE_1588
vtss_phy_api.h, 827
VTSS_PHY_PAGE_EXTENDED_2
vtss_phy_api.h, 826
VTSS_PHY_PAGE_EXTENDED_3
vtss_phy_api.h, 827
VTSS_PHY_PAGE_EXTENDED_4
vtss_phy_api.h, 827
VTSS_PHY_PAGE_EXTENDED
vtss_phy_api.h, 826
VTSS_PHY_PAGE_GPIO
vtss_phy_api.h, 827
VTSS_PHY_PAGE_MACSEC
vtss_phy_api.h, 827
VTSS_PHY_PAGE_STANDARD
vtss_phy_api.h, 826
VTSS_PHY_PAGE_TEST
vtss_phy_api.h, 828
VTSS_PHY_PAGE_TR
vtss_phy_api.h, 828
VTSS_PHY_POWER_ACTIPHY_BIT
phy.h, 536
vtss_phy_api.h, 824
VTSS_PHY_POWER_DYNAMIC_BIT
phy.h, 536
vtss_phy_api.h, 825
VTSS_PHY_RECOV_CLK1
vtss_phy_api.h, 825
VTSS_PHY_RECOV_CLK2
vtss_phy_api.h, 826
VTSS_PHY_RECOV_CLK_NUM
vtss_phy_api.h, 826
VTSS_PHY_REG_EXTENDED
vtss_phy_api.h, 828
VTSS_PHY_REG_GPIO
vtss_phy_api.h, 829
VTSS_PHY_REG_STANDARD
vtss_phy_api.h, 828
VTSS_PHY_REG_TEST
vtss_phy_api.h, 829
VTSS_PHY_REG_TR
vtss_phy_api.h, 829
VTSS_PHY_TS_8487_XAUI_SEL_0
vtss_phy_ts_api.h, 912
VTSS_PHY_TS_8487_XAUI_SEL_1
vtss_phy_ts_api.h, 912

VTSS_PHY_TS_DATA_IN_RSRVD_FIELD
vtss_phy_ts_api.h, 911
VTSS_PHY_TS_EGR_DATAPATH_RESET
vtss_phy_ts_api.h, 912
VTSS_PHY_TS_EGR_ENGINE_ERR
vtss_phy_ts_api.h, 910
VTSS_PHY_TS_EGR_FIFO_OVERFLOW
vtss_phy_ts_api.h, 911
VTSS_PHY_TS_EGR_FIFO_RESET
vtss_phy_ts_api.h, 913
VTSS_PHY_TS_EGR_LTC2_RESET
vtss_phy_ts_api.h, 913
VTSS_PHY_TS_EGR_RW_FCS_ERR
vtss_phy_ts_api.h, 910
VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED
vtss_phy_ts_api.h, 911
VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0
vtss_phy_ts_api.h, 909
VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1
vtss_phy_ts_api.h, 909
VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT
vtss_phy_ts_api.h, 904
VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_MULTI↩
CAST
vtss_phy_ts_api.h, 905
VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_UNICAST
vtss_phy_ts_api.h, 905
VTSS_PHY_TS_ETH_MATCH_DEST_ADDR
vtss_phy_ts_api.h, 905
VTSS_PHY_TS_ETH_MATCH_SRC_ADDR
vtss_phy_ts_api.h, 905
VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST
vtss_phy_ts_api.h, 905
VTSS_PHY_TS_FIFO_SIG_DEST_IP
vtss_phy_ts_api.h, 901
VTSS_PHY_TS_FIFO_SIG_DEST_MAC
vtss_phy_ts_api.h, 902
VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM
vtss_phy_ts_api.h, 901
VTSS_PHY_TS_FIFO_SIG_MSG_TYPE
vtss_phy_ts_api.h, 901
VTSS_PHY_TS_FIFO_SIG_SEQ_ID
vtss_phy_ts_api.h, 901
VTSS_PHY_TS_FIFO_SIG_SH_SPORT_ID
vtss_phy_ts_api.h, 902
VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID
vtss_phy_ts_api.h, 901
VTSS_PHY_TS_FIFO_SIG_SRC_IP
vtss_phy_ts_api.h, 900
VTSS_PHY_TS_INGR_DATAPATH_RESET
vtss_phy_ts_api.h, 912
VTSS_PHY_TS_INGR_ENGINE_ERR
vtss_phy_ts_api.h, 910
VTSS_PHY_TS_INGR_LTC1_RESET
vtss_phy_ts_api.h, 912
VTSS_PHY_TS_INGR_RW_FCS_ERR
vtss_phy_ts_api.h, 910
VTSS_PHY_TS_INGR_RW_PREAM_ERR
vtss_phy_ts_api.h, 910
VTSS_PHY_TS_IP_MATCH_DEST
vtss_phy_ts_api.h, 908
VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST
vtss_phy_ts_api.h, 908
VTSS_PHY_TS_IP_MATCH_SRC
vtss_phy_ts_api.h, 907
VTSS_PHY_TS_IP_VER_4
vtss_phy_ts_api.h, 907
VTSS_PHY_TS_IP_VER_6
vtss_phy_ts_api.h, 907
VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD
vtss_phy_ts_api.h, 911
VTSS_PHY_TS_LTC_NEW_PPS_INTRPT
vtss_phy_ts_api.h, 911
VTSS_PHY_TS_MPLS_STACK_DEPTH_1
vtss_phy_ts_api.h, 908
VTSS_PHY_TS_MPLS_STACK_DEPTH_2
vtss_phy_ts_api.h, 908
VTSS_PHY_TS_MPLS_STACK_DEPTH_3
vtss_phy_ts_api.h, 908
VTSS_PHY_TS_MPLS_STACK_DEPTH_4
vtss_phy_ts_api.h, 909
VTSS_PHY_TS_MPLS_STACK_REF_POINT_END
vtss_phy_ts_api.h, 909
VTSS_PHY_TS_MPLS_STACK_REF_POINT_TOP
vtss_phy_ts_api.h, 909
VTSS_PHY_TS_SIG_DEST_IP_LEN
vtss_phy_ts_api.h, 903
VTSS_PHY_TS_SIG_DEST_MAC_LEN
vtss_phy_ts_api.h, 904
VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN
vtss_phy_ts_api.h, 902
VTSS_PHY_TS_SIG_LEN
vtss_phy_ts_api.h, 902
VTSS_PHY_TS_SIG_MSG_TYPE_LEN
vtss_phy_ts_api.h, 903
VTSS_PHY_TS_SIG_SEQ_ID_LEN
vtss_phy_ts_api.h, 903
VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN
vtss_phy_ts_api.h, 903
VTSS_PHY_TS_SIG_SH_SPORT_ID_LEN
vtss_phy_ts_api.h, 904
VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN
vtss_phy_ts_api.h, 903
VTSS_PHY_TS_SIG_SRC_IP_LEN
vtss_phy_ts_api.h, 904
VTSS_PHY_TS_SIG_TIME_STAMP_LEN
vtss_phy_ts_api.h, 902
VTSS_PHY_TS_SPI_CLK_THRU_PPS0
options.h, 529
VTSS_PHY_TS_TAG_RANGE_INNER
vtss_phy_ts_api.h, 907
VTSS_PHY_TS_TAG_RANGE_NONE
vtss_phy_ts_api.h, 906
VTSS_PHY_TS_TAG_RANGE_OUTER
vtss_phy_ts_api.h, 907
VTSS_PHY_TS_TAG_TYPE_B

vtss_phy_ts_api.h, 906
VTSS_PHY_TS_TAG_TYPE_C
vtss_phy_ts_api.h, 906
VTSS_PHY_TS_TAG_TYPE_I
vtss_phy_ts_api.h, 906
VTSS_PHY_TS_TAG_TYPE_S
vtss_phy_ts_api.h, 906
VTSS_PHYS_PORT_CNT
vtss_fdma_api.h, 612
VTSS_PORT_ARRAY_SIZE
types.h, 561
VTSS_PORT_COUNT
types.h, 560
VTSS_PORT_IS_PORT
types.h, 561
VTSS_PORT_NO_CPU
types.h, 561
VTSS_PORT_NO_END
types.h, 561
VTSS_PORT_NO_NONE
types.h, 560
VTSS_PORT_NO_START
types.h, 561
VTSS_PORT_POLICER_CPU_QUEUES
vtss_qos_api.h, 976
VTSS_PORT_POLICERS
vtss_qos_api.h, 976
VTSS_PORTS
types.h, 560
VTSS_PRIO_ARRAY_SIZE
types.h, 562
VTSS_PRIO_END
types.h, 562
VTSS_PRIO_NO_NONE
types.h, 562
VTSS_PRIO_START
types.h, 562
VTSS_PRIO_SUPER
vtss_packet_api.h, 795
VTSS_PRIOS
types.h, 562
VTSS_PTP_IP_1588_VERSION_2_1
vtss_phy_ts_api.h, 904
VTSS_PVLAN_ARRAY_SIZE
vtss_l2_api.h, 656
VTSS_PVLAN_NO_DEFAULT
vtss_l2_api.h, 657
VTSS_PVLAN_NO_END
vtss_l2_api.h, 656
VTSS_PVLAN_NO_START
vtss_l2_api.h, 656
VTSS_PVLANS
vtss_l2_api.h, 656
VTSS_QCE_ID_LAST
vtss_qos_api.h, 977
VTSS_QCL_ARRAY_SIZE
vtss_qos_api.h, 977
VTSS_QCL_ID_END
vtss_qos_api.h, 977
VTSS_QCL_ID_START
vtss_qos_api.h, 977
VTSS_QCL_IDS
vtss_qos_api.h, 976
VTSS_QUEUE_ARRAY_SIZE
types.h, 563
VTSS_QUEUE_END
types.h, 563
VTSS_QUEUE_START
types.h, 563
VTSS_QUEUES
types.h, 563
VTSS_SEC_NS_INTERVAL
types.h, 575
VTSS_SVL_PACKET_HDR_SIZE_BYTES
vtss_packet_api.h, 797
VTSS_SVL_RX_IFH_SIZE
vtss_packet_api.h, 797
VTSS_SYNCE_CLK_MAX
vtss_sync_api.h, 996
VTSS_SYNCE_CLK_PORT_ARRAY_SIZE
types.h, 576
VTSS_SYNCE_CLK_A
vtss_sync_api.h, 995
VTSS_SYNCE_CLK_B
vtss_sync_api.h, 995
VTSS_TIME_OF_DAY
vtss_os_ecos.h, 777
vtss_os_linux.h, 788
VTSS_TS_FIFO_MEDIA_SWAP_SYNC
options.h, 528
VTSS_TS_FIFO_SYNC_LOOPBACK
options.h, 528
VTSS_VCE_ID_LAST
vtss_l2_api.h, 653
VTSS_VCL_ARRAY_SIZE
vtss_l2_api.h, 653
VTSS_VCL_ID_END
vtss_l2_api.h, 653
VTSS_VCL_ID_START
vtss_l2_api.h, 653
VTSS_VCL_IDS
vtss_l2_api.h, 652
VTSS_VID_ALL
types.h, 567
VTSS_VID_DEFAULT
types.h, 566
VTSS_VID_NULL
types.h, 566
VTSS_VID_RESERVED
types.h, 566
VTSS_VIDS
types.h, 567
VTSS_VLAN_TRANS_FIRST_GROUP_ID
vtss_l2_api.h, 654
VTSS_VLAN_TRANS_GROUP_MAX_CNT
vtss_l2_api.h, 653

VTSS_VLAN_TRANS_LAST_GROUP_ID
vtss_l2_api.h, 655
VTSS_VLAN_TRANS_MAX_CNT
vtss_l2_api.h, 654
VTSS_VLAN_TRANS_MAX_VLAN_ID
vtss_l2_api.h, 654
VTSS_VLAN_TRANS_NULL_CHECK
vtss_l2_api.h, 655
VTSS_VLAN_TRANS_NULL_GROUP_ID
vtss_l2_api.h, 654
VTSS_VLAN_TRANS_PORT_BF_SIZE
vtss_l2_api.h, 656
VTSS_VLAN_TRANS_VALID_GROUP_CHECK
vtss_l2_api.h, 655
VTSS_VLAN_TRANS_VALID_VLAN_CHECK
vtss_l2_api.h, 655
VTSS_VLAN_TRANS_VID_START
vtss_l2_api.h, 654
val
vtss_eee_port_state_t, 112
vtss_phy_conf_1g_t, 300
vtss_phy_ts_eth_conf_t, 341
vtss_phy_ts_ptp_conf_t, 370
vtss_phy_ts_y1731_oam_conf_t, 379
value
vtss_mpls_label_t, 179
vtss_phy_ts_ach_conf_t, 328, 329
vtss_phy_ts_eth_conf_t, 341
vtss_phy_ts_ptp_conf_t, 371
vtss_phy_ts_y1731_oam_conf_t, 380
vtss_sgpio_port_data_t, 459
vtss_vcap_ip_t, 475
vtss_vcap_u128_t, 477
vtss_vcap_u16_t, 478
vtss_vcap_u24_t, 479
vtss_vcap_u32_t, 480
vtss_vcap_u40_t, 481
vtss_vcap_u48_t, 482
vtss_vcap_u8_t, 483
vtss_vcap_vid_t, 485
vtss_vcap_vr_t, 486
version
vtss_phy_ts_ach_conf_t, 328
vtss_phy_ts_oam_engine_action_t, 366
vga_adc_debug
vtss_phy_api.h, 864
vid
vtss_ace_vlan_t, 67
vtss_ece_inner_tag_t, 98
vtss_ece_outer_tag_t, 104
vtss_ece_tag_t, 107
vtss_evc_pb_conf_t, 120
vtss_mce_action_t, 166
vtss_mce_outer_tag_t, 170
vtss_mce_tag_t, 174
vtss_mpls_l2_t, 177
vtss_packet_frame_info_t, 256
vtss_packet_port_info_t, 259
vtss_qce_tag_t, 435
vtss_tci_t, 463
vtss_vce_action_t, 488
vtss_vce_tag_t, 499
vtss_vid_mac_t, 502
vtss_vlan_tag_t, 506
vtss_vlan_trans_grp2vlan_conf_t, 507
vid_mac
vtss_mac_table_entry_t, 160
vlan
vtss_ace_t, 65
vtss_routing_entry_t, 453
vlan_check
vtss_phy_ts_eth_conf_t, 339
vml_format
vtss_debug_info_t, 83
voe_id
vtss_oam_ts_id_s, 215
voe_idx
vtss_evc_oam_port_conf_t, 119
vtss_mce_action_t, 164
voe_mask
vtss_oam_event_mask_t, 197
voe_sq
vtss_oam_ts_id_s, 216
voe_type
vtss_oam_voe_conf_t, 225
vr
vtss_vcap_vr_t, 487
vtss_ace_add
vtss_security_api.h, 992
vtss_ace_bit_t
vtss_security_api.h, 988
vtss_ace_counter_clear
vtss_security_api.h, 993
vtss_ace_counter_get
vtss_security_api.h, 993
vtss_ace_del
vtss_security_api.h, 993
vtss_ace_frame_arp_t, 44
arp, 45
dip, 47
dmac_match, 46
ethernet, 46
ip, 46
length, 46
req, 45
sip, 46
smac, 45
smac_match, 45
unknown, 45
vtss_ace_frame_etype_t, 47
data, 48
dmac, 47
etype, 48
ptp, 48
smac, 48
vtss_ace_frame_ipv4_t, 49

data, 51
dip, 50
dport, 51
ds, 50
fragment, 49
options, 50
proto, 50
ptp, 53
seq_zero, 53
sip, 50
sip_eq_dip, 52
sip_smac, 53
sport, 51
sport_eq_dport, 53
tcp_ack, 52
tcp_fin, 51
tcp_psh, 52
tcp_rst, 52
tcp_syn, 51
tcp_urg, 52
ttl, 49
vtss_ace_frame_ipv6_t, 54
data, 55
dport, 55
ds, 55
proto, 54
ptp, 57
seq_zero, 57
sip, 54
sip_eq_dip, 57
sport, 55
sport_eq_dport, 57
tcp_ack, 56
tcp_fin, 56
tcp_psh, 56
tcp_rst, 56
tcp_syn, 56
tcp_urg, 57
ttl, 55
vtss_ace_frame_llc_t, 58
dmac, 58
llc, 59
smac, 58
vtss_ace_frame_snap_t, 59
dmac, 59
smac, 60
snap, 60
vtss_ace_init
vtss_security_api.h, 992
vtss_ace_ptp_t, 60
enable, 61
header, 61
vtss_ace_sip_smac_t, 61
enable, 62
sip, 62
smac, 62
vtss_ace_t, 62
action, 64
arp, 66
dmac_bc, 65
dmac_mc, 65
etype, 65
frame, 66
id, 63
ipv4, 66
ipv6, 66
isdx_disable, 64
isdx_enable, 63
llc, 65
lookup, 63
policy, 64
port_list, 64
snap, 66
type, 64
vlan, 65
vtss_ace_type_t
vtss_security_api.h, 987
vtss_ace_vlan_t, 67
cfi, 68
tagged, 68
usr_prio, 67
vid, 67
vtss_acl_action_t, 68
cpu, 69
cpu_once, 69
cpu_queue, 69
evc_police, 70
evc_policer_id, 70
learn, 70
lm_cnt_disable, 71
mac_swap, 71
mirror, 71
police, 69
policer_no, 69
port_action, 70
port_list, 70
ptp_action, 71
vtss_acl_policer_conf_get
vtss_security_api.h, 989
vtss_acl_policer_conf_set
vtss_security_api.h, 990
vtss_acl_policer_conf_t, 72
bit_rate, 72
bit_rate_enable, 72
rate, 72
vtss_acl_port_action_t
vtss_security_api.h, 987
vtss_acl_port_conf_get
vtss_security_api.h, 990
vtss_acl_port_conf_set
vtss_security_api.h, 990
vtss_acl_port_conf_t, 73
action, 73
policy_no, 73
vtss_acl_port_counter_clear
vtss_security_api.h, 991

vtss_acl_port_counter_get
vtss_security_api.h, 991
vtss_acl_ptp_action_t
vtss_security_api.h, 987
vtss_afi_alloc
vtss_packet_api.h, 809
vtss_afi_free
vtss_packet_api.h, 809
vtss_afi_frm_dscr_t, 74
actual_fps, 74
fps, 74
vtss_aggr_mode_get
vtss_l2_api.h, 686
vtss_aggr_mode_set
vtss_l2_api.h, 686
vtss_aggr_mode_t, 75
dmac_enable, 76
sip_dip_enable, 76
smac_enable, 75
sport_dport_enable, 76
vtss_aggr_port_members_get
vtss_l2_api.h, 685
vtss_aggr_port_members_set
vtss_l2_api.h, 686
vtss_aneg_t, 76
generate_pause, 77
obey_pause, 77
vtss_api/include/vtss/api/l2_types.h, 511
vtss_api/include/vtss/api/options.h, 512
vtss_api/include/vtss/api/phy.h, 535
vtss_api/include/vtss/api/port.h, 537
vtss_api/include/vtss/api/types.h, 550
vtss_api/include/vtss_ae_api.h, 590
vtss_api/include/vtss_afi_api.h, 591
vtss_api/include/vtss_aneg_api.h, 591
vtss_api/include/vtss_api.h, 592
vtss_api/include/vtss_evc_api.h, 592
vtss_api/include/vtss_fdma_api.h, 610
vtss_api/include/vtss_gfp_api.h, 626
vtss_api/include/vtss_hqos_api.h, 627
vtss_api/include/vtss_i2c_api.h, 631
vtss_api/include/vtss_init_api.h, 631
vtss_api/include/vtss_l2_api.h, 644
vtss_api/include/vtss_l3_api.h, 699
vtss_api/include/vtss_mac10g_api.h, 699
vtss_api/include/vtss_misc_api.h, 700
vtss_api/include/vtss_mpls_api.h, 733
vtss_api/include/vtss_oam_api.h, 753
vtss_api/include/vtss_oha_api.h, 770
vtss_api/include/vtss_os.h, 770
vtss_api/include/vtss_os_custom.h, 770
vtss_api/include/vtss_os_ecos.h, 775
vtss_api/include/vtss_os_linux.h, 785
vtss_api/include/vtss_otn_api.h, 792
vtss_api/include/vtss_packet_api.h, 792
vtss_api/include/vtss_pcs_10gbase_r_api.h, 814
vtss_api/include/vtss_phy_10g_api.h, 814
vtss_api/include/vtss_phy_api.h, 814
vtss_api/include/vtss_phy_ts_api.h, 891
vtss_api/include/vtss_port_api.h, 961
vtss_api/include/vtss_qos_api.h, 974
vtss_api/include/vtss_rab_api.h, 983
vtss_api/include/vtss_security_api.h, 983
vtss_api/include/vtss_sfi4_api.h, 994
vtss_api/include/vtss_sync_api.h, 994
vtss_api/include/vtss_tfi5_api.h, 998
vtss_api/include/vtss_ts_api.h, 998
vtss_api/include/vtss_upi_api.h, 1017
vtss_api/include/vtss_wis_api.h, 1017
vtss_api/include/vtss_xaui_api.h, 1018
vtss_api/include/vtss_xfi_api.h, 1018
vtss_api_lock_t, 77
file, 78
function, 78
inst, 78
line, 78
vtss_apvlan_port_members_get
vtss_l2_api.h, 682
vtss_apvlan_port_members_set
vtss_l2_api.h, 682
vtss_auth_port_state_get
vtss_security_api.h, 988
vtss_auth_port_state_set
vtss_security_api.h, 989
vtss_auth_state_t
vtss_security_api.h, 986
vtss_basic_counters_t, 79
rx_frames, 79
tx_frames, 79
vtss_callout_free
vtss_os_ecos.h, 784
vtss_callout_lock
vtss_misc_api.h, 714
vtss_callout_malloc
vtss_os_ecos.h, 784
vtss_callout_trace_hex_dump
vtss_misc_api.h, 713
vtss_callout_trace_printf
vtss_misc_api.h, 712
vtss_callout_unlock
vtss_misc_api.h, 714
vtss_chip_id_get
vtss_misc_api.h, 717
vtss_chip_id_t, 80
part_number, 80
revision, 80
vtss_counter_pair_t, 81
bytes, 81
frames, 81
vtss_debug_group_t
vtss_misc_api.h, 708
vtss_debug_info_get
vtss_misc_api.h, 713
vtss_debug_info_print
vtss_misc_api.h, 714
vtss_debug_info_t, 82

chip_no, 83
clear, 83
full, 83
group, 82
layer, 82
port_list, 83
vml_format, 83
vtss_debug_layer_t
vtss_misc_api.h, 708
vtss_debug_lock
vtss_misc_api.h, 715
vtss_debug_lock_t, 84
chip_no, 84
vtss_debug_reg_check_set
vtss_misc_api.h, 732
vtss_debug_unlock
vtss_misc_api.h, 715
vtss_dev_all_event_enable
vtss_misc_api.h, 719
vtss_dev_all_event_poll
vtss_misc_api.h, 719
vtss_dgroup_port_conf_get
vtss_l2_api.h, 683
vtss_dgroup_port_conf_set
vtss_l2_api.h, 683
vtss_dgroup_port_conf_t, 85
dgroup_no, 85
vtss_dlb_policer_conf_t, 85
cbs, 87
cf, 86
cir, 86
ebs, 87
eir, 87
enable, 86
line_rate, 86
type, 86
vtss_dscp_emode_t
vtss_qos_api.h, 979
vtss_dscp_mode_t
vtss_qos_api.h, 978
vtss_ece_action_t, 87
dir, 88
dp, 90
dp_enable, 90
evc_id, 89
inner_tag, 89
outer_tag, 89
policer_id, 89
policy_no, 90
pop_tag, 89
prio, 90
prio_enable, 90
rule, 88
tx_lookup, 88
vtss_ece_add
vtss_evc_api.h, 604
vtss_ece_counters_clear
vtss_evc_api.h, 609
vtss_ece_counters_get
vtss_evc_api.h, 608
vtss_ece_dei_mode_t
types.h, 589
vtss_ece_del
vtss_evc_api.h, 605
vtss_ece_dir_t
types.h, 588
vtss_ece_frame_etype_t, 91
data, 91
etype, 91
vtss_ece_frame_ipv4_t, 92
dip, 93
dport, 93
dscp, 92
fragment, 92
proto, 93
sip, 93
sport, 93
vtss_ece_frame_ipv6_t, 94
dip, 95
dport, 95
dscp, 94
proto, 94
sip, 95
sport, 95
vtss_ece_frame_llc_t, 96
data, 96
vtss_ece_frame_snap_t, 96
data, 97
vtss_ece_init
vtss_evc_api.h, 604
vtss_ece_inner_tag_t, 97
dei, 98
dei_mode, 98
pcp, 98
pcp_mode, 98
type, 97
vid, 98
vtss_ece_inner_tag_type_t
types.h, 590
vtss_ece_key_t, 99
etype, 101
frame, 102
inner_tag, 100
ipv4, 101
ipv6, 101
llc, 101
lookup, 100
mac, 100
port_list, 99
snap, 101
tag, 100
type, 100
vtss_ece_mac_t, 102
dmac, 102
dmac_bc, 103
dmac_mc, 103

smac, 103
vtss_ece_outer_tag_t, 103
dei, 105
dei_mode, 105
enable, 104
pcp, 104
pcp_mode, 104
vid, 104
vtss_ece_pcp_mode_t
types.h, 589
vtss_ece_pop_tag_t
types.h, 588
vtss_ece_port_t
vtss_evc_api.h, 598
vtss_ece_rule_t
types.h, 588
vtss_ece_t, 105
action, 106
id, 106
key, 106
vtss_ece_tag_t, 106
dei, 107
pcp, 107
s_tagged, 107
tagged, 107
vid, 107
vtss_ece_tx_lookup_t
types.h, 589
vtss_ece_type_t
vtss_evc_api.h, 598
vtss_eee_port_conf_set
vtss_misc_api.h, 731
vtss_eee_port_conf_t, 108
eee_ena, 108
eee_fast_queues, 108
lp_advertisement, 109
optimized_for_power, 109
tx_tw, 109
vtss_eee_port_counter_get
vtss_misc_api.h, 732
vtss_eee_port_counter_t, 109
fill_level, 110
fill_level_get, 110
fill_level_thres, 110
tx_out_bytes, 111
tx_out_bytes_get, 110
vtss_eee_port_state_set
vtss_misc_api.h, 732
vtss_eee_port_state_t, 111
select, 111
val, 112
vtss_eee_state_select_t
vtss_misc_api.h, 711
vtss_eps_port_conf_get
vtss_l2_api.h, 695
vtss_eps_port_conf_set
vtss_l2_api.h, 696
vtss_eps_port_conf_t, 112
port_no, 113
type, 112
vtss_eps_port_selector_get
vtss_l2_api.h, 696
vtss_eps_port_selector_set
vtss_l2_api.h, 697
vtss_eps_port_type_t
vtss_l2_api.h, 659
vtss_eps_selector_t
vtss_l2_api.h, 660
vtss_erps_port_state_get
vtss_l2_api.h, 698
vtss_erps_port_state_set
vtss_l2_api.h, 698
vtss_erps_state_t
vtss_l2_api.h, 660
vtss_erps_vlan_member_get
vtss_l2_api.h, 697
vtss_erps_vlan_member_set
vtss_l2_api.h, 698
vtss_evc_add
vtss_evc_api.h, 601
vtss_evc_api.h
VTSS_ECE_ID_LAST, 597
VTSS_EVC_ID_NONE, 596
VTSS_EVC_MPLS_PW_CNT, 596
VTSS_EVC_POLICER_ID_DISCARD, 596
VTSS_EVC_POLICER_ID_EVC, 596
VTSS_EVC_POLICER_ID_NONE, 596
VTSS_EVC_POLICERS, 596
VTSS_ISDX_CPU_TX, 597
VTSS_MCE_ID_LAST, 597
VTSS_MCE_ISDX_NEW, 597
VTSS_MCE_ISDX_NONE, 597
VTSS_MCE_POP_NONE, 598
vtss_ece_add, 604
vtss_ece_counters_clear, 609
vtss_ece_counters_get, 608
vtss_ece_del, 605
vtss_ece_init, 604
vtss_ece_port_t, 598
vtss_ece_type_t, 598
vtss_evc_add, 601
vtss_evc_counters_clear, 608
vtss_evc_counters_get, 608
vtss_evc_del, 602
vtss_evc_get, 602
vtss_evc_hqos_map_get, 603
vtss_evc_hqos_map_set, 603
vtss_evc_oam_port_conf_get, 605
vtss_evc_oam_port_conf_set, 605
vtss_evc_policer_conf_get, 601
vtss_evc_policer_conf_set, 601
vtss_evc_port_conf_get, 600
vtss_evc_port_conf_set, 600
vtss_mce_add, 606
vtss_mce_counters_clear, 610
vtss_mce_counters_get, 609

vtss_mce_del, 607
vtss_mce_init, 606
vtss_mce_oam_detect_t, 599
vtss_mce_port_info_get, 607
vtss_mce_rule_t, 599
vtss_mce_tx_lookup_t, 599
vtss_evc_conf_t, 113
learning, 114
mpls_tp, 114
network, 114
pb, 114
policer_id, 113
vtss_evc_counters_clear
vtss_evc_api.h, 608
vtss_evc_counters_get
vtss_evc_api.h, 608
vtss_evc_counters_t, 115
rx_discard, 116
rx_green, 115
rx_red, 115
rx_yellow, 115
tx_discard, 116
tx_green, 116
tx_yellow, 116
vtss_evc_del
vtss_evc_api.h, 602
vtss_evc_get
vtss_evc_api.h, 602
vtss_evc_hqos_map_get
vtss_evc_api.h, 603
vtss_evc_hqos_map_set
vtss_evc_api.h, 603
vtss_evc_mpls_pw_info_t, 117
egress_xc, 117
ingress_xc, 117
pw_num, 118
split_horizon, 117
vtss_evc_mpls_tp_conf_t, 118
pw, 118
vtss_evc_oam_port_conf_get
vtss_evc_api.h, 605
vtss_evc_oam_port_conf_set
vtss_evc_api.h, 605
vtss_evc_oam_port_conf_t, 119
voe_idx, 119
vtss_evc_pb_conf_t, 120
ivid, 120
leaf, 121
leaf_ivid, 121
leaf_vid, 121
nni, 120
vid, 120
vtss_evc_policer_conf_get
vtss_evc_api.h, 601
vtss_evc_policer_conf_set
vtss_evc_api.h, 601
vtss_evc_port_conf_get
vtss_evc_api.h, 600
vtss_evc_port_conf_set
vtss_evc_api.h, 600
vtss_evc_port_conf_t, 121
dmac_dip, 122
key_type, 122
vtss_fan_conf_t, 122
fan_low_pol, 123
fan_open_col, 123
fan_pwm_freq, 123
ppr, 124
type, 123
vtss_fan_controller_init
vtss_misc_api.h, 730
vtss_fan_cool_lvl_get
vtss_misc_api.h, 731
vtss_fan_cool_lvl_set
vtss_misc_api.h, 730
vtss_fan_rotation_get
vtss_misc_api.h, 729
vtss_fdma_afi_cancel
vtss_fdma_api.h, 622
vtss_fdma_afi_type_t
vtss_fdma_api.h, 617
vtss_fdma_api.h
VTSS_AFI_FPS_MAX, 615
VTSS_FDMA_CH_CNT, 612
VTSS_FDMA_DCB_SIZE_BYTES, 613
VTSS_FDMA_HDR_SIZE_BYTES, 613
VTSS_FDMA_MAX_DATA_PER_DCB_BYTES, 613
VTSS_FDMA_MAX_FRAME_SIZE_BYTES, 614
VTSS_FDMA_MIN_DATA_PER_INJ_SOF_DCB_BYTES, 613
VTSS_FDMA_MIN_DATA_PER_NON_SOF_DCB_BYTES, 614
VTSS_FDMA_MIN_DATA_PER_XTR_SOF_DCB_BYTES, 614
VTSS_FDMA_MIN_FRAME_SIZE_BYTES, 613
VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED, 614
VTSS_OS_DCACHE_LINE_SIZE_BYTES, 614
VTSS_PHYS_PORT_CNT, 612
vtss_fdma_afi_cancel, 622
vtss_fdma_afi_type_t, 617
vtss_fdma_cfg, 619
vtss_fdma_ch_t, 615
vtss_fdma_ch_usage_t, 618
vtss_fdma_dcb_get, 622
vtss_fdma_dcb_release, 620
vtss_fdma_dcb_type_t, 618
vtss_fdma_irq_handler, 625
vtss_fdma_list_t, 617
vtss_fdma_stats_clr, 625
vtss_fdma_throttle_cfg_get, 623
vtss_fdma_throttle_cfg_set, 624
vtss_fdma_throttle_tick, 624
vtss_fdma_tx, 620
vtss_fdma_tx_info_init, 621

vtss_fdma_uninit, 618
vtss_fdma_cfg
vtss_fdma_api.h, 619
vtss_fdma_cfg_t, 124
afi_buf_cnt, 128
afi_done_cb, 129
enable, 125
rx_alloc_cb, 125
rx_allow_multiple_dcbs, 127
rx_allow_vlan_tag_mismatch, 127
rx_buf_cnt, 125
rx_cb, 127
rx_dont_reinsert_vlan_tag, 127
rx_dont_strip_vlan_tag, 126
rx_mtu, 125
tx_buf_cnt, 128
tx_done_cb, 128
vtss_fdma_ch_cfg_t, 129
chip_no, 132
inj_grp_mask, 130
list, 131
prio, 132
usage, 130
xtr_cb, 131
xtr_grp, 130
vtss_fdma_ch_t
vtss_fdma_api.h, 615
vtss_fdma_ch_usage_t
vtss_fdma_api.h, 618
vtss_fdma_dcb_get
vtss_fdma_api.h, 622
vtss_fdma_dcb_release
vtss_fdma_api.h, 620
vtss_fdma_dcb_type_t
vtss_fdma_api.h, 618
vtss_fdma_irq_handler
vtss_fdma_api.h, 625
vtss_fdma_list_t
vtss_fdma_api.h, 617
vtss_fdma_stats_clr
vtss_fdma_api.h, 625
vtss_fdma_throttle_cfg_get
vtss_fdma_api.h, 623
vtss_fdma_throttle_cfg_set
vtss_fdma_api.h, 624
vtss_fdma_throttle_cfg_t, 133
byte_limit_per_tick, 134
frm_limit_per_tick, 134
suspend_tick_cnt, 134
vtss_fdma_throttle_tick
vtss_fdma_api.h, 624
vtss_fdma_tx
vtss_fdma_api.h, 620
vtss_fdma_tx_info_init
vtss_fdma_api.h, 621
vtss_fdma_tx_info_t, 134
afi_fps, 136
afi_type, 136
pre_cb, 135
pre_cb_ctxt1, 135
pre_cb_ctxt2, 135
vtss_fdma_uninit
vtss_fdma_api.h, 618
vtss_fefi_mode_t
vtss_phy_api.h, 845
vtss_fiber_port_speed_t
port.h, 549
vtss_gpio_direction_set
vtss_misc_api.h, 720
vtss_gpio_event_enable
vtss_misc_api.h, 722
vtss_gpio_event_poll
vtss_misc_api.h, 722
vtss_gpio_mode_set
vtss_misc_api.h, 720
vtss_gpio_mode_t
vtss_misc_api.h, 709
vtss_gpio_read
vtss_misc_api.h, 721
vtss_gpio_write
vtss_misc_api.h, 721
vtss_hqos_add
vtss_hqos_api.h, 629
vtss_hqos_api.h
vtss_hqos_add, 629
vtss_hqos_del, 629
vtss_hqos_get, 629
vtss_hqos_min_rate_calc, 630
vtss_hqos_port_conf_get, 628
vtss_hqos_port_conf_set, 628
vtss_hqos_port_min_rate_calc, 630
vtss_hqos_conf_t, 137
dwrr_cnt, 138
min_rate, 138
port_no, 137
queue_pct, 138
shaper, 137
shaper_queue, 138
vtss_hqos_del
vtss_hqos_api.h, 629
vtss_hqos_get
vtss_hqos_api.h, 629
vtss_hqos_min_rate_calc
vtss_hqos_api.h, 630
vtss_hqos_port_conf_get
vtss_hqos_api.h, 628
vtss_hqos_port_conf_set
vtss_hqos_api.h, 628
vtss_hqos_port_conf_t, 139
sch_mode, 139
vtss_hqos_port_min_rate_calc
vtss_hqos_api.h, 630
vtss_hqos_sch_mode_t
types.h, 590
vtss_i2c_read_t
vtss_init_api.h, 635

vtss_i2c_write_t
vtss_init_api.h, 635
vtss_init_api.h
VTSS_I2C_NO_MULTIPLEXER, 633
vtss_i2c_read_t, 635
vtss_i2c_write_t, 635
vtss_init_conf_get, 642
vtss_init_conf_set, 642
vtss_inst_create, 641
vtss_inst_destroy, 642
vtss_inst_get, 641
vtss_miim_read_t, 637
vtss_miim_write_t, 638
vtss_mmd_read_inc_t, 639
vtss_mmd_read_t, 638
vtss_mmd_write_t, 639
vtss_reg_read_t, 634
vtss_reg_write_t, 634
vtss_restart_conf_end, 643
vtss_restart_conf_get, 643
vtss_restart_conf_set, 644
vtss_restart_status_get, 643
vtss_restart_t, 640
vtss_spi_32bit_read_write_t, 636
vtss_spi_64bit_read_write_t, 637
vtss_spi_read_write_t, 636
vtss_target_type_t, 640
vtss_init_conf_get
vtss_init_api.h, 642
vtss_init_conf_set
vtss_init_api.h, 642
vtss_init_conf_t, 139
miim_read, 140
miim_write, 141
mmd_read, 141
mmd_read_inc, 141
mmd_write, 141
pi, 143
reg_read, 140
reg_write, 140
restart_info_port, 142
restart_info_src, 142
serdes, 143
spi_32bit_read_write, 142
spi_64bit_read_write, 142
spi_read_write, 141
warm_start_enable, 142
vtss_inst_create
vtss_init_api.h, 641
vtss_inst_create_t, 143
target, 144
vtss_inst_destroy
vtss_init_api.h, 642
vtss_inst_get
vtss_init_api.h, 641
vtss_internal_bw_t
vtss_port_api.h, 965
vtss_intr_cfg
vtss_misc_api.h, 725
vtss_intr_mask_set
vtss_misc_api.h, 725
vtss_intr_pol_negation
vtss_misc_api.h, 726
vtss_intr_status_get
vtss_misc_api.h, 726
vtss_intr_sticky_clear
vtss_misc_api.h, 717
vtss_intr_t, 144
link_change, 144
vtss_ip_addr_t, 145
addr, 146
ipv4, 145
ipv6, 145
type, 145
vtss_ip_network_t, 146
address, 146
prefix_size, 146
vtss_ip_type_t
types.h, 586
vtss_ipv4_mc_flood_members_get
vtss_l2_api.h, 693
vtss_ipv4_mc_flood_members_set
vtss_l2_api.h, 693
vtss_ipv4_network_t, 147
address, 147
prefix_size, 147
vtss_ipv4_uc_t, 148
destination, 148
network, 148
vtss_ipv6_mc_ctrl_flood_get
vtss_l2_api.h, 695
vtss_ipv6_mc_ctrl_flood_set
vtss_l2_api.h, 695
vtss_ipv6_mc_flood_members_get
vtss_l2_api.h, 694
vtss_ipv6_mc_flood_members_set
vtss_l2_api.h, 694
vtss_ipv6_network_t, 149
address, 149
prefix_size, 149
vtss_ipv6_t, 150
addr, 150
vtss_ipv6_uc_t, 151
destination, 151
network, 151
vtss_irq_conf_get
vtss_misc_api.h, 727
vtss_irq_conf_set
vtss_misc_api.h, 727
vtss_irq_conf_t, 151
destination, 152
external, 152
vtss_irq_enable
vtss_misc_api.h, 728
vtss_irq_status_get_and_mask
vtss_misc_api.h, 727

vtss_irq_status_t, 152
active, 153
raw_ident, 153
raw_mask, 153
raw_status, 153
vtss_irq_t
vtss_misc_api.h, 710
vtss_isdx_t
types.h, 578
vtss_isolated_port_members_get
vtss_l2_api.h, 680
vtss_isolated_port_members_set
vtss_l2_api.h, 681
vtss_isolated_vlan_get
vtss_l2_api.h, 679
vtss_isolated_vlan_set
vtss_l2_api.h, 680
vtss_l2_api.h
VTSS_ERPI_ARRAY_SIZE, 657
VTSS_ERPI_END, 657
VTSS_ERPI_START, 657
VTSS_ERPIS, 657
VTSS_MAC_ADDRS, 652
VTSS_MSTI_ARRAY_SIZE, 652
VTSS_MSTI_END, 652
VTSS_MSTI_START, 652
VTSS_MSTIS, 652
VTSS_PVLAN_ARRAY_SIZE, 656
VTSS_PVLAN_NO_DEFAULT, 657
VTSS_PVLAN_NO_END, 656
VTSS_PVLAN_NO_START, 656
VTSS_PVLANS, 656
VTSS_VCE_ID_LAST, 653
VTSS_VCL_ARRAY_SIZE, 653
VTSS_VCL_ID_END, 653
VTSS_VCL_ID_START, 653
VTSS_VCL_IDS, 652
VTSS_VLAN_TRANS_FIRST_GROUP_ID, 654
VTSS_VLAN_TRANS_GROUP_MAX_CNT, 653
VTSS_VLAN_TRANS_LAST_GROUP_ID, 655
VTSS_VLAN_TRANS_MAX_CNT, 654
VTSS_VLAN_TRANS_MAX_VLAN_ID, 654
VTSS_VLAN_TRANS_NULL_CHECK, 655
VTSS_VLAN_TRANS_NULL_GROUP_ID, 654
VTSS_VLAN_TRANS_PORT_BF_SIZE, 656
VTSS_VLAN_TRANS_VALID_GROUP_CHECK, 655
VTSS_VLAN_TRANS_VALID_VLAN_CHECK, 655
VTSS_VLAN_TRANS_VID_START, 654
vtss_aggr_mode_get, 686
vtss_aggr_mode_set, 686
vtss_aggr_port_members_get, 685
vtss_aggr_port_members_set, 686
vtss_apvlan_port_members_get, 682
vtss_apvlan_port_members_set, 682
vtss_dgroup_port_conf_get, 683
vtss_dgroup_port_conf_set, 683
vtss_eps_port_conf_get, 695
vtss_eps_port_conf_set, 696
vtss_eps_port_selector_get, 696
vtss_eps_port_selector_set, 697
vtss_eps_port_type_t, 659
vtss_eps_selector_t, 660
vtss_erps_port_state_get, 698
vtss_erps_port_state_set, 698
vtss_erps_state_t, 660
vtss_erps_vlan_member_get, 697
vtss_erps_vlan_member_set, 698
vtss_ipv4_mc_flood_members_get, 693
vtss_ipv4_mc_flood_members_set, 693
vtss_ipv6_mc_ctrl_flood_get, 695
vtss_ipv6_mc_ctrl_flood_set, 695
vtss_ipv6_mc_flood_members_get, 694
vtss_ipv6_mc_flood_members_set, 694
vtss_isolated_port_members_get, 680
vtss_isolated_port_members_set, 681
vtss_isolated_vlan_get, 679
vtss_isolated_vlan_set, 680
vtss_learn_port_mode_get, 666
vtss_learn_port_mode_set, 666
vtss_mac_table_add, 660
vtss_mac_table_age, 663
vtss_mac_table_age_time_get, 662
vtss_mac_table_age_time_set, 663
vtss_mac_table_del, 661
vtss_mac_table_flush, 664
vtss_mac_table_get, 661
vtss_mac_table_get_next, 662
vtss_mac_table_port_flush, 664
vtss_mac_table_status_get, 665
vtss_mac_table_vlan_age, 663
vtss_mac_table_vlan_flush, 664
vtss_mac_table_vlan_port_flush, 665
vtss_mc_flood_members_get, 692
vtss_mc_flood_members_set, 693
vtss_mirror_conf_get, 687
vtss_mirror_conf_set, 687
vtss_mirror_cpu_egress_get, 691
vtss_mirror_cpu_egress_set, 691
vtss_mirror_cpu_ingress_get, 690
vtss_mirror_cpu_ingress_set, 690
vtss_mirror_egress_ports_get, 689
vtss_mirror_egress_ports_set, 689
vtss_mirror_ingress_ports_get, 688
vtss_mirror_ingress_ports_set, 689
vtss_mirror_monitor_port_get, 688
vtss_mirror_monitor_port_set, 688
vtss_mstp_port_msti_state_get, 669
vtss_mstp_port_msti_state_set, 670
vtss_mstp_vlan_msti_get, 668
vtss_mstp_vlan_msti_set, 669
vtss_port_state_get, 667
vtss_port_state_set, 667
vtss_pvlan_port_members_get, 681
vtss_pvlan_port_members_set, 682

vtss_sflow_port_conf_get, 684
vtss_sflow_port_conf_set, 684
vtss_sflow_sampling_rate_convert, 685
vtss_stp_port_state_get, 667
vtss_stp_port_state_set, 668
vtss_stp_state_t, 658
vtss_uc_flood_members_get, 692
vtss_uc_flood_members_set, 692
vtss_vcap_port_conf_get, 679
vtss_vcap_port_conf_set, 679
vtss_vce_add, 676
vtss_vce_del, 676
vtss_vce_init, 675
vtss_vce_type_t, 659
vtss_vcl_port_conf_get, 674
vtss_vcl_port_conf_set, 675
vtss_vlan_conf_get, 670
vtss_vlan_conf_set, 670
vtss_vlan_port_conf_get, 671
vtss_vlan_port_conf_set, 671
vtss_vlan_port_members_get, 672
vtss_vlan_port_members_set, 672
vtss_vlan_port_type_t, 658
vtss_vlan_trans_group_add, 676
vtss_vlan_trans_group_del, 677
vtss_vlan_trans_group_get, 677
vtss_vlan_trans_group_to_port_get, 678
vtss_vlan_trans_group_to_port_set, 678
vtss_vlan_tx_tag_get, 673
vtss_vlan_tx_tag_set, 674
vtss_vlan_tx_tag_t, 659
vtss_vlan_vid_conf_get, 673
vtss_vlan_vid_conf_set, 673
vtss_vt_id_t, 658
vtss_l3_counters_t, 154
ipv4uc_received_frames, 154
ipv4uc_received_octets, 154
ipv4uc_transmitted_frames, 155
ipv4uc_transmitted_octets, 155
ipv6uc_received_frames, 155
ipv6uc_received_octets, 155
ipv6uc_transmitted_frames, 156
ipv6uc_transmitted_octets, 155
vtss_lcpll_status_t, 156
cal_done, 157
cal_error, 157
fsm_lock, 157
fsm_stat, 157
gain_stat, 157
lock_status, 156
vtss_learn_mode_t, 158
automatic, 158
cpu, 158
discard, 159
vtss_learn_port_mode_get
vtss_l2_api.h, 666
vtss_learn_port_mode_set
vtss_l2_api.h, 666
vtss_mac_addr_t
types.h, 578
vtss_mac_t, 159
addr, 159
vtss_mac_table_add
vtss_l2_api.h, 660
vtss_mac_table_age
vtss_l2_api.h, 663
vtss_mac_table_age_time_get
vtss_l2_api.h, 662
vtss_mac_table_age_time_set
vtss_l2_api.h, 663
vtss_mac_table_del
vtss_l2_api.h, 661
vtss_mac_table_entry_t, 160
aged, 161
copy_to_cpu, 161
cpu_queue, 161
destination, 160
locked, 161
vid_mac, 160
vtss_mac_table_flush
vtss_l2_api.h, 664
vtss_mac_table_get
vtss_l2_api.h, 661
vtss_mac_table_get_next
vtss_l2_api.h, 662
vtss_mac_table_port_flush
vtss_l2_api.h, 664
vtss_mac_table_status_get
vtss_l2_api.h, 665
vtss_mac_table_status_t, 162
aged, 163
learned, 162
moved, 162
replaced, 162
vtss_mac_table_vlan_age
vtss_l2_api.h, 663
vtss_mac_table_vlan_flush
vtss_l2_api.h, 664
vtss_mac_table_vlan_port_flush
vtss_l2_api.h, 665
vtss_mc_flood_members_get
vtss_l2_api.h, 692
vtss_mc_flood_members_set
vtss_l2_api.h, 693
vtss_mce_action_t, 163
evc_counting, 164
evc_etree, 164
evc_id, 164
isdx, 165
oam_detect, 165
outer_tag, 165
policer_id, 166
policy_no, 166
pop_cnt, 167
port_list, 164
prio, 166

prio_enable, 166
rule, 165
tx_lookup, 165
vid, 166
voe_idx, 164
vtss_mce_add
vtss_evc_api.h, 606
vtss_mce_counters_clear
vtss_evc_api.h, 610
vtss_mce_counters_get
vtss_evc_api.h, 609
vtss_mce_del
vtss_evc_api.h, 607
vtss_mce_init
vtss_evc_api.h, 606
vtss_mce_key_t, 167
dmac, 169
dmac_mc, 169
injected, 168
inner_tag, 168
lookup, 169
mel, 168
port_cpu, 168
port_list, 168
service_detect, 169
tag, 168
vtss_mce_oam_detect_t
vtss_evc_api.h, 599
vtss_mce_outer_tag_t, 170
dei, 171
dei_mode, 171
enable, 170
pcp, 171
pcp_mode, 170
vid, 170
vtss_mce_port_info_get
vtss_evc_api.h, 607
vtss_mce_port_info_t, 171
isdx, 172
vtss_mce_rule_t
vtss_evc_api.h, 599
vtss_mce_t, 172
action, 173
id, 173
key, 173
tt_loop, 173
vtss_mce_tag_t, 174
dei, 175
pcp, 175
s_tagged, 174
tagged, 174
vid, 174
vtss_mce_tx_lookup_t
vtss_evc_api.h, 599
vtss_mem_flags_t
types.h, 582
vtss_miim_controller_t
vtss_port_api.h, 964
vtss_miim_read
vtss_port_api.h, 973
vtss_miim_read_t
vtss_init_api.h, 637
vtss_miim_write
vtss_port_api.h, 973
vtss_miim_write_t
vtss_init_api.h, 638
vtss_mirror_conf_get
vtss_l2_api.h, 687
vtss_mirror_conf_set
vtss_l2_api.h, 687
vtss_mirror_conf_t, 175
fwd_enable, 176
port_no, 176
vtss_mirror_cpu_egress_get
vtss_l2_api.h, 691
vtss_mirror_cpu_egress_set
vtss_l2_api.h, 691
vtss_mirror_cpu_ingress_get
vtss_l2_api.h, 690
vtss_mirror_cpu_ingress_set
vtss_l2_api.h, 690
vtss_mirror_egress_ports_get
vtss_l2_api.h, 689
vtss_mirror_egress_ports_set
vtss_l2_api.h, 689
vtss_mirror_ingress_ports_get
vtss_l2_api.h, 688
vtss_mirror_ingress_ports_set
vtss_l2_api.h, 689
vtss_mirror_monitor_port_get
vtss_l2_api.h, 688
vtss_mirror_monitor_port_set
vtss_l2_api.h, 688
vtss_misc_api.h
tod_get_ns_cnt_cb_t, 706
VTSS_OS_TIMESTAMP_TYPE, 705
VTSS_OS_TIMESTAMP, 706
vtss_callout_lock, 714
vtss_callout_trace_hex_dump, 713
vtss_callout_trace_printf, 712
vtss_callout_unlock, 714
vtss_chip_id_get, 717
vtss_debug_group_t, 708
vtss_debug_info_get, 713
vtss_debug_info_print, 714
vtss_debug_layer_t, 708
vtss_debug_lock, 715
vtss_debug_reg_check_set, 732
vtss_debug_unlock, 715
vtss_dev_all_event_enable, 719
vtss_dev_all_event_poll, 719
vtss_eee_port_conf_set, 731
vtss_eee_port_counter_get, 732
vtss_eee_port_state_set, 732
vtss_eee_state_select_t, 711
vtss_fan_controller_init, 730

vtss_fan_cool_lvl_get, 731
vtss_fan_cool_lvl_set, 730
vtss_fan_rotation_get, 729
vtss_gpio_direction_set, 720
vtss_gpio_event_enable, 722
vtss_gpio_event_poll, 722
vtss_gpio_mode_set, 720
vtss_gpio_mode_t, 709
vtss_gpio_read, 721
vtss_gpio_write, 721
vtss_intr_cfg, 725
vtss_intr_mask_set, 725
vtss_intr_pol_negation, 726
vtss_intr_status_get, 726
vtss_intr_sticky_clear, 717
vtss_irq_conf_get, 727
vtss_irq_conf_set, 727
vtss_irq_enable, 728
vtss_irq_status_get_and_mask, 727
vtss_irq_t, 710
vtss_poll_1sec, 718
vtss_ptp_event_enable, 718
vtss_ptp_event_poll, 718
vtss_reg_read, 715
vtss_reg_write, 716
vtss_reg_write_masked, 716
vtss_sgpio_bmode_t, 710
vtss_sgpio_conf_get, 723
vtss_sgpio_conf_set, 723
vtss_sgpio_event_enable, 724
vtss_sgpio_event_poll, 724
vtss_sgpio_mode_t, 710
vtss_sgpio_read, 723
vtss_temp_sensor_get, 729
vtss_temp_sensor_init, 729
vtss_tod_get_ns_cnt, 728
vtss_tod_set_ns_cnt_cb, 728
vtss_trace_conf_get, 711
vtss_trace_conf_set, 712
vtss_trace_group_t, 707
vtss_trace_layer_t, 706
vtss_trace_level_t, 707
vtss_mll_tagtype_t
vtss_mpls_api.h, 739
vtss_mmd_read_inc_t
vtss_init_api.h, 639
vtss_mmd_read_t
vtss_init_api.h, 638
vtss_mmd_write_t
vtss_init_api.h, 639
vtss_mpls_api.h
VTSS_MPLS_IDX_IS_DEF, 736
VTSS_MPLS_IDX_IS_UNDEF, 736
VTSS_MPLS_IDX_UNDEFINED, 736
VTSS_MPLS_IDX_UNDEF, 737
VTSS_MPLS_LABEL_VALUE_DONTCARE, 736
VTSS_MPLS_QOS_TO_TC_ENTRY_CNT, 737
VTSS_MPLS_QOS_TO_TC_MAP_CNT, 737
VTSS_MPLS_TC_TO_QOS_ENTRY_CNT, 737
VTSS_MPLS_TC_TO_QOS_MAP_CNT, 737
VTSS_MPLS_TC_VALUE_DONTCARE, 736
vtss_mll_tagtype_t, 739
vtss_mpls_cos_t, 738
vtss_mpls_l2_alloc, 741
vtss_mpls_l2_free, 742
vtss_mpls_l2_get, 742
vtss_mpls_l2_idx_t, 739
vtss_mpls_l2_segment_attach, 743
vtss_mpls_l2_segment_detach, 744
vtss_mpls_l2_set, 742
vtss_mpls_label_value_t, 738
vtss_mpls_oam_type_t, 741
vtss_mpls_qos_t, 738
vtss_mpls_segment_alloc, 744
vtss_mpls_segment_free, 745
vtss_mpls_segment_get, 745
vtss_mpls_segment_idx_t, 738
vtss_mpls_segment_server_attach, 746
vtss_mpls_segment_server_detach, 747
vtss_mpls_segment_set, 745
vtss_mpls_segment_state_t, 739
vtss_mpls_segment_status_get, 746
vtss_mpls_tc_conf_get, 751
vtss_mpls_tc_conf_set, 752
vtss_mpls_tc_t, 738
vtss_mpls_tunnel_mode_t, 740
vtss_mpls_xc_alloc, 747
vtss_mpls_xc_counters_clear, 752
vtss_mpls_xc_counters_get, 752
vtss_mpls_xc_free, 748
vtss_mpls_xc_get, 748
vtss_mpls_xc_idx_t, 739
vtss_mpls_xc_segment_attach, 750
vtss_mpls_xc_segment_detach, 751
vtss_mpls_xc_set, 750
vtss_mpls_xc_type_t, 740
vtss_mpls_cos_t
vtss_mpls_api.h, 738
vtss_mpls_l2_alloc
vtss_mpls_api.h, 741
vtss_mpls_l2_free
vtss_mpls_api.h, 742
vtss_mpls_l2_get
vtss_mpls_api.h, 742
vtss_mpls_l2_idx_t
vtss_mpls_api.h, 739
vtss_mpls_l2_segment_attach
vtss_mpls_api.h, 743
vtss_mpls_l2_segment_detach
vtss_mpls_api.h, 744
vtss_mpls_l2_set
vtss_mpls_api.h, 742
vtss_mpls_l2_t, 176
dei, 178
pcp, 177
peer_mac, 177

port, 177
section_oam, 178
self_mac, 177
tag_type, 177
vid, 177
vtss_mpls_label_t, 178
tc, 179
ttl, 179
value, 179
vtss_mpls_label_value_t
vtss_mpls_api.h, 738
vtss_mpls_oam_conf_t, 179
type, 180
vtss_mpls_oam_type_t
vtss_mpls_api.h, 741
vtss_mpls_pw_conf_t, 180
cw, 181
is_pw, 180
process_cw, 181
vtss_mpls_qos_t
vtss_mpls_api.h, 738
vtss_mpls_qos_to_tc_map_entry_t, 181
dp0_tc, 182
dp1_tc, 182
vtss_mpls_segment_alloc
vtss_mpls_api.h, 744
vtss_mpls_segment_free
vtss_mpls_api.h, 745
vtss_mpls_segment_get
vtss_mpls_api.h, 745
vtss_mpls_segment_idx_t
vtss_mpls_api.h, 738
vtss_mpls_segment_server_attach
vtss_mpls_api.h, 746
vtss_mpls_segment_server_detach
vtss_mpls_api.h, 747
vtss_mpls_segment_set
vtss_mpls_api.h, 745
vtss_mpls_segment_state_t
vtss_mpls_api.h, 739
vtss_mpls_segment_status_get
vtss_mpls_api.h, 746
vtss_mpls_segment_status_t, 182
oam_active, 183
state, 182
vtss_mpls_segment_t, 183
e_lsp, 184
hqos_id, 185
is_in, 184
l2_idx, 184
l_lsp_cos, 184
label, 185
oam_conf, 185
pw_conf, 185
server_idx, 186
tc_qos_map_idx, 184
upstream, 185
xc_idx, 186
vtss_mpls_tc_conf_get
vtss_mpls_api.h, 751
vtss_mpls_tc_conf_set
vtss_mpls_api.h, 752
vtss_mpls_tc_conf_t, 186
qos_to_tc_map, 187
tc_to_qos_map, 187
vtss_mpls_tc_t
vtss_mpls_api.h, 738
vtss_mpls_tc_to_qos_map_entry_t, 187
dp, 188
qos, 187
vtss_mpls_tunnel_mode_t
vtss_mpls_api.h, 740
vtss_mpls_xc_alloc
vtss_mpls_api.h, 747
vtss_mpls_xc_counters_clear
vtss_mpls_api.h, 752
vtss_mpls_xc_counters_get
vtss_mpls_api.h, 752
vtss_mpls_xc_counters_t, 188
rx_discard, 189
rx_green, 188
rx_red, 189
rx_yellow, 189
tx_discard, 189
tx_green, 189
tx_yellow, 190
vtss_mpls_xc_free
vtss_mpls_api.h, 748
vtss_mpls_xc_get
vtss_mpls_api.h, 748
vtss_mpls_xc_idx_t
vtss_mpls_api.h, 739
vtss_mpls_xc_segment_attach
vtss_mpls_api.h, 750
vtss_mpls_xc_segment_detach
vtss_mpls_api.h, 751
vtss_mpls_xc_set
vtss_mpls_api.h, 750
vtss_mpls_xc_t, 190
flow_handle, 191
in_seg_idx, 191
out_seg_idx, 191
tc_tunnel_mode, 191
ttl_tunnel_mode, 191
type, 191
vtss_mpls_xc_type_t
vtss_mpls_api.h, 740
vtss_mstp_port_msti_state_get
vtss_l2_api.h, 669
vtss_mstp_port_msti_state_set
vtss_l2_api.h, 670
vtss_mstp_vlan_msti_get
vtss_l2_api.h, 668
vtss_mstp_vlan_msti_set
vtss_l2_api.h, 669
vtss_mtimer_t, 192

now, 192
timeout, 192
vtss_os_custom.h, 775
vtss_os_ecos.h, 783
vtss_npi_conf_get
vtss_packet_api.h, 801
vtss_npi_conf_set
vtss_packet_api.h, 802
vtss_npi_conf_t, 193
enable, 193
port_no, 193
vtss_oam_api.h
VTSS_OAM_CNT_ALL, 761
VTSS_OAM_CNT_CCM, 760
VTSS_OAM_CNT_LB, 761
VTSS_OAM_CNT_LM, 760
VTSS_OAM_CNT_SEL, 761
VTSS_OAM_CNT_TST, 761
VTSS_OAM_EVENT_MASK_ARRAY_SIZE, 758
VTSS_OAM_GENERIC_OPCODE_CFG_CNT, 758
VTSS_OAM_PATH_SERVICE_VOE_CNT, 757
VTSS_OAM_PORT_VOE_BASE_IDX, 757
VTSS_OAM_PORT_VOE_CNT, 757
VTSS_OAM_VOE_CNT, 758
VTSS_OAM_VOE_EVENT_CCM_LOC, 759
VTSS_OAM_VOE_EVENT_CCM_MEG_ID, 760
VTSS_OAM_VOE_EVENT_CCM_MEP_ID, 759
VTSS_OAM_VOE_EVENT_CCM_PERIOD, 758
VTSS_OAM_VOE_EVENT_CCM_PRIORITY, 759
VTSS_OAM_VOE_EVENT_CCM_RX_RDI, 759
VTSS_OAM_VOE_EVENT_CCM_ZERO_PERI↩OD, 759
VTSS_OAM_VOE_EVENT_MASK, 760
VTSS_OAM_VOE_EVENT_MEG_LEVEL, 760
VTSS_OAM_VOE_IDX_NONE, 758
vtss_oam_ccm_status_get, 766
vtss_oam_event_poll, 762
vtss_oam_ipt_conf_get, 769
vtss_oam_ipt_conf_set, 769
vtss_oam_pdu_seen_status_get, 766
vtss_oam_proc_status_get, 767
vtss_oam_voe_alloc, 762
vtss_oam_voe_ccm_arm_hitme, 765
vtss_oam_voe_ccm_set_rdi_flag, 765
vtss_oam_voe_conf_get, 763
vtss_oam_voe_conf_set, 764
vtss_oam_voe_counter_clear, 768
vtss_oam_voe_counter_get, 768
vtss_oam_voe_counter_update, 767
vtss_oam_voe_counter_update_serval, 768
vtss_oam_voe_event_enable, 764
vtss_oam_voe_event_poll, 765
vtss_oam_voe_free, 763
vtss_oam_vop_conf_get, 761
vtss_oam_vop_conf_set, 762
vtss_oam_ccm_counter_t, 194
rx_invalid_count, 194
rx_invalid_seq_no, 195
rx_valid_count, 194
vtss_oam_ccm_status_get
vtss_oam_api.h, 766
vtss_oam_ccm_status_t, 195
loc, 196
meg_id_err, 197
mep_id_err, 196
period_err, 195
priority_err, 196
rx_rdi, 196
zero_period_err, 196
vtss_oam_event_mask_t, 197
voe_mask, 197
vtss_oam_event_poll
vtss_oam_api.h, 762
vtss_oam_ipt_conf_get
vtss_oam_api.h, 769
vtss_oam_ipt_conf_set
vtss_oam_api.h, 769
vtss_oam_ipt_conf_t, 198
enable, 198
service_voe_idx, 198
vtss_oam_lb_counter_t, 199
rx_lbr, 199
rx_lbr_trans_id_err, 200
tx_lbm, 199
vtss_oam_lm_counter_t, 200
ccm_lm, 202
ccm_lm_sample_lost, 203
ccm_lm_valid, 203
down_mep, 201
lm_count, 204
lmm, 202
lmm_sample_lost, 203
lmm_valid, 202
lmr, 202
lmr_sample_lost, 203
lmr_valid, 203
rx_FCl, 201
rx_ccm_lm_sample_seq_no, 202
rx_lmm_sample_seq_no, 201
rx_lmr_sample_seq_no, 201
tx_FCf, 201
up_mep, 204
vtss_oam_meg_id_t, 204
data, 205
vtss_oam_pdu_seen_status_get
vtss_oam_api.h, 766
vtss_oam_pdu_seen_status_t, 205
ccm_lm_seen, 208
ccm_seen, 208
dmm_seen, 207
dmr_seen, 207
generic_seen, 206
lbm_seen, 207
lbr_seen, 207
lmm_seen, 206

lmr_seen, 206
ltm_seen, 206
ltr_seen, 206
one_dm_seen, 207
unknown_seen, 206
vtss_oam_proc_conf_t, 208
ccm_check_only, 209
copy_next_only, 209
copy_on_ccm_err, 209
copy_on_ccm_more_than_one_tlv, 210
copy_on_dmac_err, 210
copy_on_mel_too_low_err, 210
dmac_check_type, 209
meg_level, 209
vtss_oam_proc_status_get
vtss_oam_api.h, 767
vtss_oam_proc_status_t, 210
rx_ccm_seq_no, 211
rx_dmac_err_seen, 212
rx_lbr_transaction_id, 211
rx_meg_level_err, 212
rx_meg_level_err_seen, 212
rx_tst_seq_no, 212
tx_meg_level_err_seen, 213
tx_next_ccm_seq_no, 211
tx_next_lbm_transaction_id, 212
vtss_oam_rx_tx_counter_t, 213
rx, 213
tx, 214
vtss_oam_sel_counter_t, 214
nonselected_frames, 215
selected_frames, 214
vtss_oam_timestamp_get
vtss_ts_api.h, 1015
vtss_oam_ts_id_s, 215
voe_id, 215
voe_sq, 216
vtss_oam_ts_timestamp_s, 216
port_no, 217
ts, 216
ts_valid, 217
vtss_oam_tst_counter_t, 217
rx_tst, 218
rx_tst_trans_id_err, 218
tx_tst, 218
vtss_oam_voe_alloc
vtss_oam_api.h, 762
vtss_oam_voe_alloc_cfg_t, 218
phys_port, 219
vtss_oam_voe_ccm_arm_hitme
vtss_oam_api.h, 765
vtss_oam_voe_ccm_conf_t, 219
copy_to_cpu, 220
count_as_data, 220
count_as_selected, 220
enable, 220
forward, 220
megid, 221
mepid, 220
rx_period, 222
rx_priority, 222
rx_seq_no, 221
rx_seq_no_check, 221
tx_seq_no, 221
tx_seq_no_auto_upd_op, 221
vtss_oam_voe_ccm_lm_conf_t, 222
copy_to_cpu, 223
count, 223
count_as_selected, 223
enable, 223
forward, 223
period, 224
vtss_oam_voe_ccm_set_rdi_flag
vtss_oam_api.h, 765
vtss_oam_voe_conf_get
vtss_oam_api.h, 763
vtss_oam_voe_conf_set
vtss_oam_api.h, 764
vtss_oam_voe_conf_t, 224
ccm, 227
ccm_lm, 228
dm, 228
generic, 227
lb, 228
loop_isdx_p, 226
loop_isdx_w, 226
loop_portidx_p, 226
lt, 229
mep_type, 225
proc, 227
sdlb_enable, 226
sdlb_idx, 227
single_ended_lm, 228
svc_to_path, 225
svc_to_path_idx_p, 226
svc_to_path_idx_w, 225
tst, 228
unicast_mac, 225
unknown, 227
upmep, 229
voe_type, 225
vtss_oam_voe_counter_clear
vtss_oam_api.h, 768
vtss_oam_voe_counter_get
vtss_oam_api.h, 768
vtss_oam_voe_counter_t, 229
ccm, 230
lb, 230
lm, 230
sel, 230
tst, 230
vtss_oam_voe_counter_update
vtss_oam_api.h, 767
vtss_oam_voe_counter_update_serval
vtss_oam_api.h, 768
vtss_oam_voe_dm_conf_t, 231

copy_1dm_to_cpu, 232
copy_dmm_to_cpu, 232
copy_dmr_to_cpu, 232
count_as_data, 233
count_as_selected, 233
enable_1dm, 231
enable_dmm, 231
forward_1dm, 232
forward_dmm, 232
forward_dmr, 233
vtss_oam_voe_event_enable
vtss_oam_api.h, 764
vtss_oam_voe_event_poll
vtss_oam_api.h, 765
vtss_oam_voe_free
vtss_oam_api.h, 763
vtss_oam_voe_generic_conf_t, 233
copy_to_cpu, 234
count_as_data, 235
count_as_selected, 234
enable, 234
forward, 234
vtss_oam_voe_lb_conf_t, 235
copy_lbm_to_cpu, 236
copy_lbr_to_cpu, 236
count_as_data, 237
count_as_selected, 236
enable, 236
forward_lbm, 236
forward_lbr, 236
rx_transaction_id, 237
tx_transaction_id, 237
tx_update_transaction_id, 237
vtss_oam_voe_lm_counter_conf_t, 238
priority_mask, 238
yellow, 238
vtss_oam_voe_lt_conf_t, 238
copy_ltm_to_cpu, 239
copy_ltr_to_cpu, 239
count_as_data, 240
count_as_selected, 240
enable, 239
forward_ltm, 239
forward_ltr, 240
vtss_oam_voe_se_lm_conf_t, 240
copy_lmm_to_cpu, 241
copy_lmr_to_cpu, 241
count, 242
count_as_data, 242
count_as_selected, 242
enable, 241
forward_lmm, 241
forward_lmr, 242
vtss_oam_voe_tst_conf_t, 243
copy_to_cpu, 243
count_as_data, 244
count_as_selected, 244
enable, 243
forward, 243
rx_seq_no, 244
tx_seq_no, 244
tx_seq_no_auto_update, 244
vtss_oam_voe_unknown_conf_t, 245
copy_to_cpu, 245
count_as_data, 246
count_as_selected, 246
enable, 245
vtss_oam_voe_up_mep_conf_t, 246
discard_rx, 247
loopback, 247
port, 247
vtss_oam_vop_conf_get
vtss_oam_api.h, 761
vtss_oam_vop_conf_set
vtss_oam_api.h, 762
vtss_oam_vop_conf_t, 247
ccm_lm_enable_rx_fcf_in_reserved_field, 248
common_multicast_dmac, 248
down_mep_lmr_proprietary_fcf_use, 248
enable_all_voe, 248
external_cpu_portmask, 248
pdu_type, 249
sdlb_cpy_copy_idx, 249
vtss_oam_vop_extract_conf_t, 249
rx_queue, 250
to_front, 250
vtss_oam_vop_generic_opcode_conf_t, 250
check_dmac, 251
extract, 251
opcode, 251
vtss_oam_vop_pdu_type_conf_t, 251
ccm, 252
ccm_lm, 252
dmm, 253
dmr, 253
err, 254
generic, 252
lbm, 254
lbr, 254
lmm, 253
lmr, 253
lt, 253
other, 254
vtss_os_custom.h
uint, 771
ulong, 771
VTSS_DIV64, 772
VTSS_LABS, 773
VTSS_LLABS, 773
VTSS_MOD64, 772
VTSS_MSLEEP, 771
VTSS_MTIMER_CANCEL, 772
VTSS_MTIMER_START, 772
VTSS_MTIMER_TIMEOUT, 772
VTSS_OS_CTZ64, 773
VTSS_OS_CTZ, 773

VTSS_OS_FREE, 774
VTSS_OS_MALLOC, 774
VTSS_OS_RAND, 774
vtss_mtimer_t, 775
vtss_os_ecos.h
llabs, 784
VTSS_DIV64, 777
VTSS_LABS, 778
VTSS_LLABS, 778
VTSS_MOD64, 778
VTSS_MSLEEP, 776
VTSS_MTIMER_CANCEL, 777
VTSS_MTIMER_START, 777
VTSS_MTIMER_TIMEOUT, 777
VTSS_NSLEEP, 776
VTSS_OS_BIG_ENDIAN, 781
VTSS_OS_COMPILER_ATTRIBUTE_ALIGNED, 780
VTSS_OS_CTZ64, 779
VTSS_OS_CTZ, 778
VTSS_OS_DCACHE_FLUSH, 781
VTSS_OS_DCACHE_INVALIDATE, 781
VTSS_OS_DCACHE_LINE_SIZE_BYTES, 780
VTSS_OS_FREE, 779
VTSS_OS_INTERRUPT_DISABLE, 783
VTSS_OS_INTERRUPT_FLAGS, 783
VTSS_OS_INTERRUPT_RESTORE, 783
VTSS_OS_MALLOC, 779
VTSS_OS_NTOHL, 782
VTSS_OS_RAND, 780
VTSS_OS_REORDER_BARRIER, 780
VTSS_OS_SCHEDULER_FLAGS, 782
VTSS_OS_SCHEDULER_LOCK, 782
VTSS_OS_SCHEDULER_UNLOCK, 782
VTSS_OS_VIRT_TO_PHYS, 781
VTSS_TIME_OF_DAY, 777
vtss_callout_free, 784
vtss_callout_malloc, 784
vtss_mtimer_t, 783
vtss_os_linux.h
VTSS_DIV64, 789
VTSS_LABS, 789
VTSS_LLABS, 790
VTSS_MOD64, 789
VTSS_MSLEEP, 787
VTSS_MTIMER_CANCEL, 788
VTSS_MTIMER_START, 787
VTSS_MTIMER_TIMEOUT, 787
VTSS_NSLEEP, 786
VTSS_OS_BIG_ENDIAN, 786
VTSS_OS_CTZ64, 790
VTSS_OS_CTZ, 790
VTSS_OS_FREE, 791
VTSS_OS_MALLOC, 791
VTSS_OS_NTOHL, 786
VTSS_OS_RAND, 791
VTSS_OS_SCHEDULER_FLAGS, 788
VTSS_OS_SCHEDULER_LOCK, 788
VTSS_OS_SCHEDULER_UNLOCK, 789
VTSS_TIME_OF_DAY, 788
vtss_os_timestamp_t, 255
hw_cnt, 255
vtss_packet_api.h
VTSS_AFI_FPS_MAX, 796
VTSS_AFI_ID_NONE, 796
VTSS_AFI_SLOT_CNT, 796
VTSS_JR1_PACKET_HDR_SIZE_BYTES, 796
VTSS_JR1_RX_IFH_SIZE, 797
VTSS_JR2_PACKET_HDR_SIZE_BYTES, 796
VTSS_JR2_RX_IFH_SIZE, 798
VTSS_L26_PACKET_HDR_SIZE_BYTES, 797
VTSS_L26_RX_IFH_SIZE, 798
VTSS_PACKET_HDR_SIZE_BYTES, 797
VTSS_PACKET_TX_IFH_MAX, 798
VTSS_PRIO_SUPER, 795
VTSS_SVL_PACKET_HDR_SIZE_BYTES, 797
VTSS_SVL_RX_IFH_SIZE, 797
vtss_afi_alloc, 809
vtss_afi_free, 809
vtss_npi_conf_get, 801
vtss_npi_conf_set, 802
vtss_packet_dma_conf_get, 812
vtss_packet_dma_conf_set, 813
vtss_packet_dma_offset, 813
vtss_packet_filter_t, 798
vtss_packet_frame_filter, 807
vtss_packet_frame_info_init, 807
vtss_packet_oam_type_t, 799
vtss_packet_port_filter_get, 808
vtss_packet_port_info_init, 808
vtss_packet_ptp_action_t, 799
vtss_packet_rx_conf_get, 802
vtss_packet_rx_conf_set, 803
vtss_packet_rx_frame_discard, 805
vtss_packet_rx_frame_get, 804
vtss_packet_rx_frame_get_raw, 804
vtss_packet_rx_hdr_decode, 809
vtss_packet_rx_hints_t, 800
vtss_packet_rx_port_conf_get, 803
vtss_packet_rx_port_conf_set, 803
vtss_packet_tx_frame, 811
vtss_packet_tx_frame_port, 806
vtss_packet_tx_frame_port_vlan, 806
vtss_packet_tx_frame_vlan, 807
vtss_packet_tx_hdr_compile, 811
vtss_packet_tx_hdr_encode, 810
vtss_packet_tx_info_init, 812
vtss_packet_tx_vstax_t, 801
vtss_tag_type_t, 800
vtss_packet_dma_conf_get
vtss_packet_api.h, 812
vtss_packet_dma_conf_set
vtss_packet_api.h, 813
vtss_packet_dma_conf_t, 255
dma_enable, 256
vtss_packet_dma_offset

vtss_packet_api.h, 813
vtss_packet_filter_t
vtss_packet_api.h, 798
vtss_packet_frame_filter
vtss_packet_api.h, 807
vtss_packet_frame_info_init
vtss_packet_api.h, 807
vtss_packet_frame_info_t, 256
aggr_rx_disable, 257
aggr_tx_disable, 257
port_no, 256
port_tx, 257
vid, 256
vtss_packet_oam_type_t
vtss_packet_api.h, 799
vtss_packet_port_filter_get
vtss_packet_api.h, 808
vtss_packet_port_filter_t, 257
filter, 258
tpid, 258
vtss_packet_port_info_init
vtss_packet_api.h, 808
vtss_packet_port_info_t, 258
aggr_rx_disable, 259
aggr_tx_disable, 259
port_no, 259
vid, 259
vtss_packet_ptp_action_t
vtss_packet_api.h, 799
vtss_packet_reg_type_t
types.h, 586
vtss_packet_rx_conf_get
vtss_packet_api.h, 802
vtss_packet_rx_conf_set
vtss_packet_api.h, 803
vtss_packet_rx_conf_t, 260
grp_map, 261
map, 261
queue, 260
reg, 260
shaper_rate, 261
vtss_packet_rx_frame_discard
vtss_packet_api.h, 805
vtss_packet_rx_frame_get
vtss_packet_api.h, 804
vtss_packet_rx_frame_get_raw
vtss_packet_api.h, 804
vtss_packet_rx_grp_t
types.h, 579
vtss_packet_rx_hdr_decode
vtss_packet_api.h, 809
vtss_packet_rx_header_t, 261
arrived_tagged, 263
learn, 262
length, 262
port_no, 262
queue_mask, 262
tag, 263
vtss_packet_rx_hints_t
vtss_packet_api.h, 800
vtss_packet_rx_info_t, 263
acl_hit, 268
acl_idx, 268
cos, 268
glag_no, 265
hints, 264
hw_tstamp, 270
hw_tstamp_decoded, 270
isdx, 272
length, 265
oam_info, 271
oam_info_decoded, 272
port_no, 265
sflow_port_no, 271
sflow_type, 270
stripped_tag, 267
sw_tstamp, 269
tag, 266
tag_type, 266
tstamp_id, 269
tstamp_id_decoded, 269
xtr_qu_mask, 267
vtss_packet_rx_meta_t, 273
chip_no, 274
etype, 275
fcs, 275
length, 276
no_wait, 273
sw_tstamp, 276
xtr_qu, 274
vtss_packet_rx_port_conf_get
vtss_packet_api.h, 803
vtss_packet_rx_port_conf_set
vtss_packet_api.h, 803
vtss_packet_rx_port_conf_t, 277
bpdu_reg, 278
garp_reg, 278
igmp_reg, 277
ipmc_ctrl_reg, 277
mld_reg, 278
vtss_packet_rx_queue_conf_t, 278
npi, 279
size, 279
vtss_packet_rx_queue_map_t, 279
bpdu_queue, 280
garp_queue, 280
igmp_queue, 280
ipmc_ctrl_queue, 281
learn_queue, 280
lrn_all_queue, 281
mac_vid_queue, 281
sflow_queue, 281
stack_queue, 281
vtss_packet_rx_queue_npi_conf_t, 282
enable, 282
vtss_packet_rx_reg_t, 283

bpdu_cpu_only, 283
garp_cpu_only, 283
igmp_cpu_only, 284
ipmc_ctrl_cpu_copy, 283
mld_cpu_only, 284
vtss_packet_tx_frame
vtss_packet_api.h, 811
vtss_packet_tx_frame_port
vtss_packet_api.h, 806
vtss_packet_tx_frame_port_vlan
vtss_packet_api.h, 806
vtss_packet_tx_frame_vlan
vtss_packet_api.h, 807
vtss_packet_tx_grp_t
types.h, 579
vtss_packet_tx_hdr_compile
vtss_packet_api.h, 811
vtss_packet_tx_hdr_encode
vtss_packet_api.h, 810
vtss_packet_tx_ifh_t, 284
ifh, 285
length, 285
vtss_packet_tx_info_init
vtss_packet_api.h, 812
vtss_packet_tx_info_t, 285
afi_id, 294
aggr_code, 288
cos, 289
dp, 293
dst_port_mask, 286
frm_len, 287
isdx, 292
isdx_dont_use, 292
latch_timestamp, 291
masquerade_port, 293
oam_type, 291
pdu_offset, 294
ptp_action, 289
ptp_id, 290
ptp_timestamp, 290
switch_frm, 286
tag, 287
vtss_packet_tx_vstax_t
vtss_packet_api.h, 801
vtss_phy_10g_fifo_sync_t, 295
bypass_in_api, 295
pr, 296
skip_rev_check, 295
vtss_phy_1588_csr_reg_read
vtss_phy_ts_api.h, 957
vtss_phy_1588_csr_reg_write
vtss_phy_ts_api.h, 957
vtss_phy_1588_debug_reg_read
vtss_phy_ts_api.h, 960
vtss_phy_1g_ts_fifo_sync
vtss_phy_ts_api.h, 959
vtss_phy_aneg_t, 296
asymmetric_pause, 298
speed_100m_fdx, 297
speed_100m_hdx, 297
speed_10m_fdx, 297
speed_10m_hdx, 297
speed_1g_fdx, 297
speed_1g_hdx, 297
symmetric_pause, 298
tx_remote_fault, 298
vtss_phy_api.h
lb_type, 844
MAX_CFG_BUF_SIZE, 824
MAX_STAT_BUF_SIZE, 824
MAX_WOL_MAC_ADDR_SIZE, 835
MAX_WOL_PASSWD_SIZE, 835
MIN_WOL_PASSWD_SIZE, 835
rgmii_skew_delay_psec_t, 838
VTSS_PHY_ACTIPHY_PWR, 825
VTSS_PHY_LINK_AMS_EV, 830
VTSS_PHY_LINK_AUTO_NEG_COMPLETE_EV, 830
VTSS_PHY_LINK_AUTO_NEG_ERROR_EV, 830
VTSS_PHY_LINK_DOWN_PWR, 825
VTSS_PHY_LINK_EXT_EEE_LINKFAIL_EV, 833
VTSS_PHY_LINK_EXT_EEE_WAIT_RX_TQ_EV, 833
VTSS_PHY_LINK_EXT_EEE_WAIT_TS_EV, 833
VTSS_PHY_LINK_EXT_EEE_WAKE_ERR_EV, 833
VTSS_PHY_LINK_EXT_MACSEC_EGRESS_EV, 834
VTSS_PHY_LINK_EXT_MACSEC_FC_BUFF_↩EV, 834
VTSS_PHY_LINK_EXT_MACSEC_HOST_MAC↩_EV, 834
VTSS_PHY_LINK_EXT_MACSEC_INGRESS_↩EV, 834
VTSS_PHY_LINK_EXT_MACSEC_LINE_MAC_↩EV, 834
VTSS_PHY_LINK_EXT_MEM_INT_RING_EV, 835
VTSS_PHY_LINK_EXT_RR_SW_COMPL_EV, 833
VTSS_PHY_LINK_EXTENDED_REG_INT_EV, 832
VTSS_PHY_LINK_FALSE_CARRIER_INT_EV, 831
VTSS_PHY_LINK_FDX_STATE_CHANGE_EV, 830
VTSS_PHY_LINK_FFAIL_EV, 829
VTSS_PHY_LINK_INLINE_POW_DEV_DETEC↩T_EV, 831
VTSS_PHY_LINK_LINK_SPEED_DS_DETECT↩_EV, 832
VTSS_PHY_LINK_LOS_EV, 829
VTSS_PHY_LINK_MASTER_SLAVE_RES_ER↩R_EV, 832
VTSS_PHY_LINK_RX_ER_INT_EV, 832
VTSS_PHY_LINK_RX_FIFO_OVERFLOW_INT↩

_EV, 831
VTSS_PHY_LINK_SPEED_STATE_CHANGE_↩ EV, 830
VTSS_PHY_LINK_SYMBOL_ERR_INT_EV, 831
VTSS_PHY_LINK_TX_FIFO_OVERFLOW_INT↩ _EV, 831
VTSS_PHY_LINK_UP_FULL_PWR, 825
VTSS_PHY_LINK_WAKE_ON_LAN_INT_EV, 832
VTSS_PHY_PAGE_0x2DAF, 828
VTSS_PHY_PAGE_1588, 827
VTSS_PHY_PAGE_EXTENDED_2, 826
VTSS_PHY_PAGE_EXTENDED_3, 827
VTSS_PHY_PAGE_EXTENDED_4, 827
VTSS_PHY_PAGE_EXTENDED, 826
VTSS_PHY_PAGE_GPIO, 827
VTSS_PHY_PAGE_MACSEC, 827
VTSS_PHY_PAGE_STANDARD, 826
VTSS_PHY_PAGE_TEST, 828
VTSS_PHY_PAGE_TR, 828
VTSS_PHY_POWER_ACTIPHY_BIT, 824
VTSS_PHY_POWER_DYNAMIC_BIT, 825
VTSS_PHY_RECOV_CLK1, 825
VTSS_PHY_RECOV_CLK2, 826
VTSS_PHY_RECOV_CLK_NUM, 826
VTSS_PHY_REG_EXTENDED, 828
VTSS_PHY_REG_GPIO, 829
VTSS_PHY_REG_STANDARD, 828
VTSS_PHY_REG_TEST, 829
VTSS_PHY_REG_TR, 829
vga_adc_debug, 864
vtss_fefi_mode_t, 845
vtss_phy_atom12_patch_setttings_get, 874
vtss_phy_cfg_ib_cterm, 874
vtss_phy_cfg_ob_post0, 873
vtss_phy_chip_temp_get, 847
vtss_phy_chip_temp_init, 848
vtss_phy_cl37_lp_abil_get, 849
vtss_phy_clk_source_t, 843
vtss_phy_clk_squelch, 843
vtss_phy_clock_conf_get, 853
vtss_phy_clock_conf_set, 853
vtss_phy_coma_mode_disable, 863
vtss_phy_coma_mode_enable, 864
vtss_phy_conf_1g_get, 850
vtss_phy_conf_1g_set, 851
vtss_phy_conf_get, 848
vtss_phy_conf_set, 849
vtss_phy_csr_rd, 869
vtss_phy_csr_wr, 869
vtss_phy_debug_phyinfo_print, 877
vtss_phy_debug_regdump_print, 879
vtss_phy_debug_register_dump, 877
vtss_phy_debug_stat_print, 876
vtss_phy_detect_base_ports, 878
vtss_phy_do_page_chk_get, 872
vtss_phy_do_page_chk_set, 871
vtss_phy_eee_conf_get, 865
vtss_phy_eee_conf_set, 866
vtss_phy_eee_ena, 865
vtss_phy_eee_link_partner_advertisements_get, 867
vtss_phy_eee_power_save_state_get, 866
vtss_phy_enhanced_led_control_init, 862
vtss_phy_enhanced_led_control_init_get, 863
vtss_phy_event_enable_get, 867
vtss_phy_event_enable_set, 867
vtss_phy_event_poll, 868
vtss_phy_ext_connector_loopback, 878
vtss_phy_ext_event_poll, 889
vtss_phy_fast_link_fail_t, 839
vtss_phy_fefi_detect, 887
vtss_phy_fefi_get, 886
vtss_phy_fefi_set, 886
vtss_phy_flowcontrol_decode_status, 875
vtss_phy_forced_reset_t, 838
vtss_phy_freq_t, 843
vtss_phy_gpio_get, 859
vtss_phy_gpio_mode, 859
vtss_phy_gpio_mode_t, 844
vtss_phy_gpio_set, 860
vtss_phy_i2c_read, 854
vtss_phy_i2c_write, 855
vtss_phy_id_get, 850
vtss_phy_is_8051_crc_ok, 873
vtss_phy_is_viper_revB, 889
vtss_phy_lcpll_status_get, 883
vtss_phy_led_intensity, 836
vtss_phy_led_intensity_get, 862
vtss_phy_led_intensity_set, 861
vtss_phy_led_mode_set, 861
vtss_phy_led_mode_t, 836
vtss_phy_loopback_get, 872
vtss_phy_loopback_set, 872
vtss_phy_mac_media_inhibit_odd_start, 885
vtss_phy_mac_serd_pcs_sgmii_pre, 840
vtss_phy_macsec_csr_sd6g_rd, 890
vtss_phy_macsec_csr_sd6g_wr, 890
vtss_phy_mdi_t, 838
vtss_phy_media_force_ams_sel_t, 842
vtss_phy_media_interface_t, 837
vtss_phy_media_rem_fault_t, 842
vtss_phy_mmd_read, 856
vtss_phy_mmd_write, 857
vtss_phy_mode_t, 839
vtss_phy_mse_1000m_get, 888
vtss_phy_mse_100m_get, 887
vtss_phy_patch_settings_get, 881
vtss_phy_pkt_mode_t, 839
vtss_phy_port_eee_capable, 864
vtss_phy_post_reset, 846
vtss_phy_power_conf_get, 852
vtss_phy_power_conf_set, 852
vtss_phy_power_status_get, 852
vtss_phy_pre_reset, 845
vtss_phy_pre_system_reset, 846
vtss_phy_read, 855

vtss_phy_read_page, 856
vtss_phy_read_tr_addr, 888
vtss_phy_recov_clk_t, 836
vtss_phy_reg_decode_status, 875
vtss_phy_reset, 846
vtss_phy_reset_get, 847
vtss_phy_reset_lcpll, 883
vtss_phy_sd6g_mac_serdes_conf, 891
vtss_phy_sd6g_ob_lev_rd, 885
vtss_phy_sd6g_ob_lev_wr, 885
vtss_phy_sd6g_ob_post_rd, 884
vtss_phy_sd6g_ob_post_wr, 884
vtss_phy_serdes1g_rcpll_status_get, 882
vtss_phy_serdes6g_rcpll_status_get, 882
vtss_phy_serdes_fmedia_loopback, 879
vtss_phy_serdes_sgmii_loopback, 878
vtss_phy_sigdet_polarity_t, 840
vtss_phy_statistic_get, 871
vtss_phy_status_1g_get, 851
vtss_phy_status_get, 849
vtss_phy_status_inst_poll, 890
vtss_phy_unidirectional_t, 840
vtss_phy_veriphy_get, 861
vtss_phy_veriphy_start, 860
vtss_phy_warm_start_failed_get, 876
vtss_phy_wol_conf_get, 880
vtss_phy_wol_conf_set, 881
vtss_phy_wol_enable, 880
vtss_phy_write, 857
vtss_phy_write_masked, 858
vtss_phy_write_masked_page, 858
vtss_squelch_workaround, 868
vtss_wol_passwd_len_type_t, 845
vtss_phy_atom12_patch_setttings_get
vtss_phy_api.h, 874
vtss_phy_cfg_ib_cterm
vtss_phy_api.h, 874
vtss_phy_cfg_ob_post0
vtss_phy_api.h, 873
vtss_phy_chip_temp_get
vtss_phy_api.h, 847
vtss_phy_chip_temp_init
vtss_phy_api.h, 848
vtss_phy_cl37_lp_abil_get
vtss_phy_api.h, 849
vtss_phy_clk_source_t
vtss_phy_api.h, 843
vtss_phy_clk_squelch
vtss_phy_api.h, 843
vtss_phy_clock_conf_get
vtss_phy_api.h, 853
vtss_phy_clock_conf_set
vtss_phy_api.h, 853
vtss_phy_clock_conf_t, 298
freq, 299
squelch, 299
src, 299
vtss_phy_coma_mode_disable
vtss_phy_api.h, 863
vtss_phy_coma_mode_enable
vtss_phy_api.h, 864
vtss_phy_conf_1g_get
vtss_phy_api.h, 850
vtss_phy_conf_1g_set
vtss_phy_api.h, 851
vtss_phy_conf_1g_t, 300
cfg, 300
master, 300
val, 300
vtss_phy_conf_get
vtss_phy_api.h, 848
vtss_phy_conf_set
vtss_phy_api.h, 849
vtss_phy_conf_t, 301
aneg, 302
flf, 302
force_ams_sel, 303
forced, 301
mac_if_pcs, 303
mdi, 302
media_if_pcs, 303
mode, 301
sigdet, 302
skip_coma, 303
unidir, 302
vtss_phy_csr_rd
vtss_phy_api.h, 869
vtss_phy_csr_wr
vtss_phy_api.h, 869
vtss_phy_daisy_chain_conf_t, 304
spi_daisy_input, 304
spi_daisy_output, 304
vtss_phy_daisy_conf_get
vtss_phy_ts_api.h, 937
vtss_phy_daisy_conf_set
vtss_phy_ts_api.h, 937
vtss_phy_debug_phyinfo_print
vtss_phy_api.h, 877
vtss_phy_debug_regdump_print
vtss_phy_api.h, 879
vtss_phy_debug_register_dump
vtss_phy_api.h, 877
vtss_phy_debug_stat_print
vtss_phy_api.h, 876
vtss_phy_detect_base_ports
vtss_phy_api.h, 878
vtss_phy_do_page_chk_get
vtss_phy_api.h, 872
vtss_phy_do_page_chk_set
vtss_phy_api.h, 871
vtss_phy_eee_conf_get
vtss_phy_api.h, 865
vtss_phy_eee_conf_set
vtss_phy_api.h, 866
vtss_phy_eee_conf_t, 304
eee_ena_phy, 305

eee_mode, 305
vtss_phy_eee_ena
vtss_phy_api.h, 865
vtss_phy_eee_link_partner_advertisements_get
vtss_phy_api.h, 867
vtss_phy_eee_power_save_state_get
vtss_phy_api.h, 866
vtss_phy_enhanced_led_control_init
vtss_phy_api.h, 862
vtss_phy_enhanced_led_control_init_get
vtss_phy_api.h, 863
vtss_phy_enhanced_led_control_t, 305
ser_led_frame_rate, 306
ser_led_output_1, 306
ser_led_output_2, 306
ser_led_select, 306
vtss_phy_event_enable_get
vtss_phy_api.h, 867
vtss_phy_event_enable_set
vtss_phy_api.h, 867
vtss_phy_event_poll
vtss_phy_api.h, 868
vtss_phy_ext_connector_loopback
vtss_phy_api.h, 878
vtss_phy_ext_event_poll
vtss_phy_api.h, 889
vtss_phy_fast_link_fail_t
vtss_phy_api.h, 839
vtss_phy_fefi_detect
vtss_phy_api.h, 887
vtss_phy_fefi_get
vtss_phy_api.h, 886
vtss_phy_fefi_set
vtss_phy_api.h, 886
vtss_phy_flowcontrol_decode_status
vtss_phy_api.h, 875
vtss_phy_forced_reset_t
vtss_phy_api.h, 838
vtss_phy_forced_t, 307
fdx, 307
speed, 307
vtss_phy_freq_t
vtss_phy_api.h, 843
vtss_phy_gpio_get
vtss_phy_api.h, 859
vtss_phy_gpio_mode
vtss_phy_api.h, 859
vtss_phy_gpio_mode_t
vtss_phy_api.h, 844
vtss_phy_gpio_set
vtss_phy_api.h, 860
vtss_phy_i2c_read
vtss_phy_api.h, 854
vtss_phy_i2c_write
vtss_phy_api.h, 855
vtss_phy_id_get
vtss_phy_api.h, 850
vtss_phy_is_8051_crc_ok
vtss_phy_api.h, 873
vtss_phy_is_viper_revB
vtss_phy_api.h, 889
vtss_phy_lcpll_status_get
vtss_phy_api.h, 883
vtss_phy_led_intensity
vtss_phy_api.h, 836
vtss_phy_led_intensity_get
vtss_phy_api.h, 862
vtss_phy_led_intensity_set
vtss_phy_api.h, 861
vtss_phy_led_mode_select_t, 308
mode, 308
number, 308
vtss_phy_led_mode_set
vtss_phy_api.h, 861
vtss_phy_led_mode_t
vtss_phy_api.h, 836
vtss_phy_loopback_get
vtss_phy_api.h, 872
vtss_phy_loopback_set
vtss_phy_api.h, 872
vtss_phy_loopback_t, 309
connector_enable, 310
far_end_enable, 309
mac_serdes_equipment_enable, 310
mac_serdes_facility_enable, 310
mac_serdes_input_enable, 310
media_serdes_equipment_enable, 311
media_serdes_facility_enable, 311
media_serdes_input_enable, 310
near_end_enable, 309
vtss_phy_ltc_freq_synth_s, 311
enable, 312
high_duty_cycle, 312
low_duty_cycle, 312
vtss_phy_mac_media_inhibit_odd_start
vtss_phy_api.h, 885
vtss_phy_mac_serd_pcs_cntl_t, 312
aneg_restart, 314
disable, 313
fast_link_stat_ena, 315
force_adv_ability, 314
inhibit_odd_start, 315
pd_enable, 313
restart, 313
serdes_aneg_ena, 314
serdes_pol_inv_in, 315
serdes_pol_inv_out, 315
sgmii_in_pre, 314
sgmii_out_pre, 314
vtss_phy_mac_serd_pcs_sgmii_pre
vtss_phy_api.h, 840
vtss_phy_macsec_csr_sd6g_rd
vtss_phy_api.h, 890
vtss_phy_macsec_csr_sd6g_wr
vtss_phy_api.h, 890
vtss_phy_mdi_t

vtss_phy_api.h, 838
vtss_phy_media_force_ams_sel_t
vtss_phy_api.h, 842
vtss_phy_media_interface_t
vtss_phy_api.h, 837
vtss_phy_media_rem_fault_t
vtss_phy_api.h, 842
vtss_phy_media_serd_pcs_cntl_t, 316
aneg_pd_detect, 316
force_adv_ability, 316
force_fefi, 317
force_fefi_value, 318
force_hls, 317
inhibit_odd_start, 317
remote_fault, 316
serdes_pol_inv_in, 317
serdes_pol_inv_out, 317
vtss_phy_mmd_read
vtss_phy_api.h, 856
vtss_phy_mmd_write
vtss_phy_api.h, 857
vtss_phy_mode_t
vtss_phy_api.h, 839
vtss_phy_mse_1000m_get
vtss_phy_api.h, 888
vtss_phy_mse_100m_get
vtss_phy_api.h, 887
vtss_phy_patch_settings_get
vtss_phy_api.h, 881
vtss_phy_pkt_mode_t
vtss_phy_api.h, 839
vtss_phy_port_eee_capable
vtss_phy_api.h, 864
vtss_phy_post_reset
vtss_phy_api.h, 846
vtss_phy_power_conf_get
vtss_phy_api.h, 852
vtss_phy_power_conf_set
vtss_phy_api.h, 852
vtss_phy_power_conf_t, 318
mode, 318
vtss_phy_power_mode_t
phy.h, 536
vtss_phy_power_status_get
vtss_phy_api.h, 852
vtss_phy_power_status_t, 319
level, 319
vtss_phy_pre_reset
vtss_phy_api.h, 845
vtss_phy_pre_system_reset
vtss_phy_api.h, 846
vtss_phy_read
vtss_phy_api.h, 855
vtss_phy_read_page
vtss_phy_api.h, 856
vtss_phy_read_tr_addr
vtss_phy_api.h, 888
vtss_phy_recov_clk_t
vtss_phy_api.h, 836
vtss_phy_reg_decode_status
vtss_phy_api.h, 875
vtss_phy_reset
vtss_phy_api.h, 846
vtss_phy_reset_conf_t, 320
force, 321
i_cpu_en, 321
mac_if, 320
media_if, 320
pkt_mode, 321
rgmii, 320
tbi, 321
vtss_phy_reset_get
vtss_phy_api.h, 847
vtss_phy_reset_lcpll
vtss_phy_api.h, 883
vtss_phy_rgmii_conf_t, 322
rx_clk_skew_ps, 322
tx_clk_skew_ps, 322
vtss_phy_sd6g_mac_serdes_conf
vtss_phy_api.h, 891
vtss_phy_sd6g_ob_lev_rd
vtss_phy_api.h, 885
vtss_phy_sd6g_ob_lev_wr
vtss_phy_api.h, 885
vtss_phy_sd6g_ob_post_rd
vtss_phy_api.h, 884
vtss_phy_sd6g_ob_post_wr
vtss_phy_api.h, 884
vtss_phy_serdes1g_rcpll_status_get
vtss_phy_api.h, 882
vtss_phy_serdes6g_rcpll_status_get
vtss_phy_api.h, 882
vtss_phy_serdes_fmedia_loopback
vtss_phy_api.h, 879
vtss_phy_serdes_sgmii_loopback
vtss_phy_api.h, 878
vtss_phy_sigdet_polarity_t
vtss_phy_api.h, 840
vtss_phy_statistic_get
vtss_phy_api.h, 871
vtss_phy_statistic_t, 322
cu_bad, 323
cu_good, 323
media_mac_serdes_crc, 324
media_mac_serdes_good, 324
rx_err_cnt_base_tx, 324
serdes_tx_bad, 323
serdes_tx_good, 323
vtss_phy_status_1g_get
vtss_phy_api.h, 851
vtss_phy_status_1g_t, 325
master, 325
master_cfg_fault, 325
vtss_phy_status_get
vtss_phy_api.h, 849
vtss_phy_status_inst_poll

vtss_phy_api.h, 890
vtss_phy_tbi_conf_t, 325
aneg_enable, 326
vtss_phy_timestamp_t, 326
high, 327
low, 327
nanoseconds, 327
seconds, 327
vtss_phy_ts_10g_extended_fifo_sync
vtss_phy_ts_api.h, 958
vtss_phy_ts_8487_xaui_sel_t
vtss_phy_ts_api.h, 914
vtss_phy_ts_ach_conf_t, 327
channel_type, 329
comm_opt, 329
mask, 328, 329
proto_id, 329
value, 328, 329
version, 328
vtss_phy_ts_action_format
vtss_phy_ts_api.h, 919
vtss_phy_ts_alt_clock_mode_get
vtss_phy_ts_api.h, 923
vtss_phy_ts_alt_clock_mode_s, 330
ls_lpbk, 330
ls_pps_lpbk, 330
pps_ls_lpbk, 330
vtss_phy_ts_alt_clock_mode_set
vtss_phy_ts_api.h, 924
vtss_phy_ts_alt_clock_mode_t
vtss_phy_ts_api.h, 913
vtss_phy_ts_alt_clock_saved_get
vtss_phy_ts_api.h, 923
vtss_phy_ts_api.h
VTSS_PHY_TS_8487_XAUI_SEL_0, 912
VTSS_PHY_TS_8487_XAUI_SEL_1, 912
VTSS_PHY_TS_DATA_IN_RSRVD_FIELD, 911
VTSS_PHY_TS_EGR_DATAPATH_RESET, 912
VTSS_PHY_TS_EGR_ENGINE_ERR, 910
VTSS_PHY_TS_EGR_FIFO_OVERFLOW, 911
VTSS_PHY_TS_EGR_FIFO_RESET, 913
VTSS_PHY_TS_EGR_LTC2_RESET, 913
VTSS_PHY_TS_EGR_RW_FCS_ERR, 910
VTSS_PHY_TS_EGR_TIMESTAMP_CAPTURED, 911
VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH0, 909
VTSS_PHY_TS_ENG_FLOW_VALID_FOR_CH1, 909
VTSS_PHY_TS_ETH_ADDR_MATCH_48BIT, 904
VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_M↩ULTICAST, 905
VTSS_PHY_TS_ETH_ADDR_MATCH_ANY_U↩NICAST, 905
VTSS_PHY_TS_ETH_MATCH_DEST_ADDR, 905
VTSS_PHY_TS_ETH_MATCH_SRC_ADDR, 905
VTSS_PHY_TS_ETH_MATCH_SRC_OR_DEST, 905
VTSS_PHY_TS_FIFO_SIG_DEST_IP, 901
VTSS_PHY_TS_FIFO_SIG_DEST_MAC, 902
VTSS_PHY_TS_FIFO_SIG_DOMAIN_NUM, 901
VTSS_PHY_TS_FIFO_SIG_MSG_TYPE, 901
VTSS_PHY_TS_FIFO_SIG_SEQ_ID, 901
VTSS_PHY_TS_FIFO_SIG_SH_SPORT_ID, 902
VTSS_PHY_TS_FIFO_SIG_SOURCE_PORT_ID, 901
VTSS_PHY_TS_FIFO_SIG_SRC_IP, 900
VTSS_PHY_TS_INGR_DATAPATH_RESET, 912
VTSS_PHY_TS_INGR_ENGINE_ERR, 910
VTSS_PHY_TS_INGR_LTC1_RESET, 912
VTSS_PHY_TS_INGR_RW_FCS_ERR, 910
VTSS_PHY_TS_INGR_RW_PREAM_ERR, 910
VTSS_PHY_TS_IP_MATCH_DEST, 908
VTSS_PHY_TS_IP_MATCH_SRC_OR_DEST, 908
VTSS_PHY_TS_IP_MATCH_SRC, 907
VTSS_PHY_TS_IP_VER_4, 907
VTSS_PHY_TS_IP_VER_6, 907
VTSS_PHY_TS_LTC_LOAD_SAVE_NEW_TOD, 911
VTSS_PHY_TS_LTC_NEW_PPS_INTRPT, 911
VTSS_PHY_TS_MPLS_STACK_DEPTH_1, 908
VTSS_PHY_TS_MPLS_STACK_DEPTH_2, 908
VTSS_PHY_TS_MPLS_STACK_DEPTH_3, 908
VTSS_PHY_TS_MPLS_STACK_DEPTH_4, 909
VTSS_PHY_TS_MPLS_STACK_REF_POINT_↩END, 909
VTSS_PHY_TS_MPLS_STACK_REF_POINT_↩TOP, 909
VTSS_PHY_TS_SIG_DEST_IP_LEN, 903
VTSS_PHY_TS_SIG_DEST_MAC_LEN, 904
VTSS_PHY_TS_SIG_DOMAIN_NUM_LEN, 902
VTSS_PHY_TS_SIG_LEN, 902
VTSS_PHY_TS_SIG_MSG_TYPE_LEN, 903
VTSS_PHY_TS_SIG_SEQ_ID_LEN, 903
VTSS_PHY_TS_SIG_SEQUENCE_ID_LEN, 903
VTSS_PHY_TS_SIG_SH_SPORT_ID_LEN, 904
VTSS_PHY_TS_SIG_SOURCE_PORT_ID_LEN, 903
VTSS_PHY_TS_SIG_SRC_IP_LEN, 904
VTSS_PHY_TS_SIG_TIME_STAMP_LEN, 902
VTSS_PHY_TS_TAG_RANGE_INNER, 907
VTSS_PHY_TS_TAG_RANGE_NONE, 906
VTSS_PHY_TS_TAG_RANGE_OUTER, 907
VTSS_PHY_TS_TAG_TYPE_B, 906
VTSS_PHY_TS_TAG_TYPE_C, 906
VTSS_PHY_TS_TAG_TYPE_I, 906
VTSS_PHY_TS_TAG_TYPE_S, 906
VTSS_PTP_IP_1588_VERSION_2_1, 904
vtss_phy_1588_csr_reg_read, 957
vtss_phy_1588_csr_reg_write, 957
vtss_phy_1588_debug_reg_read, 960
vtss_phy_1g_ts_fifo_sync, 959
vtss_phy_daisy_conf_get, 937
vtss_phy_daisy_conf_set, 937
vtss_phy_ts_10g_extended_fifo_sync, 958

vtss_phy_ts_8487_xaui_sel_t, 914
vtss_phy_ts_action_format, 919
vtss_phy_ts_alt_clock_mode_get, 923
vtss_phy_ts_alt_clock_mode_set, 924
vtss_phy_ts_alt_clock_mode_t, 913
vtss_phy_ts_alt_clock_saved_get, 923
vtss_phy_ts_block_soft_reset, 955
vtss_phy_ts_clock_rateadj_get, 933
vtss_phy_ts_clock_rateadj_ppm_get, 934
vtss_phy_ts_clock_rateadj_ppm_set, 934
vtss_phy_ts_clock_rateadj_set, 933
vtss_phy_ts_clock_src_t, 920
vtss_phy_ts_clockfreq_t, 919
vtss_phy_ts_correction_overflow_get, 952
vtss_phy_ts_delay_asymmetry_get, 929
vtss_phy_ts_delay_asymmetry_set, 928
vtss_phy_ts_egress_engine_action_get, 948
vtss_phy_ts_egress_engine_action_set, 948
vtss_phy_ts_egress_engine_clear, 943
vtss_phy_ts_egress_engine_conf_get, 945
vtss_phy_ts_egress_engine_conf_set, 945
vtss_phy_ts_egress_engine_init, 942
vtss_phy_ts_egress_engine_init_conf_get, 943
vtss_phy_ts_egress_latency_get, 927
vtss_phy_ts_egress_latency_set, 926
vtss_phy_ts_encap_t, 915
vtss_phy_ts_engine_channel_map_t, 914
vtss_phy_ts_engine_flow_match_t, 916
vtss_phy_ts_engine_t, 916
vtss_phy_ts_event_enable_get, 950
vtss_phy_ts_event_enable_set, 950
vtss_phy_ts_event_poll, 951
vtss_phy_ts_event_t, 914
vtss_phy_ts_fifo_empty, 939
vtss_phy_ts_fifo_mode_t, 921
vtss_phy_ts_fifo_read, 914
vtss_phy_ts_fifo_read_cb_get, 940
vtss_phy_ts_fifo_read_install, 939
vtss_phy_ts_fifo_sig_get, 938
vtss_phy_ts_fifo_sig_mask_t, 913
vtss_phy_ts_fifo_sig_set, 938
vtss_phy_ts_fifo_status_t, 915
vtss_phy_ts_fifo_timestamp_len_t, 921
vtss_phy_ts_flow_clear_cf_set, 960
vtss_phy_ts_hiacc_get, 955
vtss_phy_ts_hiacc_set, 954
vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t, 918
vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t, 919
vtss_phy_ts_ingress_engine_action_get, 946
vtss_phy_ts_ingress_engine_action_set, 946
vtss_phy_ts_ingress_engine_clear, 941
vtss_phy_ts_ingress_engine_conf_get, 944
vtss_phy_ts_ingress_engine_conf_set, 943
vtss_phy_ts_ingress_engine_init, 940
vtss_phy_ts_ingress_engine_init_conf_get, 941
vtss_phy_ts_ingress_latency_get, 926
vtss_phy_ts_ingress_latency_set, 925
vtss_phy_ts_init, 953
vtss_phy_ts_init_conf_get, 953
vtss_phy_ts_load_ptptime_get, 931
vtss_phy_ts_loadpulse_delay_get, 933
vtss_phy_ts_loadpulse_delay_set, 932
vtss_phy_ts_ltc_freq_synth_pulse_get, 936
vtss_phy_ts_ltc_freq_synth_pulse_set, 936
vtss_phy_ts_mode_get, 952
vtss_phy_ts_mode_set, 952
vtss_phy_ts_new_spi_mode_get, 956
vtss_phy_ts_new_spi_mode_set, 956
vtss_phy_ts_nphase_sampler_t, 922
vtss_phy_ts_nphase_status_get, 954
vtss_phy_ts_ongoing_adjustment, 936
vtss_phy_ts_path_delay_get, 928
vtss_phy_ts_path_delay_set, 927
vtss_phy_ts_pps_conf_get, 925
vtss_phy_ts_pps_conf_set, 924
vtss_phy_ts_ptp_clock_mode_t, 917
vtss_phy_ts_ptp_delaym_type_t, 917
vtss_phy_ts_ptp_message_type_t, 922
vtss_phy_ts_ptptime_adj1ns, 935
vtss_phy_ts_ptptime_arm, 930
vtss_phy_ts_ptptime_get, 930
vtss_phy_ts_ptptime_set, 929
vtss_phy_ts_ptptime_set_done, 930
vtss_phy_ts_rxtimestamp_len_t, 920
vtss_phy_ts_rxtimestamp_pos_t, 920
vtss_phy_ts_scaled_ppb_t, 913
vtss_phy_ts_sertod_get, 932
vtss_phy_ts_sertod_set, 931
vtss_phy_ts_soft_reset_t, 914
vtss_phy_ts_stats_get, 951
vtss_phy_ts_status_check, 958
vtss_phy_ts_tc_op_mode_t, 922
vtss_phy_ts_tesla_tsp_fifo_sync, 959
vtss_phy_ts_timeofday_offset_set, 935
vtss_phy_ts_todadj_status_t, 915
vtss_phy_ts_viper_fifo_reset, 958
vtss_phy_ts_y1731_oam_delaym_type_t, 918
vtss_phy_ts_block_soft_reset
vtss_phy_ts_api.h, 955
vtss_phy_ts_clock_rateadj_get
vtss_phy_ts_api.h, 933
vtss_phy_ts_clock_rateadj_ppm_get
vtss_phy_ts_api.h, 934
vtss_phy_ts_clock_rateadj_ppm_set
vtss_phy_ts_api.h, 934
vtss_phy_ts_clock_rateadj_set
vtss_phy_ts_api.h, 933
vtss_phy_ts_clock_src_t
vtss_phy_ts_api.h, 920
vtss_phy_ts_clockfreq_t
vtss_phy_ts_api.h, 919
vtss_phy_ts_correction_overflow_get
vtss_phy_ts_api.h, 952
vtss_phy_ts_delay_asymmetry_get
vtss_phy_ts_api.h, 929

vtss_phy_ts_delay_asymmetry_set
vtss_phy_ts_api.h, 928
vtss_phy_ts_egress_engine_action_get
vtss_phy_ts_api.h, 948
vtss_phy_ts_egress_engine_action_set
vtss_phy_ts_api.h, 948
vtss_phy_ts_egress_engine_clear
vtss_phy_ts_api.h, 943
vtss_phy_ts_egress_engine_conf_get
vtss_phy_ts_api.h, 945
vtss_phy_ts_egress_engine_conf_set
vtss_phy_ts_api.h, 945
vtss_phy_ts_egress_engine_init
vtss_phy_ts_api.h, 942
vtss_phy_ts_egress_engine_init_conf_get
vtss_phy_ts_api.h, 943
vtss_phy_ts_egress_latency_get
vtss_phy_ts_api.h, 927
vtss_phy_ts_egress_latency_set
vtss_phy_ts_api.h, 926
vtss_phy_ts_encap_t
vtss_phy_ts_api.h, 915
vtss_phy_ts_eng_init_conf_t, 331
encap_type, 331
eng_used, 331
flow_end_index, 332
flow_match_mode, 332
flow_st_index, 332
vtss_phy_ts_engine_action_t, 332
action, 334
action_gen, 333
action_ptp, 333
gen_conf, 334
oam_conf, 334
ptp_conf, 333
vtss_phy_ts_engine_channel_map_t
vtss_phy_ts_api.h, 914
vtss_phy_ts_engine_flow_conf_t, 334
channel_map, 335
eng_mode, 335
flow_conf, 336
gen, 336
oam, 336
ptp, 335
vtss_phy_ts_engine_flow_match_t
vtss_phy_ts_api.h, 916
vtss_phy_ts_engine_t
vtss_phy_ts_api.h, 916
vtss_phy_ts_eth_conf_t, 336
addr_match_mode, 338
addr_match_select, 339
comm_opt, 338
etype, 338
flow_en, 338
flow_opt, 342
i_tag, 342
inner_tag, 342
inner_tag_type, 340
lower, 340
mac_addr, 339
mask, 341, 342
num_tag, 339
outer_tag, 341
outer_tag_type, 339
pbb_en, 338
range, 340, 341
tag_range_mode, 340
tpid, 338
upper, 340
val, 341
value, 341
vlan_check, 339
vtss_phy_ts_event_enable_get
vtss_phy_ts_api.h, 950
vtss_phy_ts_event_enable_set
vtss_phy_ts_api.h, 950
vtss_phy_ts_event_poll
vtss_phy_ts_api.h, 951
vtss_phy_ts_event_t
vtss_phy_ts_api.h, 914
vtss_phy_ts_fifo_conf_t, 342
detect_only, 343
eng_minE, 343
eng_recov, 343
skip_rev_check, 343
vtss_phy_ts_fifo_empty
vtss_phy_ts_api.h, 939
vtss_phy_ts_fifo_mode_t
vtss_phy_ts_api.h, 921
vtss_phy_ts_fifo_read
vtss_phy_ts_api.h, 914
vtss_phy_ts_fifo_read_cb_get
vtss_phy_ts_api.h, 940
vtss_phy_ts_fifo_read_install
vtss_phy_ts_api.h, 939
vtss_phy_ts_fifo_sig_get
vtss_phy_ts_api.h, 938
vtss_phy_ts_fifo_sig_mask_t
vtss_phy_ts_api.h, 913
vtss_phy_ts_fifo_sig_set
vtss_phy_ts_api.h, 938
vtss_phy_ts_fifo_sig_t, 344
dest_ip, 345
dest_mac, 346
domain_num, 345
msg_type, 344
sequence_id, 345
sig_mask, 344
src_ip, 345
src_port_identity, 345
vtss_phy_ts_fifo_status_t
vtss_phy_ts_api.h, 915
vtss_phy_ts_fifo_timestamp_len_t
vtss_phy_ts_api.h, 921
vtss_phy_ts_flow_clear_cf_set
vtss_phy_ts_api.h, 960

vtss_phy_ts_gen_conf_t, 346
comm_opt, 347
data, 347
flow_en, 347
flow_offset, 347
flow_opt, 348
mask, 347
next_prot_offset, 347
vtss_phy_ts_generic_action_t, 348
channel_map, 349
data, 349
enable, 348
flow_id, 349
mask, 349
ts_offset, 350
ts_type, 349
vtss_phy_ts_generic_flow_conf_t, 350
eth1_opt, 350
gen_opt, 351
vtss_phy_ts_hiacc_get
vtss_phy_ts_api.h, 955
vtss_phy_ts_hiacc_set
vtss_phy_ts_api.h, 954
vtss_phy_ts_ietf_mpls_ach_oam_conf_t, 351
delaym_type, 351
ds, 352
ts_format, 352
vtss_phy_ts_ietf_mpls_ach_oam_delaym_type_t
vtss_phy_ts_api.h, 918
vtss_phy_ts_ietf_mpls_ach_oam_ts_format_t
vtss_phy_ts_api.h, 919
vtss_phy_ts_ingress_engine_action_get
vtss_phy_ts_api.h, 946
vtss_phy_ts_ingress_engine_action_set
vtss_phy_ts_api.h, 946
vtss_phy_ts_ingress_engine_clear
vtss_phy_ts_api.h, 941
vtss_phy_ts_ingress_engine_conf_get
vtss_phy_ts_api.h, 944
vtss_phy_ts_ingress_engine_conf_set
vtss_phy_ts_api.h, 943
vtss_phy_ts_ingress_engine_init
vtss_phy_ts_api.h, 940
vtss_phy_ts_ingress_engine_init_conf_get
vtss_phy_ts_api.h, 941
vtss_phy_ts_ingress_latency_get
vtss_phy_ts_api.h, 926
vtss_phy_ts_ingress_latency_set
vtss_phy_ts_api.h, 925
vtss_phy_ts_init
vtss_phy_ts_api.h, 953
vtss_phy_ts_init_conf_get
vtss_phy_ts_api.h, 953
vtss_phy_ts_init_conf_t, 352
auto_clear_ls, 355
chk_ing_modified, 355
clk_freq, 353
clk_src, 353
macsec_ena, 355
one_step_txfifo, 355
rx_ts_len, 353
rx_ts_pos, 353
tc_op_mode, 354
tx_fifo_hi_clk_cycs, 354
tx_fifo_lo_clk_cycs, 354
tx_fifo_mode, 353
tx_fifo_spi_conf, 354
tx_ts_len, 354
vtss_phy_ts_ip_conf_t, 356
addr, 358
comm_opt, 357
dport_mask, 357
dport_val, 357
flow_en, 357
flow_opt, 359
ip_addr, 359
ip_mode, 356
ipv4, 358
ipv6, 358
mask, 358
match_mode, 358
sport_mask, 357
sport_val, 356
vtss_phy_ts_load_ptptime_get
vtss_phy_ts_api.h, 931
vtss_phy_ts_loadpulse_delay_get
vtss_phy_ts_api.h, 933
vtss_phy_ts_loadpulse_delay_set
vtss_phy_ts_api.h, 932
vtss_phy_ts_ltc_freq_synth_pulse_get
vtss_phy_ts_api.h, 936
vtss_phy_ts_ltc_freq_synth_pulse_set
vtss_phy_ts_api.h, 936
vtss_phy_ts_mode_get
vtss_phy_ts_api.h, 952
vtss_phy_ts_mode_set
vtss_phy_ts_api.h, 952
vtss_phy_ts_mpls_conf_t, 359
bottom_up, 362
comm_opt, 360
cw_en, 360
end, 362
flow_en, 360
flow_opt, 363
frst_lvl_after_top, 361
frst_lvl_before_end, 362
snd_lvl_after_top, 361
snd_lvl_before_end, 362
stack_depth, 360
stack_level, 362
stack_ref_point, 360
thrd_lvl_after_top, 361
thrd_lvl_before_end, 362
top, 361
top_down, 361
vtss_phy_ts_mpls_lvl_rng_t, 363

lower, 363
upper, 363
vtss_phy_ts_new_spi_mode_get
vtss_phy_ts_api.h, 956
vtss_phy_ts_new_spi_mode_set
vtss_phy_ts_api.h, 956
vtss_phy_ts_nphase_sampler_t
vtss_phy_ts_api.h, 922
vtss_phy_ts_nphase_status_get
vtss_phy_ts_api.h, 954
vtss_phy_ts_nphase_status_t, 364
CALIB_DONE, 365
CALIB_ERR, 364
enable, 364
vtss_phy_ts_oam_engine_action_t, 365
channel_map, 366
enable, 366
ietf_oam_conf, 366
oam_conf, 367
version, 366
y1731_en, 366
y1731_oam_conf, 366
vtss_phy_ts_oam_engine_flow_conf_t, 367
ach_opt, 368
eth1_opt, 367
eth2_opt, 368
mpls_opt, 368
vtss_phy_ts_ongoing_adjustment
vtss_phy_ts_api.h, 936
vtss_phy_ts_path_delay_get
vtss_phy_ts_api.h, 928
vtss_phy_ts_path_delay_set
vtss_phy_ts_api.h, 927
vtss_phy_ts_pps_conf_get
vtss_phy_ts_api.h, 925
vtss_phy_ts_pps_conf_set
vtss_phy_ts_api.h, 924
vtss_phy_ts_pps_config_s, 368
pps_offset, 369
pps_output_enable, 369
pps_width_adj, 369
vtss_phy_ts_ptp_clock_mode_t
vtss_phy_ts_api.h, 917
vtss_phy_ts_ptp_conf_t, 370
domain, 371
lower, 371
mask, 370
range, 371
range_en, 370
upper, 371
val, 370
value, 371
vtss_phy_ts_ptp_delaym_type_t
vtss_phy_ts_api.h, 917
vtss_phy_ts_ptp_engine_action_t, 372
cf_update, 373
channel_map, 372
clk_mode, 373
delaym_type, 373
enable, 372
ptp_conf, 372
vtss_phy_ts_ptp_engine_flow_conf_t, 373
ach_opt, 375
eth1_opt, 374
eth2_opt, 374
ip1_opt, 374
ip2_opt, 374
mpls_opt, 375
vtss_phy_ts_ptp_message_type_t
vtss_phy_ts_api.h, 922
vtss_phy_ts_ptptime_adj1ns
vtss_phy_ts_api.h, 935
vtss_phy_ts_ptptime_arm
vtss_phy_ts_api.h, 930
vtss_phy_ts_ptptime_get
vtss_phy_ts_api.h, 930
vtss_phy_ts_ptptime_set
vtss_phy_ts_api.h, 929
vtss_phy_ts_ptptime_set_done
vtss_phy_ts_api.h, 930
vtss_phy_ts_rxtimestamp_len_t
vtss_phy_ts_api.h, 920
vtss_phy_ts_rxtimestamp_pos_t
vtss_phy_ts_api.h, 920
vtss_phy_ts_scaled_ppb_t
vtss_phy_ts_api.h, 913
vtss_phy_ts_sertod_conf_t, 375
ip_enable, 376
ls_inv, 376
op_enable, 376
vtss_phy_ts_sertod_get
vtss_phy_ts_api.h, 932
vtss_phy_ts_sertod_set
vtss_phy_ts_api.h, 931
vtss_phy_ts_soft_reset_t
vtss_phy_ts_api.h, 914
vtss_phy_ts_stats_get
vtss_phy_ts_api.h, 951
vtss_phy_ts_stats_t, 376
egr_fcs_err, 377
egr_frm_mod_cnt, 378
egr_pream_shrink_err, 377
ingr_fcs_err, 377
ingr_frm_mod_cnt, 378
ingr_pream_shrink_err, 377
ts_fifo_drop_cnt, 378
ts_fifo_tx_cnt, 378
vtss_phy_ts_status_check
vtss_phy_ts_api.h, 958
vtss_phy_ts_tc_op_mode_t
vtss_phy_ts_api.h, 922
vtss_phy_ts_tesla_tsp_fifo_sync
vtss_phy_ts_api.h, 959
vtss_phy_ts_timeofday_offset_set
vtss_phy_ts_api.h, 935
vtss_phy_ts_todadj_status_t

vtss_phy_ts_api.h, 915
vtss_phy_ts_viper_fifo_reset
vtss_phy_ts_api.h, 958
vtss_phy_ts_y1731_oam_conf_t, 379
delaym_type, 379
lower, 380
mask, 380
meg_level, 380
range, 380
range_en, 379
upper, 380
val, 379
value, 380
vtss_phy_ts_y1731_oam_delaym_type_t
vtss_phy_ts_api.h, 918
vtss_phy_type_t, 381
base_port_no, 382
channel_id, 382
part_number, 381
phy_api_base_no, 382
port_cnt, 382
revision, 381
vtss_phy_unidirectional_t
vtss_phy_api.h, 840
vtss_phy_veriphy_get
vtss_phy_api.h, 861
vtss_phy_veriphy_result_t, 383
length, 383
link, 383
status, 383
vtss_phy_veriphy_start
vtss_phy_api.h, 860
vtss_phy_veriphy_status_t
phy.h, 537
vtss_phy_warm_start_failed_get
vtss_phy_api.h, 876
vtss_phy_wol_conf_get
vtss_phy_api.h, 880
vtss_phy_wol_conf_set
vtss_phy_api.h, 881
vtss_phy_wol_conf_t, 384
magic_pkt_cnt, 385
secure_on_enable, 384
wol_mac, 384
wol_pass, 385
wol_passwd_len, 385
vtss_phy_wol_enable
vtss_phy_api.h, 880
vtss_phy_write
vtss_phy_api.h, 857
vtss_phy_write_masked
vtss_phy_api.h, 858
vtss_phy_write_masked_page
vtss_phy_api.h, 858
vtss_pi_conf_t, 385
cs_wait_ns, 386
vtss_policer_ext_t, 386
flow_control, 387
frame_rate, 387
vtss_policer_t, 387
level, 387
rate, 388
vtss_policer_type_t
types.h, 584
vtss_poll_1sec
vtss_misc_api.h, 718
vtss_port_api.h
CHIP_PORT_UNUSED, 963
VTSS_FRAME_GAP_DEFAULT, 963
VTSS_MAX_FRAME_LENGTH_MAX, 964
VTSS_MAX_FRAME_LENGTH_STANDARD, 964
vtss_internal_bw_t, 965
vtss_miim_controller_t, 964
vtss_miim_read, 973
vtss_miim_write, 973
vtss_port_basic_counters_get, 971
vtss_port_clause_37_control_get, 967
vtss_port_clause_37_control_set, 968
vtss_port_clause_37_remote_fault_t, 965
vtss_port_conf_get, 969
vtss_port_conf_set, 968
vtss_port_counters_clear, 970
vtss_port_counters_get, 970
vtss_port_counters_update, 969
vtss_port_forward_state_get, 971
vtss_port_forward_state_set, 972
vtss_port_forward_t, 966
vtss_port_ifh_conf_get, 972
vtss_port_ifh_conf_set, 972
vtss_port_loop_t, 966
vtss_port_map_get, 967
vtss_port_map_set, 967
vtss_port_max_tags_t, 965
vtss_port_status_get, 969
vtss_port_basic_counters_get
vtss_port_api.h, 971
vtss_port_bridge_counters_t, 388
dot1dTpPortInDiscards, 388
vtss_port_clause_37_adv_t, 389
acknowledge, 390
asymmetric_pause, 390
fdx, 389
hdx, 389
next_page, 390
remote_fault, 390
symmetric_pause, 390
vtss_port_clause_37_control_get
vtss_port_api.h, 967
vtss_port_clause_37_control_set
vtss_port_api.h, 968
vtss_port_clause_37_control_t, 391
advertisement, 391
enable, 391
vtss_port_clause_37_remote_fault_t
vtss_port_api.h, 965
vtss_port_conf_get

vtss_port_api.h, 969
vtss_port_conf_set
vtss_port_api.h, 968
vtss_port_conf_t, 392
exc_col_cont, 395
fdx, 394
flow_control, 394
frame_gaps, 393
frame_length_chk, 394
if_type, 392
loop, 395
max_frame_length, 394
max_tags, 395
power_down, 393
sd_active_high, 393
sd_enable, 393
sd_internal, 393
serdes, 396
speed, 394
xaui_rx_lane_flip, 395
xaui_tx_lane_flip, 395
vtss_port_counters_clear
vtss_port_api.h, 970
vtss_port_counters_get
vtss_port_api.h, 970
vtss_port_counters_t, 396
bridge, 397
ethernet_like, 397
if_group, 397
prop, 397
rmon, 396
vtss_port_counters_update
vtss_port_api.h, 969
vtss_port_ethernet_like_counters_t, 398
dot3InPauseFrames, 398
dot3OutPauseFrames, 398
vtss_port_flow_control_conf_t, 398
generate, 399
obey, 399
pfc, 399
smac, 399
vtss_port_forward_state_get
vtss_port_api.h, 971
vtss_port_forward_state_set
vtss_port_api.h, 972
vtss_port_forward_t
vtss_port_api.h, 966
vtss_port_frame_gaps_t, 400
fdx_gap, 401
hdx_gap_1, 400
hdx_gap_2, 400
vtss_port_if_group_counters_t, 401
ifInBroadcastPkts, 402
ifInDiscards, 402
ifInErrors, 403
ifInMulticastPkts, 402
ifInNUcastPkts, 402
ifInOctets, 402
ifInUcastPkts, 402
ifOutBroadcastPkts, 403
ifOutDiscards, 404
ifOutErrors, 404
ifOutMulticastPkts, 403
ifOutNUcastPkts, 404
ifOutOctets, 403
ifOutUcastPkts, 403
vtss_port_ifh_conf_get
vtss_port_api.h, 972
vtss_port_ifh_conf_set
vtss_port_api.h, 972
vtss_port_ifh_t, 404
ena_inj_header, 405
ena_xtr_header, 405
vtss_port_interface_t
types.h, 583
vtss_port_loop_t
vtss_port_api.h, 966
vtss_port_map_get
vtss_port_api.h, 967
vtss_port_map_set
vtss_port_api.h, 967
vtss_port_map_t, 405
chip_no, 406
chip_port, 406
miim_addr, 406
miim_chip_no, 407
miim_controller, 406
vtss_port_max_tags_t
vtss_port_api.h, 965
vtss_port_proprietary_counters_t, 407
rx_prio, 407
tx_prio, 408
vtss_port_rmon_counters_t, 408
rx_etherStatsBroadcastPkts, 409
rx_etherStatsCRCAlignErrors, 410
rx_etherStatsDropEvents, 409
rx_etherStatsFragments, 410
rx_etherStatsJabbers, 410
rx_etherStatsMulticastPkts, 409
rx_etherStatsOctets, 409
rx_etherStatsOversizePkts, 410
rx_etherStatsPkts, 409
rx_etherStatsPkts1024to1518Octets, 412
rx_etherStatsPkts128to255Octets, 411
rx_etherStatsPkts1519toMaxOctets, 412
rx_etherStatsPkts256to511Octets, 411
rx_etherStatsPkts512to1023Octets, 411
rx_etherStatsPkts64Octets, 411
rx_etherStatsPkts65to127Octets, 411
rx_etherStatsUndersizePkts, 410
tx_etherStatsBroadcastPkts, 413
tx_etherStatsCollisions, 413
tx_etherStatsDropEvents, 412
tx_etherStatsMulticastPkts, 413
tx_etherStatsOctets, 412
tx_etherStatsPkts, 412

tx_etherStatsPkts1024to1518Octets, 414
tx_etherStatsPkts128to255Octets, 414
tx_etherStatsPkts1519toMaxOctets, 414
tx_etherStatsPkts256to511Octets, 414
tx_etherStatsPkts512to1023Octets, 414
tx_etherStatsPkts64Octets, 413
tx_etherStatsPkts65to127Octets, 413
vtss_port_serdes_conf_t, 415
sfp_dac, 415
vtss_port_sgmii_aneg_t, 416
aneg_complete, 417
fdx, 416
hdx, 416
link, 416
speed_100M, 417
speed_10M, 417
speed_1G, 417
vtss_port_speed_t
port.h, 549
vtss_port_state_get
vtss_l2_api.h, 667
vtss_port_state_set
vtss_l2_api.h, 667
vtss_port_status_get
vtss_port_api.h, 969
vtss_port_status_t, 418
aneg, 419
aneg_complete, 419
copper, 420
fdx, 419
fiber, 420
link, 418
link_down, 418
mdi_cross, 420
remote_fault, 419
speed, 418
unidirectional_ability, 419
vtss_ptp_event_enable
vtss_misc_api.h, 718
vtss_ptp_event_poll
vtss_misc_api.h, 718
vtss_pvlan_port_members_get
vtss_l2_api.h, 681
vtss_pvlan_port_members_set
vtss_l2_api.h, 682
vtss_qce_action_t, 420
dei, 422
dp, 421
dp_enable, 421
dscp, 422
dscp_enable, 422
pcp, 422
pcp_dei_enable, 422
policy_no, 423
policy_no_enable, 423
prio, 421
prio_enable, 421
vtss_qce_add
vtss_qos_api.h, 982
vtss_qce_del
vtss_qos_api.h, 982
vtss_qce_frame_etype_t, 423
data, 424
etype, 424
vtss_qce_frame_ipv4_t, 424
dip, 425
dport, 426
dscp, 425
fragment, 425
proto, 425
sip, 425
sport, 425
vtss_qce_frame_ipv6_t, 426
dip, 427
dport, 427
dscp, 426
proto, 427
sip, 427
sport, 427
vtss_qce_frame_llc_t, 428
data, 428
vtss_qce_frame_snap_t, 429
data, 429
vtss_qce_init
vtss_qos_api.h, 981
vtss_qce_key_t, 429
etype, 431
frame, 432
inner_tag, 431
ipv4, 432
ipv6, 432
llc, 431
mac, 430
port_list, 430
snap, 431
tag, 430
type, 431
vtss_qce_mac_t, 432
dmac, 433
dmac_bc, 433
dmac_mc, 433
smac, 433
vtss_qce_t, 434
action, 435
id, 434
key, 434
vtss_qce_tag_t, 435
dei, 436
pcp, 436
s_tag, 436
tagged, 436
vid, 435
vtss_qce_type_t
vtss_qos_api.h, 979
vtss_qos_api.h
VTSS_PORT_POLICER_CPU_QUEUES, 976

VTSS_PORT_POLICERS, 976
VTSS_QCE_ID_LAST, 977
VTSS_QCL_ARRAY_SIZE, 977
VTSS_QCL_ID_END, 977
VTSS_QCL_ID_START, 977
VTSS_QCL_IDS, 976
vtss_dscp_emode_t, 979
vtss_dscp_mode_t, 978
vtss_qce_add, 982
vtss_qce_del, 982
vtss_qce_init, 981
vtss_qce_type_t, 979
vtss_qos_conf_get, 980
vtss_qos_conf_set, 980
vtss_qos_port_conf_get, 980
vtss_qos_port_conf_set, 981
vtss_qos_shaper_calibrate, 982
vtss_shaper_mode_t, 978
vtss_tag_remark_mode_t, 978
vtss_wred_v2_max_t, 977
vtss_qos_conf_get
vtss_qos_api.h, 980
vtss_qos_conf_set
vtss_qos_api.h, 980
vtss_qos_conf_t, 437
dscp_dp_level_map, 438
dscp_qos_class_map, 438
dscp_qos_map, 438
dscp_qos_map_dp1, 438
dscp_remap, 439
dscp_remap_dp1, 439
dscp_remark, 438
dscp_translate_map, 439
dscp_trust, 437
policer_bc, 440
policer_bc_frame_rate, 441
policer_bc_mode, 441
policer_mc, 440
policer_mc_frame_rate, 440
policer_mc_mode, 440
policer_uc, 439
policer_uc_frame_rate, 439
policer_uc_mode, 440
prios, 437
red_v2, 441
vtss_qos_port_conf_get
vtss_qos_api.h, 980
vtss_qos_port_conf_set
vtss_qos_api.h, 981
vtss_qos_port_conf_t, 441
default_dei, 444
default_dpl, 444
default_prio, 443
dmac_dip, 447
dp_level_map, 445
dscp_class_enable, 445
dscp_emode, 445
dscp_mode, 445
dscp_translate, 445
dwrr_cnt, 447
dwrr_enable, 447
excess_enable, 443
key_type, 447
policer_ext_port, 442
policer_port, 442
policer_queue, 443
qos_class_map, 444
queue_pct, 447
shaper_port, 443
shaper_queue, 443
tag_class_enable, 444
tag_default_dei, 446
tag_default_pcp, 446
tag_dei_map, 446
tag_pcp_map, 446
tag_remark_mode, 446
usr_prio, 444
vtss_qos_shaper_calibrate
vtss_qos_api.h, 982
vtss_rcpll_status_t, 448
cal_error, 448
cal_not_done, 449
out_of_range, 448
vtss_red_v2_t, 449
enable, 449
max, 450
max_unit, 450
min_fl, 450
vtss_reg_read
vtss_misc_api.h, 715
vtss_reg_read_t
vtss_init_api.h, 634
vtss_reg_write
vtss_misc_api.h, 716
vtss_reg_write_masked
vtss_misc_api.h, 716
vtss_reg_write_t
vtss_init_api.h, 634
vtss_restart_conf_end
vtss_init_api.h, 643
vtss_restart_conf_get
vtss_init_api.h, 643
vtss_restart_conf_set
vtss_init_api.h, 644
vtss_restart_status_get
vtss_init_api.h, 643
vtss_restart_status_t, 450
cur_version, 451
prev_version, 451
restart, 451
vtss_restart_t
vtss_init_api.h, 640
vtss_routing_entry_t, 452
ipv4_uc, 452
ipv6_uc, 452
route, 453

type, 452
vlan, 453
vtss_rx_master_timestamp_get
vtss_ts_api.h, 1015
vtss_rx_timestamp_get
vtss_ts_api.h, 1014
vtss_rx_timestamp_id_release
vtss_ts_api.h, 1015
vtss_secure_on_passwd_t, 453
passwd, 454
vtss_security_api.h
VTSS_ACE_ID_LAST, 986
vtss_ace_add, 992
vtss_ace_bit_t, 988
vtss_ace_counter_clear, 993
vtss_ace_counter_get, 993
vtss_ace_del, 993
vtss_ace_init, 992
vtss_ace_type_t, 987
vtss_acl_policer_conf_get, 989
vtss_acl_policer_conf_set, 990
vtss_acl_port_action_t, 987
vtss_acl_port_conf_get, 990
vtss_acl_port_conf_set, 990
vtss_acl_port_counter_clear, 991
vtss_acl_port_counter_get, 991
vtss_acl_ptp_action_t, 987
vtss_auth_port_state_get, 988
vtss_auth_port_state_set, 989
vtss_auth_state_t, 986
vtss_serdes_macro_conf_t, 454
ib_cterm_ena, 455
qsgmii, 455
serdes1g_vdd, 454
serdes6g_vdd, 454
vtss_serdes_mode_t
types.h, 583
vtss_sflow_port_conf_get
vtss_l2_api.h, 684
vtss_sflow_port_conf_set
vtss_l2_api.h, 684
vtss_sflow_port_conf_t, 455
sampling_rate, 456
type, 456
vtss_sflow_sampling_rate_convert
vtss_l2_api.h, 685
vtss_sflow_type_t
l2_types.h, 511
vtss_sgpio_bmode_t
vtss_misc_api.h, 710
vtss_sgpio_conf_get
vtss_misc_api.h, 723
vtss_sgpio_conf_set
vtss_misc_api.h, 723
vtss_sgpio_conf_t, 456
bit_count, 457
bmode, 457
port_conf, 457
vtss_sgpio_event_enable
vtss_misc_api.h, 724
vtss_sgpio_event_poll
vtss_misc_api.h, 724
vtss_sgpio_mode_t
vtss_misc_api.h, 710
vtss_sgpio_port_conf_t, 457
enabled, 458
int_pol_high, 458
mode, 458
vtss_sgpio_port_data_t, 459
value, 459
vtss_sgpio_read
vtss_misc_api.h, 723
vtss_shaper_mode_t
vtss_qos_api.h, 978
vtss_shaper_t, 459
ebs, 460
eir, 460
level, 460
mode, 460
rate, 460
vtss_spi_32bit_read_write_t
vtss_init_api.h, 636
vtss_spi_64bit_read_write_t
vtss_init_api.h, 637
vtss_spi_read_write_t
vtss_init_api.h, 636
vtss_squelch_workaround
vtss_phy_api.h, 868
vtss_storm_policer_mode_t
types.h, 584
vtss_stp_port_state_get
vtss_l2_api.h, 667
vtss_stp_port_state_set
vtss_l2_api.h, 668
vtss_stp_state_t
vtss_l2_api.h, 658
vtss_sync_api.h
VTSS_SYNCE_CLK_MAX, 996
VTSS_SYNCE_CLK_A, 995
VTSS_SYNCE_CLK_B, 995
vtss_synce_clock_in_get, 998
vtss_synce_clock_in_set, 997
vtss_synce_clock_out_get, 997
vtss_synce_clock_out_set, 996
vtss_synce_divider_t, 996
vtss_synce_clock_in_get
vtss_sync_api.h, 998
vtss_synce_clock_in_set
vtss_sync_api.h, 997
vtss_synce_clock_in_t, 461
enable, 462
port_no, 461
squelsh, 461
vtss_synce_clock_out_get
vtss_sync_api.h, 997
vtss_synce_clock_out_set

vtss_sync_api.h, 996
vtss_synce_clock_out_t, 462
divider, 462
enable, 463
vtss_synce_divider_t
vtss_sync_api.h, 996
vtss_tag_remark_mode_t
vtss_qos_api.h, 978
vtss_tag_type_t
vtss_packet_api.h, 800
vtss_target_type_t
vtss_init_api.h, 640
vtss_tci_t, 463
cfi, 464
tagprio, 464
vid, 463
vtss_temp_sensor_get
vtss_misc_api.h, 729
vtss_temp_sensor_init
vtss_misc_api.h, 729
vtss_timeofday_t, 464
sec, 465
vtss_timestamp_age
vtss_ts_api.h, 1016
vtss_timestamp_t, 465
nanoseconds, 466
sec_msb, 466
seconds, 466
vtss_tod_get_ns_cnt
vtss_misc_api.h, 728
vtss_tod_set_ns_cnt_cb
vtss_misc_api.h, 728
vtss_trace_conf_get
vtss_misc_api.h, 711
vtss_trace_conf_set
vtss_misc_api.h, 712
vtss_trace_conf_t, 466
level, 467
vtss_trace_group_t
vtss_misc_api.h, 707
vtss_trace_layer_t
vtss_misc_api.h, 706
vtss_trace_level_t
vtss_misc_api.h, 707
vtss_ts_adjtimer_get
vtss_ts_api.h, 1005
vtss_ts_adjtimer_one_sec
vtss_ts_api.h, 1003
vtss_ts_adjtimer_set
vtss_ts_api.h, 1005
vtss_ts_alt_clock_mode_get
vtss_ts_api.h, 1006
vtss_ts_alt_clock_mode_set
vtss_ts_api.h, 1007
vtss_ts_alt_clock_mode_t, 467
load, 468
one_pps_in, 468
one_pps_out, 467
save, 468
vtss_ts_api.h, 1002
vtss_ts_alt_clock_saved_get
vtss_ts_api.h, 1006
vtss_ts_api.h
vtss_oam_timestamp_get, 1015
vtss_rx_master_timestamp_get, 1015
vtss_rx_timestamp_get, 1014
vtss_rx_timestamp_id_release, 1015
vtss_timestamp_age, 1016
vtss_ts_adjtimer_get, 1005
vtss_ts_adjtimer_one_sec, 1003
vtss_ts_adjtimer_set, 1005
vtss_ts_alt_clock_mode_get, 1006
vtss_ts_alt_clock_mode_set, 1007
vtss_ts_alt_clock_mode_t, 1002
vtss_ts_alt_clock_saved_get, 1006
vtss_ts_delay_asymmetry_get, 1012
vtss_ts_delay_asymmetry_set, 1011
vtss_ts_egress_latency_get, 1011
vtss_ts_egress_latency_set, 1010
vtss_ts_external_clock_mode_get, 1008
vtss_ts_external_clock_mode_set, 1008
vtss_ts_external_clock_saved_get, 1008
vtss_ts_freq_offset_get, 1006
vtss_ts_ingress_latency_get, 1009
vtss_ts_ingress_latency_set, 1009
vtss_ts_internal_mode_get, 1013
vtss_ts_internal_mode_set, 1013
vtss_ts_ongoing_adjustment, 1004
vtss_ts_operation_mode_get, 1013
vtss_ts_operation_mode_set, 1012
vtss_ts_p2p_delay_get, 1010
vtss_ts_p2p_delay_set, 1010
vtss_ts_status_change, 1017
vtss_ts_timeofday_get, 1004
vtss_ts_timeofday_next_pps_get, 1005
vtss_ts_timeofday_next_pps_set, 1007
vtss_ts_timeofday_offset_set, 1003
vtss_ts_timeofday_set, 1002
vtss_ts_timeofday_set_delta, 1003
vtss_tx_timestamp_idx_alloc, 1016
vtss_tx_timestamp_update, 1014
vtss_ts_delay_asymmetry_get
vtss_ts_api.h, 1012
vtss_ts_delay_asymmetry_set
vtss_ts_api.h, 1011
vtss_ts_egress_latency_get
vtss_ts_api.h, 1011
vtss_ts_egress_latency_set
vtss_ts_api.h, 1010
vtss_ts_ext_clock_mode_t, 468
enable, 469
freq, 469
one_pps_mode, 469
vtss_ts_external_clock_mode_get
vtss_ts_api.h, 1008
vtss_ts_external_clock_mode_set

vtss_ts_api.h, 1008
vtss_ts_external_clock_saved_get
vtss_ts_api.h, 1008
vtss_ts_freq_offset_get
vtss_ts_api.h, 1006
vtss_ts_id_t, 470
ts_id, 470
vtss_ts_ingress_latency_get
vtss_ts_api.h, 1009
vtss_ts_ingress_latency_set
vtss_ts_api.h, 1009
vtss_ts_internal_mode_get
vtss_ts_api.h, 1013
vtss_ts_internal_mode_set
vtss_ts_api.h, 1013
vtss_ts_internal_mode_t, 470
int_fmt, 471
vtss_ts_ongoing_adjustment
vtss_ts_api.h, 1004
vtss_ts_operation_mode_get
vtss_ts_api.h, 1013
vtss_ts_operation_mode_set
vtss_ts_api.h, 1012
vtss_ts_operation_mode_t, 471
mode, 471
vtss_ts_p2p_delay_get
vtss_ts_api.h, 1010
vtss_ts_p2p_delay_set
vtss_ts_api.h, 1010
vtss_ts_status_change
vtss_ts_api.h, 1017
vtss_ts_timeofday_get
vtss_ts_api.h, 1004
vtss_ts_timeofday_next_pps_get
vtss_ts_api.h, 1005
vtss_ts_timeofday_next_pps_set
vtss_ts_api.h, 1007
vtss_ts_timeofday_offset_set
vtss_ts_api.h, 1003
vtss_ts_timeofday_set
vtss_ts_api.h, 1002
vtss_ts_timeofday_set_delta
vtss_ts_api.h, 1003
vtss_ts_timestamp_alloc_t, 472
cb, 473
context, 472
port_mask, 472
vtss_ts_timestamp_t, 473
context, 474
id, 474
ts, 473
ts_valid, 474
vtss_tx_timestamp_idx_alloc
vtss_ts_api.h, 1016
vtss_tx_timestamp_update
vtss_ts_api.h, 1014
vtss_uc_flood_members_get
vtss_l2_api.h, 692
vtss_uc_flood_members_set
vtss_l2_api.h, 692
vtss_vcap_bit_t
types.h, 587
vtss_vcap_ip_t, 474
mask, 475
value, 475
vtss_vcap_key_type_t
types.h, 587
vtss_vcap_port_conf_get
vtss_l2_api.h, 679
vtss_vcap_port_conf_set
vtss_l2_api.h, 679
vtss_vcap_port_conf_t, 475
dmac_dip_1, 476
key_type_is1_1, 476
vtss_vcap_u128_t, 476
mask, 477
value, 477
vtss_vcap_u16_t, 477
mask, 478
value, 478
vtss_vcap_u24_t, 478
mask, 479
value, 479
vtss_vcap_u32_t, 479
mask, 480
value, 480
vtss_vcap_u40_t, 480
mask, 481
value, 481
vtss_vcap_u48_t, 481
mask, 482
value, 482
vtss_vcap_u8_t, 482
mask, 483
value, 483
vtss_vcap_udp_tcp_t, 483
high, 484
in_range, 484
low, 484
vtss_vcap_vid_t, 484
mask, 485
value, 485
vtss_vcap_vr_t, 485
high, 487
low, 487
mask, 486
r, 487
type, 486
v, 487
value, 486
vr, 487
vtss_vcap_vr_type_t
types.h, 587
vtss_vce_action_t, 488
policy_no, 488
vid, 488

vtss_vce_add
vtss_l2_api.h, 676
vtss_vce_del
vtss_l2_api.h, 676
vtss_vce_frame_etype_t, 488
data, 489
etype, 489
vtss_vce_frame_ipv4_t, 489
dport, 491
dscp, 490
fragment, 490
options, 490
proto, 490
sip, 491
vtss_vce_frame_ipv6_t, 491
dport, 492
dscp, 492
proto, 492
sip, 492
vtss_vce_frame_llc_t, 493
data, 493
vtss_vce_frame_snap_t, 493
data, 494
vtss_vce_init
vtss_l2_api.h, 675
vtss_vce_key_t, 494
etype, 495
frame, 496
ipv4, 496
ipv6, 496
llc, 495
mac, 495
port_list, 495
snap, 496
tag, 495
type, 495
vtss_vce_mac_t, 497
dmac_bc, 497
dmac_mc, 497
smac, 497
vtss_vce_t, 498
action, 499
id, 498
key, 498
vtss_vce_tag_t, 499
dei, 500
pcp, 500
s_tag, 500
tagged, 500
vid, 499
vtss_vce_type_t
vtss_l2_api.h, 659
vtss_vcl_port_conf_get
vtss_l2_api.h, 674
vtss_vcl_port_conf_set
vtss_l2_api.h, 675
vtss_vcl_port_conf_t, 501
dmac_dip, 501
vtss_vdd_t
types.h, 586
vtss_vid_mac_t, 501
mac, 502
vid, 502
vtss_vlan_conf_get
vtss_l2_api.h, 670
vtss_vlan_conf_set
vtss_l2_api.h, 670
vtss_vlan_conf_t, 502
s_etype, 503
vtss_vlan_frame_t
types.h, 584
vtss_vlan_port_conf_get
vtss_l2_api.h, 671
vtss_vlan_port_conf_set
vtss_l2_api.h, 671
vtss_vlan_port_conf_t, 503
frame_type, 504
ingress_filter, 504
port_type, 503
pvid, 504
s_etype, 504
untagged_vid, 504
vtss_vlan_port_members_get
vtss_l2_api.h, 672
vtss_vlan_port_members_set
vtss_l2_api.h, 672
vtss_vlan_port_type_t
vtss_l2_api.h, 658
vtss_vlan_tag_t, 505
dei, 506
pcp, 505
tpid, 505
vid, 506
vtss_vlan_trans_group_add
vtss_l2_api.h, 676
vtss_vlan_trans_group_del
vtss_l2_api.h, 677
vtss_vlan_trans_group_get
vtss_l2_api.h, 677
vtss_vlan_trans_group_to_port_get
vtss_l2_api.h, 678
vtss_vlan_trans_group_to_port_set
vtss_l2_api.h, 678
vtss_vlan_trans_grp2vlan_conf_t, 506
group_id, 507
trans_vid, 507
vid, 507
vtss_vlan_trans_port2grp_conf_t, 507
group_id, 508
ports, 508
vtss_vlan_tx_tag_get
vtss_l2_api.h, 673
vtss_vlan_tx_tag_set
vtss_l2_api.h, 674
vtss_vlan_tx_tag_t
vtss_l2_api.h, 659

vtss_vlan_vid_conf_get
vtss_l2_api.h, 673
vtss_vlan_vid_conf_set
vtss_l2_api.h, 673
vtss_vlan_vid_conf_t, 508
fid, 509
learning, 509
mirror, 509
vtss_vt_id_t
vtss_l2_api.h, 658
vtss_wol_mac_addr_t, 509
addr, 510
vtss_wol_passwd_len_type_t
vtss_phy_api.h, 845
vtss_wred_v2_max_t
vtss_qos_api.h, 977

warm_start_enable
vtss_init_conf_t, 142
wol_mac
vtss_phy_wol_conf_t, 384
wol_pass
vtss_phy_wol_conf_t, 385
wol_passwd_len
vtss_phy_wol_conf_t, 385

xaui_rx_lane_flip
vtss_port_conf_t, 395
xaui_tx_lane_flip
vtss_port_conf_t, 395
xc_idx
vtss_mpls_segment_t, 186
xtr_cb
vtss_fdma_ch_cfg_t, 131
xtr_grp
vtss_fdma_ch_cfg_t, 130
xtr_qu
vtss_packet_rx_meta_t, 274
xtr_qu_mask
vtss_packet_rx_info_t, 267

y1731_en
vtss_phy_ts_oam_engine_action_t, 366
y1731_oam_conf
vtss_phy_ts_oam_engine_action_t, 366
yellow
vtss_oam_voe_lm_counter_conf_t, 238

zero_period_err
vtss_oam_ccm_status_t, 196