ZBOSS v1.0 API Manual

Generated by Doxygen 1.8.1.2

Wed Oct 30 2013 18:41:52

Contents

1 ZBOSS v1.0 **1**

2 Module Index **3**

2.1 API sections 3

3 Data Structure Index **5**

3.1 Data Structures 5

4 Module Documentation **9**

4.1 Stack initialization API 9

4.1.1 Detailed Description 9

4.1.2 Function Documentation 9

4.1.2.1 zb_init 9

4.2 ZDO init and main() structure 10

4.2.1 Detailed Description 10

4.2.2 Function Documentation 10

4.2.2.1 zdo_dev_start 10

4.2.2.2 zdo_main_loop 10

4.2.2.3 zb_zdo_startup_complete 11

4.3 ZDO API 12

4.3.1 Detailed Description 12

4.4 ZDO Informational Base 13

4.4.1 Detailed Description 13

4.5 ZDO base constants and definitions 14

4.5.1 Detailed Description 14

4.5.2 Typedef Documentation 14

4.5.2.1 zb_zdp_status_t 14

4.5.3 Enumeration Type Documentation 14

4.5.3.1 zb_zdp_status_e 14

4.6 ZDO discovery services 16

4.6.1 Detailed Description 18

4.6.2 Function Documentation 18

4.6.2.1 zb_zdo_nwk_addr_req 18
4.6.2.2 zb_zdo_ieee_addr_req 19
4.6.2.3 zb_zdo_node_desc_req 20
4.6.2.4 zb_zdo_power_desc_req 21
4.6.2.5 zb_zdo_simple_desc_req 22
4.6.2.6 zb_zdo_active_ep_req 23
4.6.2.7 zb_zdo_match_desc_req 23
4.6.2.8 zb_zdo_system_server_discovery_req 24
4.6.3 Macro Definition Documentation 25
4.6.3.1 ZB_ZDO_SINGLE_DEVICE_RESP 25
4.6.3.2 ZB_ZDO_EXTENDED_DEVICE_RESP 25
4.6.4 Typedef Documentation 25
4.6.4.1 zb_zdo_nwk_addr_req_t 25
4.6.4.2 zb_zdo_nwk_addr_req_param_t 25
4.6.4.3 zb_zdo_ieee_addr_req_t 25
4.6.4.4 zb_zdo_node_desc_req_t 26
4.6.4.5 zb_zdo_desc_resp_hdr_t 26
4.6.4.6 zb_zdo_node_desc_resp_t 26
4.6.4.7 zb_zdo_simple_desc_resp_hdr_t 26
4.6.4.8 zb_zdo_simple_desc_resp_t 26
4.6.4.9 zb_zdo_power_desc_resp_t 26
4.6.4.10 zb_zdo_power_desc_req_t 26
4.6.4.11 zb_zdo_simple_desc_req_t 26
4.6.4.12 zb_zdo_active_ep_req_t 26
4.6.4.13 zb_zdo_ep_resp_t 26
4.6.4.14 zb_zdo_match_desc_param_t 26
4.6.4.15 zb_zdo_match_desc_req_head_t 27
4.6.4.16 zb_zdo_match_desc_req_tail_t 27
4.6.4.17 zb_zdo_match_desc_resp_t 27
4.6.4.18 zb_zdo_system_server_discovery_req_t 27
4.6.4.19 zb_zdo_system_server_discovery_param_t 27
4.6.4.20 zb_zdo_system_server_discovery_resp_t 27
4.7 ZDO management services 28
4.7.1 Detailed Description 30
4.7.2 Function Documentation 30
4.7.2.1 zb_zdo_mgmt_nwk_update_req 30
4.7.2.2 zb_zdo_mgmt_lqi_req 31
4.7.2.3 zb_zdo_bind_req 32
4.7.2.4 zb_zdo_unbind_req 33
4.7.2.5 zdo_mgmt_leave_req 34

4.7.2.6 zb_zdo_add_group_req 35
4.7.3 Typedef Documentation 35
4.7.3.1 zb_zdo_mgmt_nwk_update_req_hdr_t 35
4.7.3.2 zb_zdo_mgmt_nwk_update_req_t 35
4.7.3.3 zb_zdo_mgmt_nwk_update_notify_hdr_t 35
4.7.3.4 zb_zdo_mgmt_nwk_update_notify_param_t 35
4.7.3.5 zb_zdo_mgmt_lqi_param_t 35
4.7.3.6 zb_zdo_mgmt_lqi_req_t 36
4.7.3.7 zb_zdo_mgmt_lqi_resp_t 36
4.7.3.8 zb_zdo_neighbor_table_record_t 36
4.7.3.9 zb_zdo_bind_req_param_t 36
4.7.3.10 zb_zdo_bind_req_head_t 36
4.7.3.11 zb_zdo_bind_req_tail_1_t 36
4.7.3.12 zb_zdo_bind_req_tail_2_t 36
4.7.3.13 zb_zdo_mgmt_leave_param_t 36
4.7.3.14 zb_zdo_mgmt_leave_req_t 36
4.7.3.15 zb_zdo_mgmt_leave_res_t 36
4.7.3.16 zb_zdo_end_device_bind_req_head_t 36
4.7.3.17 zb_zdo_end_device_bind_req_tail_t 37
4.7.3.18 zb_end_device_bind_req_param_t 37
4.7.3.19 zb_zdo_mgmt_permit_joining_req_t 37
4.7.3.20 zb_zdo_mgmt_permit_joining_req_param_t 37
4.8 AF functions visible to applications 38
4.8.1 Detailed Description 38
4.8.2 Function Documentation 38
4.8.2.1 zb_af_set_data_indication 38
4.9 APS functions visible to applications 39
4.9.1 Detailed Description 40
4.9.2 Function Documentation 40
4.9.2.1 zb_apsde_data_request 40
4.9.3 Macro Definition Documentation 41
4.9.3.1 ZB_APS_HDR_CUT_P 41
4.9.3.2 ZB_APS_HDR_CUT 41
4.9.4 Typedef Documentation 41
4.9.4.1 zb_apsde_data_req_t 41
4.9.4.2 zb_apsme_binding_req_t 42
4.9.4.3 zb_aps_hdr_t 42
4.9.4.4 zb_apsde_data_indication_t 42
4.9.4.5 zb_apsme_add_group_req_t 42
4.9.4.6 zb_apsme_add_group_conf_t 42

4.9.5 Enumeration Type Documentation 42
4.9.5.1 zb_aps_addr_mode_e 42
4.9.5.2 zb_aps_status_e 42
4.9.5.3 zb_apsde_tx_opt_e 43
4.10 APS Informational Base 44
4.10.1 Detailed Description 45
4.10.2 Function Documentation 45
4.10.2.1 zb_apsme_get_request 45
4.10.2.2 zb_apsme_get_confirm 45
4.10.2.3 zb_apsme_set_request 45
4.10.2.4 zb_apsme_set_confirm 45
4.10.3 Typedef Documentation 45
4.10.3.1 zb_aps_aib_attr_id_t 45
4.10.3.2 zb_apsme_get_request_t 45
4.10.3.3 zb_apsme_get_confirm_t 45
4.10.3.4 zb_apsme_set_request_t 45
4.10.3.5 zb_apsme_set_confirm_t 45
4.10.4 Enumeration Type Documentation 45
4.10.4.1 zb_aps_aib_attr_id_e 45
4.11 NWK functions visible to applications 47
4.11.1 Detailed Description 48
4.11.2 Function Documentation 48
4.11.2.1 zb_nlde_data_request 48
4.11.2.2 zb_nlme_send_status 49
4.11.3 Macro Definition Documentation 49
4.11.3.1 ZB_NWK_IS_ADDRESS_BROADCAST 49
4.11.3.2 ZB_NWK_COMMAND_STATUS_FRAME_SECURITY_FAILED 50
4.11.3.3 ZB_NWK_COMMAND_STATUS_IS_SECURE 50
4.11.4 Typedef Documentation 50
4.11.4.1 zb_nwk_broadcast_address_t 50
4.11.4.2 zb_nwk_status_t 50
4.11.4.3 zb_nwk_command_status_t 50
4.11.4.4 zb_nlde_data_req_t 50
4.11.4.5 zb_nlme_status_indication_t 50
4.11.4.6 zb_nlme_send_status_t 50
4.11.5 Enumeration Type Documentation 51
4.11.5.1 zb_nwk_broadcast_address_e 51
4.11.5.2 zb_nwk_status_e 51
4.11.5.3 zb_nwk_command_status_e 52
4.12 NWK Informational Base 53

4.12.1 Detailed Description 54
4.12.2 Function Documentation 54
4.12.2.1 zb_nlme_get_request 54
4.12.2.2 zb_nlme_get_confirm 55
4.12.2.3 zb_nlme_set_request 55
4.12.2.4 zb_nlme_set_confirm 55
4.12.3 Typedef Documentation 56
4.12.3.1 zb_nlme_get_request_t 56
4.12.3.2 zb_nlme_get_confirm_t 56
4.12.3.3 zb_nlme_set_request_t 56
4.12.3.4 zb_nlme_set_confirm_t 56
4.12.3.5 zb_nib_attribute_t 56
4.12.4 Enumeration Type Documentation 56
4.12.4.1 zb_nib_attribute_e 56
4.13 MAC API 57
4.13.1 Detailed Description 59
4.13.2 Function Documentation 59
4.13.2.1 zb_mlme_get_request 59
4.13.2.2 zb_mlme_get_confirm 59
4.13.2.3 zb_mlme_set_request 59
4.13.2.4 zb_mlme_set_confirm 59
4.13.3 Macro Definition Documentation 59
4.13.3.1 MAC_PIB 59
4.13.3.2 ZB_PIB_SHORT_PAN_ID 59
4.13.3.3 ZB_PIB_SHORT_ADDRESS 59
4.13.3.4 ZB_PIB_EXTENDED_ADDRESS 59
4.13.3.5 ZB_PIB_COORD_SHORT_ADDRESS 59
4.13.3.6 ZB_PIB_RX_ON_WHEN_IDLE 60
4.13.3.7 ZB_MAC_DSN 60
4.13.3.8 ZB_MAC_BSN 60
4.13.3.9 ZB_INC_MAC_DSN 60
4.13.3.10 ZB_INC_MAC_BSN 60
4.13.3.11 ZB_PIB_BEACON_PAYLOAD 60
4.13.3.12 ZB_MLME_BUILD_GET_REQ 60
4.13.4 Typedef Documentation 60
4.13.4.1 zb_mac_status_t 60
4.13.4.2 zb_mlme_get_request_t 60
4.13.4.3 zb_mlme_get_confirm_t 61
4.13.4.4 zb_mlme_set_request_t 61
4.13.4.5 zb_mlme_set_confirm_t 61

4.13.5 Enumeration Type Documentation 61
4.13.5.1 zb_mac_status_e 61
4.13.5.2 zb_mac_pib_attr_t 62
4.14 Security subsystem API 63
4.14.1 Detailed Description 63
4.14.2 Function Documentation 63
4.14.2.1 zb_secur_setup_preconfigured_key 63
4.14.2.2 zb_secur_send_nwk_key_update_br 63
4.14.2.3 zb_secur_send_nwk_key_switch 63
4.14.2.4 secur_clear_preconfigured_key 63
4.15 Low level API 64
4.15.1 Detailed Description 64
4.16 Compile-time configuration parameters 65
4.16.1 Detailed Description 68
4.16.2 Macro Definition Documentation 68
4.16.2.1 NO_NVRAM 68
4.16.2.2 ZB_INIT_HAS_ARGS 69
4.16.2.3 ZB_SECURITY 69
4.16.2.4 ZB_TRAFFIC_DUMP_ON 69
4.16.2.5 ZB_WORD_SIZE_4 69
4.16.2.6 ZB_LITTLE_ENDIAN 69
4.16.2.7 ZB_TRANSPORT_LINUX_PIPES 69
4.16.2.8 ZB_LINUX_PIPE_TRANSPORT_TIMEOUT 69
4.16.2.9 ZB_NS_BUILD 69
4.16.2.10 ZB_MANUAL_ACK 69
4.16.2.11 ZB_UDP_PORT_REAL 69
4.16.2.12 ZB_UDP_PORT_NS 69
4.16.2.13 ZB_COORDINATOR_ROLE 70
4.16.2.14 ZB_STACK_PROFILE 70
4.16.2.15 ZB_STACK_PROFILE_2007 70
4.16.2.16 ZB_PROTOCOL_VERSION 70
4.16.2.17 ZB_SCHEDULER_Q_SIZE 70
4.16.2.18 ZB_BUF_Q_SIZE 70
4.16.2.19 ZB_IO_BUF_SIZE 70
4.16.2.20 ZB_IOBUF_POOL_SIZE 70
4.16.2.21 ZB_MAC_MAX_REQUESTS 70
4.16.2.22 ZB_MAC_RESPONSE_WAIT_TIME 70
4.16.2.23 ZB_MAX_FRAME_TOTAL_WAIT_TIME 70
4.16.2.24 ZB_MAC_MAX_FRAME_RETRIES 71
4.16.2.25 ZB_APS_DUP_CHECK_TIMEOUT 71

4.16.2.26 ZB_APS_POLL_AFTER_REQ_TMO 71
4.16.2.27 ZB_APS_SRC_BINDING_TABLE_SIZE 71
4.16.2.28 ZB_APS_DST_BINDING_TABLE_SIZE 71
4.16.2.29 ZB_APS_GROUP_TABLE_SIZE 71
4.16.2.30 ZB_APS_ENDPOINTS_IN_GROUP_TABLE 71
4.16.2.31 ZB_APS_GROUP_UP_Q_SIZE 71
4.16.2.32 ZB_APS_RETRANS_ACK_Q_SIZE 71
4.16.2.33 ZB_N_APS_RETRANS_ENTRIES 71
4.16.2.34 ZB_N_APS_MAX_FRAME_ENTRIES 71
4.16.2.35 ZB_N_APS_ACK_WAIT_DURATION 72
4.16.2.36 ZB_IEEE_ADDR_TABLE_SIZE 72
4.16.2.37 ZB_NEIGHBOR_TABLE_SIZE 72
4.16.2.38 ZB_PANID_TABLE_SIZE 72
4.16.2.39 ZB_NWK_DISTRIBUTED_ADDRESS_ASSIGN 72
4.16.2.40 ZB_NWK_ROUTING 72
4.16.2.41 N_SECUR_MATERIAL 72
4.16.2.42 ZB_NWK_TREE_ROUTING 72
4.16.2.43 ZB_NWK_MAX_CHILDREN 72
4.16.2.44 ZB_NWK_MAX_ROUTERS 72
4.16.2.45 ZB_NWK_MAX_DEPTH 72
4.16.2.46 ZB_NWK_ROUTING_TABLE_SIZE 72
4.16.2.47 ZB_NWK_ROUTE_DISCOVERY_TABLE_SIZE 73
4.16.2.48 ZB_NWK_REJOIN_REQUEST_TABLE_SIZE 73
4.16.2.49 ZB_DEFAULT_SCAN_DURATION 73
4.16.2.50 ZB_DEFAULT_PRMIT_JOINING_DURATION 73
4.16.2.51 ZB_DEFAULT_MAX_CHILDREN 73
4.16.2.52 ZB_APS_COMMAND_RADIUS 73
4.16.2.53 ZB_STANDARD_SECURITY 73
4.16.2.54 ZB_TC_GENERATES_KEYS 73
4.16.2.55 ZB_TC_AT_ZC 73
4.16.2.56 ZB_CCM_KEY_SIZE 73
4.16.2.57 ZB_SECURITY_LEVEL 73
4.16.2.58 ZB_CCM_L 74
4.16.2.59 ZB_CCM_NONCE_LEN 74
4.16.2.60 ZB_CCM_M 74
4.16.2.61 ZB_SECUR_NWK_COUNTER_LIMIT 74
4.16.2.62 ZB_DEFAULT_SECURE_ALL_FRAMES 74
4.16.2.63 ZB_ZCL_CLUSTER_NUM 74
4.16.2.64 ZB_ZDO_INDIRECT_POLL_TIMER 74
4.16.2.65 ZB_ZDO_MAX_PARENT_THRESHOLD_RETRY 74

4.16.2.66 ZB_ZDO_MIN_SCAN_DURATION 74
4.16.2.67 ZB_ZDO_MAX_SCAN_DURATION 74
4.16.2.68 ZB_ZDO_NEW_ACTIVE_CHANNEL 74
4.16.2.69 ZB_ZDO_NEW_CHANNEL_MASK 75
4.16.2.70 ZB_ZDO_CHANNEL_CHECK_TIMEOUT 75
4.16.2.71 ZB_ZDO_APS_CHANEL_TIMER 75
4.16.2.72 ZB_ZDO_15_MIN_TIMEOUT 75
4.16.2.73 ZB_ZDO_1_MIN_TIMEOUT 75
4.16.2.74 ZB_ZDO_NWK_SCAN_ATTEMPTS 75
4.16.2.75 ZB_ZDO_NWK_TIME_BTWN_SCANS 75
4.16.2.76 ZB_ZDO_ENDDEV_BIND_TIMEOUT 75
4.16.2.77 ZDO_TRAN_TABLE_SIZE 75
4.16.2.78 ZB_ZDO_PENDING_LEAVE_SIZE 75
4.16.2.79 ZB_ZDO_PARENT_LINK_FAILURE_CNT 75
4.17 Base typedefs 76
4.17.1 Detailed Description 78
4.17.2 Function Documentation 78
4.17.2.1 zb_put_next_htole16 78
4.17.3 Macro Definition Documentation 79
4.17.3.1 ZB_SHORT_MIN 79
4.17.3.2 ZB_IS_64BIT_ADDR_ZERO 79
4.17.3.3 ZB_64BIT_ADDR_ZERO 79
4.17.3.4 ZB_64BIT_ADDR_COPY 79
4.17.3.5 ZB_64BIT_ADDR_CMP 79
4.17.3.6 ZB_ADDR_CMP 79
4.17.3.7 ZB_INT8_C 79
4.17.3.8 ZB_HTOLE16 79
4.17.3.9 ZB_LETOH16 80
4.17.4 Typedef Documentation 80
4.17.4.1 zb_bool_t 80
4.17.4.2 zb_char_t 80
4.17.4.3 zb_uchar_t 80
4.17.4.4 zb_uint8_t 80
4.17.4.5 zb_int8_t 80
4.17.4.6 zb_uint16_t 80
4.17.4.7 zb_int16_t 80
4.17.4.8 zb_uint32_t 80
4.17.4.9 zb_int32_t 80
4.17.4.10 zb_bitfield_t 80
4.17.4.11 zb_sbitfield_t 81

4.17.4.12 zb_short_t 81
4.17.4.13 zb_ushort_t 81
4.17.4.14 zb_int_t 81
4.17.4.15 zb_uint_t 81
4.17.4.16 zb_long_t 81
4.17.4.17 zb_ulong_t 81
4.17.4.18 zb_voidp_t 81
4.17.4.19 zb_64bit_addr_t 81
4.17.4.20 zb_ieee_addr_t 81
4.17.4.21 zb_ext_pan_id_t 81
4.17.5 Enumeration Type Documentation 81
4.17.5.1 zb_bool_e 81
4.18 Packet buffers pool 82
4.18.1 Detailed Description 83
4.18.2 Function Documentation 83
4.18.2.1 zb_buf_initial_alloc 83
4.18.2.2 zb_get_buf_tail 83
4.18.2.3 zb_buf_assign_param 84
4.18.2.4 zb_buf_reuse 84
4.18.2.5 zb_init_buffers 84
4.18.2.6 zb_get_in_buf 84
4.18.2.7 zb_get_out_buf 84
4.18.2.8 zb_free_buf 85
4.18.2.9 zb_get_in_buf_delayed 85
4.18.2.10 zb_get_out_buf_delayed 85
4.18.3 Macro Definition Documentation 85
4.18.3.1 ZB_BUF_BEGIN 85
4.18.3.2 ZB_BUF_LEN 85
4.18.3.3 ZB_BUF_OFFSET 86
4.18.3.4 ZB_BUF_ALLOC_LEFT 86
4.18.3.5 ZB_BUF_ALLOC_RIGHT 86
4.18.3.6 ZB_BUF_CUT_LEFT 86
4.18.3.7 ZB_BUF_CUT_LEFT2 86
4.18.3.8 ZB_BUF_CUT_RIGHT 86
4.18.3.9 ZB_BUF_COPY 87
4.18.4 Typedef Documentation 87
4.18.4.1 zb_buf_hdr_t 87
4.18.4.2 zb_buf_s_t 87
4.19 Scheduler 88
4.19.1 Detailed Description 89

4.19.2 Function Documentation 89
4.19.2.1 ZB_RING_BUFFER_DECLARE 89
4.19.2.2 zb_sched_init 89
4.19.2.3 zb_sched_loop_iteration 89
4.19.2.4 zb_schedule_callback 90
4.19.2.5 zb_schedule_mac_cb 90
4.19.2.6 zb_schedule_alarm 90
4.19.2.7 zb_schedule_alarm_cancel 90
4.19.3 Macro Definition Documentation 91
4.19.3.1 ZB_ALARM_ANY_PARAM 91
4.19.3.2 ZB_ALARM_ALL_CB 91
4.19.3.3 ZB_SCHED_HAS_PENDING_CALLBACKS 91
4.19.3.4 ZB_SCHED_WAIT_COND 91
4.19.3.5 ZB_SCHED_GLOBAL_LOCK 91
4.19.3.6 ZB_SCHED_GLOBAL_UNLOCK 91
4.19.3.7 ZB_SCHED_GLOBAL_LOCK_INT 91
4.19.3.8 ZB_SCHED_GLOBAL_UNLOCK_INT 92
4.19.4 Typedef Documentation 92
4.19.4.1 zb_callback_t 92
4.19.4.2 zb_cb_q_ent_t 92
4.19.4.3 zb_tm_q_ent_t 92
4.19.4.4 zb_sched_globals_t 92
4.20 Time 93
4.20.1 Detailed Description 93
4.20.2 Macro Definition Documentation 93
4.20.2.1 ZB_TIMER_GET 93
4.20.2.2 ZB_TIME_SUBTRACT 93
4.20.2.3 ZB_TIME_ADD 94
4.20.2.4 ZB_TIME_GE 94
4.20.2.5 ZB_BEACON_INTERVAL_USEC 94
4.20.2.6 ZB_TIME_ONE_SECOND 94
4.20.2.7 ZB_TIME_BEACON_INTERVAL_TO_MSEC 94
4.20.2.8 ZB_MILLISECONDS_TO_BEACON_INTERVAL 95
4.20.2.9 ZB_TIMER_START 95
4.20.3 Typedef Documentation 95
4.20.3.1 zb_time_t 95
4.21 Debug trace 96
4.21.1 Detailed Description 99
4.21.2 Macro Definition Documentation 99
4.21.2.1 TRACE_MSG 99

4.21.2.2 TRACE_FORMAT_64 100
4.21.2.3 TRACE_ERROR 100
4.21.2.4 TRACE_SUBSYSTEM_COMMON 100
4.21.2.5 TRACE_COMMON1 100

5 Data Structure Documentation 101
5.1 zb_addr64_struct_s Struct Reference 101
5.2 zb_addr_u Union Reference 101
5.2.1 Detailed Description 101
5.3 zb_aps_hdr_s Struct Reference 101
5.3.1 Detailed Description 102
5.4 zb_apsde_data_req_s Struct Reference 102
5.4.1 Detailed Description 102
5.4.2 Field Documentation 102
5.4.2.1 dst_addr 102
5.4.2.2 profileid 102
5.4.2.3 clusterid 103
5.4.2.4 dst_endpoint 103
5.4.2.5 src_endpoint 103
5.4.2.6 radius 103
5.4.2.7 addr_mode 103
5.4.2.8 tx_options 103
5.5 zb_apsme_add_group_conf_s Struct Reference 103
5.5.1 Detailed Description 104
5.5.2 Field Documentation 104
5.5.2.1 group_address 104
5.5.2.2 endpoint 104
5.6 zb_apsme_add_group_req_s Struct Reference 104
5.6.1 Detailed Description 104
5.6.2 Field Documentation 104
5.6.2.1 group_address 104
5.6.2.2 endpoint 104
5.7 zb_apsme_binding_req_s Struct Reference 104
5.7.1 Detailed Description 105
5.7.2 Field Documentation 105
5.7.2.1 src_addr 105
5.7.2.2 src_endpoint 105
5.7.2.3 clusterid 105
5.7.2.4 addr_mode 105
5.7.2.5 dst_addr 105

5.7.2.6 dst_endpoint 105
5.8 zb_apsme_get_confirm_s Struct Reference 106
5.8.1 Detailed Description 106
5.8.2 Field Documentation 106
5.8.2.1 status 106
5.8.2.2 aib_attr 106
5.8.2.3 aib_length 106
5.9 zb_apsme_get_request_s Struct Reference 106
5.9.1 Detailed Description 106
5.9.2 Field Documentation 107
5.9.2.1 aib_attr 107
5.10 zb_apsme_set_confirm_s Struct Reference 107
5.10.1 Detailed Description 107
5.10.2 Field Documentation 107
5.10.2.1 status 107
5.10.2.2 aib_attr 107
5.11 zb_apsme_set_request_s Struct Reference 107
5.11.1 Detailed Description 108
5.11.2 Field Documentation 108
5.11.2.1 aib_attr 108
5.11.2.2 aib_length 108
5.12 zb_buf_hdr_s Struct Reference 108
5.12.1 Detailed Description 108
5.12.2 Field Documentation 108
5.12.2.1 len 108
5.12.2.2 data_offset 108
5.12.2.3 handle 109
5.12.2.4 status 109
5.12.2.5 is_in_buf 109
5.12.2.6 encrypt_type 109
5.12.2.7 use_same_key 109
5.12.2.8 zdo_cmd_no_resp 109
5.13 zb_buf_q_ent_s Struct Reference 109
5.13.1 Field Documentation 109
5.13.1.1 func 109
5.14 zb_buf_s Struct Reference 110
5.14.1 Detailed Description 110
5.15 zb_cb_q_ent_s Struct Reference 110
5.15.1 Detailed Description 110
5.15.2 Field Documentation 110

5.15.2.1 func 110
5.15.2.2 param 110
5.16 zb_end_device_bind_req_param_s Struct Reference 111
5.16.1 Detailed Description 111
5.16.2 Field Documentation 111
5.16.2.1 dst_addr 111
5.16.2.2 head_param 111
5.16.2.3 tail_param 111
5.16.2.4 cluster_list 111
5.17 zb_mac_cb_ent_s Struct Reference 111
5.18 zb_mac_device_table_s Struct Reference 112
5.19 zb_mlme_get_confirm_s Struct Reference 112
5.19.1 Detailed Description 112
5.20 zb_mlme_get_request_s Struct Reference 112
5.20.1 Detailed Description 112
5.21 zb_mlme_set_confirm_s Struct Reference 113
5.21.1 Detailed Description 113
5.22 zb_mlme_set_request_s Struct Reference 113
5.22.1 Detailed Description 113
5.23 zb_nlde_data_req_s Struct Reference 113
5.23.1 Detailed Description 114
5.23.2 Field Documentation 114
5.23.2.1 dst_addr 114
5.23.2.2 radius 114
5.23.2.3 addr_mode 114
5.23.2.4 nonmember_radius 114
5.23.2.5 discovery_route 114
5.23.2.6 security_enable 114
5.23.2.7 ndsu_handle 114
5.24 zb_nlme_get_confirm_s Struct Reference 115
5.24.1 Detailed Description 115
5.24.2 Field Documentation 115
5.24.2.1 status 115
5.24.2.2 nib_attribute 115
5.24.2.3 attribute_length 115
5.25 zb_nlme_get_request_s Struct Reference 115
5.25.1 Detailed Description 115
5.25.2 Field Documentation 116
5.25.2.1 nib_attribute 116
5.26 zb_nlme_send_status_s Struct Reference 116

5.26.1 Detailed Description 116
5.26.2 Field Documentation 116
5.26.2.1 dest_addr 116
5.26.2.2 status 116
5.26.2.3 ndsu_handle 116
5.27 zb_nlme_set_confirm_s Struct Reference 117
5.27.1 Detailed Description 117
5.27.2 Field Documentation 117
5.27.2.1 status 117
5.27.2.2 nib_attribute 117
5.28 zb_nlme_set_request_s Struct Reference 117
5.28.1 Detailed Description 117
5.28.2 Field Documentation 118
5.28.2.1 nib_attribute 118
5.29 zb_nlme_status_indication_s Struct Reference 118
5.29.1 Detailed Description 118
5.29.2 Field Documentation 118
5.29.2.1 status 118
5.29.2.2 network_addr 118
5.30 ZB_PACKED_STRUCT Struct Reference 118
5.30.1 Detailed Description 119
5.30.2 Field Documentation 119
5.30.2.1 mac_ack_wait_duration 119
5.30.2.2 mac_association_permit 119
5.30.2.3 mac_auto_request 119
5.30.2.4 mac_batt_life_ext 119
5.30.2.5 mac_beacon_payload 119
5.30.2.6 mac_beacon_payload_length 120
5.30.2.7 mac_beacon_order 120
5.30.2.8 mac_bsn 120
5.30.2.9 mac_coord_extended_address 120
5.30.2.10 mac_coord_short_address 120
5.30.2.11 mac_dsn 120
5.30.2.12 mac_pan_id 120
5.30.2.13 mac_rx_on_when_idle 120
5.30.2.14 mac_short_address 120
5.30.2.15 mac_superframe_order 120
5.30.2.16 mac_max_frame_retries 120
5.30.2.17 mac_extended_address 120
5.31 zb_sched_globals_s Struct Reference 121

5.31.1 Detailed Description 121
5.31.2 Member Function Documentation 121
5.31.2.1 ZB_LIST_DEFINE 121
5.31.2.2 ZB_STK_DEFINE 121
5.31.3 Field Documentation 121
5.31.3.1 cb_q 121
5.31.3.2 tm_buffer 121
5.32 zb_tm_q_ent_s Struct Reference 122
5.32.1 Detailed Description 122
5.32.2 Field Documentation 122
5.32.2.1 func 122
5.32.2.2 param 122
5.32.2.3 run_time 122
5.33 zb_zdo_active_ep_req_s Struct Reference 122
5.33.1 Detailed Description 123
5.33.2 Field Documentation 123
5.33.2.1 nwk_addr 123
5.34 zb_zdo_bind_req_head_s Struct Reference 123
5.34.1 Detailed Description 123
5.34.2 Field Documentation 123
5.34.2.1 src_address 123
5.34.2.2 src_endp 123
5.34.2.3 cluster_id 123
5.34.2.4 dst_addr_mode 123
5.35 zb_zdo_bind_req_param_s Struct Reference 124
5.35.1 Detailed Description 124
5.35.2 Field Documentation 124
5.35.2.1 src_address 124
5.35.2.2 src_endp 124
5.35.2.3 cluster_id 124
5.35.2.4 dst_addr_mode 124
5.35.2.5 dst_address 124
5.35.2.6 dst_endp 125
5.35.2.7 req_dst_addr 125
5.36 zb_zdo_bind_req_tail_1_s Struct Reference 125
5.36.1 Detailed Description 125
5.36.2 Field Documentation 125
5.36.2.1 dst_addr 125
5.37 zb_zdo_bind_req_tail_2_s Struct Reference 125
5.37.1 Detailed Description 125

5.37.2 Field Documentation 126
5.37.2.1 dst_addr 126
5.37.2.2 dst_endp 126
5.38 zb_zdo_bind_resp_s Struct Reference 126
5.39 zb_zdo_configuration_attributes_e Struct Reference 126
5.39.1 Field Documentation 126
5.39.1.1 permit_join_duration 126
5.40 zb_zdo_desc_resp_hdr_s Struct Reference 126
5.40.1 Detailed Description 127
5.40.2 Field Documentation 127
5.40.2.1 status 127
5.40.2.2 nwk_addr 127
5.41 zb_zdo_end_device_bind_req_head_s Struct Reference 127
5.41.1 Detailed Description 127
5.41.2 Field Documentation 127
5.41.2.1 binding_target 127
5.41.2.2 src_ieee_addr 127
5.41.2.3 src_endp 128
5.41.2.4 profile_id 128
5.41.2.5 num_in_cluster 128
5.42 zb_zdo_end_device_bind_req_tail_s Struct Reference 128
5.42.1 Detailed Description 128
5.42.2 Field Documentation 128
5.42.2.1 num_out_cluster 128
5.43 zb_zdo_end_device_bind_resp_s Struct Reference 128
5.44 zb_zdo_ep_resp_s Struct Reference 129
5.44.1 Detailed Description 129
5.44.2 Field Documentation 129
5.44.2.1 status 129
5.44.2.2 nwk_addr 129
5.44.2.3 ep_count 129
5.45 zb_zdo_ieee_addr_req_s Struct Reference 129
5.45.1 Detailed Description 129
5.45.2 Field Documentation 130
5.45.2.1 nwk_addr 130
5.45.2.2 request_type 130
5.45.2.3 start_index 130
5.46 zb_zdo_match_desc_param_s Struct Reference 130
5.46.1 Detailed Description 130
5.46.2 Field Documentation 130

5.46.2.1 nwk_addr 130
5.46.2.2 profile_id 130
5.46.2.3 num_in_clusters 130
5.46.2.4 num_out_clusters 131
5.46.2.5 cluster_list 131
5.47 zb_zdo_match_desc_req_head_s Struct Reference 131
5.47.1 Detailed Description 131
5.47.2 Field Documentation 131
5.47.2.1 nwk_addr 131
5.47.2.2 profile_id 131
5.47.2.3 num_in_clusters 131
5.48 zb_zdo_match_desc_req_tail_s Struct Reference 132
5.48.1 Detailed Description 132
5.48.2 Field Documentation 132
5.48.2.1 num_out_clusters 132
5.49 zb_zdo_match_desc_resp_s Struct Reference 132
5.49.1 Detailed Description 132
5.49.2 Field Documentation 132
5.49.2.1 status 132
5.49.2.2 nwk_addr 132
5.49.2.3 match_len 133
5.50 zb_zdo_mgmt_leave_param_s Struct Reference 133
5.50.1 Detailed Description 133
5.50.2 Field Documentation 133
5.50.2.1 device_address 133
5.50.2.2 dst_addr 133
5.51 zb_zdo_mgmt_leave_req_s Struct Reference 133
5.51.1 Detailed Description 134
5.51.2 Field Documentation 134
5.51.2.1 device_address 134
5.52 zb_zdo_mgmt_leave_res_s Struct Reference 134
5.52.1 Detailed Description 134
5.53 zb_zdo_mgmt_lqi_param_s Struct Reference 134
5.53.1 Detailed Description 135
5.53.2 Field Documentation 135
5.53.2.1 start_index 135
5.53.2.2 dst_addr 135
5.54 zb_zdo_mgmt_lqi_req_s Struct Reference 135
5.54.1 Detailed Description 135
5.54.2 Field Documentation 135

5.54.2.1 start_index 135
5.55 zb_zdo_mgmt_lqi_resp_s Struct Reference 135
5.55.1 Detailed Description 136
5.55.2 Field Documentation 136
5.55.2.1 status 136
5.55.2.2 neighbor_table_entries 136
5.55.2.3 start_index 136
5.55.2.4 neighbor_table_list_count 136
5.56 zb_zdo_mgmt_nwk_update_notify_hdr_s Struct Reference 136
5.56.1 Detailed Description 136
5.56.2 Field Documentation 137
5.56.2.1 status 137
5.56.2.2 scanned_channels 137
5.56.2.3 total_transmissions 137
5.56.2.4 transmission_failures 137
5.56.2.5 scanned_channels_list_count 137
5.57 zb_zdo_mgmt_nwk_update_notify_param_s Struct Reference 137
5.57.1 Detailed Description 137
5.57.2 Field Documentation 137
5.57.2.1 hdr 137
5.57.2.2 energy_values 138
5.57.2.3 dst_addr 138
5.57.2.4 tsn 138
5.58 zb_zdo_mgmt_nwk_update_req_hdr_s Struct Reference 138
5.58.1 Detailed Description 138
5.58.2 Field Documentation 138
5.58.2.1 scan_channels 138
5.58.2.2 scan_duration 138
5.59 zb_zdo_mgmt_nwk_update_req_s Struct Reference 138
5.59.1 Detailed Description 139
5.59.2 Field Documentation 139
5.59.2.1 hdr 139
5.59.2.2 scan_count 139
5.59.2.3 update_id 139
5.59.2.4 manager_addr 139
5.59.2.5 dst_addr 139
5.60 zb_zdo_mgmt_permit_joining_req_param_s Struct Reference 139
5.60.1 Detailed Description 140
5.61 zb_zdo_mgmt_permit_joining_req_s Struct Reference 140
5.61.1 Detailed Description 140

5.62 zb_zdo_neighbor_table_record_s Struct Reference 140
5.62.1 Detailed Description 140
5.62.2 Field Documentation 140
5.62.2.1 ext_pan_id 140
5.62.2.2 ext_addr 141
5.62.2.3 network_addr 141
5.62.2.4 type_flags 141
5.62.2.5 permit_join 141
5.62.2.6 depth 141
5.62.2.7 lqi 141
5.63 zb_zdo_node_desc_req_s Struct Reference 141
5.63.1 Detailed Description 141
5.63.2 Field Documentation 141
5.63.2.1 nwk_addr 141
5.64 zb_zdo_node_desc_resp_s Struct Reference 142
5.64.1 Detailed Description 142
5.64.2 Field Documentation 142
5.64.2.1 hdr 142
5.64.2.2 node_desc 142
5.65 zb_zdo_nwk_addr_req_param_s Struct Reference 142
5.65.1 Detailed Description 142
5.65.2 Field Documentation 142
5.65.2.1 dst_addr 142
5.65.2.2 ieee_addr 143
5.65.2.3 request_type 143
5.65.2.4 start_index 143
5.66 zb_zdo_nwk_addr_req_s Struct Reference 143
5.66.1 Detailed Description 143
5.66.2 Field Documentation 143
5.66.2.1 ieee_addr 143
5.66.2.2 request_type 143
5.66.2.3 start_index 143
5.67 zb_zdo_nwk_addr_resp_head_s Struct Reference 144
5.67.1 Field Documentation 144
5.67.1.1 status 144
5.67.1.2 ieee_addr 144
5.67.1.3 nwk_addr 144
5.68 zb_zdo_power_desc_req_s Struct Reference 144
5.68.1 Detailed Description 144
5.68.2 Field Documentation 144

5.68.2.1 nwk_addr 144
5.69 zb_zdo_power_desc_resp_s Struct Reference 145
5.69.1 Detailed Description 145
5.69.2 Field Documentation 145
5.69.2.1 hdr 145
5.69.2.2 power_desc 145
5.70 zb_zdo_simple_desc_req_s Struct Reference 145
5.70.1 Detailed Description 145
5.70.2 Field Documentation 145
5.70.2.1 nwk_addr 145
5.70.2.2 endpoint 146
5.71 zb_zdo_simple_desc_resp_hdr_s Struct Reference 146
5.71.1 Detailed Description 146
5.71.2 Field Documentation 146
5.71.2.1 status 146
5.71.2.2 nwk_addr 146
5.71.2.3 length 146
5.72 zb_zdo_simple_desc_resp_s Struct Reference 146
5.72.1 Detailed Description 147
5.72.2 Field Documentation 147
5.72.2.1 hdr 147
5.72.2.2 simple_desc 147
5.73 zb_zdo_system_server_discovery_req_s Struct Reference 147
5.73.1 Detailed Description 147
5.73.2 Field Documentation 147
5.73.2.1 server_mask 147
5.74 zb_zdo_system_server_discovery_resp_s Struct Reference 147
5.74.1 Detailed Description 148
5.74.2 Field Documentation 148
5.74.2.1 status 148
5.74.2.2 server_mask 148

Chapter 1

ZBOSS v1.0

ZBOSS v1.0 is the open-source *ZigBee®* protocol stack implementing *ZigBee® 2007* specification
certified by the *ZigBee® Alliance*. **ZBOSS** is a high-performance, small memory footprint, cross-platform solution.
This document provides *ZBOSS v1.0 API manual*, go to `API sections` or `Data structure` tabs for details.

Chapter 2

Module Index

2.1 API sections

Here is a list of all modules:

Stack initialization API 9
    ZDO init and main() structure 10
ZDO API 12
    ZDO Informational Base 13
    ZDO base constants and definitions 14
    ZDO discovery services 16
    ZDO management services 28
AF functions visible to applications 38
APS functions visible to applications 39
    APS Informational Base 44
NWK functions visible to applications 47
    NWK Informational Base 53
MAC API 57
Security subsystem API 63
Low level API 64
    Compile-time configuration parameters 65
    Base typedefs 76
    Packet buffers pool 82
    Scheduler 88
    Time 93
    Debug trace 96

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

zb_addr64_struct_s 101
zb_addr_u
Union to address either long or short address 101
zb_aps_hdr_s
Parsed APS header This data structure passed to zb_aps_hdr_parse() 101
zb_apsde_data_req_s
APSDE data request structure 102
zb_apsme_add_group_conf_s
APSME-ADD-GROUP.confirm primitive parameters 103
zb_apsme_add_group_req_s
APSME-ADD-GROUP.request primitive parameters 104
zb_apsme_binding_req_s
APSME binding structure 104
zb_apsme_get_confirm_s
APSME GET confirm structure 106
zb_apsme_get_request_s
APSME GET request structure 106
zb_apsme_set_confirm_s
APSME SET confirm structure 107
zb_apsme_set_request_s
APSME SET request structure 107
zb_buf_hdr_s
Packet buffer header 108
zb_buf_q_ent_s 109
zb_buf_s
Packet buffer 110
zb_cb_q_ent_s
Immediate pending callbacks queue entry 110
zb_end_device_bind_req_param_s
Parameters for 2.4.3.2.1 End_Device_Bind_req 111
zb_mac_cb_ent_s 111
zb_mac_device_table_s 112
zb_mlme_get_confirm_s
Defines MLME-GET.confirm primitive 112
zb_mlme_get_request_s
Defines MLME-GET.request primitive 112
zb_mlme_set_confirm_s
Defines MLME-SET.confirm primitive 113

zb_mlme_set_request_s
Defines MLME-SET.request primitive 113
zb_nlde_data_req_s
Parameters for NLDE-DATA.request primitive 113
zb_nlme_get_confirm_s
Arguments of the NLME-GET.confirm routine 115
zb_nlme_get_request_s
Arguments of the NLME-GET.request routine 115
zb_nlme_send_status_s
Arguments of the NLME-SEND-STATUS.confirm routine 116
zb_nlme_set_confirm_s
Arguments of the NLME-SET.confirm routine 117
zb_nlme_set_request_s
Arguments of the NLME-SET.request routine 117
zb_nlme_status_indication_s
Arguments of the NLME-STATUS.request routine 118
ZB_PACKED_STRUCT
MAC PIB 118
zb_sched_globals_s
Data structures for the delayed execution 121
zb_tm_q_ent_s
Delayed (scheduled to run after timeout) callbacks queue entry 122
zb_zdo_active_ep_req_s
Parameters of Active_desc_req primitive 122
zb_zdo_bind_req_head_s
2.4.3.2.2 Bind_req request head send to the remote 123
zb_zdo_bind_req_param_s
Parameters for 2.4.3.2.2 Bind_req API call 124
zb_zdo_bind_req_tail_1_s
2.4.3.2.2 Bind_req request tail 1st variant send to the remote 125
zb_zdo_bind_req_tail_2_s
2.4.3.2.2 Bind_req request tail 2nd variant send to the remote 125
zb_zdo_bind_resp_s 126
zb_zdo_configuration_attributes_e 126
zb_zdo_desc_resp_hdr_s
Header of Node_desc_resp primitive 126
zb_zdo_end_device_bind_req_head_s
2.4.3.2.1 End_Device_Bind_req command head 127
zb_zdo_end_device_bind_req_tail_s
2.4.3.2.1 End_Device_Bind_req command head 128
zb_zdo_end_device_bind_resp_s 128
zb_zdo_ep_resp_s
Active EP response 129
zb_zdo_ieee_addr_req_s
Parameters of IEEE_addr_req primitive 129
zb_zdo_match_desc_param_s
Parameters of match_desc_req primitive 130
zb_zdo_match_desc_req_head_s
Match_desc_req head 131
zb_zdo_match_desc_req_tail_s
Match_desc_req tail 132
zb_zdo_match_desc_resp_s
2.4.4.1.7 Match_Desc_rsp response structure 132
zb_zdo_mgmt_leave_param_s
Request for 2.4.3.3.5 Mgmt_Leave_req 133
zb_zdo_mgmt_leave_req_s
Request for 2.4.3.3.5 Mgmt_Leave_req 133

zb_zdo_mgmt_leave_res_s
Response for 2.4.4.3.5 Mgmt_Leave_rsp 134
zb_zdo_mgmt_lqi_param_s
Parameters for 2.4.3.3.2 Mgmt_Lqi_req 134
zb_zdo_mgmt_lqi_req_s
Request for 2.4.3.3.2 Mgmt_Lqi_req 135
zb_zdo_mgmt_lqi_resp_s
Response for 2.4.4.3.2 Mgmt_Lqi_rsp 135
zb_zdo_mgmt_nwk_update_notify_hdr_s
Header parameters for mgmt_nwk_update_notify 136
zb_zdo_mgmt_nwk_update_notify_param_s
Parameters for mgmt_nwk_update_notify 137
zb_zdo_mgmt_nwk_update_req_hdr_s
Header of parameters for Mgmt_NWK_Update_req 138
zb_zdo_mgmt_nwk_update_req_s
Parameters for Mgmt_NWK_Update_req 138
zb_zdo_mgmt_permit_joining_req_param_s
Parameters for zb_zdo_mgmt_permit_joining_req 139
zb_zdo_mgmt_permit_joining_req_s
Parameters for 2.4.3.3.7 Mgmt_Permit_Joining_req 140
zb_zdo_neighbor_table_record_s
NeighborTableList Record Format for mgmt_lqi_resp 140
zb_zdo_node_desc_req_s
Parameters of Node_desc_req primitive 141
zb_zdo_node_desc_resp_s
Parameters of Node_desc_resp primitive 142
zb_zdo_nwk_addr_req_param_s
Parameters for nwk_addr_req command 142
zb_zdo_nwk_addr_req_s
NWK_addr_req command primitive 143
zb_zdo_nwk_addr_resp_head_s 144
zb_zdo_power_desc_req_s
Parameters of Power_desc_req primitive 144
zb_zdo_power_desc_resp_s
Parameters of Power_desc_resp primitive 145
zb_zdo_simple_desc_req_s
Parameters of Power_desc_req primitive 145
zb_zdo_simple_desc_resp_hdr_s
Header of Node_desc_resp primitive 146
zb_zdo_simple_desc_resp_s
Parameters of simple_desc_resp primitive 146
zb_zdo_system_server_discovery_req_s
Request parameters for 2.4.3.1.13 System_Server_Discovery_req 147
zb_zdo_system_server_discovery_resp_s
Response parameters for 2.4.4.1.10 System_Server_Discovery_rsp 147

Chapter 4

Module Documentation

4.1 Stack initialization API

Functions

- void **zb_init** () ZB_CALLBACK

  Global stack initialization.
- void **zb_handle_parms_before_start** ()

Modules

- **ZDO init and main() structure**

Macros

- #define **ZB_INIT**(a, b, c) **zb_init**()

4.1.1 Detailed Description

4.1.2 Function Documentation

4.1.2.1 void zb_init ( )

Global stack initialization.

To be called from main() at start.

Usual initialization sequence: **zb_init()** (p. 9), then assign some IB values, then zdo_startup().

Parameters

trace_comment	- trace file name component (for Unix)
rx_pipe	- rx pipe name (for Unix/ns build) or node number (for ns build in 8051 simulator)
tx_pipe	- tx pipe (for Unix)

Example:

```
#ifndef ZB8051
  zb_init("zdo_zc", argv[1], argv[2]);
#else
  zb_init("zdo_zc", "1", "1");
#endif
```

4.2 ZDO init and main() structure

Functions

- zb_ret_t **zdo_dev_start** () ZB_SDCC_REENTRANT

  Typical device start: init, load some parameters from nvram and proceed with startup.
- void **zdo_main_loop** ()

  Application main loop.
- void **zb_zdo_startup_complete** (**zb_uint8_t** param) ZB_CALLBACK

  Callback which will be called after device startup complete.

4.2.1 Detailed Description

4.2.2 Function Documentation

4.2.2.1 zb_ret_t zdo_dev_start ( )

Typical device start: init, load some parameters from nvram and proceed with startup.

Startup means either Formation (for ZC), rejoin or discovery/association join. After startup complete zb_zdo_-startup_complete callback is called, so application will know when to do some useful things.

Precondition: stack must be inited by **zb_init()** (p. 9) call. **zb_init()** (p. 9) loads IB from NVRAM or set its defaults, so caller has a chanse to change some parameters. Note: ZB is not looped in this routine. Instead, it schedules callback and returns. Caller must run **zdo_main_loop()** (p. 10) after this routine.

Example:

```
zb_init("zdo_zc", "1", "1");
ZB_AIB().aps_designated_coordinator = 1;
ZB_IEEE_ADDR_COPY(ZB_PIB_EXTENDED_ADDRESS(), &g_zc_addr);
MAC_PIB().mac_pan_id = 0x1aaa;
ZG->nwk.nib.max_children = 1;
if (zdo_dev_start() != RET_OK)
{
  TRACE_MSG(TRACE_ERROR, "zdo_dev_start failed", (FMT__0));
}
else
{
  zdo_main_loop();
}
```

4.2.2.2 void zdo_main_loop ( )

Application main loop.

Must be called after **zb_init()** (p. 9) and **zdo_dev_start()** (p. 10).

Example:

```
zb_init("zdo_zc", "1", "1");
ZB_AIB().aps_designated_coordinator = 1;
ZB_IEEE_ADDR_COPY(ZB_PIB_EXTENDED_ADDRESS(), &g_zc_addr);
MAC_PIB().mac_pan_id = 0x1aaa;
ZG->nwk.nib.max_children = 1;
if (zdo_dev_start() != RET_OK)
{
  TRACE_MSG(TRACE_ERROR, "zdo_dev_start failed", (FMT__0));
}
else
{
  zdo_main_loop();
}
```

4.2.2.3 void zb_zdo_startup_complete (zb_uint8_t *param*)

Callback which will be called after device startup complete.

Must be defined in the application.

Parameters

param	- ref to buffer with startup status

Example:

```
void zb_zdo_startup_complete(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  TRACE_MSG(TRACE_APS3, ">>zb_zdo_startup_complete status %hd", (FMT__D, buf->u
      .hdr.status));
  if (buf->u.hdr.status == 0)
  {
    TRACE_MSG(TRACE_APS1, "Device STARTED OK", (FMT__0));
    zb_af_set_data_indication(data_indication);
  }
  else
  {
    TRACE_MSG(TRACE_ERROR, "Device STARTE FAILED status %hd", (FMT__D, buf->u.
      hdr.status));
  }
  zb_free_buf(buf);
}
```

4.3 ZDO API

Modules

- **ZDO Informational Base**
- **ZDO base constants and definitions**
- **ZDO discovery services**
- **ZDO management services**

4.3.1 Detailed Description

4.4 ZDO Informational Base

Data Structures

- struct **zb_zdo_configuration_attributes_e**

Macros

- #define **ZB_ZDO_NODE_DESC**() (&ZG->zdo.conf_attr.node_desc)
- #define **ZB_ZDO_NODE_POWER_DESC**() (&ZG->zdo.conf_attr.node_power_desc)
- #define **ZB_ZDO_SIMPLE_DESC**() (&ZG->zdo.conf_attr.zdo_simple_desc)
- #define **ZB_ZDO_SIMPLE_DESC_LIST**() (ZG->zdo.conf_attr.simple_desc_list)
- #define **ZB_ZDO_SIMPLE_DESC_NUMBER**() (ZG->zdo.conf_attr.simple_desc_number)

Typedefs

- typedef struct
  zb_zdo_configuration_attributes_e zb_zdo_configuration_attributes_t

4.4.1 Detailed Description

4.5 ZDO base constants and definitions

Typedefs

- typedef enum **zb_zdp_status_e zb_zdp_status_t**

  ZDP status values (2.4.5 ZDP Enumeration Description)

Enumerations

- enum **zb_zdp_status_e** {
  ZB_ZDP_STATUS_SUCCESS = 0x00, **ZB_ZDP_STATUS_INV_REQUESTTYPE** = 0x80, **ZB_ZDP_STAT-US_DEVICE_NOT_FOUND** = 0x81, **ZB_ZDP_STATUS_INVALID_EP** = 0x82,
  ZB_ZDP_STATUS_NOT_ACTIVE = 0x83, **ZB_ZDP_STATUS_NOT_SUPPORTED** = 0x84, **ZB_ZDP_STA-TUS_TIMEOUT** = 0x85, **ZB_ZDP_STATUS_NO_MATCH** = 0x86,
  ZB_ZDP_STATUS_NO_ENTRY = 0x88, **ZB_ZDP_STATUS_NO_DESCRIPTOR** = 0x89, **ZB_ZDP_STATU-S_INSUFFICIENT_SPACE** = 0x8a, **ZB_ZDP_STATUS_NOT_PERMITTED** = 0x8b,
  ZB_ZDP_STATUS_TABLE_FULL = 0x8c, **ZB_ZDP_STATUS_NOT_AUTHORIZED** = 0x8d }

  ZDP status values (2.4.5 ZDP Enumeration Description)

4.5.1 Detailed Description

4.5.2 Typedef Documentation

4.5.2.1 typedef enum zb_zdp_status_e zb_zdp_status_t

ZDP status values (2.4.5 ZDP Enumeration Description)

Device start

Startup procedure as defined in 2.5.5.5.6.2 Startup Procedure

4.5.3 Enumeration Type Documentation

4.5.3.1 enum zb_zdp_status_e

ZDP status values (2.4.5 ZDP Enumeration Description)

Device start

Startup procedure as defined in 2.5.5.5.6.2 Startup Procedure

Enumerator:

ZB_ZDP_STATUS_SUCCESS The requested operation or transmission was completed successfully

ZB_ZDP_STATUS_INV_REQUESTTYPE The supplied request type was invalid.

ZB_ZDP_STATUS_DEVICE_NOT_FOUND The requested device did not exist on a device following a child descriptor request to a parent.

ZB_ZDP_STATUS_INVALID_EP The supplied endpoint was equal to 0x00 or between 0xf1 and 0xff.

ZB_ZDP_STATUS_NOT_ACTIVE The requested endpoint is not described by a simple descriptor.

ZB_ZDP_STATUS_NOT_SUPPORTED The requested optional feature is not supported on the target device.

ZB_ZDP_STATUS_TIMEOUT A timeout has occurred with the requested operation.

ZB_ZDP_STATUS_NO_MATCH The end device bind request was unsuccessful due to a failure to match any suitable clusters.

ZB_ZDP_STATUS_NO_ENTRY The unbind request was unsuccessful due to the coordinator or source device not having an entry in its binding table to unbind.

ZB_ZDP_STATUS_NO_DESCRIPTOR A child descriptor was not available following a discovery request to a parent.

ZB_ZDP_STATUS_INSUFFICIENT_SPACE The device does not have storage space to support the requested operation.

ZB_ZDP_STATUS_NOT_PERMITTED The device is not in the proper state to support the requested operation.

ZB_ZDP_STATUS_TABLE_FULL The device does not have table space to support the operation.

ZB_ZDP_STATUS_NOT_AUTHORIZED The permissions configuration table on the target indicates that the request is not authorized from this device.

4.6 ZDO discovery services

Functions

- void **zb_zdo_nwk_addr_req** (**zb_uint8_t** param, **zb_callback_t** cb) ZB_SDCC_REENTRANT

  NWK_addr_req primitive.
- void **zb_zdo_ieee_addr_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  IEEE_addr_req primitive.
- void **zb_zdo_node_desc_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  Node_desc_req primitive.
- void **zb_zdo_power_desc_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  Power_desc_req primitive.
- void **zb_zdo_simple_desc_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  Simple_desc_req primitive.
- void **zb_zdo_active_ep_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  Active_desc_req primitive.
- void **zb_zdo_match_desc_req** (**zb_uint8_t** param, **zb_callback_t** cb) ZB_SDCC_REENTRANT

  Match_desc_req primitive.
- void **zb_zdo_system_server_discovery_req** (**zb_uint8_t** param, **zb_callback_t** cb) ZB_SDCC_REENTR-ANT

  Performs System_Server_Discovery_req.

Data Structures

- struct **zb_zdo_nwk_addr_req_s**

  NWK_addr_req command primitive.
- struct **zb_zdo_nwk_addr_req_param_s**

  Parameters for nwk_addr_req command.
- struct **zb_zdo_nwk_addr_resp_head_s**
- struct **zb_zdo_ieee_addr_req_s**

  Parameters of IEEE_addr_req primitive.
- struct **zb_zdo_node_desc_req_s**

  Parameters of Node_desc_req primitive.
- struct **zb_zdo_desc_resp_hdr_s**

  Header of Node_desc_resp primitive.
- struct **zb_zdo_node_desc_resp_s**

  Parameters of Node_desc_resp primitive.
- struct **zb_zdo_simple_desc_resp_hdr_s**

  Header of Node_desc_resp primitive.
- struct **zb_zdo_simple_desc_resp_s**

  Parameters of simple_desc_resp primitive.
- struct **zb_zdo_power_desc_resp_s**

  Parameters of Power_desc_resp primitive.
- struct **zb_zdo_power_desc_req_s**

  Parameters of Power_desc_req primitive.
- struct **zb_zdo_simple_desc_req_s**

  Parameters of Power_desc_req primitive.
- struct **zb_zdo_active_ep_req_s**

  Parameters of Active_desc_req primitive.
- struct **zb_zdo_ep_resp_s**

  Active EP response.

- struct **zb_zdo_match_desc_param_s**

  Parameters of match_desc_req primitive.
- struct **zb_zdo_match_desc_req_head_s**

  Match_desc_req head.
- struct **zb_zdo_match_desc_req_tail_s**

  Match_desc_req tail.
- struct **zb_zdo_match_desc_resp_s**

  2.4.4.1.7 Match_Desc_rsp response structure
- struct **zb_zdo_system_server_discovery_req_s**

  Request parameters for 2.4.3.1.13 System_Server_Discovery_req.
- struct **zb_zdo_system_server_discovery_resp_s**

  Response parameters for 2.4.4.1.10 System_Server_Discovery_rsp.

Macros

- #define **ZB_ZDO_SINGLE_DEVICE_RESP** 0

  2.4.3.1, 2.4.4.1
- #define **ZB_ZDO_EXTENDED_DEVICE_RESP** 1

Typedefs

- typedef struct
  zb_zdo_nwk_addr_req_s zb_zdo_nwk_addr_req_t

  NWK_addr_req command primitive.
- typedef struct
  zb_zdo_nwk_addr_req_param_s zb_zdo_nwk_addr_req_param_t

  Parameters for nwk_addr_req command.
- typedef struct
  zb_zdo_nwk_addr_resp_head_s zb_zdo_nwk_addr_resp_head_t
- typedef struct
  zb_zdo_ieee_addr_req_s zb_zdo_ieee_addr_req_t

  Parameters of IEEE_addr_req primitive.
- typedef struct
  zb_zdo_node_desc_req_s zb_zdo_node_desc_req_t

  Parameters of Node_desc_req primitive.
- typedef struct
  zb_zdo_desc_resp_hdr_s zb_zdo_desc_resp_hdr_t

  Header of Node_desc_resp primitive.
- typedef struct
  zb_zdo_node_desc_resp_s zb_zdo_node_desc_resp_t

  Parameters of Node_desc_resp primitive.
- typedef struct
  zb_zdo_simple_desc_resp_hdr_s zb_zdo_simple_desc_resp_hdr_t

  Header of Node_desc_resp primitive.
- typedef struct
  zb_zdo_simple_desc_resp_s zb_zdo_simple_desc_resp_t

  Parameters of simple_desc_resp primitive.
- typedef struct
  zb_zdo_power_desc_resp_s zb_zdo_power_desc_resp_t

  Parameters of Power_desc_resp primitive.
- typedef struct
  zb_zdo_power_desc_req_s zb_zdo_power_desc_req_t

Parameters of Power_desc_req primitive.

- typedef struct **zb_zdo_simple_desc_req_s zb_zdo_simple_desc_req_t**

  Parameters of Power_desc_req primitive.
- typedef struct **zb_zdo_active_ep_req_s zb_zdo_active_ep_req_t**

  Parameters of Active_desc_req primitive.
- typedef struct **zb_zdo_ep_resp_s zb_zdo_ep_resp_t**

  Active EP response.
- typedef struct **zb_zdo_match_desc_param_s zb_zdo_match_desc_param_t**

  Parameters of match_desc_req primitive.
- typedef struct **zb_zdo_match_desc_req_head_s zb_zdo_match_desc_req_head_t**

  Match_desc_req head.
- typedef struct **zb_zdo_match_desc_req_tail_s zb_zdo_match_desc_req_tail_t**

  Match_desc_req tail.
- typedef struct **zb_zdo_match_desc_resp_s zb_zdo_match_desc_resp_t**

  2.4.4.1.7 Match_Desc_rsp response structure
- typedef struct **zb_zdo_system_server_discovery_req_s zb_zdo_system_server_discovery_req_t**

  Request parameters for 2.4.3.1.13 System_Server_Discovery_req.
- typedef **zb_zdo_system_server_discovery_req_t zb_zdo_system_server_discovery_param_t**

  Parameters for 2.4.3.1.13 System_Server_Discovery_req call.
- typedef struct **zb_zdo_system_server_discovery_resp_s zb_zdo_system_server_discovery_resp_t**

  Response parameters for 2.4.4.1.10 System_Server_Discovery_rsp.

4.6.1 Detailed Description

4.6.2 Function Documentation

4.6.2.1 void zb_zdo_nwk_addr_req (zb_uint8_t *param*, zb_callback_t *cb*)

NWK_addr_req primitive.

Parameters

param	- index of buffer with primitive parameters -

See Also

zb_zdo_nwk_addr_req_param_t (p. 25)

Parameters

cb	- user's function to call when got response from the remote.

See Also

zb_zdo_nwk_addr_resp_head_t passed to cb as parameter.

Example:

```
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_zdo_nwk_addr_req_param_t *req_param = ZB_GET_BUF_PARAM(buf,
      zb_zdo_nwk_addr_req_param_t);

  req_param->dst_addr = 0; // send req to ZC
  zb_address_ieee_by_ref(req_param->ieee_addr, short_addr);
  req_param->request_type = ZB_ZDO_SINGLE_DEVICE_RESP;
  req_param->start_index = 0;
  zb_zdo_nwk_addr_req(param, zb_get_peer_short_addr_cb);
}

void zb_get_peer_short_addr_cb(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_zdo_nwk_addr_resp_head_t *resp;
  zb_ieee_addr_t ieee_addr;
  zb_uint16_t nwk_addr;
  zb_address_ieee_ref_t addr_ref;

  TRACE_MSG(TRACE_ZDO2, "zb_get_peer_short_addr_cb param %hd", (FMT__H, param))
      ;

  resp = (zb_zdo_nwk_addr_resp_head_t*)ZB_BUF_BEGIN(buf);
  TRACE_MSG(TRACE_ZDO2, "resp status %hd, nwk addr %d", (FMT__H_D, resp->status
      , resp->nwk_addr));
  ZB_DUMP_IEEE_ADDR(resp->ieee_addr);
  if (resp->status == ZB_ZDP_STATUS_SUCCESS)
  {
    ZB_LETOH64(ieee_addr, resp->ieee_addr);
    ZB_LETOH16(&nwk_addr, &resp->nwk_addr);
    zb_address_update(ieee_addr, nwk_addr, ZB_TRUE, &addr_ref);
  }
  zb_free_buf(buf);
}
```

4.6.2.2 void zb_zdo_ieee_addr_req (zb_uint8_t *param*, zb_callback_t *cb*)

IEEE_addr_req primitive.

Parameters

param	- index of buffer with primitive parameters

See Also

zb_zdo_ieee_addr_req_t (p. 25). Parameters mut be put into buffer as data (allocated).

Parameters

cb	- user's function to call when got response from the remote.

Example:

```
{
  zb_zdo_ieee_addr_req_t *req = NULL;

  ZB_BUF_INITIAL_ALLOC(buf, sizeof(zb_zdo_ieee_addr_req_t), req);

  req->nwk_addr = ind->src_addr;
  req->request_type = ZB_ZDO_SINGLE_DEV_RESPONSE;
  req->start_index = 0;
  zb_zdo_ieee_addr_req(ZB_REF_FROM_BUF(buf), ieee_addr_callback);
}

void ieee_addr_callback(zb_uint8_t param) ZB_CALLBACK
```

```
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_zdo_nwk_addr_resp_head_t *resp;
  zb_ieee_addr_t ieee_addr;
  zb_uint16_t nwk_addr;
  zb_address_ieee_ref_t addr_ref;

  TRACE_MSG(TRACE_ZDO2, "zb_get_peer_short_addr_cb param %hd", (FMT__H, param))
      ;

  resp = (zb_zdo_nwk_addr_resp_head_t*)ZB_BUF_BEGIN(buf);
  TRACE_MSG(TRACE_ZDO2, "resp status %hd, nwk addr %d", (FMT__H_D, resp->status
      , resp->nwk_addr));
  ZB_DUMP_IEEE_ADDR(resp->ieee_addr);
  if (resp->status == ZB_ZDP_STATUS_SUCCESS)
  {
    ZB_LETOH64(ieee_addr, resp->ieee_addr);
    ZB_LETOH16(&nwk_addr, &resp->nwk_addr);
    zb_address_update(ieee_addr, nwk_addr, ZB_TRUE, &addr_ref);
  }
  zb_free_buf(buf);
}
```

4.6.2.3 void zb_zdo_node_desc_req (**zb_uint8_t** *param,* **zb_callback_t** *cb*)

Node_desc_req primitive.

Parameters

param	- index of buffer with primitive parameters

See Also

zb_zdo_node_desc_req_t (p. 26). Parameters must be put into buffer as data (allocated).

Parameters

cb	- user's function to call when got response from the remote.

Example:

```
{
  ZB_BUF_INITIAL_ALLOC(asdu, sizeof(zb_zdo_node_desc_req_t), req);
  req->nwk_addr = 0; //send to coordinator
  zb_zdo_node_desc_req(ZB_REF_FROM_BUF(asdu), node_desc_callback);
}


void node_desc_callback(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_uint8_t *zdp_cmd = ZB_BUF_BEGIN(buf);
  zb_zdo_node_desc_resp_t *resp = (zb_zdo_node_desc_resp_t*)(zdp_cmd);
  zb_zdo_power_desc_req_t *req;

  TRACE_MSG(TRACE_APS1, "node_desc_callback: status %hd, addr 0x%x",
            (FMT__H_D, resp->hdr.status, resp->hdr.nwk_addr));
  if (resp->hdr.status != ZB_ZDP_STATUS_SUCCESS || resp->hdr.nwk_addr != 0x0)
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect status/addr", (FMT__0));
    g_error++;
  }

  TRACE_MSG(TRACE_APS1, "logic type %hd, aps flag %hd, frequency %hd",
            (FMT__H_H_H, ZB_GET_NODE_DESC_LOGICAL_TYPE(&resp->node_desc),
      ZB_GET_NODE_DESC_APS_FLAGS(&resp->node_desc),
             ZB_GET_NODE_DESC_FREQ_BAND(&resp->node_desc)));
  if (ZB_GET_NODE_DESC_LOGICAL_TYPE(&resp->node_desc) != 0 ||
      ZB_GET_NODE_DESC_APS_FLAGS(&resp->node_desc) != 0 ||
      ZB_GET_NODE_DESC_FREQ_BAND(&resp->node_desc) != ZB_FREQ_BAND_2400 )
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect type/flags/freq", (FMT__0));
    g_error++;
  }
  TRACE_MSG(TRACE_APS1, "mac cap 0x%hx, manufact code %hd, max buf %hd, max
       transfer %hd",
            (FMT__H_H_H_H, resp->node_desc.mac_capability_flags, resp->node_desc
      .manufacturer_code,
             resp->node_desc.max_buf_size, resp->node_desc.
      max_incoming_transfer_size));
  if ((resp->node_desc.mac_capability_flags & 0xB) != 0xB || (resp->node_desc.
      mac_capability_flags & ~0x4f) != 0 ||
      resp->node_desc.manufacturer_code != 0 ||
      resp->node_desc.max_incoming_transfer_size != 0)
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect cap/manuf code/max transfer", (
      FMT__0));
    g_error++;
  }

  zb_free_buf(buf);
}
```

4.6.2.4 void zb_zdo_power_desc_req (zb_uint8_t *param*, zb_callback_t *cb*)

Power_desc_req primitive.

Parameters

param	- index of buffer with primitive parameters

See Also

zb_zdo_power_desc_req_t (p. 26). Parameters must be put into buffer as data (allocated).

Parameters

cb	- user's function to call when got response from the remote.

Example:

```
{
  ZB_BUF_INITIAL_ALLOC(buf, sizeof(zb_zdo_power_desc_req_t), req);
  req->nwk_addr = 0; //send to coordinator
  zb_zdo_power_desc_req(ZB_REF_FROM_BUF(buf), node_power_desc_callback);
}


void node_power_desc_callback(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_uint8_t *zdp_cmd = ZB_BUF_BEGIN(buf);
  zb_zdo_power_desc_resp_t *resp = (zb_zdo_power_desc_resp_t*)(zdp_cmd);
  zb_zdo_simple_desc_req_t *req;

  TRACE_MSG(TRACE_APS1, " node_power_desc_callback status %hd, addr 0x%x",
            (FMT__H, resp->hdr.status, resp->hdr.nwk_addr));
  if (resp->hdr.status != ZB_ZDP_STATUS_SUCCESS || resp->hdr.nwk_addr != 0x0)
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect status/addr", (FMT__0));
    g_error++;
  }

  TRACE_MSG(TRACE_APS1, "power mode %hd, avail power src %hd, cur power src
       %hd, cur power level %hd",
            (FMT__H_H_H_H, ZB_GET_POWER_DESC_CUR_POWER_MODE(&resp->power_desc),
             ZB_GET_POWER_DESC_AVAIL_POWER_SOURCES(&resp->power_desc),
             ZB_GET_POWER_DESC_CUR_POWER_SOURCE(&resp->power_desc),
             ZB_GET_POWER_DESC_CUR_POWER_SOURCE_LEVEL(&resp->power_desc)));
  // PowerDescriptor=Current power mode=0b0000, Available power mode=0b0111,
       Current
  // power source=0b0001, Current power source level=0b110001
  if (ZB_GET_POWER_DESC_CUR_POWER_MODE(&resp->power_desc) != 0 ||
      ZB_GET_POWER_DESC_AVAIL_POWER_SOURCES(&resp->power_desc) != 0x7 ||
      ZB_GET_POWER_DESC_CUR_POWER_SOURCE(&resp->power_desc) != 0x1 ||
      ZB_GET_POWER_DESC_CUR_POWER_SOURCE_LEVEL(&resp->power_desc) != 0xC)
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect power desc", (FMT__0));
    g_error++;
  }
  zb_free_buf(buf);
}
```

4.6.2.5 void zb_zdo_simple_desc_req (zb_uint8_t *param*, zb_callback_t *cb*)

Simple_desc_req primitive.

Parameters

param	- index of buffer with primitive parameters

See Also

zb_zdo_simple_desc_req_t (p. 26).

Parameters

cb	- user's function to call when got response from the remote.

Example:

```
{
  zb_zdo_simple_desc_req_t *req;

  ZB_BUF_INITIAL_ALLOC(buf, sizeof(zb_zdo_simple_desc_req_t), req);
  req->nwk_addr = 0; //send to coordinator
  req->endpoint = 1;
  zb_zdo_simple_desc_req(ZB_REF_FROM_BUF(buf), simple_desc_callback);
}


void simple_desc_callback(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_uint8_t *zdp_cmd = ZB_BUF_BEGIN(buf);
  zb_zdo_simple_desc_resp_t *resp = (zb_zdo_simple_desc_resp_t*)(zdp_cmd);
  zb_uint_t i;
  zb_zdo_active_ep_req_t *req;

  TRACE_MSG(TRACE_APS1, "simple_desc_callback status %hd, addr 0x%x",
            (FMT__H, resp->hdr.status, resp->hdr.nwk_addr));
  if (resp->hdr.status != ZB_ZDP_STATUS_SUCCESS || resp->hdr.nwk_addr != 0x0)
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect status/addr", (FMT__0));
    g_error++;
  }

//simple descriptor for test SimpleDescriptor=
//Endpoint=0x01, Application profile identifier=0x0103, Application device
//identifier=0x0000, Application device version=0b0000, Application
//flags=0b0000, Application input cluster count=0x0A, Application input
//cluster list=0x00 0x03 0x04 0x38 0x54 0x70 0x8c 0xc4 0xe0 0xff,
//Application output cluster count=0x0A, Application output cluster
//list=0x00 0x01 0x02 0x1c 0x38 0x70 0x8c 0xa8 0xc4 0xff

  TRACE_MSG(TRACE_APS1, "ep %hd, app prof %d, dev id %d, dev ver %hd, input
       count 0x%hx, output count 0x%hx",
            (FMT__H_D_D_H_H_H, resp->simple_desc.endpoint, resp->simple_desc.
      app_profile_id,
            resp->simple_desc.app_device_id, resp->simple_desc.
      app_device_version,
           resp->simple_desc.app_input_cluster_count, resp->simple_desc.
      app_output_cluster_count));

  TRACE_MSG(TRACE_APS1, "clusters:", (FMT__0));
  for(i = 0; i < resp->simple_desc.app_input_cluster_count + resp->simple_desc.
      app_output_cluster_count; i++)
  {
    TRACE_MSG(TRACE_APS1, " 0x%hx", (FMT__H, *(resp->simple_desc.
      app_cluster_list + i)));
  }
  zb_free_buf(buf);
}
```

4.6.2.6 void zb_zdo_active_ep_req (zb_uint8_t *param,* zb_callback_t *cb*)

Active_desc_req primitive.

Parameters

param	- index of buffer with primitive parameters

See Also

zb_zdo_active_ep_req_t (p. 26). Parameters must be put into buffer as data (allocated).

Parameters

cb	- user's function to call when got response from the remote.

Example:

```
{
  zb_zdo_active_ep_req_t *req;

  ZB_BUF_INITIAL_ALLOC(buf, sizeof(zb_zdo_active_ep_req_t), req);
  req->nwk_addr = 0; //coord addr
  zb_zdo_active_ep_req(ZB_REF_FROM_BUF(buf), active_ep_callback);

void active_ep_callback(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_uint8_t *zdp_cmd = ZB_BUF_BEGIN(buf);
  zb_zdo_ep_resp_t *resp = (zb_zdo_ep_resp_t*)zdp_cmd;
  zb_uint8_t *ep_list = zdp_cmd + sizeof(zb_zdo_ep_resp_t);

  TRACE_MSG(TRACE_APS1, "active_ep_callback status %hd, addr 0x%x",
            (FMT__H, resp->status, resp->nwk_addr));

  if (resp->status != ZB_ZDP_STATUS_SUCCESS || resp->nwk_addr != 0x0)
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect status/addr", (FMT__0));
    g_error++;
  }

  TRACE_MSG(TRACE_APS1, " ep count %hd, ep %hd", (FMT__H_H, resp->ep_count, *
      ep_list));
  if (resp->ep_count != 1 || *ep_list != 1)
  {
    TRACE_MSG(TRACE_APS3, "Error incorrect ep count or ep value", (FMT__0));
    g_error++;
  }

  zb_free_buf(buf);
}
```

4.6.2.7 void zb_zdo_match_desc_req (zb_uint8_t *param,* zb_callback_t *cb*)

Match_desc_req primitive.

Parameters

param	- index of buffer with primitive parameters

See Also

zb_zdo_match_desc_param_t (p. 26).

Parameters

cb	- user's function to call when got response from the remote.

Example:

```
{
  zb_zdo_match_desc_param_t *req;

  ZB_BUF_INITIAL_ALLOC(buf, sizeof(zb_zdo_match_desc_param_t) + (2 + 3) *
      sizeof(zb_uint16_t), req);

  req->nwk_addr = 0; //send to coordinator
  req->profile_id = 0x103;
  req->num_in_clusters = 2;
  req->num_out_clusters = 3;
  req->cluster_list[0] = 0x54;
  req->cluster_list[1] = 0xe0;

  req->cluster_list[2] = 0x1c;
  req->cluster_list[3] = 0x38;
  req->cluster_list[4] = 0xa8;

  zb_zdo_match_desc_req(param, match_desc_callback);
}

void match_desc_callback(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_uint8_t *zdp_cmd = ZB_BUF_BEGIN(buf);
  zb_zdo_match_desc_resp_t *resp = (zb_zdo_match_desc_resp_t*)zdp_cmd;
  zb_uint8_t *match_list = (zb_uint8_t*)(resp + 1);

  TRACE_MSG(TRACE_APS1, "match_desc_callback status %hd, addr 0x%x",
            (FMT__H, resp->status, resp->nwk_addr));
  if (resp->status != ZB_ZDP_STATUS_SUCCESS || resp->nwk_addr != 0x0)
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect status/addr", (FMT__0));
    g_error++;
  }
  //asdu=Match_Descr_rsp(Status=0x00=Success, NWKAddrOfInterest=0x0000,
  //MatchLength=0x01, MatchList=0x01)
  TRACE_MSG(TRACE_APS1, "match_len %hd, list %hd ", (FMT__H_H, resp->match_len,
       *match_list));
  if (resp->match_len != 1 || *match_list != 1)
  {
    TRACE_MSG(TRACE_APS1, "Error incorrect match result", (FMT__0));
    g_error++;
  }
  zb_free_buf(buf);
}
```

4.6.2.8 void zb_zdo_system_server_discovery_req (zb_uint8_t *param*, zb_callback_t *cb*)

Performs System_Server_Discovery_req.

Parameters

param	- index of buffer with request parameters

See Also

zb_zdo_system_server_discovery_param_t (p. 27)

Parameters

cb	- user's function to call when got response from the remote.

See Also

zb_zdo_system_server_discovery_resp_t (p. 27)

Example:

```
{
  zb_zdo_system_server_discovery_param_t *req_param;

  req_param = ZB_GET_BUF_PARAM(asdu, zb_zdo_system_server_discovery_param_t);
  req_param->server_mask = ZB_NETWORK_MANAGER;

  zb_zdo_system_server_discovery_req(ZB_REF_FROM_BUF(asdu), get_nwk_manager_cb)
      ;
}

void get_nwk_manager_cb(zb_uint8_t param)
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_uint8_t *zdp_cmd = ZB_BUF_BEGIN(buf);
  zb_zdo_system_server_discovery_resp_t *resp = (
      zb_zdo_system_server_discovery_resp_t*)(zdp_cmd);

  if (resp->status == ZB_ZDP_STATUS_SUCCESS && resp->server_mask &
      ZB_NETWORK_MANAGER )
  {
    TRACE_MSG(TRACE_APS3, "system_server_discovery received, status: OK", (
      FMT__0));
  }
  else
  {
    TRACE_MSG(TRACE_ERROR, "ERROR receiving system_server_discovery status %x,
       mask %x",
              (FMT__D_D, resp->status, resp->server_mask));
  }
  zb_free_buf(buf);
}
```

4.6.3 Macro Definition Documentation

4.6.3.1 #define ZB_ZDO_SINGLE_DEVICE_RESP 0

2.4.3.1, 2.4.4.1

Single device response

4.6.3.2 #define ZB_ZDO_EXTENDED_DEVICE_RESP 1

Extended response

4.6.4 Typedef Documentation

4.6.4.1 typedef struct zb_zdo_nwk_addr_req_s zb_zdo_nwk_addr_req_t

NWK_addr_req command primitive.

4.6.4.2 typedef struct zb_zdo_nwk_addr_req_param_s zb_zdo_nwk_addr_req_param_t

Parameters for nwk_addr_req command.

4.6.4.3 typedef struct zb_zdo_ieee_addr_req_s zb_zdo_ieee_addr_req_t

Parameters of IEEE_addr_req primitive.

To be put into buffer as data (means - after space alloc).

4.6.4.4 typedef struct zb_zdo_node_desc_req_s zb_zdo_node_desc_req_t

Parameters of Node_desc_req primitive.

To be put into buffer as data (means - after space alloc).

4.6.4.5 typedef struct zb_zdo_desc_resp_hdr_s zb_zdo_desc_resp_hdr_t

Header of Node_desc_resp primitive.

4.6.4.6 typedef struct zb_zdo_node_desc_resp_s zb_zdo_node_desc_resp_t

Parameters of Node_desc_resp primitive.

4.6.4.7 typedef struct zb_zdo_simple_desc_resp_hdr_s zb_zdo_simple_desc_resp_hdr_t

Header of Node_desc_resp primitive.

4.6.4.8 typedef struct zb_zdo_simple_desc_resp_s zb_zdo_simple_desc_resp_t

Parameters of simple_desc_resp primitive.

4.6.4.9 typedef struct zb_zdo_power_desc_resp_s zb_zdo_power_desc_resp_t

Parameters of Power_desc_resp primitive.

4.6.4.10 typedef struct zb_zdo_power_desc_req_s zb_zdo_power_desc_req_t

Parameters of Power_desc_req primitive.

To be put into buffer as data (means - after space alloc).

4.6.4.11 typedef struct zb_zdo_simple_desc_req_s zb_zdo_simple_desc_req_t

Parameters of Power_desc_req primitive.

To be put into buffer as data (means - after space alloc).

4.6.4.12 typedef struct zb_zdo_active_ep_req_s zb_zdo_active_ep_req_t

Parameters of Active_desc_req primitive.

To be put into buffer as data (means - after space alloc).

4.6.4.13 typedef struct zb_zdo_ep_resp_s zb_zdo_ep_resp_t

Active EP response.

4.6.4.14 typedef struct zb_zdo_match_desc_param_s zb_zdo_match_desc_param_t

Parameters of match_desc_req primitive.

To be put into buffer as data (means - after space alloc).

4.6.4.15 typedef struct zb_zdo_match_desc_req_head_s zb_zdo_match_desc_req_head_t

Match_desc_req head.

4.6.4.16 typedef struct zb_zdo_match_desc_req_tail_s zb_zdo_match_desc_req_tail_t

Match_desc_req tail.

4.6.4.17 typedef struct zb_zdo_match_desc_resp_s zb_zdo_match_desc_resp_t

2.4.4.1.7 Match_Desc_rsp response structure

4.6.4.18 typedef struct zb_zdo_system_server_discovery_req_s zb_zdo_system_server_discovery_req_t

Request parameters for 2.4.3.1.13 System_Server_Discovery_req.

4.6.4.19 typedef zb_zdo_system_server_discovery_req_t zb_zdo_system_server_discovery_param_t

Parameters for 2.4.3.1.13 System_Server_Discovery_req call.

4.6.4.20 typedef struct zb_zdo_system_server_discovery_resp_s zb_zdo_system_server_discovery_resp_t

Response parameters for 2.4.4.1.10 System_Server_Discovery_rsp.

4.7 ZDO management services

Functions

- void **zb_zdo_mgmt_nwk_update_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  Performs Mgmt_NWK_Update_req request.
- void **zb_zdo_mgmt_lqi_req** (**zb_uint8_t** param, **zb_callback_t** cb) ZB_SDCC_REENTRANT

  Sends 2.4.3.3.2 Mgmt_Lqi_req.
- void **zb_zdo_bind_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  Bind_req request.
- void **zb_zdo_unbind_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  Unbind_req request.
- void **zdo_mgmt_leave_req** (**zb_uint8_t** param, **zb_callback_t** cb) ZB_SDCC_REENTRANT

  Sends 2.4.3.3.2 Mgmt_Leave_req.
- void **zb_zdo_add_group_req** (**zb_uint8_t** param, **zb_callback_t** cb)

  ZDO interface for ADD-GROUP.request.

Data Structures

- struct **zb_zdo_mgmt_nwk_update_req_hdr_s**

  Header of parameters for Mgmt_NWK_Update_req.
- struct **zb_zdo_mgmt_nwk_update_req_s**

  Parameters for Mgmt_NWK_Update_req.
- struct **zb_zdo_mgmt_nwk_update_notify_hdr_s**

  Header parameters for mgmt_nwk_update_notify.
- struct **zb_zdo_mgmt_nwk_update_notify_param_s**

  Parameters for mgmt_nwk_update_notify.
- struct **zb_zdo_mgmt_lqi_param_s**

  Parameters for 2.4.3.3.2 Mgmt_Lqi_req.
- struct **zb_zdo_mgmt_lqi_req_s**

  Request for 2.4.3.3.2 Mgmt_Lqi_req.
- struct **zb_zdo_mgmt_lqi_resp_s**

  Response for 2.4.4.3.2 Mgmt_Lqi_rsp.
- struct **zb_zdo_neighbor_table_record_s**

  NeighborTableList Record Format for mgmt_lqi_resp.
- struct **zb_zdo_bind_req_param_s**

  Parameters for 2.4.3.2.2 Bind_req API call.
- struct **zb_zdo_bind_req_head_s**

  2.4.3.2.2 Bind_req request head send to the remote
- struct **zb_zdo_bind_req_tail_1_s**

  2.4.3.2.2 Bind_req request tail 1st variant send to the remote
- struct **zb_zdo_bind_req_tail_2_s**

  2.4.3.2.2 Bind_req request tail 2nd variant send to the remote
- struct **zb_zdo_bind_resp_s**
- struct **zb_zdo_mgmt_leave_param_s**

  Request for 2.4.3.3.5 Mgmt_Leave_req.
- struct **zb_zdo_mgmt_leave_req_s**

  Request for 2.4.3.3.5 Mgmt_Leave_req.
- struct **zb_zdo_mgmt_leave_res_s**

  Response for 2.4.4.3.5 Mgmt_Leave_rsp.
- struct **zb_zdo_end_device_bind_req_head_s**

2.4.3.2.1 End_Device_Bind_req command head

- struct **zb_zdo_end_device_bind_req_tail_s**

  2.4.3.2.1 End_Device_Bind_req command head
- struct **zb_end_device_bind_req_param_s**

  Parameters for 2.4.3.2.1 End_Device_Bind_req.
- struct **zb_zdo_end_device_bind_resp_s**
- struct **zb_zdo_mgmt_permit_joining_req_s**

  Parameters for 2.4.3.3.7 Mgmt_Permit_Joining_req.
- struct **zb_zdo_mgmt_permit_joining_req_param_s**

  Parameters for zb_zdo_mgmt_permit_joining_req.

Macros

- #define **ZB_ZDO_RECORD_SET_DEVICE_TYPE**(var, type) (var &= ∼3, var |= type)
- #define **ZB_ZDO_RECORD_GET_DEVICE_TYPE**(var) (var & 3)
- #define **ZB_ZDO_RECORD_SET_RX_ON_WHEN_IDLE**(var, type) (var &= ∼0xC, var |= (type << 2))
- #define **ZB_ZDO_RECORD_GET_RX_ON_WHEN_IDLE**(var) ((var & 0xC) >> 2)
- #define **ZB_ZDO_RECORD_SET_RELATIONSHIP**(var, type) (var &= ∼0x70, var |= (type << 4))
- #define **ZB_ZDO_RECORD_GET_RELATIONSHIP**(var) ((var & 0x70) >> 4)

Typedefs

- typedef struct
  zb_zdo_mgmt_nwk_update_req_hdr_s zb_zdo_mgmt_nwk_update_req_hdr_t

  Header of parameters for Mgmt_NWK_Update_req.
- typedef struct
  zb_zdo_mgmt_nwk_update_req_s zb_zdo_mgmt_nwk_update_req_t

  Parameters for Mgmt_NWK_Update_req.
- typedef struct
  zb_zdo_mgmt_nwk_update_notify_hdr_s zb_zdo_mgmt_nwk_update_notify_hdr_t

  Header parameters for mgmt_nwk_update_notify.
- typedef struct
  zb_zdo_mgmt_nwk_update_notify_param_s zb_zdo_mgmt_nwk_update_notify_param_t

  Parameters for mgmt_nwk_update_notify.
- typedef struct
  zb_zdo_mgmt_lqi_param_s zb_zdo_mgmt_lqi_param_t

  Parameters for 2.4.3.3.2 Mgmt_Lqi_req.
- typedef struct
  zb_zdo_mgmt_lqi_req_s zb_zdo_mgmt_lqi_req_t

  Request for 2.4.3.3.2 Mgmt_Lqi_req.
- typedef struct
  zb_zdo_mgmt_lqi_resp_s zb_zdo_mgmt_lqi_resp_t

  Response for 2.4.4.3.2 Mgmt_Lqi_rsp.
- typedef struct
  zb_zdo_neighbor_table_record_s zb_zdo_neighbor_table_record_t

  NeighborTableList Record Format for mgmt_lqi_resp.
- typedef struct
  zb_zdo_bind_req_param_s zb_zdo_bind_req_param_t

  Parameters for 2.4.3.2.2 Bind_req API call.
- typedef struct
  zb_zdo_bind_req_head_s zb_zdo_bind_req_head_t

  2.4.3.2.2 Bind_req request head send to the remote

- typedef struct **zb_zdo_bind_req_tail_1_s zb_zdo_bind_req_tail_1_t**

  2.4.3.2.2 Bind_req request tail 1st variant send to the remote
- typedef struct **zb_zdo_bind_req_tail_2_s zb_zdo_bind_req_tail_2_t**

  2.4.3.2.2 Bind_req request tail 2nd variant send to the remote
- typedef struct **zb_zdo_bind_resp_s zb_zdo_bind_resp_t**
- typedef struct **zb_zdo_mgmt_leave_param_s zb_zdo_mgmt_leave_param_t**

  Request for 2.4.3.3.5 Mgmt_Leave_req.
- typedef struct **zb_zdo_mgmt_leave_req_s zb_zdo_mgmt_leave_req_t**

  Request for 2.4.3.3.5 Mgmt_Leave_req.
- typedef struct **zb_zdo_mgmt_leave_res_s zb_zdo_mgmt_leave_res_t**

  Response for 2.4.4.3.5 Mgmt_Leave_rsp.
- typedef struct **zb_zdo_end_device_bind_req_head_s zb_zdo_end_device_bind_req_head_t**

  2.4.3.2.1 End_Device_Bind_req command head
- typedef struct **zb_zdo_end_device_bind_req_tail_s zb_zdo_end_device_bind_req_tail_t**

  2.4.3.2.1 End_Device_Bind_req command head
- typedef struct **zb_end_device_bind_req_param_s zb_end_device_bind_req_param_t**

  Parameters for 2.4.3.2.1 End_Device_Bind_req.
- typedef struct **zb_zdo_end_device_bind_resp_s zb_zdo_end_device_bind_resp_t**
- typedef struct **zb_zdo_mgmt_permit_joining_req_s zb_zdo_mgmt_permit_joining_req_t**

  Parameters for 2.4.3.3.7 Mgmt_Permit_Joining_req.
- typedef struct **zb_zdo_mgmt_permit_joining_req_param_s zb_zdo_mgmt_permit_joining_req_param_t**

  Parameters for zb_zdo_mgmt_permit_joining_req.

4.7.1 Detailed Description

4.7.2 Function Documentation

4.7.2.1 void zb_zdo_mgmt_nwk_update_req (**zb_uint8_t** *param*, **zb_callback_t** *cb*)

Performs Mgmt_NWK_Update_req request.

Parameters

param	- index of buffer with call parameters. Parameters mut be put into buffer as parameters.

See Also

zb_zdo_mgmt_nwk_update_req_t (p. 35)

Parameters

cb	- user's function to call when got response from the remote.

See Also

zb_zdo_mgmt_nwk_update_notify_hdr_t (p. 35)

Example:

```
{
  zb_zdo_mgmt_nwk_update_req_t *req;

  req = ZB_GET_BUF_PARAM(buf, zb_zdo_mgmt_nwk_update_req_t);

  req->hdr.scan_channels = ZB_MAC_ALL_CHANNELS_MASK;
  req->hdr.scan_duration = TEST_SCAN_DURATION;
  req->scan_count = TEST_SCAN_COUNT;
  req->update_id = ZB_NIB_UPDATE_ID();

  req->dst_addr = 0;

  zb_zdo_mgmt_nwk_update_req(param, mgmt_nwk_update_ok_cb);
}


void mgmt_nwk_update_ok_cb(zb_uint8_t param)
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_uint8_t *zdp_cmd = ZB_BUF_BEGIN(buf);
  zb_zdo_mgmt_nwk_update_notify_hdr_t *notify_resp = (
      zb_zdo_mgmt_nwk_update_notify_hdr_t *)zdp_cmd;

  TRACE_MSG(TRACE_APS3,
            "notify_resp status %hd, scanned_channels %x %x,
       total_transmissions %hd, "
            "transmission_failures %hd, scanned_channels_list_count %hd, buf
       len %hd",
            (FMT__H_D_D_H_H_H_H, notify_resp->status, (zb_uint16_t)notify_resp
      ->scanned_channels,
             *((zb_uint16_t*)&notify_resp->scanned_channels + 1),
             notify_resp->total_transmissions, notify_resp->
      transmission_failures,
             notify_resp->scanned_channels_list_count, ZB_BUF_LEN(buf)));

  if (notify_resp->status == ZB_ZDP_STATUS_SUCCESS)
  {
    TRACE_MSG(TRACE_APS3, "mgmt_nwk_update_notify received, Ok", (FMT__0));
  }
  else
  {
    TRACE_MSG(TRACE_ERROR, "mgmt_nwk_update_notify received, ERROR incorrect
       status %x",
              (FMT__D, notify_resp->status));
  }

  zb_free_buf(buf);
}
```

4.7.2.2 void zb_zdo_mgmt_lqi_req (zb_uint8_t *param*, zb_callback_t *cb*)

Sends 2.4.3.3.2 Mgmt_Lqi_req.

Parameters

param	- index of buffer with Lqi request parameters.

See Also

zb_zdo_mgmt_lqi_param_t (p. 35)

Parameters

cb	- user's function to call when got response from the remote.

See Also

zb_zdo_mgmt_lqi_resp_t (p. 36)
zb_zdo_neighbor_table_record_t (p. 36)

Example:

```
{
  zb_zdo_mgmt_lqi_param_t *req_param;

  req_param = ZB_GET_BUF_PARAM(buf, zb_zdo_mgmt_lqi_param_t);
  req_param->start_index = 0;
  req_param->dst_addr = 0; //coord short addr

  zb_zdo_mgmt_lqi_req(ZB_REF_FROM_BUF(buf), get_lqi_cb);
}


void get_lqi_cb(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_uint8_t *zdp_cmd = ZB_BUF_BEGIN(buf);
  zb_zdo_mgmt_lqi_resp_t *resp = (zb_zdo_mgmt_lqi_resp_t*)(zdp_cmd);
  zb_zdo_neighbor_table_record_t *record = (zb_zdo_neighbor_table_record_t*)(
      resp + 1);
  zb_uint_t i;

  TRACE_MSG(TRACE_APS1, "get_lqi_cb status %hd, neighbor_table_entries %hd,
       start_index %hd, neighbor_table_list_count %d",
            (FMT__H_H_H_H, resp->status, resp->neighbor_table_entries, resp->
      start_index, resp->neighbor_table_list_count));

  for (i = 0; i < resp->neighbor_table_list_count; i++)
  {
    TRACE_MSG(TRACE_APS1, "#%hd: long addr " TRACE_FORMAT_64 " pan id "
      TRACE_FORMAT_64,
              (FMT__H_A_A, i, TRACE_ARG_64(record->ext_addr), TRACE_ARG_64(
      record->ext_pan_id)));

    TRACE_MSG(TRACE_APS1,
      "#%hd: network_addr %d, dev_type %hd, rx_on_wen_idle %hd, relationship
       %hd, permit_join %hd, depth %hd, lqi %hd",
      (FMT_H_D_H_H_H_H_H_H, i, record->network_addr,
       ZB_ZDO_RECORD_GET_DEVICE_TYPE(record->type_flags),
       ZB_ZDO_RECORD_GET_RX_ON_WHEN_IDLE(record->type_flags),
       ZB_ZDO_RECORD_GET_RELATIONSHIP(record->type_flags),
       record->permit_join, record->depth, record->lqi));

    record++;
  }
}
```

4.7.2.3 void zb_zdo_bind_req (zb_uint8_t *param*, zb_callback_t *cb*)

Bind_req request.

Parameters

param	- index of buffer with request.

See Also

zb_apsme_binding_req_t (p. 42)

Parameters

cb	- user's function to call when got response from the remote.

See Also

zb_zdo_bind_resp_t

Example:

```
{
  zb_apsme_binding_req_t *req;

  req = ZB_GET_BUF_PARAM(buf, zb_apsme_binding_req_t);
  ZB_MEMCPY(&req->src_addr, &g_ieee_addr, sizeof(zb_ieee_addr_t));
  req->src_endpoint = i;
  req->clusterid = 1;
  req->addr_mode = ZB_APS_ADDR_MODE_64_ENDP_PRESENT;
  ZB_MEMCPY(&req->dst_addr.addr_long, &g_ieee_addr_d, sizeof(zb_ieee_addr_t));
  req->dst_endpoint = 240;

  zb_zdo_bind_req(ZB_REF_FROM_BUF(buf), zb_bind_callback);
  ret = buf->u.hdr.status;
  if (ret == RET_TABLE_FULL)
  {
    TRACE_MSG(TRACE_ERROR, "TABLE FULL %d", (FMT__D, ret));
  }
}

void zb_bind_callback(zb_uint8_t param)
{
  zb_ret_t ret = RET_OK;
  zb_buf_t *buf = (zb_buf_t *)ZB_BUF_FROM_REF(param);
  zb_uint8_t *aps_body = NULL;

  aps_body = ZB_BUF_BEGIN(buf);
  ZB_MEMCPY(&ret, aps_body, sizeof(ret));

  TRACE_MSG(TRACE_INFO1, "zb_bind_callback %hd", (FMT__H, ret));
  if (ret == RET_OK)
  {
    // bind ok
  }
}
```

4.7.2.4 void zb_zdo_unbind_req (zb_uint8_t *param*, zb_callback_t *cb*)

Unbind_req request.

Parameters

param	- index of buffer with request.

See Also

zb_zdo_bind_req_param_t (p. 36)

Parameters

cb	- user's function to call when got response from the remote.

See Also

zb_zdo_bind_resp_t

Example:

```
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_zdo_bind_req_param_t *bind_param;

  TRACE_MSG(TRACE_COMMON1, "unbind_device_1", (FMT__0));

  zb_buf_initial_alloc(buf, 0);
  bind_param = ZB_GET_BUF_PARAM(buf, zb_zdo_bind_req_param_t);
  ZB_MEMCPY(bind_param->src_address, g_ieee_addr_ed1, sizeof(zb_ieee_addr_t));
  bind_param->src_endp = TEST_ED1_EP;
  bind_param->cluster_id = TP_BUFFER_TEST_REQUEST_CLID;
  bind_param->dst_addr_mode = ZB_APS_ADDR_MODE_64_ENDP_PRESENT;
  ZB_MEMCPY(bind_param->dst_address.addr_long, g_ieee_addr_ed2, sizeof(
      zb_ieee_addr_t));
  bind_param->dst_endp = TEST_ED2_EP;
  bind_param->req_dst_addr = zb_address_short_by_ieee(g_ieee_addr_ed1);
  TRACE_MSG(TRACE_COMMON1, "dst addr %d", (FMT__D, bind_param->req_dst_addr));

  zb_zdo_unbind_req(param, unbind_device1_cb);
}


void unbind_device1_cb(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_zdo_bind_resp_t *bind_resp = (zb_zdo_bind_resp_t*)ZB_BUF_BEGIN(buf);

  TRACE_MSG(TRACE_COMMON1, "unbind_device1_cb resp status %hd", (FMT__H,
      bind_resp->status));
  if (bind_resp->status != ZB_ZDP_STATUS_SUCCESS)
  {
    TRACE_MSG(TRACE_COMMON1, "Error bind device 1. Test status failed", (FMT__0
      ));
  }
  zb_free_buf(buf);

}
```

4.7.2.5 void zdo_mgmt_leave_req (zb_uint8_t *param*, zb_callback_t *cb*)

Sends 2.4.3.3.2 Mgmt_Leave_req.

Parameters

param	- index of buffer with Lqi request parameters.

See Also

zb_zdo_mgmt_leave_param_t (p. 36)

Parameters

cb	- user's function to call when got response from the remote.

Example:

```
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  zb_zdo_mgmt_leave_param_t *req = NULL;
  zb_ret_t ret = RET_OK;

  TRACE_MSG(TRACE_ERROR, "zb_leave_req", (FMT__0));

  req = ZB_GET_BUF_PARAM(buf, zb_zdo_mgmt_leave_param_t);

  ZB_MEMSET(req->device_address, 0, sizeof(zb_ieee_addr_t));
  req->remove_children = ZB_FALSE;
  req->rejoin = ZB_FALSE;
  req->dst_addr = 1;
  zdo_mgmt_leave_req(param, leave_callback);
}

void leave_callback(zb_uint8_t param)
{
  zb_uint8_t *ret = (zb_uint8_t *)ZB_BUF_BEGIN(ZB_BUF_FROM_REF(param));

  TRACE_MSG(TRACE_ERROR, "LEAVE CALLBACK status %hd", (FMT__H, *ret));
}
```

4.7.2.6 void zb_zdo_add_group_req (zb_uint8_t *param*, zb_callback_t *cb*)

ZDO interface for ADD-GROUP.request.

Note that zb_apsme_add_group_request does not call comfirm callback.

Parameters

param	- (in/out) buffer with parameters • in - **zb_apsme_add_group_req_t** (p. 42) • out - **zb_apsme_add_group_conf_t** (p. 42)
cb	- user's callback to be used as APSME-ADD-GROUP.confirm.

See Also

zb_apsme_add_group_conf_t (p. 42)

Example

```
{
  zb_apsme_add_group_req_t *req;
  zb_buf_reuse(buf);
  req = ZB_GET_BUF_PARAM(buf, zb_apsme_add_group_req_t);
  req->group_address = 10;
  req->endpoint = 66;
  zb_zdo_add_group_req(param, group_add_conf1);
}

void group_add_conf1(zb_uint8_t param) ZB_CALLBACK
{
  zb_apsme_add_group_conf_t *conf = ZB_GET_BUF_PARAM(ZB_BUF_FROM_REF(param),
      zb_apsme_add_group_conf_t);
  conf->status = status;

  TRACE_MSG(TRACE_ERROR, "group add status %hd", (FMT__H, conf->status));
}
```

4.7.3 Typedef Documentation

4.7.3.1 typedef struct zb_zdo_mgmt_nwk_update_req_hdr_s zb_zdo_mgmt_nwk_update_req_hdr_t

Header of parameters for Mgmt_NWK_Update_req.

4.7.3.2 typedef struct zb_zdo_mgmt_nwk_update_req_s zb_zdo_mgmt_nwk_update_req_t

Parameters for Mgmt_NWK_Update_req.

4.7.3.3 typedef struct zb_zdo_mgmt_nwk_update_notify_hdr_s zb_zdo_mgmt_nwk_update_notify_hdr_t

Header parameters for mgmt_nwk_update_notify.

4.7.3.4 typedef struct zb_zdo_mgmt_nwk_update_notify_param_s zb_zdo_mgmt_nwk_update_notify_param-_t

Parameters for mgmt_nwk_update_notify.

4.7.3.5 typedef struct zb_zdo_mgmt_lqi_param_s zb_zdo_mgmt_lqi_param_t

Parameters for 2.4.3.3.2 Mgmt_Lqi_req.

4.7.3.6 typedef struct zb_zdo_mgmt_lqi_req_s zb_zdo_mgmt_lqi_req_t

Request for 2.4.3.3.2 Mgmt_Lqi_req.

4.7.3.7 typedef struct zb_zdo_mgmt_lqi_resp_s zb_zdo_mgmt_lqi_resp_t

Response for 2.4.4.3.2 Mgmt_Lqi_rsp.

4.7.3.8 typedef struct zb_zdo_neighbor_table_record_s zb_zdo_neighbor_table_record_t

NeighborTableList Record Format for mgmt_lqi_resp.

4.7.3.9 typedef struct zb_zdo_bind_req_param_s zb_zdo_bind_req_param_t

Parameters for 2.4.3.2.2 Bind_req API call.

4.7.3.10 typedef struct zb_zdo_bind_req_head_s zb_zdo_bind_req_head_t

2.4.3.2.2 Bind_req request head send to the remote

4.7.3.11 typedef struct zb_zdo_bind_req_tail_1_s zb_zdo_bind_req_tail_1_t

2.4.3.2.2 Bind_req request tail 1st variant send to the remote

4.7.3.12 typedef struct zb_zdo_bind_req_tail_2_s zb_zdo_bind_req_tail_2_t

2.4.3.2.2 Bind_req request tail 2nd variant send to the remote

4.7.3.13 typedef struct zb_zdo_mgmt_leave_param_s zb_zdo_mgmt_leave_param_t

Request for 2.4.3.3.5 Mgmt_Leave_req.

Problem in the specification: in 2.4.3.3.5 Mgmt_Leave_req only one DeviceAddress exists. But, in such case it is impossible to satisfy 2.4.3.3.5.1: "The Mgmt_Leave_req is generated from a Local Device requesting that a Remote Device leave the network or to request that another device leave the network." Also, in the PRO TC document, 14.2-TP/NWK/BV-04 ZR-ZDO-APL RX Join/Leave is following note: "gZC sends Mgmt_Leave.request with DevAddr=all zero, DstAddr=ZR"

4.7.3.14 typedef struct zb_zdo_mgmt_leave_req_s zb_zdo_mgmt_leave_req_t

Request for 2.4.3.3.5 Mgmt_Leave_req.

4.7.3.15 typedef struct zb_zdo_mgmt_leave_res_s zb_zdo_mgmt_leave_res_t

Response for 2.4.4.3.5 Mgmt_Leave_rsp.

4.7.3.16 typedef struct zb_zdo_end_device_bind_req_head_s zb_zdo_end_device_bind_req_head_t

2.4.3.2.1 End_Device_Bind_req command head

4.7.3.17 typedef struct zb_zdo_end_device_bind_req_tail_s zb_zdo_end_device_bind_req_tail_t

2.4.3.2.1 End_Device_Bind_req command head

4.7.3.18 typedef struct zb_end_device_bind_req_param_s zb_end_device_bind_req_param_t

Parameters for 2.4.3.2.1 End_Device_Bind_req.

4.7.3.19 typedef struct zb_zdo_mgmt_permit_joining_req_s zb_zdo_mgmt_permit_joining_req_t

Parameters for 2.4.3.3.7 Mgmt_Permit_Joining_req.

4.7.3.20 typedef struct zb_zdo_mgmt_permit_joining_req_param_s zb_zdo_mgmt_permit_joining_req_-param_t

Parameters for zb_zdo_mgmt_permit_joining_req.

4.8 AF functions visible to applications

Functions

- void **zb_af_set_data_indication** (**zb_callback_t** cb)
  This function setup user callback to be called for APS data packets not parsed internally.

4.8.1 Detailed Description

4.8.2 Function Documentation

4.8.2.1 void zb_af_set_data_indication (**zb_callback_t** *cb*)

This function setup user callback to be called for APS data packets not parsed internally.

To be used mainly for tests.

Parameters

cb	- callback to call when AF got APS packet to the endpoint is has no explicit handler for.

See Also

zb_apsde_data_indication_t (p. 42)

Example:

```
void zb_zdo_startup_complete(zb_uint8_t param) ZB_CALLBACK
{
  zb_buf_t *buf = ZB_BUF_FROM_REF(param);
  TRACE_MSG(TRACE_APS3, ">>zb_zdo_startup_complete status %hd", (FMT__D, buf->u
      .hdr.status));
  if (buf->u.hdr.status == 0)
  {
    TRACE_MSG(TRACE_APS1, "Device STARTED OK", (FMT__0));
    zb_af_set_data_indication(data_indication);
  }
  else
  {
    TRACE_MSG(TRACE_ERROR, "Device start FAILED status %hd", (FMT__D, buf->u.
      hdr.status));
  }
  zb_free_buf(buf);
}

void data_indication(zb_uint8_t param) ZB_CALLBACK
{
  zb_ushort_t i;
  zb_uint8_t *ptr;
  zb_buf_t *asdu = (zb_buf_t *)ZB_BUF_FROM_REF(param);
  zb_apsde_data_indication_t *ind = ZB_GET_BUF_PARAM(asdu,
      zb_apsde_data_indication_t);

  ptr = ZB_APS_HDR_CUT(asdu);

  TRACE_MSG(TRACE_APS3, "apsde_data_indication: packet %p len %hd status 0x%hx
       from %d",
            (FMT__P_D_D_D, asdu, ZB_BUF_LEN(asdu), asdu->u.hdr.status, ind->
      src_addr));

  for (i = 0 ; i < ZB_BUF_LEN(asdu) ; ++i)
  {
    TRACE_MSG(TRACE_APS3, "%x %c", (FMT__D_C, (int)ptr[i], ptr[i]));
  }
  zb_free_buf(apsdu);
}
```

4.9 APS functions visible to applications

Functions

- void **zb_apsde_data_request** (**zb_uint8_t** param) ZB_CALLBACK

    NLDE-DATA.request primitive.

Data Structures

- struct **zb_apsde_data_req_s**

    APSDE data request structure.
- struct **zb_apsme_binding_req_s**

    APSME binding structure.
- struct **zb_aps_hdr_s**

    Parsed APS header This data structure passed to zb_aps_hdr_parse()
- struct **zb_apsme_add_group_req_s**

    APSME-ADD-GROUP.request primitive parameters.
- struct **zb_apsme_add_group_conf_s**

    APSME-ADD-GROUP.confirm primitive parameters.

Modules

- **APS Informational Base**

Macros

- #define **ZB_MIN_ENDPOINT_NUMBER** 1
- #define **ZB_MAX_ENDPOINT_NUMBER** 240
- #define **ZB_APS_HDR_CUT_P**(packet, ptr) **ZB_BUF_CUT_LEFT**(packet, zb_aps_full_hdr_size(**ZB_BUF_-BEGIN**(packet)), ptr)

    Remove APS header from the packet.
- #define **ZB_APS_HDR_CUT**(packet) zb_buf_cut_left(packet, zb_aps_full_hdr_size(**ZB_BUF_BEGI-N**(packet)))

    Remove APS header from the packet.

Typedefs

- typedef enum **zb_aps_status_e zb_aps_status_t**
- typedef struct **zb_apsde_data_req_s zb_apsde_data_req_t**

    APSDE data request structure.
- typedef struct
zb_apsme_binding_req_s zb_apsme_binding_req_t

    APSME binding structure.
- typedef struct **zb_aps_hdr_s zb_aps_hdr_t**

    Parsed APS header This data structure passed to zb_aps_hdr_parse()
- typedef **zb_aps_hdr_t zb_apsde_data_indication_t**

    Parameters of the APSDE-DATA.indication primitive.
- typedef struct
zb_apsme_add_group_req_s zb_apsme_add_group_req_t

    APSME-ADD-GROUP.request primitive parameters.
- typedef struct
zb_apsme_add_group_conf_s zb_apsme_add_group_conf_t

    APSME-ADD-GROUP.confirm primitive parameters.

Enumerations

- enum **zb_aps_addr_mode_e** { **ZB_APS_ADDR_MODE_DST_ADDR_ENDP_NOT_PRESENT** = 0, **ZB_A-PS_ADDR_MODE_16_GROUP_ENDP_NOT_PRESENT** = 1, **ZB_APS_ADDR_MODE_16_ENDP_PRES-ENT** = 2, **ZB_APS_ADDR_MODE_64_ENDP_PRESENT** = 3 }

  APS addressing mode constants.
- enum **zb_aps_status_e** {
  ZB_APS_STATUS_SUCCESS = 0x00, **ZB_APS_STATUS_INVALID_BINDING** = 0xa4, **ZB_APS_STATU-S_INVALID_GROUP** = 0xa5, **ZB_APS_STATUS_INVALID_PARAMETER** = 0xa6,
  ZB_APS_STATUS_NO_BOUND_DEVICE = 0xa8, **ZB_APS_STATUS_NO_SHORT_ADDRESS** = 0xa9, **Z-B_APS_STATUS_NOT_SUPPORTED** = 0xaa, **ZB_APS_STATUS_SECURED_LINK_KEY** = 0xab,
  ZB_APS_STATUS_SECURED_NWK_KEY = 0xac, **ZB_APS_STATUS_SECURITY_FAIL** = 0xad, **ZB_AP-S_STATUS_TABLE_FULL** = 0xae, **ZB_APS_STATUS_UNSECURED** = 0xaf,
  ZB_APS_STATUS_UNSUPPORTED_ATTRIBUTE = 0xb0 }
- enum **zb_apsde_tx_opt_e** { **ZB_APSDE_TX_OPT_SECURITY_ENABLED** = 1, **ZB_APSDE_TX_OPT_U-SE_NWK_KEY** = 2, **ZB_APSDE_TX_OPT_ACK_TX** = 4, **ZB_APSDE_TX_OPT_FRAG_PERMITTED** = 8 }

  The transmission options for the ASDU to be transferred.

4.9.1 Detailed Description

4.9.2 Function Documentation

4.9.2.1 void zb_apsde_data_request (zb_uint8_t *param*)

NLDE-DATA.request primitive.

This function can be called via scheduler, returns immediatly. Later zb_nlde_data_confirm will be called to pass NLDE-DATA.request result up.

Parameters

apsdu	- packet to send (

See Also

zb_buf_t) and parameters at buffer tail
zb_nlde_data_req_t (p. 50)

Example:

```
{
  zb_apsde_data_req_t *req;
  zb_ushort_t i;

  buf = ZB_BUF_FROM_REF(param);
  ZB_BUF_INITIAL_ALLOC(buf, 10, ptr);
  for (i = 0 ; i < 10 ; ++i)
  {
    ptr[i] = i % 32 + '0';
  }

  req = ZB_GET_BUF_TAIL(buf, sizeof(zb_apsde_data_req_t));
  req->dst_addr.addr_short = 0; // ZC
  req->addr_mode = ZB_APS_ADDR_MODE_16_ENDP_PRESENT;
  req->tx_options = ZB_APSDE_TX_OPT_ACK_TX;
  req->radius = 5;
  req->profileid = 2;
  req->src_endpoint = 10;
  req->dst_endpoint = 10;
  buf->u.hdr.handle = 0x11;
  TRACE_MSG(TRACE_APS3, "Sending apsde_data.request", (FMT__0));
  ZB_SCHEDULE_CALLBACK(zb_apsde_data_request, ZB_REF_FROM_BUF(buf));
}
```

4.9.3 Macro Definition Documentation

4.9.3.1 #define ZB_APS_HDR_CUT_P(*packet,* *ptr*) ZB_BUF_CUT_LEFT(packet, zb_aps_full_hdr_size(ZB_BUF_BEGIN(packet)), ptr)

Remove APS header from the packet.

Parameters

packet	- APS packet
ptr	- (out) pointer to the APS data begin

Example:

```
void data_indication(zb_uint8_t param)
{
  zb_ushort_t i;
  zb_uint8_t *ptr;
  zb_buf_t *asdu = (zb_buf_t *)ZB_BUF_FROM_REF(param);

  ptr = ZB_APS_HDR_CUT_P(asdu);

  TRACE_MSG(TRACE_APS3, "data_indication: packet %p len %d handle 0x%x", (
      FMT__P_D_D,
                         asdu, (int)ZB_BUF_LEN(asdu), asdu->u.hdr.status));

  for (i = 0 ; i < ZB_BUF_LEN(asdu) ; ++i)
  {
    TRACE_MSG(TRACE_APS3, "%x %c", (FMT__D_C, (int)ptr[i], ptr[i]));
  }

  zb_free_buf(asdu);
}
```

4.9.3.2 #define ZB_APS_HDR_CUT(*packet*) zb_buf_cut_left(packet, zb_aps_full_hdr_size(ZB_BUF_BEGIN(packet)))

Remove APS header from the packet.

Parameters

packet	- APS packet
ptr	- (out) pointer to the APS data begin

Example:

```
void data_indication(zb_uint8_t param)
{
  zb_ushort_t i;
  zb_buf_t *asdu = (zb_buf_t *)ZB_BUF_FROM_REF(param);

  ZB_APS_HDR_CUT(asdu);

  TRACE_MSG(TRACE_APS3, "data_indication: packet %p len %d handle 0x%x", (
      FMT__P_D_D,
                         asdu, (int)ZB_BUF_LEN(asdu), asdu->u.hdr.status));

  zb_free_buf(asdu);
}
```

4.9.4 Typedef Documentation

4.9.4.1 typedef struct zb_apsde_data_req_s zb_apsde_data_req_t

APSDE data request structure.

This data structure passed to **zb_apsde_data_request()** (p. 40) in the packet buffer (at its tail).

4.9.4.2 typedef struct zb_apsme_binding_req_s zb_apsme_binding_req_t

APSME binding structure.

This data structure passed to zb_apsme_bind_request()

4.9.4.3 typedef struct zb_aps_hdr_s zb_aps_hdr_t

Parsed APS header This data structure passed to zb_aps_hdr_parse()

4.9.4.4 typedef zb_aps_hdr_t zb_apsde_data_indication_t

Parameters of the APSDE-DATA.indication primitive.

4.9.4.5 typedef struct zb_apsme_add_group_req_s zb_apsme_add_group_req_t

APSME-ADD-GROUP.request primitive parameters.

4.9.4.6 typedef struct zb_apsme_add_group_conf_s zb_apsme_add_group_conf_t

APSME-ADD-GROUP.confirm primitive parameters.

4.9.5 Enumeration Type Documentation

4.9.5.1 enum zb_aps_addr_mode_e

APS addressing mode constants.

Enumerator:

ZB_APS_ADDR_MODE_DST_ADDR_ENDP_NOT_PRESENT 0x00 = DstAddress and DstEndpoint not present

ZB_APS_ADDR_MODE_16_GROUP_ENDP_NOT_PRESENT 0x01 = 16-bit group address for DstAddress; DstEndpoint not present

ZB_APS_ADDR_MODE_16_ENDP_PRESENT 0x02 = 16-bit address for DstAddress and DstEndpoint present

ZB_APS_ADDR_MODE_64_ENDP_PRESENT 0x03 = 64-bit extended address for DstAddress and Dst-Endpoint present

4.9.5.2 enum zb_aps_status_e

Enumerator:

ZB_APS_STATUS_SUCCESS A request has been executed successfully.

ZB_APS_STATUS_INVALID_BINDING An APSME-UNBIND.request failed due to the requested binding link not existing in the binding table.

ZB_APS_STATUS_INVALID_GROUP An APSME-REMOVE-GROUP.request has been issued with a group identifier that does not appear in the group table.

ZB_APS_STATUS_INVALID_PARAMETER A parameter value was invalid or out of range. ZB_APS_STATUS_NO_ACK 0xa7 An APSDE-DATA.request requesting acknowledged transmission failed due to no acknowledgement being received.

ZB_APS_STATUS_NO_BOUND_DEVICE An APSDE-DATA.request with a destination addressing mode set to 0x00 failed due to there being no devices bound to this device.

ZB_APS_STATUS_NO_SHORT_ADDRESS An APSDE-DATA.request with a destination addressing mode set to 0x03 failed due to no corresponding short address found in the address map table.

ZB_APS_STATUS_NOT_SUPPORTED An APSDE-DATA.request with a destination addressing mode set to 0x00 failed due to a binding table not being supported on the device.

ZB_APS_STATUS_SECURED_LINK_KEY An ASDU was received that was secured using a link key.

ZB_APS_STATUS_SECURED_NWK_KEY An ASDU was received that was secured using a network key.

ZB_APS_STATUS_SECURITY_FAIL An APSDE-DATA.request requesting security has resulted in an error during the corresponding security processing.

ZB_APS_STATUS_TABLE_FULL An APSME-BIND.request or APSME.ADD- GROUP.request issued when the binding or group tables, respectively, were full.

ZB_APS_STATUS_UNSECURED An ASDU was received without any security

ZB_APS_STATUS_UNSUPPORTED_ATTRIBUTE An APSME-GET.request or APSME- SET.request has been issued with an unknown attribute identifier.

4.9.5.3 enum zb_apsde_tx_opt_e

The transmission options for the ASDU to be transferred.

These are a bitwise OR of one or more.

Enumerator:

ZB_APSDE_TX_OPT_SECURITY_ENABLED 0x01 = Security enabled transmission

ZB_APSDE_TX_OPT_USE_NWK_KEY 0x02 = Use NWK key

ZB_APSDE_TX_OPT_ACK_TX 0x04 = Acknowledged transmission

ZB_APSDE_TX_OPT_FRAG_PERMITTED 0x08 = Fragmentation permitted

4.10 APS Informational Base

Functions

- void **zb_apsme_get_request** (**zb_uint8_t** param) ZB_CALLBACK

    APSME GET request primitive.
- void **zb_apsme_get_confirm** (**zb_uint8_t** param) ZB_CALLBACK

    APSME GET confirm primitive.
- void **zb_apsme_set_request** (**zb_uint8_t** param) ZB_CALLBACK

    APSME SET request primitive.
- void **zb_apsme_set_confirm** (**zb_uint8_t** param) ZB_CALLBACK

    APSME SET confirm primitive.

Data Structures

- struct **zb_apsme_get_request_s**

    APSME GET request structure.
- struct **zb_apsme_get_confirm_s**

    APSME GET confirm structure.
- struct **zb_apsme_set_request_s**

    APSME SET request structure.
- struct **zb_apsme_set_confirm_s**

    APSME SET confirm structure.

Typedefs

- typedef enum **zb_aps_aib_attr_id_e** **zb_aps_aib_attr_id_t**

    APS Information Base constants.
- typedef struct
  zb_apsme_get_request_s **zb_apsme_get_request_t**

    APSME GET request structure.
- typedef struct
  zb_apsme_get_confirm_s **zb_apsme_get_confirm_t**

    APSME GET confirm structure.
- typedef struct
  zb_apsme_set_request_s **zb_apsme_set_request_t**

    APSME SET request structure.
- typedef struct
  zb_apsme_set_confirm_s **zb_apsme_set_confirm_t**

    APSME SET confirm structure.

Enumerations

- enum **zb_aps_aib_attr_id_e** {
  ZB_APS_AIB_BINDING = 0xc1, **ZB_APS_AIB_DESIGNATED_COORD** = 0xc2, **ZB_APS_AIB_CHANNE-L_MASK** = 0xc3, **ZB_APS_AIB_USE_EXT_PANID** = 0xc4,
  ZB_APS_AIB_GROUP_TABLE = 0xc5, **ZB_APS_AIB_NONMEMBER_RADIUS** = 0xc6, **ZB_APS_AIB_P-ERMISSION_CONFIG** = 0xc7, **ZB_APS_AIB_USE_INSECURE_JOIN** = 0xc8,
  ZB_APS_AIB_INTERFRAME_DELAY = 0xc9, **ZB_APS_AIB_LAST_CHANNEL_ENERGY** = 0xca, **ZB_A-PS_AIB_LAST_CHANNEL_FAILURE_RATE** = 0xcb, **ZB_APS_AIB_CHANNEL_TIMER** = 0xcc }

    APS Information Base constants.

4.10.1 Detailed Description

4.10.2 Function Documentation

4.10.2.1 void zb_apsme_get_request (zb_uint8_t *param*)

APSME GET request primitive.

4.10.2.2 void zb_apsme_get_confirm (zb_uint8_t *param*)

APSME GET confirm primitive.

4.10.2.3 void zb_apsme_set_request (zb_uint8_t *param*)

APSME SET request primitive.

4.10.2.4 void zb_apsme_set_confirm (zb_uint8_t *param*)

APSME SET confirm primitive.

4.10.3 Typedef Documentation

4.10.3.1 typedef enum zb_aps_aib_attr_id_e zb_aps_aib_attr_id_t

APS Information Base constants.

4.10.3.2 typedef struct zb_apsme_get_request_s zb_apsme_get_request_t

APSME GET request structure.

4.10.3.3 typedef struct zb_apsme_get_confirm_s zb_apsme_get_confirm_t

APSME GET confirm structure.

4.10.3.4 typedef struct zb_apsme_set_request_s zb_apsme_set_request_t

APSME SET request structure.

4.10.3.5 typedef struct zb_apsme_set_confirm_s zb_apsme_set_confirm_t

APSME SET confirm structure.

4.10.4 Enumeration Type Documentation

4.10.4.1 enum zb_aps_aib_attr_id_e

APS Information Base constants.

Enumerator:

ZB_APS_AIB_BINDING	The current set of binding table entries in the device (see subclause 2.2.8.2.1).

ZB_APS_AIB_DESIGNATED_COORD TRUE if the device should become the ZigBee Coordinator on startup, FALSE if otherwise.

ZB_APS_AIB_CHANNEL_MASK The mask of allowable channels for this device to use for network operations.

ZB_APS_AIB_USE_EXT_PANID The 64-bit address of a network to form or to join.

ZB_APS_AIB_GROUP_TABLE The current set of group table entries (see Table 2.25).

ZB_APS_AIB_NONMEMBER_RADIUS The value to be used for the NonmemberRadius parameter when using NWK layer multicast.

ZB_APS_AIB_PERMISSION_CONFIG The current set of permission configuration items.

ZB_APS_AIB_USE_INSECURE_JOIN A flag controlling the use of insecure join at startup.

ZB_APS_AIB_INTERFRAME_DELAY Fragmentation parameter - the standard delay, in milliseconds, between sending two blocks of a fragmented transmission (see subclause 2.2.8.4.5).

ZB_APS_AIB_LAST_CHANNEL_ENERGY The energy measurement for the channel energy scan performed on the previous channel just before a channel change (in accordance with [B1]).

ZB_APS_AIB_LAST_CHANNEL_FAILURE_RATE The latest percentage of transmission network transmission failures for the previous channel just before a channel change (in percentage of failed transmissions to the total number of transmissions attempted)

ZB_APS_AIB_CHANNEL_TIMER A countdown timer (in hours) indicating the time to the next permitted frequency agility channel change. A value of NULL indicates the channel has not been changed previously.

4.11 NWK functions visible to applications

Functions

- void **zb_nlde_data_request** (**zb_uint8_t** param) ZB_CALLBACK

  NLDE-DATA.request primitive.
- void **call_status_indication** (**zb_uint8_t** param) ZB_CALLBACK
- void **zb_nlme_send_status** (**zb_uint8_t** param) ZB_CALLBACK

  Send status indication primitive.

Data Structures

- struct **zb_nlde_data_req_s**

  Parameters for NLDE-DATA.request primitive.
- struct **zb_nlme_status_indication_s**

  Arguments of the NLME-STATUS.request routine.
- struct **zb_nlme_send_status_s**

  Arguments of the NLME-SEND-STATUS.confirm routine.

Modules

- **NWK Informational Base**

Macros

- #define **ZB_NWK_IS_ADDRESS_BROADCAST**(addr) (((addr) & 0xFFF0) == 0xFFF0)

  Check that address is broadcast.
- #define **ZB_NWK_COMMAND_STATUS_FRAME_SECURITY_FAILED ZB_NWK_COMMAND_STATUS-_BAD_KEY_SEQUENCE_NUMBER**

  'frame security failed' status mentioned in 4.3.1.2 Security Processing of Incoming Frames but not defined in the table 3.42 Status Codes for Network Status Command Frame
- #define **ZB_NWK_COMMAND_STATUS_IS_SECURE**(st) ((st) == **ZB_NWK_COMMAND_STATUS_BAD-_FRAME_COUNTER** || (st) == **ZB_NWK_COMMAND_STATUS_BAD_KEY_SEQUENCE_NUMBER**)

  Check that NWK command status is security-related.

Typedefs

- typedef enum
  zb_nwk_broadcast_address_e zb_nwk_broadcast_address_t

  Network broadcast addresses types.
- typedef enum **zb_nwk_status_e zb_nwk_status_t**

  NWK layer status values.
- typedef enum
  zb_nwk_command_status_e zb_nwk_command_status_t

  Network command status codes.
- typedef struct **zb_nlde_data_req_s zb_nlde_data_req_t**

  Parameters for NLDE-DATA.request primitive.
- typedef struct
  zb_nlme_status_indication_s zb_nlme_status_indication_t

  Arguments of the NLME-STATUS.request routine.
- typedef struct
  zb_nlme_send_status_s zb_nlme_send_status_t

  Arguments of the NLME-SEND-STATUS.confirm routine.

Enumerations

- enum **zb_nwk_broadcast_address_e** {
ZB_NWK_BROADCAST_ALL_DEVICES = 0xFFFF, **ZB_NWK_BROADCAST_RESERVED** = 0xFFFE, **ZB-_NWK_BROADCAST_RX_ON_WHEN_IDLE** = 0xFFFD, **ZB_NWK_BROADCAST_ROUTER_COORDINA-TOR** = 0xFFFC,
ZB_NWK_BROADCAST_LOW_POWER_ROUTER = 0xFFFB }

  Network broadcast addresses types.
- enum **zb_nwk_status_e** {
ZB_NWK_STATUS_SUCCESS = 0x00, **ZB_NWK_STATUS_INVALID_PARAMETER** = 0xC1, **ZB_NWK_-STATUS_INVALID_REQUEST** = 0xC2, **ZB_NWK_STATUS_NOT_PERMITTED** = 0xC3,
ZB_NWK_STATUS_STARTUP_FAILURE = 0xC4, **ZB_NWK_STATUS_ALREADY_PRESENT** = 0xC5, **Z-B_NWK_STATUS_SYNC_FAILURE** = 0xC6, **ZB_NWK_STATUS_NEIGHBOR_TABLE_FULL** = 0xC7,
ZB_NWK_STATUS_UNKNOWN_DEVICE = 0xC8, **ZB_NWK_STATUS_UNSUPPORTED_ATTRIBUTE** = 0xC9, **ZB_NWK_STATUS_NO_NETWORKS** = 0xCA, **ZB_NWK_STATUS_MAX_FRM_COUNTER** = 0xC-C,
ZB_NWK_STATUS_NO_KEY = 0xCD, **ZB_NWK_STATUS_BAD_CCM_OUTPUT** = 0xCE, **ZB_NWK_STA-TUS_NO_ROUTING_CAPACITY** = 0xCF, **ZB_NWK_STATUS_ROUTE_DISCOVERY_FAILED** = 0xD0,
ZB_NWK_STATUS_ROUTE_ERROR = 0xD1, **ZB_NWK_STATUS_BT_TABLE_FULL** = 0xD2, **ZB_NWK-_STATUS_FRAME_NOT_BUFFERED** = 0xD3 }

  NWK layer status values.
- enum **zb_nwk_command_status_e** {
ZB_NWK_COMMAND_STATUS_NO_ROUTE_AVAILABLE = 0x00, **ZB_NWK_COMMAND_STATUS_T-REE_LINK_FAILURE** = 0x01, **ZB_NWK_COMMAND_STATUS_NONE_TREE_LINK_FAILURE** = 0x02, **ZB_NWK_COMMAND_STATUS_LOW_BATTERY_LEVEL** = 0x03,
ZB_NWK_COMMAND_STATUS_NO_ROUTING_CAPACITY = 0x04, **ZB_NWK_COMMAND_STATUS_N-O_INDIRECT_CAPACITY** = 0x05, **ZB_NWK_COMMAND_STATUS_INDIRECT_TRANSACTION_EXPIRY** = 0x06, **ZB_NWK_COMMAND_STATUS_TARGET_DEVICE_UNAVAILABLE** = 0x07,
ZB_NWK_COMMAND_STATUS_TARGET_ADDRESS_UNALLOCATED = 0x08, **ZB_NWK_COMMAN-D_STATUS_PARENT_LINK_FAILURE** = 0x09, **ZB_NWK_COMMAND_STATUS_VALIDATE_ROUTE** = 0x0a, **ZB_NWK_COMMAND_STATUS_SOURCE_ROUTE_FAILURE** = 0x0b,
ZB_NWK_COMMAND_STATUS_MANY_TO_ONE_ROUTE_FAILURE = 0x0c, **ZB_NWK_COMMAND_-STATUS_ADDRESS_CONFLICT** = 0x0d, **ZB_NWK_COMMAND_STATUS_VERIFY_ADDRESS** = 0x0e, **ZB_NWK_COMMAND_STATUS_PAN_IDENTIFIER_UPDATE** = 0x0f,
ZB_NWK_COMMAND_STATUS_NETWORK_ADDRESS_UPDATE = 0x10, **ZB_NWK_COMMAND_STA-TUS_BAD_FRAME_COUNTER** = 0x11, **ZB_NWK_COMMAND_STATUS_BAD_KEY_SEQUENCE_NUM-BER** = 0x12 }

  Network command status codes.

4.11.1 Detailed Description

4.11.2 Function Documentation

4.11.2.1 void zb_nlde_data_request (zb_uint8_t *param*)

NLDE-DATA.request primitive.

This function return immediatly. Later zb_nlde_data_confirm will be called to pass NLDE-DATA.request result up.

Parameters

nldereq	- parameters structure -

See Also

zb_nlde_data_req_t (p. 50) This variable does not pass to other levels, so it can be local variable in the caller.

Example:

```
{
  zb_nlde_data_req_t *req;
  zb_uint16_t dst_addr;

  req = ZB_GET_BUF_TAIL(buf, sizeof(zb_nlde_data_req_t));
  // send to parent
  zb_address_short_by_ref(&dst_addr, ZG->nwk.handle.parent);
  TRACE_MSG(TRACE_APS3, "parent %hd parent_addr %d", (FMT__H_D, ZG->nwk.handle.
      parent, dst_addr));

  req->dst_addr = dst_addr;
  req->radius = 0; // use default
  req->addr_mode = ZB_ADDR_16BIT_DEV_OR_BROADCAST;
  req->discovery_route = 0;
  req->security_enable = 0;
  req->ndsu_handle = 10;

  TRACE_MSG(TRACE_APS3, "Sending nlde_data.request", (FMT__0));
  ZB_SCHEDULE_CALLBACK(zb_nlde_data_request, ZB_REF_FROM_BUF(buf));
}
```

4.11.2.2 void zb_nlme_send_status (zb_uint8_t *param*)

Send status indication primitive.

Send status to the remote device

Parameters

v_buf	- request params -

See Also

zb_nlme_send_status_t (p. 50)

Returns

nothing

Example:

```
{
  zb_nlme_send_status_t *request = ZB_GET_BUF_PARAM(ZB_BUF_FROM_REF(param),
      zb_nlme_send_status_t);

  request->dest_addr = 0; // send status indication to the coordinator
  request->status.status = ZB_NWK_COMMAND_STATUS_LOW_BATTERY_LEVEL;
  request->status.network_addr = ZB_NIB_NETWORK_ADDRESS();
  request->ndsu_handle = 0;

  ZB_SCHEDULE_CALLBACK(zb_nlme_send_status, param);
}
```

4.11.3 Macro Definition Documentation

4.11.3.1 #define ZB_NWK_IS_ADDRESS_BROADCAST(*addr*) (((addr) & 0xFFF0) == 0xFFF0)

Check that address is broadcast.

Parameters

addr	- 16-bit address

Returns

TRUE if address is broadcast, FALSE otherwhise

4.11.3.2 #define ZB_NWK_COMMAND_STATUS_FRAME_SECURITY_FAILED ZB_NWK_COMMAND_STATUS_BAD_KEY-_SEQUENCE_NUMBER

'frame security failed' status mentioned in 4.3.1.2 Security Processing of Incoming Frames but not defined in the table 3.42 Status Codes for Network Status Command Frame

Really need this status for for intra-pan portrability procedure (AZD601,602). Let's use other security status code.

4.11.3.3 #define ZB_NWK_COMMAND_STATUS_IS_SECURE(*st*) ((st) == ZB_NWK_COMMAND_STATUS_BAD_FRAM-E_COUNTER || (st) == ZB_NWK_COMMAND_STATUS_BAD_KEY_SEQUENCE_NUMBER)

Check that NWK command status is security-related.

Parameters

st	- status code

Returns

1 if NWK command status is security-related

4.11.4 Typedef Documentation

4.11.4.1 typedef enum zb_nwk_broadcast_address_e zb_nwk_broadcast_address_t

Network broadcast addresses types.

4.11.4.2 typedef enum zb_nwk_status_e zb_nwk_status_t

NWK layer status values.

Got from 3.7

4.11.4.3 typedef enum zb_nwk_command_status_e zb_nwk_command_status_t

Network command status codes.

4.11.4.4 typedef struct zb_nlde_data_req_s zb_nlde_data_req_t

Parameters for NLDE-DATA.request primitive.

4.11.4.5 typedef struct zb_nlme_status_indication_s zb_nlme_status_indication_t

Arguments of the NLME-STATUS.request routine.

4.11.4.6 typedef struct zb_nlme_send_status_s zb_nlme_send_status_t

Arguments of the NLME-SEND-STATUS.confirm routine.

4.11.5 Enumeration Type Documentation

4.11.5.1 enum **zb_nwk_broadcast_address_e**

Network broadcast addresses types.

Enumerator:

ZB_NWK_BROADCAST_ALL_DEVICES All devices in PAN

ZB_NWK_BROADCAST_RX_ON_WHEN_IDLE macRxOnWhenIdle = TRUE

ZB_NWK_BROADCAST_ROUTER_COORDINATOR All routers and coordinator

ZB_NWK_BROADCAST_LOW_POWER_ROUTER Low power routers only

4.11.5.2 enum **zb_nwk_status_e**

NWK layer status values.

Got from 3.7

Enumerator:

ZB_NWK_STATUS_SUCCESS A request has been executed successfully.

ZB_NWK_STATUS_INVALID_PARAMETER An invalid or out-of-range parameter has been passed to a primitive from the next higher layer.

ZB_NWK_STATUS_INVALID_REQUEST The next higher layer has issued a request that is invalid or cannot be executed given the current state of the NWK layer.

ZB_NWK_STATUS_NOT_PERMITTED An NLME-JOIN.request has been disallowed.

ZB_NWK_STATUS_STARTUP_FAILURE An NLME-NETWORK-FORMATION.request has failed to start a network.

ZB_NWK_STATUS_ALREADY_PRESENT A device with the address supplied to the NLMEDIRECT-JOIN.-request is already present in the neighbor table of the device on which the NLMEDIRECT-JOIN.request was issued.

ZB_NWK_STATUS_SYNC_FAILURE Used to indicate that an NLME-SYNC.request has failed at the MAC layer.

ZB_NWK_STATUS_NEIGHBOR_TABLE_FULL An NLME-JOIN-DIRECTLY.request has failed because there is no more room in the neighbor table.

ZB_NWK_STATUS_UNKNOWN_DEVICE An NLME-LEAVE.request has failed because the device addressed in the parameter list is not in the neighbor table of the issuing device.

ZB_NWK_STATUS_UNSUPPORTED_ATTRIBUTE An NLME-GET.request or NLME-SET.request has been issued with an unknown attribute identifier.

ZB_NWK_STATUS_NO_NETWORKS An NLME-JOIN.request has been issued in an environment where no networks are detectable.

ZB_NWK_STATUS_MAX_FRM_COUNTER Security processing has been attempted on an outgoing frame, and has failed because the frame counter has reached its maximum value.

ZB_NWK_STATUS_NO_KEY Security processing has been attempted on an outgoing frame, and has failed because no key was available with which to process it.

ZB_NWK_STATUS_BAD_CCM_OUTPUT Security processing has been attempted on an outgoing frame, and has failed because the security engine produced erroneous output.

ZB_NWK_STATUS_NO_ROUTING_CAPACITY An attempt to discover a route has failed due to a lack of routing table or discovery table capacity.

ZB_NWK_STATUS_ROUTE_DISCOVERY_FAILED An attempt to discover a route has failed due to a reason other than a lack of routing capacity.

ZB_NWK_STATUS_ROUTE_ERROR An NLDE-DATA.request has failed due to a routing failure on the sending device.

ZB_NWK_STATUS_BT_TABLE_FULL An attempt to send a broadcast frame or member mode multicast has failed due to the fact that there is no room in the BTT.

ZB_NWK_STATUS_FRAME_NOT_BUFFERED An NLDE-DATA.request has failed due to insufficient buffering available. A non-member mode multicast frame was discarded pending route discovery.

4.11.5.3 enum zb_nwk_command_status_e

Network command status codes.

Enumerator:

ZB_NWK_COMMAND_STATUS_NO_ROUTE_AVAILABLE No route available

ZB_NWK_COMMAND_STATUS_TREE_LINK_FAILURE Tree link failure

ZB_NWK_COMMAND_STATUS_NONE_TREE_LINK_FAILURE None-tree link failure

ZB_NWK_COMMAND_STATUS_LOW_BATTERY_LEVEL Low battery level

ZB_NWK_COMMAND_STATUS_NO_ROUTING_CAPACITY No routing capacity

ZB_NWK_COMMAND_STATUS_NO_INDIRECT_CAPACITY No indirect capacity

ZB_NWK_COMMAND_STATUS_INDIRECT_TRANSACTION_EXPIRY Indirect transaction expiry

ZB_NWK_COMMAND_STATUS_TARGET_DEVICE_UNAVAILABLE Target device unavailable

ZB_NWK_COMMAND_STATUS_TARGET_ADDRESS_UNALLOCATED Target address unallocated

ZB_NWK_COMMAND_STATUS_PARENT_LINK_FAILURE Parent link failure

ZB_NWK_COMMAND_STATUS_VALIDATE_ROUTE Validate route

ZB_NWK_COMMAND_STATUS_SOURCE_ROUTE_FAILURE Source route failure

ZB_NWK_COMMAND_STATUS_MANY_TO_ONE_ROUTE_FAILURE Many-to-one route failure

ZB_NWK_COMMAND_STATUS_ADDRESS_CONFLICT Address conflict

ZB_NWK_COMMAND_STATUS_VERIFY_ADDRESS Verify address

ZB_NWK_COMMAND_STATUS_PAN_IDENTIFIER_UPDATE Pan identifier update

ZB_NWK_COMMAND_STATUS_NETWORK_ADDRESS_UPDATE Network address update

ZB_NWK_COMMAND_STATUS_BAD_FRAME_COUNTER Bad frame counter

ZB_NWK_COMMAND_STATUS_BAD_KEY_SEQUENCE_NUMBER Bad key sequence number

4.12 NWK Informational Base

Functions

- void **zb_nlme_get_request** (**zb_uint8_t** param) ZB_CALLBACK

    NLME-GET.request primitive.
- void **zb_nlme_get_confirm** (**zb_uint8_t** param) ZB_CALLBACK

    NLME-GET.confirm primitive.
- void **zb_nlme_set_request** (**zb_uint8_t** param) ZB_CALLBACK

    NLME-SET.request primitive.
- void **zb_nlme_set_confirm** (**zb_uint8_t** param) ZB_CALLBACK

    NLME-SET.confirm primitive.

Data Structures

- struct **zb_nlme_get_request_s**

    Arguments of the NLME-GET.request routine.
- struct **zb_nlme_get_confirm_s**

    Arguments of the NLME-GET.confirm routine.
- struct **zb_nlme_set_request_s**

    Arguments of the NLME-SET.request routine.
- struct **zb_nlme_set_confirm_s**

    Arguments of the NLME-SET.confirm routine.

Macros

- #define **ZB_NIB_SEQUENCE_NUMBER**() ZG->nwk.nib.sequence_number
- #define **ZB_NIB_SEQUENCE_NUMBER_INC**() (ZG->nwk.nib.sequence_number++)
- #define **ZB_NIB_MAX_DEPTH**() ZG->nwk.nib.max_depth
- #define **ZB_NIB_DEPTH**() ZG->nwk.nib.depth
- #define **ZB_NIB_DEVICE_TYPE**() ZG->nwk.nib.device_type
- #define **ZB_NIB_NETWORK_ADDRESS**() ZG->mac.pib.mac_short_address
- #define **ZB_NIB_PAN_ID**() ZG->mac.pib.mac_pan_id
- #define **ZB_NIB_EXT_PAN_ID**() ZG->nwk.nib.extended_pan_id
- #define **ZB_NIB_UPDATE_ID**() ZG->nwk.nib.update_id
- #define **ZB_NIB_SECURITY_LEVEL**() ZG->nwk.nib.security_level
- #define **ZB_NIB_GET_USE_TREE_ROUTING**() ZG->nwk.nib.use_tree_routing
- #define **ZB_NIB_SET_USE_TREE_ROUTING**(v) (ZG->nwk.nib.use_tree_routing = (v))
- #define **ZB_NIB_SECURITY_MATERIAL**() ZG->nwk.nib.secur_material_set
- #define **ZB_NIB_NWK_MANAGER_ADDR**() ZG->nwk.nib.nwk_manager_addr /* TODO: init it correctly */
- #define **ZB_NIB_NWK_TX_TOTAL**() ZG->nwk.nib.nwk_tx_total
- #define **ZB_NIB_NWK_TX_FAIL**() ZG->nwk.nib.nwk_tx_fail

Typedefs

- typedef struct
  zb_nlme_get_request_s zb_nlme_get_request_t

    Arguments of the NLME-GET.request routine.
- typedef struct
  zb_nlme_get_confirm_s zb_nlme_get_confirm_t

    Arguments of the NLME-GET.confirm routine.

- typedef struct
  zb_nlme_set_request_s zb_nlme_set_request_t

  Arguments of the NLME-SET.request routine.
- typedef struct
  zb_nlme_set_confirm_s zb_nlme_set_confirm_t

  Arguments of the NLME-SET.confirm routine.
- typedef enum **zb_nib_attribute_e zb_nib_attribute_t**

  NWK NIB Attributes.

Enumerations

- enum **zb_nib_attribute_e** {
  ZB_NIB_ATTRIBUTE_SEQUENCE_NUMBER = 0X81, **ZB_NIB_ATTRIBUTE_PASSIVE_ASK_TIMEOUT** = 0X82, **ZB_NIB_ATTRIBUTE_MAX_BROADCAST_RETRIES** = 0X83, **ZB_NIB_ATTRIBUTE_MAX_CHI-LDREN** = 0X84,
  ZB_NIB_ATTRIBUTE_MAX_DEPTH = 0X85, **ZB_NIB_ATTRIBUTE_MAX_ROUTERS** = 0X86, **ZB_NIB_A-TTRIBUTE_NEIGHBOR_TABLE** = 0X87, **ZB_NIB_ATTRIBUTE_BROADCAST_DELIVERY_TIME** = 0X88,
  ZB_NIB_ATTRIBUTE_REPORT_CONSTANT_COST = 0X89, **ZB_NIB_ATTRIBUTE_ROUTE_DISCOVE-RY_RETRIES_PERMITTED** = 0X8A, **ZB_NIB_ATTRIBUTE_ROUTE_TABLE** = 0X8B, **ZB_NIB_ATTRIBU-TE_SYM_LINK** = 0X8E,
  ZB_NIB_ATTRIBUTE_CAPABILITY_INFORMATION = 0X8F, **ZB_NIB_ATTRIBUTE_ADDR_ALLOC** = 0-X90, **ZB_NIB_ATTRIBUTE_USE_TREE_ROUTING** = 0X91, **ZB_NIB_ATTRIBUTE_MANAGER_ADDR** = 0X92,
  ZB_NIB_ATTRIBUTE_MAX_SOURCE_ROUTE = 0X93, **ZB_NIB_ATTRIBUTE_UPDATE_ID** = 0X94, **ZB_-NIB_ATTRIBUTE_TRANSACTION_PERSISTENCE_TIME** = 0X95, **ZB_NIB_ATTRIBUTE_NETWORK_A-DDRESS** = 0X96,
  ZB_NIB_ATTRIBUTE_STACK_PROFILE = 0X97, **ZB_NIB_ATTRIBUTE_BROADCAST_TRANSACTION-_TABLE** = 0X98, **ZB_NIB_ATTRIBUTE_GROUP_ID_TABLE** = 0X99, **ZB_NIB_ATTRIBUTE_EXTENDED-_PANID** = 0X9A,
  ZB_NIB_ATTRIBUTE_USE_MULTICAST = 0X9B, **ZB_NIB_ATTRIBUTE_ROUTE_RECORD_TABLE** = 0-X9C, **ZB_NIB_ATTRIBUTE_IS_CONCENTRATOR** = 0X9D, **ZB_NIB_ATTRIBUTE_CONCENTRATOR_R-ADIUS** = 0X9E,
  ZB_NIB_ATTRIBUTE_CONCENTRATOR_DESCOVERY_TIME = 0X9F, **ZB_NIB_ATTRIBUTE_SECURIT-Y_LEVEL** = 0XA0, **ZB_NIB_ATTRIBUTE_SECURITY_MATERIAL_SET** = 0XA1, **ZB_NIB_ATTRIBUTE_A-CTIVE_KEY_SEQ_NUMBER** = 0XA2,
  ZB_NIB_ATTRIBUTE_ALL_FRESH = 0XA3, **ZB_NIB_ATTRIBUTE_SECURE_ALL_FRAMES** = 0XA5, **Z-B_NIB_ATTRIBUTE_LINK_STATUS_PERIOD** = 0XA6, **ZB_NIB_ATTRIBUTE_ROUTER_AGE_LIMIT** = 0-XA7,
  ZB_NIB_ATTRIBUTE_UNIQUE_ADDR = 0XA8, **ZB_NIB_ATTRIBUTE_ADDRESS_MAP** = 0XA9, **ZB_NI-B_ATTRIBUTE_TIME_STAMP** = 0X8C, **ZB_NIB_ATTRIBUTE_PAN_ID** = 0X80,
  ZB_NIB_ATTRIBUTE_TX_TOTAL = 0X8D }

  NWK NIB Attributes.

4.12.1 Detailed Description

4.12.2 Function Documentation

4.12.2.1 void zb_nlme_get_request (zb_uint8_t *param*)

NLME-GET.request primitive.

Perform get NIB attribute

Parameters

v_buf	- buffer containing parameters -

See Also

zb_nlme_get_request_t (p. 56)

Returns

RET_OK on success, error code otherwise.

4.12.2.2 void zb_nlme_get_confirm (zb_uint8_t *param*)

NLME-GET.confirm primitive.

Report the results of reading attribute from NIB.

Parameters

v_buf	- buffer containing results -

See Also

zb_nlme_get_confirm_t (p. 56)

Returns

RET_OK on success, error code otherwise.

4.12.2.3 void zb_nlme_set_request (zb_uint8_t *param*)

NLME-SET.request primitive.

Perform set NIB attribute

Parameters

v_buf	- buffer containing parameters -

See Also

zb_nlme_set_request_t (p. 56)

Returns

RET_OK on success, error code otherwise.

4.12.2.4 void zb_nlme_set_confirm (zb_uint8_t *param*)

NLME-SET.confirm primitive.

Report the results of writing attribute from NIB.

Parameters

v_buf	- buffer containing results -

See Also

zb_nlme_set_confirm_t (p. 56)

Returns

RET_OK on success, error code otherwise.

4.12.3 Typedef Documentation

4.12.3.1 typedef struct zb_nlme_get_request_s zb_nlme_get_request_t

Arguments of the NLME-GET.request routine.

4.12.3.2 typedef struct zb_nlme_get_confirm_s zb_nlme_get_confirm_t

Arguments of the NLME-GET.confirm routine.

4.12.3.3 typedef struct zb_nlme_set_request_s zb_nlme_set_request_t

Arguments of the NLME-SET.request routine.

4.12.3.4 typedef struct zb_nlme_set_confirm_s zb_nlme_set_confirm_t

Arguments of the NLME-SET.confirm routine.

4.12.3.5 typedef enum zb_nib_attribute_e zb_nib_attribute_t

NWK NIB Attributes.

NWK NIB

Some NIB fields are indeed PIB fields. Use macros to access it.

4.12.4 Enumeration Type Documentation

4.12.4.1 enum zb_nib_attribute_e

NWK NIB Attributes.

NWK NIB

Some NIB fields are indeed PIB fields. Use macros to access it.

4.13 MAC API

Functions

- void **zb_mlme_get_request** (**zb_uint8_t** param) ZB_CALLBACK
  MLME-GET.request primitive.
- void **zb_mlme_get_confirm** (**zb_uint8_t** param) ZB_CALLBACK
  MLME-GET.confirm primitive.
- void **zb_mlme_set_request** (**zb_uint8_t** param) ZB_CALLBACK
  MLME-SET.request primitive.
- void **zb_mlme_set_confirm** (**zb_uint8_t** param) ZB_CALLBACK
  MLME-SET.confirm primitive.

Data Structures

- struct **zb_mac_device_table_s**
- struct **ZB_PACKED_STRUCT**
  MAC PIB.
- struct **zb_mlme_get_request_s**
  Defines MLME-GET.request primitive.
- struct **zb_mlme_get_confirm_s**
  Defines MLME-GET.confirm primitive.
- struct **zb_mlme_set_request_s**
  Defines MLME-SET.request primitive.
- struct **zb_mlme_set_confirm_s**
  Defines MLME-SET.confirm primitive.

Macros

- #define **MAC_PIB**() (ZG->mac.pib)
  Get MAC PIB.
- #define **ZB_PIB_SHORT_PAN_ID**() ZG->mac.pib.mac_pan_id
  Get mac pan id.
- #define **ZB_PIB_SHORT_ADDRESS**() ZG->mac.pib.mac_short_address
  Get mac short address.
- #define **ZB_PIB_EXTENDED_ADDRESS**() ZG->mac.pib.mac_extended_address
  Get mac extended address.
- #define **ZB_PIB_COORD_SHORT_ADDRESS**() ZG->mac.pib.mac_coord_short_address
  Get mac coord short address.
- #define **ZB_PIB_RX_ON_WHEN_IDLE**() ZG->mac.pib.mac_rx_on_when_idle
  Get mac rx on when idle.
- #define **ZB_MAC_DSN**() ZG->mac.pib.mac_dsn
  Get mac DSN.
- #define **ZB_MAC_BSN**() ZG->mac.pib.mac_bsn
  Get mac pan BSN.
- #define **ZB_INC_MAC_DSN**() (ZG->mac.pib.mac_dsn++)
  Increment mac pan DSN.
- #define **ZB_INC_MAC_BSN**() (ZG->mac.pib.mac_bsn++)
  Increment mac pan BSN.
- #define **ZB_PIB_BEACON_PAYLOAD**() ZG->mac.pib.mac_beacon_payload
  Get mac beacon payload.
- #define **ZB_MLME_BUILD_GET_REQ**(buf, pib_attr, outlen)
  Defines MLME-GET.request primitive.

Typedefs

- typedef enum **zb_mac_status_e** **zb_mac_status_t**

  MAC status.
- typedef struct **zb_mac_device_table_s** **zb_mac_device_table_t**
- typedef struct **zb_mlme_get_request_s** **zb_mlme_get_request_t**

  Defines MLME-GET.request primitive.
- typedef struct **zb_mlme_get_confirm_s** **zb_mlme_get_confirm_t**

  Defines MLME-GET.confirm primitive.
- typedef struct **zb_mlme_set_request_s** **zb_mlme_set_request_t**

  Defines MLME-SET.request primitive.
- typedef struct **zb_mlme_set_confirm_s** **zb_mlme_set_confirm_t**

  Defines MLME-SET.confirm primitive.

Enumerations

- enum **zb_mac_status_e** {
  MAC_SUCCESS = 0x0, **MAC_PAN_AT_CAPACITY** = 0x1, **MAC_PAN_ACCESS_DENIED** = 0x2, **MAC_B-EACON_LOSS** = 0xe0,
  MAC_CHANNEL_ACCESS_FAILURE = 0xe1, **MAC_COUNTER_ERROR** = 0xdB, **MAC_DENIED** = 0xe2, **MAC_DISABLE_TRX_FAILURE** = 0xe3,
  MAC_SECURITY_ERROR = 0xe4, **MAC_FRAME_TOO_LONG** = 0xe5, **MAC_IMPROPER_KEY_TYPE** = 0xdc, **MAC_IMPROPER_SECURITY_LEVEL** = 0xdd,
  MAC_INVALID_ADDRESS = 0xf5, **MAC_INVALID_GTS** = 0xe6, **MAC_INVALID_HANDLE** = 0xe7, **MAC_-INVALID_INDEX** = 0xf9,
  MAC_INVALID_PARAMETER = 0xe8, **MAC_LIMIT_REACHED** = 0xfa, **MAC_NO_ACK** = 0xe9, **MAC_NO-_BEACON** = 0xea,
  MAC_NO_DATA = 0xeb, **MAC_NO_SHORT_ADDRESS** = 0xec, **MAC_ON_TIME_TOO_LONG** = 0xf6, **MA-C_OUT_OF_CAP** = 0xed,
  MAC_PAN_ID_CONFLICT = 0xee, **MAC_PAST_TIME** = 0xf7, **MAC_READ_ONLY** = 0xfb, **MAC_REALIG-NMENT** = 0xef,
  MAC_SCAN_IN_PROGRESS = 0xfc, **MAC_SUPERFRAME_OVERLAP** = 0xfd, **MAC_TRACKING_OFF** = 0xf8, **MAC_TRANSACTION_EXPIRED** = 0xf0,
  MAC_TRANSACTION_OVERFLOW = 0xf1, **MAC_TX_ACTIVE** = 0xf2, **MAC_UNAVAILABLE_KEY** = 0xf3, **MAC_UNSUPPORTED_ATTRIBUTE** = 0xf4,
  MAC_UNSUPPORTED_LEGACY = 0xde, **MAC_UNSUPPORTED_SECURITY** = 0xdf }

  MAC status.
- enum **zb_mac_pib_attr_t** {
  ZB_PHY_PIB_CURRENT_CHANNEL = 0x00, **ZB_PHY_PIB_CURRENT_PAGE** = 0x04, **ZB_PIB_ATTRIB-UTE_ACK_WAIT_DURATION** = 0x40, **ZB_PIB_ATTRIBUTE_ASSOCIATION_PERMIT** = 0x41,
  ZB_PIB_ATTRIBUTE_AUTO_REQUEST = 0x42, **ZB_PIB_ATTRIBUTE_BATT_LIFE_EXT** = 0x43, **ZB_PI-B_ATTRIBUTE_BATT_LIFE_EXT_PERIODS** = 0x44, **ZB_PIB_ATTRIBUTE_BEACON_PAYLOAD** = 0x45,
  ZB_PIB_ATTRIBUTE_BEACON_PAYLOAD_LENGTH = 0x46, **ZB_PIB_ATTRIBUTE_BEACON_ORDER** = 0x47, **ZB_PIB_ATTRIBUTE_BEACON_TX_TIME** = 0x48, **ZB_PIB_ATTRIBUTE_BSN** = 0x49,
  ZB_PIB_ATTRIBUTE_COORD_EXTEND_ADDRESS = 0x4a, **ZB_PIB_ATTRIBUTE_COORD_SHORT_A-DDRESS** = 0x4b, **ZB_PIB_ATTRIBUTE_DSN** = 0x4c, **ZB_PIB_ATTRIBUTE_GTS_PERMIT** = 0x4d,
  ZB_PIB_ATTRIBUTE_MAX_CSMA_BACKOFFS = 0x4e, **ZB_PIB_ATTRIBUTE_MIN_BE** = 0x4f, **ZB_PIB-_ATTRIBUTE_PANID** = 0x50, **ZB_PIB_ATTRIBUTE_PROMISCUOUS_MODE** = 0x51,
  ZB_PIB_ATTRIBUTE_RX_ON_WHEN_IDLE = 0x52, **ZB_PIB_ATTRIBUTE_SHORT_ADDRESS** = 0x53,

ZB_PIB_ATTRIBUTE_SUPER_FRAME_ORDER = 0x54, **ZB_PIB_ATTRIBUTE_TRANSACTION_PERSI-STENCE_TIME** = 0x55,
ZB_PIB_ATTRIBUTE_ASSOCIATED_PAN_COORD = 0x56, **ZB_PIB_ATTRIBUTE_MAX_BE** = 0x57, **ZB-_PIB_ATTRIBUTE_MAX_FRAME_TOTAL_WAIT_TIME** = 0x58, **ZB_PIB_ATTRIBUTE_MAX_FRAME_R-ETRIES** = 0x59,
ZB_PIB_ATTRIBUTE_RESPONSE_WAIT_TIME = 0x5a, **ZB_PIB_ATTRIBUTE_SYNC_SYMBOL_OFFS-ET** = 0x5b, **ZB_PIB_ATTRIBUTE_TIMESTAMP_SUPPORTED** = 0x5c, **ZB_PIB_ATTRIBUTE_SECURITY-_ENABLED** = 0x5d }

Mac PIB attributes.

4.13.1 Detailed Description

4.13.2 Function Documentation

4.13.2.1 void zb_mlme_get_request (zb_uint8_t *param*)

MLME-GET.request primitive.

4.13.2.2 void zb_mlme_get_confirm (zb_uint8_t *param*)

MLME-GET.confirm primitive.

4.13.2.3 void zb_mlme_set_request (zb_uint8_t *param*)

MLME-SET.request primitive.

4.13.2.4 void zb_mlme_set_confirm (zb_uint8_t *param*)

MLME-SET.confirm primitive.

4.13.3 Macro Definition Documentation

4.13.3.1 #define MAC_PIB( ) (ZG->mac.pib)

Get MAC PIB.

4.13.3.2 #define ZB_PIB_SHORT_PAN_ID( ) ZG->mac.pib.mac_pan_id

Get mac pan id.

4.13.3.3 #define ZB_PIB_SHORT_ADDRESS( ) ZG->mac.pib.mac_short_address

Get mac short address.

4.13.3.4 #define ZB_PIB_EXTENDED_ADDRESS( ) ZG->mac.pib.mac_extended_address

Get mac extended address.

4.13.3.5 #define ZB_PIB_COORD_SHORT_ADDRESS( ) ZG->mac.pib.mac_coord_short_address

Get mac coord short address.

4.13.3.6 #define ZB_PIB_RX_ON_WHEN_IDLE( ) ZG->mac.pib.mac_rx_on_when_idle

Get mac rx on when idle.

4.13.3.7 #define ZB_MAC_DSN( ) ZG->mac.pib.mac_dsn

Get mac DSN.

4.13.3.8 #define ZB_MAC_BSN( ) ZG->mac.pib.mac_bsn

Get mac pan BSN.

4.13.3.9 #define ZB_INC_MAC_DSN( ) (ZG->mac.pib.mac_dsn++)

Increment mac pan DSN.

4.13.3.10 #define ZB_INC_MAC_BSN( ) (ZG->mac.pib.mac_bsn++)

Increment mac pan BSN.

4.13.3.11 #define ZB_PIB_BEACON_PAYLOAD( ) ZG->mac.pib.mac_beacon_payload

Get mac beacon payload.

4.13.3.12 #define ZB_MLME_BUILD_GET_REQ(*buf,* *pib_attr,* *outlen*)

Value:

```
do                                                        \
{                                                         \
} while( 0 )
```

Defines MLME-GET.request primitive.

Parameters

buf	- pointer to zb_buf_t
pib_attr	- one of possible values from zb_mac_pib_attr_t
outlen	- out integer variable to receive length

4.13.4 Typedef Documentation

4.13.4.1 typedef enum zb_mac_status_e zb_mac_status_t

MAC status.

4.13.4.2 typedef struct zb_mlme_get_request_s zb_mlme_get_request_t

Defines MLME-GET.request primitive.

4.13.4.3 typedef struct zb_mlme_get_confirm_s zb_mlme_get_confirm_t

Defines MLME-GET.confirm primitive.

4.13.4.4 typedef struct zb_mlme_set_request_s zb_mlme_set_request_t

Defines MLME-SET.request primitive.

4.13.4.5 typedef struct zb_mlme_set_confirm_s zb_mlme_set_confirm_t

Defines MLME-SET.confirm primitive.

4.13.5 Enumeration Type Documentation

4.13.5.1 enum zb_mac_status_e

MAC status.

Enumerator:

MAC_SUCCESS Transaction was successful.

MAC_BEACON_LOSS Beacon was lost (used in beacon'd networks)

MAC_CHANNEL_ACCESS_FAILURE Unable to transmit due to channel being busy.

MAC_COUNTER_ERROR Frame counter of received frame is invalid.

MAC_DENIED GTS request denied.

MAC_DISABLE_TRX_FAILURE Failed to disable the transceiver.

MAC_SECURITY_ERROR Frame failed decryption.

MAC_FRAME_TOO_LONG Frame exceeded maximum size.

MAC_IMPROPER_KEY_TYPE Key not allowed to be used with this frame type.

MAC_IMPROPER_SECURITY_LEVEL Frame does not meet min security level expected.

MAC_INVALID_ADDRESS Data request failed because no src or dest address.

MAC_INVALID_GTS Invalid timeslot requested (beacon'd networks)

MAC_INVALID_HANDLE Invalid frame data handle.

MAC_INVALID_INDEX Invalid index when trying to write MAC PIB.

MAC_INVALID_PARAMETER Invalid parameter passed to service.

MAC_LIMIT_REACHED Scan terminated because max pan descriptors reached.

MAC_NO_ACK ACK not received after tx with ack_req flag set.

MAC_NO_BEACON Beacon not returned after beacon request.

MAC_NO_DATA Data frame not returned after data request (indirect poll)

MAC_NO_SHORT_ADDRESS No short address allocated to this device (due to lack of address space)

MAC_ON_TIME_TOO_LONG Rx enable request failed. Spec'd number of symbols longer than beacon interval.

MAC_OUT_OF_CAP Association failed due to lack of capacity (no nbor tbl entry or no address)

MAC_PAN_ID_CONFLICT Different networks within listening range have identical PAN IDs.

MAC_PAST_TIME Rx enable failed. Too late for current superframe and unable to be deferred.

MAC_READ_ONLY PIB attribute is read only.

MAC_REALIGNMENT Coordinator realignment received.

MAC_SCAN_IN_PROGRESS Request to perform scan failed because scan already in progress.

MAC_SUPERFRAME_OVERLAP Start time of beacon overlapped transmission time of coordinator beacon.

MAC_TRACKING_OFF Device not tracking beacons but instructed to send beacons based on tracked beacons.

MAC_TRANSACTION_EXPIRED Frame buffered in indirect queue expired.

MAC_TRANSACTION_OVERFLOW Exceeded maximum amount of entries in indirect queue.

MAC_TX_ACTIVE Transmission in progress.

MAC_UNAVAILABLE_KEY Security key unavailable.

MAC_UNSUPPORTED_ATTRIBUTE Requested PIB attribute is not supported.

MAC_UNSUPPORTED_LEGACY 802.15.4 2003 security on frame, but not supported by device

MAC_UNSUPPORTED_SECURITY Security on received frame is not supported.

4.13.5.2 enum zb_mac_pib_attr_t

Mac PIB attributes.

4.14 Security subsystem API

Functions

- void **zb_secur_setup_preconfigured_key** (**zb_uint8_t** ∗key, **zb_uint8_t** i)

  Setup pre-configured key to be used by ZCP tests.
- void **zb_secur_send_nwk_key_update_br** (**zb_uint8_t** param) ZB_CALLBACK

  Send new network key to all devices in the net via broadcast.
- void **zb_secur_send_nwk_key_switch** (**zb_uint8_t** param) ZB_CALLBACK

  Generate switch key.
- void **secur_clear_preconfigured_key** ()

  Clear preconfigures key (key number 0)

4.14.1 Detailed Description

4.14.2 Function Documentation

4.14.2.1 void zb_secur_setup_preconfigured_key (zb_uint8_t ∗ *key,* zb_uint8_t *i*)

Setup pre-configured key to be used by ZCP tests.

Parameters

key	- key to be used
i	- key number (0-3)

4.14.2.2 void zb_secur_send_nwk_key_update_br (zb_uint8_t *param*)

Send new network key to all devices in the net via broadcast.

4.6.3.4 Network Key Update 4.6.3.4.1 Trust Center Operation

Parameters

param	- buffer with single parameter - short broadcast address. Valid values are 0xffff, 0xfffd

4.14.2.3 void zb_secur_send_nwk_key_switch (zb_uint8_t *param*)

Generate switch key.

According to test 14.24TP/SEC/BV-01-I Security NWK Key Switch (No Pre- configured Key)-ZR, this command can be send either to broadcast or unicast to all rx-on-when-idle from the neighbor. When send unicast, it encrypted by the new (!) key, when send proadcast - by the old key. That mean, switch our key *after* this frame transfer and secure - in the command send confirm.

Parameters

param	- packet buffer with single parameter - broadcast address. If 0, send unicast.

4.14.2.4 void secur_clear_preconfigured_key ( )

Clear preconfigures key (key number 0)

4.15 Low level API

Modules

- **Compile-time configuration parameters**
- **Base typedefs**
- **Packet buffers pool**
- **Scheduler**
- **Time**
- **Debug trace**

4.15.1 Detailed Description

4.16 Compile-time configuration parameters

Macros

- #define **NO_NVRAM**

  Define to let us work properly with Ember stack.
- #define **ZB_INIT_HAS_ARGS**

  Some additional run-time checks.
- #define **ZB_SECURITY**

  Check arrays to be verified by valgring.
- #define **ZB_TRAFFIC_DUMP_ON**

  If defined, switch on traffic dump.
- #define **UNIX**
- #define **LINUX**
- #define **ZB_WORD_SIZE_4**

  In Linux work size 4 bytes, at 8051 1 byte.
- #define **ZB_LITTLE_ENDIAN**

  If defined, we run on little-endian machine.
- #define **ZB_TRANSPORT_LINUX_PIPES**

  If defined, transport is named pipes in Linux.
- #define **ZB_LINUX_PIPE_TRANSPORT_TIMEOUT** 1

  Linux named pipes transport timeout: wait in select() for this number of seconds.
- #define **ZB_NS_BUILD**

  If defined, this is special build to work with ns-3 network simulator.
- #define **ZB_MANUAL_ACK**

  If defined (for NS build), ack is sent and checked manually.
- #define **ZB_UDP_PORT_REAL** 9998

  Port to be used for zb-over-udp when converting traffic dump into .pcap for WireShark.
- #define **ZB_UDP_PORT_NS** 9999

  Port to be used for zb-over-udp when converting traffic dump into .pcap for WireShark.
- #define **ZB_COORDINATOR_ROLE**

  If defined, ZC functionality is compiled Implies ZR role as well.
- #define **ZB_STACK_PROFILE** 1

  Stack profile constant 1 means 2007, 2 means PRO, 0 means network select.
- #define **ZB_STACK_PROFILE_2007**

  If defined, 2007 stack profile is implemented.
- #define **ZB_PROTOCOL_VERSION** 2

  Protocol version: table 1.1 - current (2006 compatible)
- #define **ZB_SCHEDULER_Q_SIZE** 16

  Scheduler callbacks queue size.
- #define **ZB_MAC_QUEUE_SIZE** 4
- #define **ZB_BUF_Q_SIZE** 16

  Size of queue for wait for free packet buffer.
- #define **ZB_IO_BUF_SIZE** 148

  Size, in bytes, of the packet buffer.
- #define **ZB_IOBUF_POOL_SIZE** 16

  Number of packet buffers.
- #define **ZB_MAC_MAX_REQUESTS** 10

  MAC transaction queue size.
- #define **ZB_DEBUG_ENLARGE_TIMEOUT** 1
- #define **ZB_MAC_RESPONSE_WAIT_TIME** 64

MAC: max time to wait for a response command frame, range 2-64 Default is 32, 64 set for better compatibility.

- #define **ZB_MAX_FRAME_TOTAL_WAIT_TIME** 800

  MAC: max time to wait for indirect data.
- #define **ZB_MAC_MAX_FRAME_RETRIES** 3

  MAC: The maximum number of retries allowed after a transmission failure 0-7.
- #define **ZB_APS_DUP_CHECK_TIMEOUT** **ZB_MILLISECONDS_TO_BEACON_INTERVAL**(1000)

  APS: dup check timeout.
- #define **ZB_APS_POLL_AFTER_REQ_TMO** **ZB_MILLISECONDS_TO_BEACON_INTERVAL**(200)

  After send APS packet, if waiting for ACK, call POLL after this timeout.
- #define **ZB_APS_SRC_BINDING_TABLE_SIZE** 32

  APS: SRC binding tble size.
- #define **ZB_APS_DST_BINDING_TABLE_SIZE** 32

  APS: DST binding tble size.
- #define **ZB_APS_GROUP_TABLE_SIZE** 16

  APS: man number of groups in the system.
- #define **ZB_APS_ENDPOINTS_IN_GROUP_TABLE** 8

  APS: max number of endpoints per group table entry.
- #define **ZB_APS_GROUP_UP_Q_SIZE** 8

  APS: size of queue to be used to pass incoming group addresses packets up.
- #define **ZB_APS_RETRANS_ACK_Q_SIZE** 4

  APS: size of the APS queue of buffers waiting for sending ACK from our side.
- #define **ZB_N_APS_RETRANS_ENTRIES** 10

  APS retransmissions.
- #define **ZB_N_APS_MAX_FRAME_ENTRIES** 3

  APS maximum of apscMaxFrameRetries times.
- #define **ZB_N_APS_ACK_WAIT_DURATION** 2∗**ZB_ZDO_INDIRECT_POLL_TIMER**

  APS: APS ACK wait time.
- #define **ZB_IEEE_ADDR_TABLE_SIZE** 101

  NWK: size of the long-short address translation table.
- #define **ZB_NEIGHBOR_TABLE_SIZE** 32

  NWK: size of the neighbor table.
- #define **ZB_PANID_TABLE_SIZE** 8

  NWK: size os the long-short panid translation table.
- #define **ZB_NWK_DISTRIBUTED_ADDRESS_ASSIGN**

  NWK: If defined, use distributed address assing for tree and for mesh routing (ZigBee 2007).
- #define **ZB_NWK_ROUTING**

  NWK: If defined, enable routing functionality.
- #define **N_SECUR_MATERIAL** 3

  Number of secure materials to store.
- #define **ZB_NWK_TREE_ROUTING**

  NWK: if defined, implement tree routing.
- #define **ZB_NWK_MESH_ROUTING**
- #define **ZB_NWK_MAX_CHILDREN** 4

  NWK: if defined, implement mesh routing.
- #define **ZB_NWK_MAX_ROUTERS** 4

  NWK: Max number of routers per node.
- #define **ZB_NWK_MAX_DEPTH** 5

  NWK: max network depth.
- #define **ZB_NWK_ROUTING_TABLE_SIZE** 5

  NWK Mesh route stuff: routing table size.
- #define **ZB_NWK_ROUTE_DISCOVERY_TABLE_SIZE** 5

NWK Mesh route stuff: route discovery table size.

- #define **ZB_NWK_EXPIRY_ROUTE_DISCOVERY** 2∗**ZB_TIME_ONE_SECOND**
- #define **ZB_NWK_ROUTE_DISCOVERY_EXPIRY** 10
- #define **ZB_MWK_INITIAL_RREQ_RETRIES** 3
- #define **ZB_MWK_RREQ_RETRIES** 2
- #define **ZB_NWK_PENDING_TABLE_SIZE** 5
- #define **ZB_NWK_PENDING_ENTRY_EXPIRY** 20
- #define **ZB_NWK_STATIC_PATH_COST** 7
- #define **ZB_NWK_BTR_TABLE_SIZE** 16
- #define **ZB_NWK_BRR_TABLE_SIZE** 8
- #define **ZB_NWK_WAIT_ALLOC_TABLE_SIZE** 5
- #define **ZB_NWK_MAX_BROADCAST_JITTER_INTERVAL ZB_MILLISECONDS_TO_BEACON_INTERVAL**(0x40)
- #define **ZB_NWK_RREQ_RETRY_INTERVAL ZB_MILLISECONDS_TO_BEACON_INTERVAL**(0xFE)
- #define **ZB_NWK_EXPIRY_PENDING** 5∗**ZB_TIME_ONE_SECOND**
- #define **ZB_NWK_MAX_BROADCAST_JITTER** 0x40∗ZB_TIME_ONE_SECOND
- #define **ZB_NWK_MAX_BROADCAST_RETRIES** 0x02
- #define **ZB_NWK_PASSIVE_ACK_TIMEOUT** 100
- #define **ZB_NWK_REJOIN_REQUEST_TABLE_SIZE** 3

  Maximum number of rejoin requests in progress.
- #define **ZB_NWK_REJOIN_TIMEOUT** ZB_MAC_PIB_RESPONSE_WAIT_TIME ∗ 5
- #define **ZB_DEFAULT_SCAN_DURATION** 3

  NWK: default energy/active scan duration.
- #define **ZB_TRANSCEIVER_ALL_CHANNELS_MASK** 0x07FFF800 /∗ 0000.0111 1111.1111 1111.1000 0000.0000∗/
- #define **ZB_DEFAULT_APS_CHANNEL_MASK** ((1l<<11)|(1l<<12))
- #define **ZB_DEFAULT_PRMIT_JOINING_DURATION** 0xff

  Default duration to permit joining (currently infinite)
- #define **ZB_DEFAULT_MAX_CHILDREN** 32

  Default value of nib.max_children - max number of children which can join to this device.
- #define **ZB_APS_COMMAND_RADIUS** 5

  NWK radius to be used when sending APS command.
- #define **ZB_STANDARD_SECURITY**

  SECUR: if defined, implement Standard security.
- #define **ZB_TC_GENERATES_KEYS**

  SECUR: If defined, generate random keys at Trust Center at start of pre-configured jey is not set.
- #define **ZB_TC_AT_ZC**

  SECUR: If defined, trust Center is at ZC (currently - always)
- #define **ZB_CCM_KEY_SIZE** 16

  SECUR: CCM key size.
- #define **ZB_SECUR_N_SECUR_MATERIAL** 3
- #define **ZB_SECURITY_LEVEL** 5

  SECUR: security level.
- #define **ZB_CCM_L** 2

  SECUR: CCM L parameter.
- #define **ZB_CCM_NONCE_LEN** 13

  SECUR: CCM nonce length.
- #define **ZB_CCM_M** 4

  SECUR: CCM M parameter.
- #define **ZB_SECUR_NWK_COUNTER_LIMIT** (((**zb_uint32_t**)∼0) - 128)

  Value of nwk packets counter which triggered nwk key switch.
- #define **ZB_DEFAULT_SECURE_ALL_FRAMES** 1

Default value for nib.secure_all_frames.

- #define **ZB_ZCL_CLUSTER_NUM** 8

  Maximum number of ZCL clusters.
- #define **ZB_ZDO_INDIRECT_POLL_TIMER** (5∗**ZB_TIME_ONE_SECOND**) /∗ **ZB_TIME_ONE_SECOND**∗10 ∗/

  ZDO Indirect poll timer.
- #define **ZB_ZDO_MAX_PARENT_THRESHOLD_RETRY** 10

  ZDO Max parent threshold retry.
- #define **ZB_ZDO_MIN_SCAN_DURATION** 0

  Min scan duration for mgmt_nwk_update_req.
- #define **ZB_ZDO_MAX_SCAN_DURATION** 5

  Max scan duration for mgmt_nwk_update_req.
- #define **ZB_ZDO_NEW_ACTIVE_CHANNEL** 0xFE

  Special value of the scan duration for mgmt_nwk_update_req: change active channel (by number)
- #define **ZB_ZDO_NEW_CHANNEL_MASK** 0xFF

  Special value of the scan duration for mgmt_nwk_update_req: change channels mask.
- #define **ZB_ZDO_CHANNEL_CHECK_TIMEOUT** (**ZB_TIME_ONE_SECOND** ∗ 60 ∗ 15)

  15 minutes timeout.
- #define **ZB_ZDO_APS_CHANEL_TIMER** (1 ∗ 60)

  A countdown timer (in minutes) indicating the time to the next permitted frequency agility channel change.
- #define **ZB_ZDO_15_MIN_TIMEOUT** (**ZB_TIME_ONE_SECOND** ∗ 60 ∗ 15)

  15 minutes timer to measure large timeouts
- #define **ZB_ZDO_1_MIN_TIMEOUT** (**ZB_TIME_ONE_SECOND** ∗ 60)

  1 minute timer to measure large timeouts
- #define **ZB_ZDO_NWK_SCAN_ATTEMPTS** 1

  Integer value representing the number of scan attempts to make before the NWK layer decides which ZigBee coordinator or router to associate with.
- #define **ZB_NWK_ONE_SCAN_ATTEMPT**
- #define **ZB_ZDO_NWK_TIME_BTWN_SCANS** 30

  Integer value representing the time duration (in milliseconds)
- #define **ZB_ZDO_ENDDEV_BIND_TIMEOUT** 30

  Timeout value in seconds employed in End Device Binding.
- #define **ZDO_TRAN_TABLE_SIZE** 16

  ZDO: transactions table size.
- #define **ZB_ZDO_PENDING_LEAVE_SIZE** 4

  Number of pending Mgmt_Leave requests allowed.
- #define **ZB_ZDO_PARENT_LINK_FAILURE_CNT** 12

  This define turns on/off test profile.
- #define **ZB_PREDEFINED_ROUTER_ADDR** 0x3344
- #define **ZB_PREDEFINED_ED_ADDR** 0x3344
- #define **ZB_DISTURBER_PANID** 0x0bad

4.16.1 Detailed Description

4.16.2 Macro Definition Documentation

4.16.2.1 #define NO_NVRAM

Define to let us work properly with Ember stack.

If defined, NVRAM not compiled

To be used near always to prevent flash damage (flash can do ∼1000 rewrites only)

4.16.2.2 #define ZB_INIT_HAS_ARGS

Some additional run-time checks.

Check arrays to be verified by valgring. Useful for Linux/PC build only. Slows down execution.

4.16.2.3 #define ZB_SECURITY

Check arrays to be verified by valgring.

If defined, security is compiled

4.16.2.4 #define ZB_TRAFFIC_DUMP_ON

If defined, switch on traffic dump.

4.16.2.5 #define ZB_WORD_SIZE_4

In Linux work size 4 bytes, at 8051 1 byte.

4.16.2.6 #define ZB_LITTLE_ENDIAN

If defined, we run on little-endian machine.

4.16.2.7 #define ZB_TRANSPORT_LINUX_PIPES

If defined, transport is named pipes in Linux.

4.16.2.8 #define ZB_LINUX_PIPE_TRANSPORT_TIMEOUT 1

Linux named pipes transport timeout: wait in select() for this number of seconds.

4.16.2.9 #define ZB_NS_BUILD

If defined, this is special build to work with ns-3 network simulator.

4.16.2.10 #define ZB_MANUAL_ACK

If defined (for NS build), ack is sent and checked manually.

4.16.2.11 #define ZB_UDP_PORT_REAL 9998

Port to be used for zb-over-udp when converting traffic dump into .pcap for WireShark.

This is for real transiver case - that is, dump contains all transiver registers access.

4.16.2.12 #define ZB_UDP_PORT_NS 9999

Port to be used for zb-over-udp when converting traffic dump into .pcap for WireShark.

This is for ns-3 build case - that is, dump contains MAC packets.

4.16.2.13 #define ZB_COORDINATOR_ROLE

If defined, ZC functionality is compiled Implies ZR role as well.

4.16.2.14 #define ZB_STACK_PROFILE 1

Stack profile constant 1 means 2007, 2 means PRO, 0 means network select.

4.16.2.15 #define ZB_STACK_PROFILE_2007

If defined, 2007 stack profile is implemented.

4.16.2.16 #define ZB_PROTOCOL_VERSION 2

Protocol version: table 1.1 - current (2006 compatible)

4.16.2.17 #define ZB_SCHEDULER_Q_SIZE 16

Scheduler callbacks queue size.

Ususlly not need to change it.

4.16.2.18 #define ZB_BUF_Q_SIZE 16

Size of queue for wait for free packet buffer.

4.16.2.19 #define ZB_IO_BUF_SIZE 148

Size, in bytes, of the packet buffer.

Be sure keep it multiple of 4 to exclude alignment problems at ARM

4.16.2.20 #define ZB_IOBUF_POOL_SIZE 16

Number of packet buffers.

More buffers - more memory. Less buffers - risk to be blocked due to buffer absence.

4.16.2.21 #define ZB_MAC_MAX_REQUESTS 10

MAC transaction queue size.

4.16.2.22 #define ZB_MAC_RESPONSE_WAIT_TIME 64

MAC: max time to wait for a response command frame, range 2-64 Default is 32, 64 set for better compatibility.

4.16.2.23 #define ZB_MAX_FRAME_TOTAL_WAIT_TIME 800

MAC: max time to wait for indirect data.

4.16.2.24 #define ZB_MAC_MAX_FRAME_RETRIES 3

MAC: The maximum number of retries allowed after a transmission failure 0-7.

4.16.2.25 #define ZB_APS_DUP_CHECK_TIMEOUT ZB_MILLISECONDS_TO_BEACON_INTERVAL(1000)

APS: dup check timeout.

APS dup checks resolution is 1s, timer entry size in the address translation table is 2b, so dup timeout is 4s.

4.16.2.26 #define ZB_APS_POLL_AFTER_REQ_TMO ZB_MILLISECONDS_TO_BEACON_INTERVAL(200)

After send APS packet, if waiting for ACK, call POLL after this timeout.

4.16.2.27 #define ZB_APS_SRC_BINDING_TABLE_SIZE 32

APS: SRC binding tble size.

4.16.2.28 #define ZB_APS_DST_BINDING_TABLE_SIZE 32

APS: DST binding tble size.

4.16.2.29 #define ZB_APS_GROUP_TABLE_SIZE 16

APS: man number of groups in the system.

4.16.2.30 #define ZB_APS_ENDPOINTS_IN_GROUP_TABLE 8

APS: max number of endpoints per group table entry.

4.16.2.31 #define ZB_APS_GROUP_UP_Q_SIZE 8

APS: size of queue to be used to pass incoming group addresses packets up.

4.16.2.32 #define ZB_APS_RETRANS_ACK_Q_SIZE 4

APS: size of the APS queue of buffers waiting for sending ACK from our side.

4.16.2.33 #define ZB_N_APS_RETRANS_ENTRIES 10

APS retransmissions.

APS: max number of packets waiting for APS ACK

4.16.2.34 #define ZB_N_APS_MAX_FRAME_ENTRIES 3

APS maximum of apscMaxFrameRetries times.

4.16.2.35 #define ZB_N_APS_ACK_WAIT_DURATION 2∗ZB_ZDO_INDIRECT_POLL_TIMER

APS: APS ACK wait time.

After this timeout resend APS packet

4.16.2.36 #define ZB_IEEE_ADDR_TABLE_SIZE 101

NWK: size of the long-short address translation table.

4.16.2.37 #define ZB_NEIGHBOR_TABLE_SIZE 32

NWK: size of the neighbor table.

4.16.2.38 #define ZB_PANID_TABLE_SIZE 8

NWK: size os the long-short panid translation table.

4.16.2.39 #define ZB_NWK_DISTRIBUTED_ADDRESS_ASSIGN

NWK: If defined, use distributed address assing for tree and for mesh routing (ZigBee 2007).

4.16.2.40 #define ZB_NWK_ROUTING

NWK: If defined, enable routing functionality.

4.16.2.41 #define N_SECUR_MATERIAL 3

Number of secure materials to store.

4.16.2.42 #define ZB_NWK_TREE_ROUTING

NWK: if defined, implement tree routing.

4.16.2.43 #define ZB_NWK_MAX_CHILDREN 4

NWK: if defined, implement mesh routing.

NWK: Max number of children per node

4.16.2.44 #define ZB_NWK_MAX_ROUTERS 4

NWK: Max number of routers per node.

4.16.2.45 #define ZB_NWK_MAX_DEPTH 5

NWK: max network depth.

4.16.2.46 #define ZB_NWK_ROUTING_TABLE_SIZE 5

NWK Mesh route stuff: routing table size.

4.16.2.47 #define ZB_NWK_ROUTE_DISCOVERY_TABLE_SIZE 5

NWK Mesh route stuff: route discovery table size.

4.16.2.48 #define ZB_NWK_REJOIN_REQUEST_TABLE_SIZE 3

Maximum number of rejoin requests in progress.

4.16.2.49 #define ZB_DEFAULT_SCAN_DURATION 3

NWK: default energy/active scan duration.

4.16.2.50 #define ZB_DEFAULT_PRMIT_JOINING_DURATION 0xff

Default duration to permit joining (currently infinite)

4.16.2.51 #define ZB_DEFAULT_MAX_CHILDREN 32

Default value of nib.max_children - max number of children which can join to this device.

4.16.2.52 #define ZB_APS_COMMAND_RADIUS 5

NWK radius to be used when sending APS command.

4.16.2.53 #define ZB_STANDARD_SECURITY

SECUR: if defined, implement Standard security.

4.16.2.54 #define ZB_TC_GENERATES_KEYS

SECUR: If defined, generate random keys at Trust Center at start of pre-configured jey is not set.

4.16.2.55 #define ZB_TC_AT_ZC

SECUR: If defined, trust Center is at ZC (currently - always)

4.16.2.56 #define ZB_CCM_KEY_SIZE 16

SECUR: CCM key size.

Hard-coded

4.16.2.57 #define ZB_SECURITY_LEVEL 5

SECUR: security level.

Now fixed to be 5

4.16.2.58 #define ZB_CCM_L 2

SECUR: CCM L parameter.

Fixed to 2 for security level 5

4.16.2.59 #define ZB_CCM_NONCE_LEN 13

SECUR: CCM nonce length.

Now fixed.

4.16.2.60 #define ZB_CCM_M 4

SECUR: CCM M parameter.

Fixed to 4 for security level 5

4.16.2.61 #define ZB_SECUR_NWK_COUNTER_LIMIT (((zb_uint32_t)∼0) - 128)

Value of nwk packets counter which triggered nwk key switch.

4.16.2.62 #define ZB_DEFAULT_SECURE_ALL_FRAMES 1

Default value for nib.secure_all_frames.

4.16.2.63 #define ZB_ZCL_CLUSTER_NUM 8

Maximum number of ZCL clusters.

4.16.2.64 #define ZB_ZDO_INDIRECT_POLL_TIMER (5∗ZB_TIME_ONE_SECOND) /∗ ZB_TIME_ONE_SECOND∗10 ∗/

ZDO Indirect poll timer.

4.16.2.65 #define ZB_ZDO_MAX_PARENT_THRESHOLD_RETRY 10

ZDO Max parent threshold retry.

4.16.2.66 #define ZB_ZDO_MIN_SCAN_DURATION 0

Min scan duration for mgmt_nwk_update_req.

4.16.2.67 #define ZB_ZDO_MAX_SCAN_DURATION 5

Max scan duration for mgmt_nwk_update_req.

4.16.2.68 #define ZB_ZDO_NEW_ACTIVE_CHANNEL 0xFE

Special value of the scan duration for mgmt_nwk_update_req: change active channel (by number)

4.16.2.69 #define ZB_ZDO_NEW_CHANNEL_MASK 0xFF

Special value of the scan duration for mgmt_nwk_update_req: change channels mask.

4.16.2.70 #define ZB_ZDO_CHANNEL_CHECK_TIMEOUT (ZB_TIME_ONE_SECOND * 60 * 15)

15 minutes timeout.

KLUDGE: it is 2 bytes value, 15 minutes is nearly maximum value that can be stored

4.16.2.71 #define ZB_ZDO_APS_CHANEL_TIMER (1 * 60)

A countdown timer (in minutes) indicating the time to the next permitted frequency agility channel change.

4.16.2.72 #define ZB_ZDO_15_MIN_TIMEOUT (ZB_TIME_ONE_SECOND * 60 * 15)

15 minutes timer to measure large timeouts

4.16.2.73 #define ZB_ZDO_1_MIN_TIMEOUT (ZB_TIME_ONE_SECOND * 60)

1 minute timer to measure large timeouts

4.16.2.74 #define ZB_ZDO_NWK_SCAN_ATTEMPTS 1

Integer value representing the number of scan attempts to make before the NWK layer decides which ZigBee coordinator or router to associate with.

4.16.2.75 #define ZB_ZDO_NWK_TIME_BTWN_SCANS 30

Integer value representing the time duration (in milliseconds)

4.16.2.76 #define ZB_ZDO_ENDDEV_BIND_TIMEOUT 30

Timeout value in seconds employed in End Device Binding.

4.16.2.77 #define ZDO_TRAN_TABLE_SIZE 16

ZDO: transactions table size.

4.16.2.78 #define ZB_ZDO_PENDING_LEAVE_SIZE 4

Number of pending Mgmt_Leave requests allowed.

4.16.2.79 #define ZB_ZDO_PARENT_LINK_FAILURE_CNT 12

This define turns on/off test profile.

- This define is for APS retransmissions test, do not use it for the normal work Comp[ile Test Profile feature This difine turnes on/off channel error mode (set errors while data sending) Number of times device failes to send packet to the parent before rejoin

4.17 Base typedefs

Functions

- void **zb_htole32** (**zb_uint32_t** ZB_XDATA ∗ptr, **zb_uint32_t** ZB_XDATA ∗val)
- void **zb_put_next_htole16** (**zb_uint8_t** ∗∗dst, **zb_uint16_t** val)

  Put next 2-bute value into buffer, move pointer.
- void **zb_get_next_letoh16** (**zb_uint16_t** ∗dst, **zb_uint8_t** ∗∗src)

Data Structures

- union **zb_addr_u**

  Union to address either long or short address.

Macros

- #define **ZB_32BIT_WORD**
- #define **ZB_XDATA**
- #define **ZB_CODE**
- #define **ZB_IAR_CODE** code
- #define **ZB_REGISTER**
- #define **ZB_VOID_ARGLIST** void
- #define **ZB_CONST** const
- #define **ZB_INLINE**
- #define **ZB_BITFIELD_CAST**(x) (x)
- #define **ZB_INT8_MIN** (-127 - 1)
- #define **ZB_INT8_MAX** 127
- #define **ZB_UINT8_MIN** 0
- #define **ZB_UINT8_MAX** 255
- #define **ZB_INT16_MIN** (-32767 - 1)
- #define **ZB_INT16_MAX** 32767
- #define **ZB_UINT16_MIN** 0
- #define **ZB_UINT16_MAX** 65535
- #define **ZB_INT32_MIN** (-2147483647L - 1)
- #define **ZB_INT32_MAX** 2147483647L
- #define **ZB_UINT32_MIN** 0UL
- #define **ZB_UINT32_MAX** 4294967295UL
- #define **ZB_UINT_MIN** 0UL
- #define **ZB_SHORT_MIN** ZB_INT32_MIN

  Max value constants per type.
- #define **ZB_SHORT_MAX** ZB_INT32_MAX
- #define **ZB_USHORT_MAX** ZB_UINT32_MAX
- #define **ZB_INT_MIN** ZB_INT32_MIN
- #define **ZB_INT_MAX** ZB_INT32_MAX
- #define **ZB_UINT_MAX** ZB_UINT32_MAX
- #define **ZB_INT_MASK** 0x7fffffff
- #define **ZB_IS_64BIT_ADDR_ZERO**(addr) (!ZB_MEMCMP((addr), g_zero_addr, 8))

  Return true if long address is zero.
- #define **ZB_64BIT_ADDR_ZERO**(addr) ZB_MEMSET((addr), 0, 8)

  Clear long address.
- #define **ZB_64BIT_ADDR_COPY**(dst, src) ZB_MEMCPY(dst, src, sizeof(**zb_64bit_addr_t**))

  Copy long address.

- #define **ZB_64BIT_ADDR_CMP**(one, two) ((**zb_bool_t**)!ZB_MEMCMP((one), (two), 8))

  Return 1 if long addresses are equal.
- #define **ZB_EXTPANID_IS_ZERO ZB_IS_64BIT_ADDR_ZERO**
- #define **ZB_EXTPANID_ZERO ZB_64BIT_ADDR_ZERO**
- #define **ZB_EXTPANID_COPY ZB_64BIT_ADDR_COPY**
- #define **ZB_EXTPANID_CMP ZB_64BIT_ADDR_CMP**
- #define **ZB_IEEE_ADDR_IS_ZERO ZB_IS_64BIT_ADDR_ZERO**
- #define **ZB_IEEE_ADDR_ZERO ZB_64BIT_ADDR_ZERO**
- #define **ZB_IEEE_ADDR_COPY ZB_64BIT_ADDR_COPY**
- #define **ZB_IEEE_ADDR_CMP ZB_64BIT_ADDR_CMP**
- #define **ZB_ADDR_CMP**(addr_mode, addr1, addr2)
- #define **ZB_INT8_C**(c) c

  definitions for constants of given type
- #define **ZB_UINT8_C**(c) c ## U
- #define **ZB_INT16_C**(c) c
- #define **ZB_UINT16_C**(c) c ## U
- #define **ZB_INT32_C**(c) c ## L
- #define **ZB_UINT32_C**(c) c ## UL
- #define **ZB_OFFSETOF**(t, f) (zb_size_t)(&((t ∗)0)->f)
- #define **ZB_OFFSETOF_VAR**(s, f) (zb_size_t)(((**zb_int8_t** ∗)(&(s)->f)) - ((**zb_int8_t** ∗)(s)))
- #define **ZB_SIZEOF_FIELD**(type, field) (sizeof(((type∗)0)->field))
- #define **ZB_ARRAY_SIZE**(arr) (sizeof((arr))/sizeof((arr)[0]))
- #define **ZB_SIGNED_SHIFT**(v, s) ((**zb_int_t**)(v) >> (s))
- #define **ZB_PACKED_STRUCT**
- #define **ZB_HTOLE16**(ptr, val)
- #define **ZB_HTOLE32**(ptr, val) zb_htole32((**zb_uint32_t**∗)(ptr), (**zb_uint32_t**∗)(val))
- #define **ZB_HTOBE16**(ptr, val) (∗(**zb_uint16_t** ∗)(ptr)) = ∗((**zb_uint16_t** ∗)(val))
- #define **ZB_HTOBE16_VAL**(ptr, val) ((**zb_uint16_t** ∗)(ptr))[0] = (val)
- #define **ZB_HTOLE64**(ptr, val) ZB_MEMCPY((ptr), (val), 8)
- #define **ZB_LETOH64** ZB_HTOLE64
- #define **ZB_LETOH16 ZB_HTOLE16**

  Convert 16-bits integer from the little endian to the host endian.
- #define **ZB_LETOH32** ZB_HTOLE32
- #define **ZB_BETOH16** ZB_HTOBE16
- #define **ZB_GET_LOW_BYTE**(val) ((val) & 0xFF)
- #define **ZB_GET_HI_BYTE**(val) (((val) >> 8) & 0xFF)
- #define **ZB_PKT_16B_ZERO_BYTE** 0
- #define **ZB_PKT_16B_FIRST_BYTE** 1

Typedefs

- typedef enum **zb_bool_e zb_bool_t**

  General purpose boolean type.
- typedef char **zb_char_t**

  project-local char type
- typedef unsigned char **zb_uchar_t**

  project-local unsigned char type
- typedef unsigned char **zb_uint8_t**

  project-local 1-byte unsigned int type
- typedef signed char **zb_int8_t**

  project-local 1-byte signed int type
- typedef unsigned short **zb_uint16_t**

  project-local 2-byte unsigned int type

- typedef signed short **zb_int16_t**

  project-local 2-byte signed int type
- typedef unsigned int **zb_uint32_t**

  project-local 4-byte unsigned int type
- typedef signed int **zb_int32_t**

  project-local 4-byte signed int type
- typedef **zb_uint32_t** **zb_bitfield_t**

  type to be used for unsigned bit fields inside structure
- typedef **zb_int32_t** **zb_sbitfield_t**

  type to be used for signed bit fields inside structure
- typedef int **zb_short_t**

  short int (can fit into single CPU register)
- typedef unsigned int **zb_ushort_t**

  unsigned short int (can fit into single CPU register)
- typedef int **zb_int_t**

  int (at least 2 bytes)
- typedef unsigned int **zb_uint_t**

  unsigned int (at least 2 bytes)
- typedef **zb_int_t** **zb_long_t**

  long int (at least 4 bytes)
- typedef **zb_uint_t** **zb_ulong_t**

  unsigned long int (at least 4 bytes)
- typedef void ∗ **zb_voidp_t**

  ptr to void
- typedef void **zb_void_t**
- typedef **zb_uint8_t** **zb_64bit_addr_t** [8]

  8-bytes address (xpanid or long device address) base type
- typedef **zb_64bit_addr_t** **zb_ieee_addr_t**

  Long (64-bit) device address.
- typedef **zb_64bit_addr_t** **zb_ext_pan_id_t**

  Long (64-bit) Extented pan id.

Enumerations

- enum **zb_bool_e** { **ZB_FALSE** = 0, **ZB_TRUE** = 1 }

  General purpose boolean type.

Variables

- **zb_64bit_addr_t g_zero_addr**

4.17.1 Detailed Description

4.17.2 Function Documentation

4.17.2.1 void zb_put_next_htole16 (zb_uint8_t ∗∗ *dst,* zb_uint16_t *val*)

Put next 2-bute value into buffer, move pointer.

To be used for headers compose.

Parameters

dst	- (in/out) address os the buffer pointer As a side effect it will be incremented by 2.

4.17.3 Macro Definition Documentation

4.17.3.1 #define ZB_SHORT_MIN ZB_INT32_MIN

Max value constants per type.

4.17.3.2 #define ZB_IS_64BIT_ADDR_ZERO(*addr*) (!ZB_MEMCMP((addr), g_zero_addr, 8))

Return true if long address is zero.

4.17.3.3 #define ZB_64BIT_ADDR_ZERO(*addr*) ZB_MEMSET((addr), 0, 8)

Clear long address.

4.17.3.4 #define ZB_64BIT_ADDR_COPY(*dst, src*) ZB_MEMCPY(dst, src, sizeof(zb_64bit_addr_t))

Copy long address.

4.17.3.5 #define ZB_64BIT_ADDR_CMP(*one, two*) ((zb_bool_t)!ZB_MEMCMP((one), (two), 8))

Return 1 if long addresses are equal.

4.17.3.6 #define ZB_ADDR_CMP(*addr_mode, addr1, addr2*)

Value:

```
((addr_mode == ZB_ADDR_16BIT_DEV_OR_BROADCAST) ?                         \
   (addr1.addr_short == addr2.addr_short) : ZB_64BIT_ADDR_CMP(addr1.addr_long,
      addr2.addr_long))
```

4.17.3.7 #define ZB_INT8_C(*c*) c

definitions for constants of given type

4.17.3.8 #define ZB_HTOLE16(*ptr, val*)

Value:

```
(((zb_uint8_t *)(ptr))[0] = ((zb_uint8_t *)(val))[1], \
   ((zb_uint8_t *)(ptr))[1] = ((zb_uint8_t *)(val))[0]  \
  )
```

macros to change words endian and access words at potentially

non-aligned pointers.

ZigBee uses little endian - see 1.2.1.3.

4.17.3.9 #define ZB_LETOH16 ZB_HTOLE16

Convert 16-bits integer from the little endian to the host endian.

Parameters

ptr	- destination pointer. It is ok if it not aligned to 2.
val	- source pointer. It is ok if it not aligned to 2.

4.17.4 Typedef Documentation

4.17.4.1 typedef enum zb_bool_e zb_bool_t

General purpose boolean type.

4.17.4.2 typedef char zb_char_t

project-local char type

4.17.4.3 typedef unsigned char zb_uchar_t

project-local unsigned char type

4.17.4.4 typedef unsigned char zb_uint8_t

project-local 1-byte unsigned int type

4.17.4.5 typedef signed char zb_int8_t

project-local 1-byte signed int type

4.17.4.6 typedef unsigned short zb_uint16_t

project-local 2-byte unsigned int type

4.17.4.7 typedef signed short zb_int16_t

project-local 2-byte signed int type

4.17.4.8 typedef unsigned int zb_uint32_t

project-local 4-byte unsigned int type

4.17.4.9 typedef signed int zb_int32_t

project-local 4-byte signed int type

4.17.4.10 typedef zb_uint32_t zb_bitfield_t

type to be used for unsigned bit fields inside structure

4.17.4.11 typedef zb_int32_t zb_sbitfield_t

type to be used for signed bit fields inside structure

4.17.4.12 typedef int zb_short_t

short int (can fit into single CPU register)

4.17.4.13 typedef unsigned int zb_ushort_t

unsigned short int (can fit into single CPU register)

4.17.4.14 typedef int zb_int_t

int (at least 2 bytes)

4.17.4.15 typedef unsigned int zb_uint_t

unsigned int (at least 2 bytes)

4.17.4.16 typedef zb_int_t zb_long_t

long int (at least 4 bytes)

4.17.4.17 typedef zb_uint_t zb_ulong_t

unsigned long int (at least 4 bytes)

4.17.4.18 typedef void* zb_voidp_t

ptr to void

4.17.4.19 typedef zb_uint8_t zb_64bit_addr_t[8]

8-bytes address (xpanid or long device address) base type

4.17.4.20 typedef zb_64bit_addr_t zb_ieee_addr_t

Long (64-bit) device address.

4.17.4.21 typedef zb_64bit_addr_t zb_ext_pan_id_t

Long (64-bit) Extented pan id.

4.17.5 Enumeration Type Documentation

4.17.5.1 enum zb_bool_e

General purpose boolean type.

4.18 Packet buffers pool

Functions

- zb_void_t ∗ **zb_buf_initial_alloc** (zb_buf_t ∗zbbuf, **zb_uint8_t** size)

  Initial allocate space in buffer.
- zb_void_t ∗ **zb_buf_smart_alloc_left** (zb_buf_t ∗zbbuf, **zb_uint8_t** size) ZB_SDCC_REENTRANT
- zb_void_t ∗ **zb_buf_smart_alloc_right** (zb_buf_t ∗zbbuf, **zb_uint8_t** size) ZB_SDCC_REENTRANT
- void ∗ **zb_buf_cut_left** (zb_buf_t ∗zbbuf, **zb_uint8_t** size)
- void **zb_buf_cut_right** (zb_buf_t ∗zbbuf, **zb_uint8_t** size)
- zb_void_t ∗ **zb_get_buf_tail** (zb_buf_t ∗zbbuf, **zb_uint8_t** size)

  Get buffer tail of size 'size'.
- void **zb_buf_assign_param** (zb_buf_t ∗zbbuf, **zb_uint8_t** ∗param, **zb_uint8_t** size) ZB_SDCC_REENTRA-NT

  Copy data to the bufefr tail - assign parameter.
- zb_void_t **zb_buf_reuse** (zb_buf_t ∗zbbuf)

  Reuse previously used buffer.
- void **zb_init_buffers** () ZB_CALLBACK

  Initialize packet buffers pool.
- zb_buf_t ∗ **zb_get_in_buf** ()

  Get IN buffer from the buffers list.
- zb_buf_t ∗ **zb_get_out_buf** ()

  Get OUT buffer from the buffers list.
- void **zb_free_buf** (zb_buf_t ∗buf)

  Free packt buffer.
- zb_ret_t **zb_get_in_buf_delayed** (**zb_callback_t** callback)

  Allocate IN buffer.
- zb_ret_t **zb_get_out_buf_delayed** (**zb_callback_t** callback)

  Allocate OUT buffer.

Data Structures

- struct **zb_buf_hdr_s**

  Packet buffer header.
- struct **zb_buf_s**

  Packet buffer.

Macros

- #define **ZB_UNDEFINED_BUFFER** (**zb_uint8_t**)(-1)
- #define **zb_buf_t** **zb_buf_s_t**
- #define **ZB_IN_BUF_AVAILABLE**() (ZG->bpool.bufs_allocated[1] < **ZB_IOBUF_POOL_SIZE**/2)
- #define **ZB_OUT_BUF_AVAILABLE**() (ZG->bpool.bufs_allocated[0] < **ZB_IOBUF_POOL_SIZE**/2)
- #define **ZB_BUF_BEGIN**(zbbuf) ((zbbuf)->buf + (zbbuf)->u.hdr.data_offset)

  Return current buffer pointer.
- #define **ZB_BUF_LEN**(zbbuf) ((zbbuf)->u.hdr.len)

  Return current buffer length.
- #define **ZB_BUF_OFFSET**(zbbuf) ((zbbuf)->u.hdr.data_offset)

  Return current buffer offset.
- #define **ZB_BUF_INITIAL_ALLOC**(zbbuf, size, ptr) (ptr) = **zb_buf_initial_alloc**((zbbuf), (size))
- #define **ZB_BUF_ALLOC_LEFT**(zbbuf, size, ptr) (ptr) = zb_buf_smart_alloc_left((zbbuf), (size))

Allocate space at buffer begin.

- #define **ZB_BUF_ALLOC_RIGHT**(zbbuf, size, ptr) (ptr) = zb_buf_smart_alloc_right((zbbuf), (size))

  Allocate space at buffer end.
- #define **ZB_BUF_CUT_LEFT**(zbbuf, size, ptr) (ptr) = zb_buf_cut_left((zbbuf), (size))

  Cut space at buffer begin.
- #define **ZB_BUF_CUT_LEFT2**(zbbuf, size)
- #define **ZB_BUF_CUT_RIGHT**(zbbuf, size) zb_buf_cut_right((zbbuf), (size))

  Cut space at buffer end.
- #define **ZB_GET_BUF_TAIL zb_get_buf_tail**
- #define **ZB_GET_BUF_PARAM**(zbbuf, type) ((type ∗)ZB_GET_BUF_TAIL((zbbuf), sizeof(type)))
- #define **ZB_SET_BUF_PARAM**(zbbuf, param, type) (∗((type ∗)ZB_GET_BUF_TAIL(zbbuf, sizeof(type))) = (param))
- #define **ZB_SET_BUF_PARAM_PTR**(zbbuf, param, type) (ZB_MEMCPY((type ∗)ZB_GET_BUF_TAIL(zbbuf, sizeof(type)), (param), sizeof(type)))
- #define **ZB_BUF_COPY**(dst_buf, src_buf)

  Copy one buffer to the other.
- #define **ZB_BUF_REUSE zb_buf_reuse**
- #define **ZB_BUF_GET_FREE_SIZE**(zbbuf) (unsigned)(**ZB_IO_BUF_SIZE** - **ZB_BUF_LEN**(zbbuf))
- #define **ZB_BUF_FROM_REF**(ref) (&ZG->bpool.pool[ref])
- #define **ZB_REF_FROM_BUF**(buf) (buf - &ZG->bpool.pool[0])
- #define **ZB_GET_IN_BUF_DELAYED zb_get_in_buf_delayed**
- #define **ZB_GET_OUT_BUF_DELAYED zb_get_out_buf_delayed**

Typedefs

- typedef struct **zb_buf_hdr_s zb_buf_hdr_t**

  Packet buffer header.
- typedef struct **zb_buf_s zb_buf_s_t**

  Packet buffer.

4.18.1 Detailed Description

4.18.2 Function Documentation

4.18.2.1 zb_void_t∗ zb_buf_initial_alloc (zb_buf_t ∗ *zbbuf,* zb_uint8_t *size*)

Initial allocate space in buffer.

Parameters

zbbuf	- buffer
size	- size to allocate

Returns

pointer to the allocated space

4.18.2.2 zb_void_t∗ zb_get_buf_tail (zb_buf_t ∗ *zbbuf,* zb_uint8_t *size*)

Get buffer tail of size 'size'.

Macro usually used to place external information (some parameters) to the buffer

Parameters

zbbuf	- buffer
size	- requested size

Returns

pointer to the buffer tail

4.18.2.3 void zb_buf_assign_param (zb_buf_t ∗ *zbbuf*, zb_uint8_t ∗ *param*, zb_uint8_t *size*)

Copy data to the bufefr tail - assign parameter.

Take care on space on the buffer tail, move data if necessary.

Parameters

zbbuf	- buffer
param	- data to copy
size	- data size

4.18.2.4 zb_void_t zb_buf_reuse (zb_buf_t ∗ *zbbuf*)

Reuse previously used buffer.

Parameters

zbbuf	- buffer

4.18.2.5 void zb_init_buffers ( )

Initialize packet buffers pool.

To be called at start time.

Returns

nothing

4.18.2.6 zb_buf_t∗ zb_get_in_buf ( )

Get IN buffer from the buffers list.

If no buffers available, does not block. To be called from the interrupt handler reading packets. If no buffer available, int handler must skip this packet.

Returns

pointer to the buffer or NULL if no buffer available.

4.18.2.7 zb_buf_t∗ zb_get_out_buf ( )

Get OUT buffer from the buffers list.

If no buffers available, does not block. To be called from the main loop routine.

Returns

pointer to the buffer.

4.18.2.8 void zb_free_buf (zb_buf_t * *buf*)

Free packt buffer.

Put packet buffer into freelist.

Can be called from the main loop.

Parameters

buf	- packet buffer.

Returns

nothing

4.18.2.9 zb_ret_t zb_get_in_buf_delayed (zb_callback_t *callback*)

Allocate IN buffer.

Call callback when buffer is available.

If buffer available, schedules callback for execution immediatly. If no buffers available now, schedule callback later, when buffer will be available.

Returns

RET_OK or error code.

4.18.2.10 zb_ret_t zb_get_out_buf_delayed (zb_callback_t *callback*)

Allocate OUT buffer.

Call callback when buffer is available.

If buffer available, schedules callback for execution immediatly. If no buffers available now, schedule callback later, when buffer will be available.

Returns

RET_OK or error code.

4.18.3 Macro Definition Documentation

4.18.3.1 #define ZB_BUF_BEGIN(*zbbuf*) ((zbbuf)->buf + (zbbuf)->u.hdr.data_offset)

Return current buffer pointer.

4.18.3.2 #define ZB_BUF_LEN(*zbbuf*) ((zbbuf)->u.hdr.len)

Return current buffer length.

4.18.3.3 #define ZB_BUF_OFFSET(*zbbuf*) ((zbbuf)->u.hdr.data_offset)

Return current buffer offset.

4.18.3.4 #define ZB_BUF_ALLOC_LEFT(*zbbuf*, *size*, *ptr*) (ptr) = zb_buf_smart_alloc_left((zbbuf), (size))

Allocate space at buffer begin.

Parameters

zbbuf	- buffer
size	- size to allocate
ptr	- (out) pointer to the new buffer begin

4.18.3.5 #define ZB_BUF_ALLOC_RIGHT(*zbbuf*, *size*, *ptr*) (ptr) = zb_buf_smart_alloc_right((zbbuf), (size))

Allocate space at buffer end.

Parameters

zbbuf	- buffer
size	- size to allocate
ptr	- (out) pointer to the space allocated

4.18.3.6 #define ZB_BUF_CUT_LEFT(*zbbuf*, *size*, *ptr*) (ptr) = zb_buf_cut_left((zbbuf), (size))

Cut space at buffer begin.

Note: removed assert from here because it can be called from SPI int handler

Parameters

zbbuf	- buffer
size	- size to cut
ptr	- (out) pointer to the new buffer begin

4.18.3.7 #define ZB_BUF_CUT_LEFT2(*zbbuf*, *size*)

Value:

```
do                                                          \
{                                                           \
  (zbbuf)->u.hdr.len -= (size);                             \
  (zbbuf)->u.hdr.data_offset += (size);                     \
} while (0)
```

4.18.3.8 #define ZB_BUF_CUT_RIGHT(*zbbuf*, *size*) zb_buf_cut_right((zbbuf), (size))

Cut space at buffer end.

Parameters

zbbuf	- buffer
size	- size to cut

4.18.3.9 #define ZB_BUF_COPY(*dst_buf, src_buf*)

Value:

```
do                                                                       \
{                                                                           zb_uint8_t is_in = (dst_buf)->u.hdr.is_in_buf;
  ZB_MEMCPY((dst_buf), (src_buf), sizeof(zb_buf_t));                     \
  (dst_buf)->u.hdr.is_in_buf = is_in;                                    \
} while (0)
```

Copy one buffer to the other.

Parameters

src_buf	- source buffer
dst_buf	- destination buffer

4.18.4 Typedef Documentation

4.18.4.1 typedef struct zb_buf_hdr_s zb_buf_hdr_t

Packet buffer header.

4.18.4.2 typedef struct zb_buf_s zb_buf_s_t

Packet buffer.

4.19 Scheduler

Functions

- **ZB_RING_BUFFER_DECLARE** (zb_cb_q, **zb_cb_q_ent_t**, **ZB_SCHEDULER_Q_SIZE**)

  Immediate pending callbacks queue (ring buffer)
- **ZB_RING_BUFFER_DECLARE** (zb_mac_tx_q, **zb_mac_cb_ent_t**, ZB_MAC_QUEUE_SIZE)
- void **zb_sched_init** () ZB_SDCC_REENTRANT

  Initialize scheduler subsystem.
- void **zb_sched_loop_iteration** () ZB_SDCC_REENTRANT

  Call all callbacks.
- zb_ret_t **zb_schedule_callback** (**zb_callback_t** func, **zb_uint8_t** param) ZB_SDCC_REENTRANT

  Schedule callback execution.
- zb_ret_t **zb_schedule_mac_cb** (**zb_callback_t** func, **zb_uint8_t** param) ZB_SDCC_REENTRANT

  Just the similar to schedule callback function, but used for mac cb queue.
- zb_ret_t **zb_schedule_alarm** (**zb_callback_t** func, **zb_uint8_t** param, **zb_time_t** timeout_bi) ZB_SDCC_R-EENTRANT

  Schedule alarm - callback to be executed after timeout.
- zb_ret_t **zb_schedule_alarm_cancel** (**zb_callback_t** func, **zb_uint8_t** param) ZB_SDCC_REENTRANT

  Cancel scheduled alarm.
- zb_ret_t **zb_schedule_tx_cb** (**zb_callback_t** func, **zb_uint8_t** param) ZB_SDCC_REENTRANT

Data Structures

- struct **zb_cb_q_ent_s**

  Immediate pending callbacks queue entry.
- struct **zb_mac_cb_ent_s**
- struct **zb_tm_q_ent_s**

  Delayed (scheduled to run after timeout) callbacks queue entry.
- struct **zb_buf_q_ent_s**
- struct **zb_sched_globals_s**

  Data structures for the delayed execution.

Macros

- #define **ZB_SCHEDULE_CALLBACK zb_schedule_callback**
- #define **ZB_SCHEDULE_AFTER_TX_CB**(cb) (MAC_CTX().tx_wait_cb = cb)
- #define **ZB_SCHEDULE_MAC_CB zb_schedule_mac_cb**
- #define **ZB_SCHEDULE_TX_CB** zb_schedule_tx_cb
- #define **ZB_SCHEDULE_ALARM zb_schedule_alarm**
- #define **ZB_ALARM_ANY_PARAM** (**zb_uint8_t**)(-1)

  *Special parameter for **zb_schedule_alarm_cancel()** (p. 90): cancel alarm once without parameter check.*
- #define **ZB_ALARM_ALL_CB** (**zb_uint8_t**)(-2)

  *Special parameter for **zb_schedule_alarm_cancel()** (p. 90): cancel alarm for all parameters.*
- #define **ZB_SCHEDULE_ALARM_CANCEL zb_schedule_alarm_cancel**
- #define **ZB_SCHED_HAS_PENDING_CALLBACKS**() !ZB_RING_BUFFER_IS_EMPTY(&ZG->sched.cb_-q)

  Return true if scheduler has any pending callbacks.
- #define **ZB_SCHED_WAIT_COND**(condition)

  Wait (block, go idle) until condition will not be true.
- #define **ZB_SCHED_GLOBAL_LOCK** ZB_OSIF_GLOBAL_LOCK

Global lock operation Protect manupulation with queues in the main loop by this macro.

- #define **ZB_SCHED_GLOBAL_UNLOCK** ZB_OSIF_GLOBAL_UNLOCK

  Global unlock operation Protect manupulation with queues by this macro.
- #define **ZB_SCHED_GLOBAL_LOCK_INT**() ZB_OSIF_GLOBAL_LOCK_INT

  Global lock operation - call from the interrupt handler.
- #define **ZB_SCHED_GLOBAL_UNLOCK_INT**() ZB_OSIF_GLOBAL_UNLOCK_INT

  Global unlock operation - call from the interrupt handler.

Typedefs

- typedef void(ZB_CODE ∗ **zb_callback_t**)(**zb_uint8_t** param) ZB_CALLBACK

  Callback function typedef.
- typedef struct **zb_cb_q_ent_s** **zb_cb_q_ent_t**

  Immediate pending callbacks queue entry.
- typedef struct **zb_mac_cb_ent_s** **zb_mac_cb_ent_t**
- typedef struct **zb_tm_q_ent_s** **zb_tm_q_ent_t**

  Delayed (scheduled to run after timeout) callbacks queue entry.
- typedef struct **zb_buf_q_ent_s** **zb_buf_q_ent_t**
- typedef struct **zb_sched_globals_s** **zb_sched_globals_t**

  Data structures for the delayed execution.

4.19.1 Detailed Description

4.19.2 Function Documentation

4.19.2.1 ZB_RING_BUFFER_DECLARE (zb_cb_q , zb_cb_q_ent_t , ZB_SCHEDULER_Q_SIZE)

Immediate pending callbacks queue (ring buffer)

4.19.2.2 void zb_sched_init ( )

Initialize scheduler subsystem.

4.19.2.3 void zb_sched_loop_iteration ( )

Call all callbacks.

All cooperative multitasking done here.

Call all callbacks from the queue. Callbacks can schedule other callbacks, so potentially stay here infinite. In practice at some point callbacks ring buffer became empty. Put device into asleep waiting for interrupts (8051) or wait for data from other source (Linux).

This function usually placed into main loop.

This function MUST be reentrant in Keil: must not share its xdata segment with functions called from it by pointers.

Returns

none

4.19.2.4 zb_ret_t zb_schedule_callback (zb_callback_t *func*, zb_uint8_t *param*)

Schedule callback execution.

Schedule execution of function 'func' in the main scheduler loop.

Parameters

func	- function to execute
param	- callback parameter - usually, but not always ref to packet buffer

Returns

RET_OK or error code.

4.19.2.5 zb_ret_t zb_schedule_mac_cb (zb_callback_t *func*, zb_uint8_t *param*)

Just the similar to schedule callback function, but used for mac cb queue.

4.19.2.6 zb_ret_t zb_schedule_alarm (zb_callback_t *func*, zb_uint8_t *param*, zb_time_t *timeout_bi*)

Schedule alarm - callback to be executed after timeout.

Function will be called via scheduler after timeout expired (maybe, plus some additional time). Timer resolution depends on implementation. Same callback can be scheduled for execution more then once.

Parameters

func	- function to call via scheduler
param	- parameter to pass to the function
timeout_bi	- timeout, in beacon intervals

Returns

RET_OK or error code

4.19.2.7 zb_ret_t zb_schedule_alarm_cancel (zb_callback_t *func*, zb_uint8_t *param*)

Cancel scheduled alarm.

This function cancel previously scheduled alarm. Function is identified by the pointer.

Parameters

func	- function to cancel
param	- parameter to cancel.

See Also

ZB_ALARM_ANY_PARAM (p. 91).
ZB_ALARM_ALL_CB (p. 91)

Returns

RET_OK or error code

4.19.3 Macro Definition Documentation

4.19.3.1 #define ZB_ALARM_ANY_PARAM (zb_uint8_t)(-1)

Special parameter for **zb_schedule_alarm_cancel()** (p. 90): cancel alarm once without parameter check.

Cancel only one alarm without check for parameter

4.19.3.2 #define ZB_ALARM_ALL_CB (zb_uint8_t)(-2)

Special parameter for **zb_schedule_alarm_cancel()** (p. 90): cancel alarm for all parameters.

4.19.3.3 #define ZB_SCHED_HAS_PENDING_CALLBACKS( ) !ZB_RING_BUFFER_IS_EMPTY(&ZG->sched.cb_q)

Return true if scheduler has any pending callbacks.

4.19.3.4 #define ZB_SCHED_WAIT_COND(*condition*)

Value:

```
do                                              \
{                                               \
  ZB_SCHED_GLOBAL_LOCK();                       \
  while ( !(condition) )                        \
  {                                             \
    ZB_SCHED_GLOBAL_UNLOCK();                   \
    ZB_GO_IDLE();                               \
    ZB_SCHED_GLOBAL_LOCK();                     \
  }                                             \
  ZB_SCHED_GLOBAL_UNLOCK();                     \
}                                               \
while(0)
```

Wait (block, go idle) until condition will not be true.

Parameters

condition	- condition to check for

4.19.3.5 #define ZB_SCHED_GLOBAL_LOCK ZB_OSIF_GLOBAL_LOCK

Global lock operation Protect manupulation with queues in the main loop by this macro.

It disables interrupts on 8051 device and locks mutex in Linux.

4.19.3.6 #define ZB_SCHED_GLOBAL_UNLOCK ZB_OSIF_GLOBAL_UNLOCK

Global unlock operation Protect manupulation with queues by this macro.

It enables interrupts on 8051 device and unlocks mutex in Linux.

4.19.3.7 #define ZB_SCHED_GLOBAL_LOCK_INT( ) ZB_OSIF_GLOBAL_LOCK_INT

Global lock operation - call from the interrupt handler.

Returns

RET_OK if success, RET_BUZY if locked by userspace

4.19.3.8 #define ZB_SCHED_GLOBAL_UNLOCK_INT( ) ZB_OSIF_GLOBAL_UNLOCK_INT

Global unlock operation - call from the interrupt handler.

4.19.4 Typedef Documentation

4.19.4.1 typedef void(ZB_CODE ∗ zb_callback_t)(zb_uint8_t param) ZB_CALLBACK

Callback function typedef.

scheduler

Use cooperative multitasking. Trivial scheduler: do all in callbacks. No 'task' primitive. Base primitive - callback call. Callback will be called indirectly, via scheduler. Callback call can be treated as event send. Callbacks schedule done via scheduler in the main scheduler loop. Can pass 1 parameter (void∗) to the callback. Callback initiated using call schedule_callback(func, param). Scheduling callback does not block currently running callback. More then one callback can be scheduled. It will be called later, when current function will return to the scheduler.

Before main loop call application-dependent initialization functions. It can schedule some callbacks. Callbacks will be called later, in the main loop.

Data structure for callbacks support - fixed-size ring buffer of callbacks control structure. Callbacks served in FIFO order, no priorities.

When no callbacks to call, scheduler put device asleep (stop CPU for 8051, wait inside select() for Linux); it can be waked by interrupt (8051) or data arrive or timeout (Linux).

There are 2 possible kinds of routines: callbacks running in the main loop and interrupt handlers. Interrupt handlers works with SPI, UART, timer, transiver interrupt (what else?). Interrupt handler can't schedule callback call.

To work with data shared between interrupt handler and main loop introduced "global lock" operation. It means interrupts disable when running not in the interrupt context. In Linux it means either mutex lock or nothing (depending on i/o implementation). Callback is function planned to execute by another function. Note that callback must be declared as reentrant for dscc.

Parameters

param	- callback parameter - usually, but not always, ref to packet buf

Returns

none.

4.19.4.2 typedef struct zb_cb_q_ent_s zb_cb_q_ent_t

Immediate pending callbacks queue entry.

4.19.4.3 typedef struct zb_tm_q_ent_s zb_tm_q_ent_t

Delayed (scheduled to run after timeout) callbacks queue entry.

4.19.4.4 typedef struct zb_sched_globals_s zb_sched_globals_t

Data structures for the delayed execution.

4.20 Time

Macros

- #define **ZB_TIMER_GET**() (ZB_TIMER_CTX().timer)

  Get current timer value (beacon intervals)
- #define **ZB_TIME_SUBTRACT**(a, b) ((**zb_time_t**)((a) - (b)) < ZB_HALF_MAX_TIME_VAL ? (**zb_time_t**)((a) - (b)) : (**zb_time_t**)((b) - (a)))

  Time subtraction: subtract 'b' from 'a'.
- #define **ZB_TIME_ADD**(a, b) (**zb_time_t**)((a) + (b))

  Time add: add 'a' to 'b'.
- #define **ZB_TIME_GE**(a, b) ((**zb_time_t**)((a) - (b)) < ZB_HALF_MAX_TIME_VAL)

  Compare times a and b - check that a >= b.
- #define **ZB_BEACON_INTERVAL_USEC** 15360 /∗ in microseconds ∗/
- #define **ZB_TIME_ONE_SECOND ZB_MILLISECONDS_TO_BEACON_INTERVAL**(1000)

  One second timeout.
- #define **ZB_TIME_BEACON_INTERVAL_TO_MSEC**(t) (**ZB_BEACON_INTERVAL_USEC** / 100 ∗ (t) / 10)

  Convert time from beacon intervals to millisecinds.
- #define **ZB_MILLISECONDS_TO_BEACON_INTERVAL**(ms) (((10l ∗ (ms) + 3) / (ZB_BEACON_INTERVA-L_USEC / 100)))

  Convert time from millisecinds to beacon intervals.
- #define **ZB_TIMER_START**(interval) zb_timer_start(interval)

  Start timer - assign time to sleep.

Typedefs

- typedef **zb_uint16_t zb_time_t**

  Timer type.

4.20.1 Detailed Description

4.20.2 Macro Definition Documentation

4.20.2.1 #define ZB_TIMER_GET( ) (ZB_TIMER_CTX().timer)

Get current timer value (beacon intervals)

4.20.2.2 #define ZB_TIME_SUBTRACT(*a*, *b*) ((zb_time_t)((a) - (b)) < ZB_HALF_MAX_TIME_VAL ? (zb_time_t)((a) - (b)) : (zb_time_t)((b) - (a)))

Time subtraction: subtract 'b' from 'a'.

Take overflow into account: change sign (subtraction order) if result > values_diapasin/2. Suppose a always >= b, so result is never negative. This macro will be used to calculate, for example, amount of time to sleep

- it is positive by definition. Do not use it to compare time values! Use **ZB_TIME_GE()** (p. 94) instead. Note that both a and b is of type **zb_time_t** (p. 95). Can't decrease time (subtract constant from it) using this macro.

Parameters

a	- time to subtract from
b	- time to subtract

Returns

subtraction result

4.20.2.3 #define ZB_TIME_ADD(*a, b*) (zb_time_t)((a) + (b))

Time add: add 'a' to 'b'.

Overflow is possible, but this is ok - it handled by subtraction and compare macros.

Parameters

a	- time to add to
b	- value to add

Returns

addition result

4.20.2.4 #define ZB_TIME_GE(*a, b*) ((zb_time_t)((a) - (b)) < ZB_HALF_MAX_TIME_VAL)

Compare times a and b - check that a >= b.

Taking into account overflow and unsigned values arithmetic and supposing difference between a and b can't be > 1/2 of the overall time values diapason, a >= b only if a - b < values_diapason/2

Parameters

a	- first time value to compare
b	- second time value to compare

Returns

1 is a >= b, 0 otherwhise

4.20.2.5 #define ZB_BEACON_INTERVAL_USEC 15360 /* in microseconds */

Time measurement unit is beacon interval.

It is both internal representation and value used in API. It is still possible to convert it to/from msec. 1 beacon interval = aBaseSuperframeDuration * symbol duration aBaseSuperframeDuration = aBaseSlotDuration * aNumSuperframeSlots aBaseSlotDuration = 60 aNumSuperframeSlots = 16 1 symbol = 16e-6 sec (mac spec 6.5.3.2 Symbol rate)

4.20.2.6 #define ZB_TIME_ONE_SECOND ZB_MILLISECONDS_TO_BEACON_INTERVAL(1000)

One second timeout.

4.20.2.7 #define ZB_TIME_BEACON_INTERVAL_TO_MSEC(*t*) (ZB_BEACON_INTERVAL_USEC / 100 * (t) / 10)

Convert time from beacon intervals to millisecinds.

Try to not cause overflow in 16-bit arithmetic (with some precision lost...)

4.20.2.8 #define ZB_MILLISECONDS_TO_BEACON_INTERVAL(*ms*) (((10l ∗ (ms) + 3) / (ZB_BEACON_INTERVAL_USEC / 100)))

Convert time from millisecinds to beacon intervals.

Try to not cause overflow in 16-bit arithmetic (with some precision lost...)

4.20.2.9 #define ZB_TIMER_START(*interval*) zb_timer_start(interval)

Start timer - assign time to sleep.

Parameters

interval	- time in internal forrmat to sleep before delayed callback run

4.20.3 Typedef Documentation

4.20.3.1 typedef zb_uint16_t zb_time_t

Timer type.

Timer functionality.

The idea is: platform has some timer which can be stopped or run. When run, it increments (or decrements - depends on platform) some counter until counter overflow (underflow), then issues interrupt - wakeups main loop if it sleeping. Time stored in ticks; time resolution is platform dependent, its usual value is 15.36 usec - 1 beacon interval. Note that time type has limited capacity (usually 16 bits) and can overflow. Macros which works with time handles overflow. It is supposed that time values will not differ to more then 1/2 of the maximum time value.

All that timer macros will not be used directly by the application code - it is scheduler internals. The only API for timer is ZB_SCHEDULE_ALARM() call.

16 bits for 8051 - it will be hw timer value. Not sure it is right to use 16 bits in Linux. But let's do it now to debug owerflow. In the future could use 32 bits in Linux.

4.21 Debug trace

Data Structures

- struct **zb_addr64_struct_s**

Macros

- #define **TRACE_MSG**(...)
- #define **TRACE_INIT**(name)
- #define **TRACE_DEINIT**(c)
- #define **TRACE_ENABLED**(m) 0
- #define **TRACE_FORMAT_64** "%A"

  Trace format for 64-bit address - single argument for 8051.
- #define **TRACE_ARG_64**(a) ∗((**zb_addr64_struct_t** ∗)a)
- #define **TRACE_ERROR** -1, 1

  General trace message definition: error.
- #define **TRACE_INFO1** -1, 2
- #define **TRACE_INFO2** -1, 3
- #define **TRACE_INFO3** -1, 4
- #define **TRACE_SUBSYSTEM_COMMON** 0x0001
- #define **TRACE_SUBSYSTEM_OSIF** 0x0002
- #define **TRACE_SUBSYSTEM_MAC** 0x0004
- #define **TRACE_SUBSYSTEM_NWK** 0x0008
- #define **TRACE_SUBSYSTEM_APS** 0x0010
- #define **TRACE_SUBSYSTEM_AF** 0x0020
- #define **TRACE_SUBSYSTEM_ZDO** 0x0040
- #define **TRACE_SUBSYSTEM_SECUR** 0x0080
- #define **TRACE_SUBSYSTEM_ZCL** 0x0100
- #define **FMT__0** __FILE__,__LINE__, 0
- #define **FMT__A** __FILE__,__LINE__, 8
- #define **FMT__A_A** __FILE__,__LINE__, 16
- #define **FMT__A_D_A_P** __FILE__,__LINE__, 21
- #define **FMT__A_D_D_P_H** __FILE__,__LINE__, 16
- #define **FMT__A_D_H** __FILE__,__LINE__, 11
- #define **FMT__C** __FILE__,__LINE__, 1
- #define **FMT__D** __FILE__,__LINE__, 2
- #define **FMT__D_A** __FILE__,__LINE__, 10
- #define **FMT__D_A_D_D_D_D_D_D_D_D** __FILE__,__LINE__, 26
- #define **FMT__D_A_D_P_H_H_H** __FILE__,__LINE__, 18
- #define **FMT__D_A_P** __FILE__,__LINE__, 13
- #define **FMT__A_P** __FILE__,__LINE__, 11
- #define **FMT__D_C** __FILE__,__LINE__, 3
- #define **FMT__D_D** __FILE__,__LINE__, 4
- #define **FMT__D_D_A_D** __FILE__,__LINE__, 14
- #define **FMT__D_D_A_D_D_D_D** __FILE__,__LINE__, 20
- #define **FMT__D_D_D** __FILE__,__LINE__, 6
- #define **FMT__D_D_D_C** __FILE__,__LINE__, 7
- #define **FMT__D_D_D_D** __FILE__,__LINE__, 8
- #define **FMT__D_D_D_D_D_D_D_D_D_D_D_D_D_D_D_D** __FILE__,__LINE__, 34
- #define **FMT__D_D_D_P** __FILE__,__LINE__, 9
- #define **FMT__D_D_P** __FILE__,__LINE__, 7
- #define **FMT__D_D_P_D** __FILE__,__LINE__, 9
- #define **FMT__D_D_P_P_P** __FILE__,__LINE__, 13

- #define **FMT__D_H** __FILE__,__LINE__, 3
- #define **FMT__D_D_H** __FILE__,__LINE__, 5
- #define **FMT__D_H_H** __FILE__,__LINE__, 4
- #define **FMT__D_H_H_H_H_H_H_D_D_D_D** __FILE__,__LINE__, 16
- #define **FMT__D_H_P** __FILE__,__LINE__, 6
- #define **FMT__D_P** __FILE__,__LINE__, 5
- #define **FMT__D_P_D** __FILE__,__LINE__, 7
- #define **FMT__D_P_H_H_D_H_H** __FILE__,__LINE__, 11
- #define **FMT__D_P_P** __FILE__,__LINE__, 8
- #define **FMT__D_P_P_D_D_H_H** __FILE__,__LINE__, 14
- #define **FMT__D_P_P_H** __FILE__,__LINE__, 9
- #define **FMT__H** __FILE__,__LINE__, 1
- #define **FMT__H_A** __FILE__,__LINE__, 9
- #define **FMT__H_A_A** __FILE__,__LINE__, 17
- #define **FMT__H_A_H_H_H_H_H_H_H_H** __FILE__,__LINE__, 17
- #define **FMT__H_C_D_C** __FILE__,__LINE__, 5
- #define **FMT__H_D** __FILE__,__LINE__, 3
- #define **FMT__H_D_A_H_D** __FILE__,__LINE__, 14
- #define **FMT__H_D_A_H_H_H_H** __FILE__,__LINE__, 15
- #define **FMT__H_D_D** __FILE__,__LINE__, 5
- #define **FMT__H_D_D_D_H_H_D** __FILE__,__LINE__, 11
- #define **FMT__H_H** __FILE__,__LINE__, 2
- #define **FMT__H_H_D** __FILE__,__LINE__, 4
- #define **FMT__H_H_H** __FILE__,__LINE__, 3
- #define **FMT__H_H_H_H** __FILE__,__LINE__, 4
- #define **FMT__H_H_P** __FILE__,__LINE__, 5
- #define **FMT__H_P** __FILE__,__LINE__, 4
- #define **FMT__L_L** __FILE__,__LINE__, 8
- #define **FMT__P** __FILE__,__LINE__, 3
- #define **FMT__P_D** __FILE__,__LINE__, 5
- #define **FMT__P_D_D** __FILE__,__LINE__, 7
- #define **FMT__P_D_D_D** __FILE__,__LINE__, 9
- #define **FMT__P_D_D_D_D_D** __FILE__,__LINE__, 13
- #define **FMT__P_D_D_D_D_D_D** __FILE__,__LINE__, 15
- #define **FMT__P_D_D_D_D_D_D_D** __FILE__,__LINE__, 17
- #define **FMT__P_D_D_D_H_D** __FILE__,__LINE__, 12
- #define **FMT__P_D_H** __FILE__,__LINE__, 6
- #define **FMT__P_D_P** __FILE__,__LINE__, 8
- #define **FMT__P_H** __FILE__,__LINE__, 4
- #define **FMT__P_H_D** __FILE__,__LINE__, 6
- #define **FMT__P_H_H** __FILE__,__LINE__, 5
- #define **FMT__P_H_H_L** __FILE__,__LINE__, 9
- #define **FMT__P_H_L** __FILE__,__LINE__, 8
- #define **FMT__P_H_P_H_L** __FILE__,__LINE__, 12
- #define **FMT__P_H_P_P** __FILE__,__LINE__, 10
- #define **FMT__P_H_P_P_P** __FILE__,__LINE__, 13
- #define **FMT__P_P** __FILE__,__LINE__, 6
- #define **FMT__P_P_D** __FILE__,__LINE__, 8
- #define **FMT__P_P_D_D_H** __FILE__,__LINE__, 11
- #define **FMT__P_P_D_H_H** __FILE__,__LINE__, 10
- #define **FMT__P_P_H** __FILE__,__LINE__, 7
- #define **FMT__P_P_P** __FILE__,__LINE__, 9
- #define **FMT__H_H_H_D_D_H_A_H_A** __FILE__,__LINE__, 25
- #define **FMT__H_H_P_P_P** __FILE__,__LINE__, 11
- #define **FMT__D_H_D_P_D** __FILE__,__LINE__, 10

- #define **FMT__D_D_D_D_D** __FILE__,__LINE__, 10
- #define **FMT__H_D_D_D_D** __FILE__,__LINE__, 9
- #define **FMT__D_D_D_D_H** __FILE__,__LINE__, 9
- #define **FMT__D_H_H_D** __FILE__,__LINE__, 6
- #define **FMT__D_P_D_D** __FILE__,__LINE__, 9
- #define **FMT__H_H_H_D** __FILE__,__LINE__, 5
- #define **FMT__H_D_H_H** __FILE__,__LINE__, 5
- #define **FMT__P_H_H_H_H_H_H_H** __FILE__,__LINE__, 10
- #define **FMT__P_H_H_H_H_H_H** __FILE__,__LINE__, 9
- #define **FMT__D_D_H_D_H** __FILE__,__LINE__, 8
- #define **FMT__H_D_D_H_H_H_H** __FILE__,__LINE__, 9
- #define **FMT__H_H_A_A** __FILE__,__LINE__, 18
- #define **FMT__P_H_P_P_H** __FILE__,__LINE__, 11
- #define **FMT__P_H_P_H** __FILE__,__LINE__, 8
- #define **FMT__A_D_D** __FILE__,__LINE__, 12
- #define **FMT__P_H_H_H** __FILE__,__LINE__, 6
- #define **FMT__P_H_P** __FILE__,__LINE__, 7
- #define **FMT__P_P_H_H** __FILE__,__LINE__, 8
- #define **FMT__D_P_H_H_D_D** __FILE__,__LINE__, 11
- #define **FMT__A_H** __FILE__,__LINE__, 9
- #define **FMT__P_H_D_L** __FILE__,__LINE__, 10
- #define **FMT__H_H_H_P** __FILE__,__LINE__, 6
- #define **FMT__A_D_P_H_H_H** __FILE__,__LINE__, 16
- #define **FMT__H_P_H_P_H_H** __FILE__,__LINE__, 10
- #define **FMT__H_P_H_P_H_H** __FILE__,__LINE__, 10
- #define **FMT__H_P_H_H_H_H** __FILE__,__LINE__, 8
- #define **FMT_H_D_H_H_H_H_H_H** __FILE__,__LINE__, 9
- #define **FMT__H_D_D_H_H_H** __FILE__,__LINE__, 8
- #define **FMT__D_D_H_H** __FILE__,__LINE__, 6
- #define **FMT__H_H_D_H** __FILE__,__LINE__, 5
- #define **FMT__D_H_H_H_H** __FILE__,__LINE__, 6
- #define **FMT__H_H_H_D_H** __FILE__,__LINE__, 6
- #define **FMT__H_D_H** __FILE__,__LINE__, 4
- #define **FMT__H_D_H_D** __FILE__,__LINE__, 6
- #define **FMT__D_H_D_H_H** __FILE__,__LINE__, 7
- #define **FMT__H_P_H_P_H** __FILE__,__LINE__, 9
- #define **FMT__H_P_H_H_H** __FILE__,__LINE__, 7
- #define **FMT__D_H_D_H** __FILE__,__LINE__, 6
- #define **FMT__D_H_H_H** __FILE__,__LINE__, 5
- #define **FMT__H_H_D_H_P** __FILE__,__LINE__, 8
- #define **FMT__H_H_H_D_H_P** __FILE__,__LINE__, 9
- #define **FMT__A_H_H** __FILE__,__LINE__, 10
- #define **FMT__P_H_H_H_H** __FILE__,__LINE__, 7
- #define **FMT__H_D_P_H_H_H_H_H** __FILE__,__LINE__, 11
- #define **FMT__P_H_H_H_L** __FILE__,__LINE__, 10
- #define **FMT__H_H_H_H_H_H_H_H** __FILE__,__LINE__, 8
- #define **FMT__H_H_H_H_H_H_H** __FILE__,__LINE__, 7
- #define **FMT__H_H_H_H_H_H** __FILE__,__LINE__, 6
- #define **FMT__H_H_H_H_H** __FILE__,__LINE__, 5
- #define **FMT__H_D_H_H_H** __FILE__,__LINE__, 6
- #define **FMT__D_D_D_D_D_D** __FILE__,__LINE__, 12
- #define **FMT__P_H_H_H_H_D** __FILE__,__LINE__, 7
- #define **FMT__H_D_D_H_D_H** __FILE__,__LINE__, 9
- #define **FMT__H_P_H** __FILE__,__LINE__, 5
- #define **FMT__H_H_D_D** __FILE__,__LINE__, 6

- #define **TRACE_COMMON1 TRACE_SUBSYSTEM_COMMON**, 1
- #define **TRACE_COMMON2 TRACE_SUBSYSTEM_COMMON**, 2
- #define **TRACE_COMMON3 TRACE_SUBSYSTEM_COMMON**, 3
- #define **TRACE_OSIF1** TRACE_SUBSYSTEM_OSIF, 1
- #define **TRACE_OSIF2** TRACE_SUBSYSTEM_OSIF, 2
- #define **TRACE_OSIF3** TRACE_SUBSYSTEM_OSIF, 3
- #define **TRACE_MAC1** TRACE_SUBSYSTEM_MAC, 1
- #define **TRACE_MAC2** TRACE_SUBSYSTEM_MAC, 2
- #define **TRACE_MAC3** TRACE_SUBSYSTEM_MAC, 3
- #define **TRACE_NWK1** TRACE_SUBSYSTEM_NWK, 1
- #define **TRACE_NWK2** TRACE_SUBSYSTEM_NWK, 2
- #define **TRACE_NWK3** TRACE_SUBSYSTEM_NWK, 3
- #define **TRACE_APS1** TRACE_SUBSYSTEM_APS, 1
- #define **TRACE_APS2** TRACE_SUBSYSTEM_APS, 2
- #define **TRACE_APS3** TRACE_SUBSYSTEM_APS, 3
- #define **TRACE_AF1** TRACE_SUBSYSTEM_AF, 1
- #define **TRACE_AF2** TRACE_SUBSYSTEM_AF, 2
- #define **TRACE_AF3** TRACE_SUBSYSTEM_AF, 3
- #define **TRACE_ZDO1** TRACE_SUBSYSTEM_ZDO, 1
- #define **TRACE_ZDO2** TRACE_SUBSYSTEM_ZDO, 2
- #define **TRACE_ZDO3** TRACE_SUBSYSTEM_ZDO, 3
- #define **TRACE_SECUR1** TRACE_SUBSYSTEM_SECUR, 1
- #define **TRACE_SECUR2** TRACE_SUBSYSTEM_SECUR, 2
- #define **TRACE_SECUR3** TRACE_SUBSYSTEM_SECUR, 3
- #define **TRACE_ZCL1** TRACE_SUBSYSTEM_ZCL, 1
- #define **TRACE_ZCL2** TRACE_SUBSYSTEM_ZCL, 2
- #define **TRACE_ZCL3** TRACE_SUBSYSTEM_ZCL, 3

Typedefs

- typedef struct **zb_addr64_struct_s zb_addr64_struct_t**

4.21.1 Detailed Description

4.21.2 Macro Definition Documentation

4.21.2.1 #define TRACE_MSG(...)

ZigBee trace subsystem.

Has 2 parameters to switch log messages on/off: mask and level. Mask used to exclude some layers trace. Level used to trace more or less detailed messages from the same layer. Trace can be switched at compile time only osung 2 defines. ZB_TRACE_LEVEL is mandatory, ZB_TRACE_MASK is optional. No trace code compiled if ZB_TRACE_LEVEL is not defined.

Trace call looks like:

TRACE_MSG(TRACE_COMMON3, "%p calling cb %p param %hd", (FMT_P_P_H, (void*)ent, ent->func, ent->param));

FMT_P_P_H and similar constants are defined in zb_trace_fmts.h and are sum of argument sizes. Actual for 8051, ignored in Unix.

See

See Also

tests/trace.c for usage example.

4.21.2.2 #define TRACE_FORMAT_64 "%A"

Trace format for 64-bit address - single argument for 8051.

4.21.2.3 #define TRACE_ERROR -1, 1

General trace message definition: error.

4.21.2.4 #define TRACE_SUBSYSTEM_COMMON 0x0001

Trace subsystems

4.21.2.5 #define TRACE_COMMON1 TRACE_SUBSYSTEM_COMMON, 1

per-subsystem trace definitions

Chapter 5

Data Structure Documentation

5.1 zb_addr64_struct_s Struct Reference

Data Fields

- **zb_64bit_addr_t addr**

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

- zb_trace.h

5.2 zb_addr_u Union Reference

Union to address either long or short address.

`#include <zb_types.h>`

Data Fields

- **zb_uint16_t addr_short**
- **zb_ieee_addr_t addr_long**

5.2.1 Detailed Description

Union to address either long or short address.

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

- zb_types.h

5.3 zb_aps_hdr_s Struct Reference

Parsed APS header This data structure passed to zb_aps_hdr_parse()

`#include <zb_aps.h>`

Data Fields

- **zb_uint8_t fc**
- **zb_uint16_t src_addr**
- **zb_uint16_t dst_addr**
- **zb_uint16_t group_addr**
- **zb_uint8_t dst_endpoint**
- **zb_uint8_t src_endpoint**
- **zb_uint16_t clusterid**
- **zb_uint16_t profileid**
- **zb_uint8_t aps_counter**

5.3.1 Detailed Description

Parsed APS header This data structure passed to zb_aps_hdr_parse()

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

- zb_aps.h

5.4 zb_apsde_data_req_s Struct Reference

APSDE data request structure.

```
#include <zb_aps.h>
```

Data Fields

- union **zb_addr_u dst_addr**
- **zb_uint16_t profileid**
- **zb_uint16_t clusterid**
- **zb_uint8_t dst_endpoint**
- **zb_uint8_t src_endpoint**
- **zb_uint8_t radius**
- **zb_uint8_t addr_mode**
- **zb_uint8_t tx_options**

5.4.1 Detailed Description

APSDE data request structure.

This data structure passed to **zb_apsde_data_request()** (p. 40) in the packet buffer (at its tail).

5.4.2 Field Documentation

5.4.2.1 union zb_addr_u zb_apsde_data_req_s::dst_addr

Destination address

5.4.2.2 zb_uint16_t zb_apsde_data_req_s::profileid

The identifier of the profile for which this frame is intended.

5.4.2.3 zb_uint16_t zb_apsde_data_req_s::clusterid

The identifier of the object for which this frame is intended.

5.4.2.4 zb_uint8_t zb_apsde_data_req_s::dst_endpoint

either the number of the individual endpoint of the entity to which the ASDU is being transferred or the broadcast endpoint (0xff).

5.4.2.5 zb_uint8_t zb_apsde_data_req_s::src_endpoint

The individual endpoint of the entity from which the ASDU is being transferred.

5.4.2.6 zb_uint8_t zb_apsde_data_req_s::radius

The distance, in hops, that a frame will be allowed to travel through the network.

5.4.2.7 zb_uint8_t zb_apsde_data_req_s::addr_mode

The type of destination address supplied by the DstAddr parameter -

See Also

zb_aps_addr_mode_e (p. 42)

5.4.2.8 zb_uint8_t zb_apsde_data_req_s::tx_options

The transmission options for the ASDU to be transferred. These are a bitwise OR of one or more of the following: 0x01 = Security enabled transmission 0x02 = Use NWK key 0x04 = Acknowledged transmission 0x08 = Fragmentation permitted.

See Also

zb_apsde_tx_opt_e (p. 43)

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

- zb_aps.h

5.5 zb_apsme_add_group_conf_s Struct Reference

APSME-ADD-GROUP.confirm primitive parameters.

```
#include <zb_aps.h>
```

Data Fields

- **zb_uint16_t group_address**
- **zb_uint8_t endpoint**
- **zb_uint8_t status**

5.5.1 Detailed Description

APSME-ADD-GROUP.confirm primitive parameters.

5.5.2 Field Documentation

5.5.2.1 zb_uint16_t zb_apsme_add_group_conf_s::group_address

The 16-bit address of the group being added.

5.5.2.2 zb_uint8_t zb_apsme_add_group_conf_s::endpoint

The endpoint to which the given group is being added.

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

- zb_aps.h

5.6 zb_apsme_add_group_req_s Struct Reference

APSME-ADD-GROUP.request primitive parameters.

`#include <zb_aps.h>`

Data Fields

- **zb_uint16_t group_address**
- **zb_uint8_t endpoint**

5.6.1 Detailed Description

APSME-ADD-GROUP.request primitive parameters.

5.6.2 Field Documentation

5.6.2.1 zb_uint16_t zb_apsme_add_group_req_s::group_address

The 16-bit address of the group being added.

5.6.2.2 zb_uint8_t zb_apsme_add_group_req_s::endpoint

The endpoint to which the given group is being added.

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

- zb_aps.h

5.7 zb_apsme_binding_req_s Struct Reference

APSME binding structure.

`#include <zb_aps.h>`

Data Fields

- **zb_ieee_addr_t src_addr**
- **zb_uint8_t src_endpoint**
- **zb_uint16_t clusterid**
- **zb_uint8_t addr_mode**
- union **zb_addr_u dst_addr**
- **zb_uint8_t dst_endpoint**

5.7.1 Detailed Description

APSME binding structure.

This data structure passed to zb_apsme_bind_request()

5.7.2 Field Documentation

5.7.2.1 zb_ieee_addr_t zb_apsme_binding_req_s::src_addr

The source IEEE address for the binding entry.

5.7.2.2 zb_uint8_t zb_apsme_binding_req_s::src_endpoint

The source endpoint for the binding entry.

5.7.2.3 zb_uint16_t zb_apsme_binding_req_s::clusterid

The identifier of the cluster on the source device that is to be bound to the destination device.

5.7.2.4 zb_uint8_t zb_apsme_binding_req_s::addr_mode

The type of destination address supplied by the DstAddr parameter -

See Also

zb_aps_addr_mode_e (p. 42)

5.7.2.5 union zb_addr_u zb_apsme_binding_req_s::dst_addr

The destination address for the binding entry.

5.7.2.6 zb_uint8_t zb_apsme_binding_req_s::dst_endpoint

This parameter will be present only if the DstAddrMode parameter has a value of 0x03 and, if present, will be the destination endpoint for the binding entry.

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

- zb_aps.h

5.8 zb_apsme_get_confirm_s Struct Reference

APSME GET confirm structure.

`#include <zb_aps.h>`

Data Fields

- zb_aps_status_t **status**
- **zb_aps_aib_attr_id_t aib_attr**
- **zb_uint8_t aib_length**

5.8.1 Detailed Description

APSME GET confirm structure.

5.8.2 Field Documentation

5.8.2.1 zb_aps_status_t zb_apsme_get_confirm_s::status

The results of the request to read an AIB attribute value.

5.8.2.2 zb_aps_aib_attr_id_t zb_apsme_get_confirm_s::aib_attr

The identifier of the AIB attribute that was read.

5.8.2.3 zb_uint8_t zb_apsme_get_confirm_s::aib_length

The length, in octets, of the attribute value being returned.

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

- zb_aps.h

5.9 zb_apsme_get_request_s Struct Reference

APSME GET request structure.

`#include <zb_aps.h>`

Data Fields

- **zb_aps_aib_attr_id_t aib_attr**

5.9.1 Detailed Description

APSME GET request structure.

5.9.2 Field Documentation

5.9.2.1 zb_aps_aib_attr_id_t zb_apsme_get_request_s::aib_attr

The identifier of the AIB attribute to read.

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

- zb_aps.h

5.10 zb_apsme_set_confirm_s Struct Reference

APSME SET confirm structure.

```
#include <zb_aps.h>
```

Data Fields

- zb_aps_status_t **status**
- **zb_aps_aib_attr_id_t aib_attr**

5.10.1 Detailed Description

APSME SET confirm structure.

5.10.2 Field Documentation

5.10.2.1 zb_aps_status_t zb_apsme_set_confirm_s::status

The result of the request to write the AIB Attribute.

5.10.2.2 zb_aps_aib_attr_id_t zb_apsme_set_confirm_s::aib_attr

The identifier of the AIB attribute that was written.

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

- zb_aps.h

5.11 zb_apsme_set_request_s Struct Reference

APSME SET request structure.

```
#include <zb_aps.h>
```

Data Fields

- **zb_aps_aib_attr_id_t aib_attr**
- **zb_uint8_t aib_length**

5.11.1 Detailed Description

APSME SET request structure.

5.11.2 Field Documentation

5.11.2.1 zb_aps_aib_attr_id_t zb_apsme_set_request_s::aib_attr

The identifier of the AIB attribute to be written.

5.11.2.2 zb_uint8_t zb_apsme_set_request_s::aib_length

The length, in octets, of the attribute value being set.

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

- zb_aps.h

5.12 zb_buf_hdr_s Struct Reference

Packet buffer header.

`#include <zb_bufpool.h>`

Data Fields

- **zb_uint8_t len**
- **zb_uint8_t data_offset**
- **zb_uint8_t handle**
- **zb_uint8_t mac_hdr_offset**
- **zb_int16_t status**
- **zb_bitfield_t is_in_buf**:1
- **zb_bitfield_t encrypt_type**:2
- **zb_bitfield_t use_same_key**:1
- **zb_bitfield_t zdo_cmd_no_resp**:1
- **zb_bitfield_t reserved**:3
- **zb_uint8_t mhr_len**

5.12.1 Detailed Description

Packet buffer header.

5.12.2 Field Documentation

5.12.2.1 zb_uint8_t zb_buf_hdr_s::len

current layer buffer length

5.12.2.2 zb_uint8_t zb_buf_hdr_s::data_offset

data offset in buffer buf

5.12.2.3 zb_uint8_t zb_buf_hdr_s::handle

The handle associated with the NSDU to be transmitted by the NWK layer entity.

5.12.2.4 zb_int16_t zb_buf_hdr_s::status

some status to be passed with packet

5.12.2.5 zb_bitfield_t zb_buf_hdr_s::is_in_buf

if 1, this is input buffer

5.12.2.6 zb_bitfield_t zb_buf_hdr_s::encrypt_type

payload must be encrypted before send, if !0.

See Also

zb_secur_buf_encr_type_e.

5.12.2.7 zb_bitfield_t zb_buf_hdr_s::use_same_key

if 1, use same nwk key# packet was encrypted by

5.12.2.8 zb_bitfield_t zb_buf_hdr_s::zdo_cmd_no_resp

if 1, this is ZDO command with no responce - call cqallback at confirm

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

- zb_bufpool.h

5.13 zb_buf_q_ent_s Struct Reference

Public Member Functions

- **ZB_SL_LIST_FIELD** (struct **zb_buf_q_ent_s** $*$, next)

Data Fields

- **zb_callback_t func**

5.13.1 Field Documentation

5.13.1.1 zb_callback_t zb_buf_q_ent_s::func

function to call

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

- zb_scheduler.h

5.14 zb_buf_s Struct Reference

Packet buffer.

`#include <zb_bufpool.h>`

Data Fields

- union {
  zb_buf_hdr_t hdr
  struct **zb_buf_s** ∗ **next**
  } **u**

- **zb_uint8_t buf** [**ZB_IO_BUF_SIZE**]

5.14.1 Detailed Description

Packet buffer.

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

- zb_bufpool.h

5.15 zb_cb_q_ent_s Struct Reference

Immediate pending callbacks queue entry.

`#include <zb_scheduler.h>`

Data Fields

- **zb_callback_t func**
- **zb_uint8_t param**

5.15.1 Detailed Description

Immediate pending callbacks queue entry.

5.15.2 Field Documentation

5.15.2.1 zb_callback_t zb_cb_q_ent_s::func

function to call

5.15.2.2 zb_uint8_t zb_cb_q_ent_s::param

parameter to pass to 'func'

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

- zb_scheduler.h

5.16 zb_end_device_bind_req_param_s Struct Reference

Parameters for 2.4.3.2.1 End_Device_Bind_req.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t dst_addr**
- **zb_zdo_end_device_bind_req_head_t head_param**
- **zb_zdo_end_device_bind_req_tail_t tail_param**
- **zb_uint16_t cluster_list** [1]

5.16.1 Detailed Description

Parameters for 2.4.3.2.1 End_Device_Bind_req.

5.16.2 Field Documentation

5.16.2.1 zb_uint16_t zb_end_device_bind_req_param_s::dst_addr

Destinition address

5.16.2.2 zb_zdo_end_device_bind_req_head_t zb_end_device_bind_req_param_s::head_param

Parameters for command head

5.16.2.3 zb_zdo_end_device_bind_req_tail_t zb_end_device_bind_req_param_s::tail_param

Parameters for command tail

5.16.2.4 zb_uint16_t zb_end_device_bind_req_param_s::cluster_list[1]

List of Input and Output ClusterIDs to be used for matching

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

- zb_zdo.h

5.17 zb_mac_cb_ent_s Struct Reference

Data Fields

- **zb_callback_t func**
- **zb_uint8_t param**

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

- zb_scheduler.h

5.18 zb_mac_device_table_s Struct Reference

Data Fields

- **zb_ieee_addr_t long_address**
- **zb_uint16_t short_address**
- **zb_uint32_t frame_counter**
- **zb_uint16_t pan_id**

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

- zb_mac.h

5.19 zb_mlme_get_confirm_s Struct Reference

Defines MLME-GET.confirm primitive.

`#include <zb_mac.h>`

Data Fields

- **zb_mac_status_t status**
- **zb_mac_pib_attr_t pib_attr**
- **zb_uint8_t pib_index**
- **zb_uint8_t pib_length**

5.19.1 Detailed Description

Defines MLME-GET.confirm primitive.

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

- zb_mac.h

5.20 zb_mlme_get_request_s Struct Reference

Defines MLME-GET.request primitive.

`#include <zb_mac.h>`

Data Fields

- **zb_mac_pib_attr_t pib_attr**
- **zb_uint8_t pib_index**

5.20.1 Detailed Description

Defines MLME-GET.request primitive.

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

- zb_mac.h

5.21 zb_mlme_set_confirm_s Struct Reference

Defines MLME-SET.confirm primitive.

`#include <zb_mac.h>`

Data Fields

- **zb_mac_status_t status**
- **zb_mac_pib_attr_t pib_attr**
- **zb_uint8_t pib_index**

5.21.1 Detailed Description

Defines MLME-SET.confirm primitive.

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

- zb_mac.h

5.22 zb_mlme_set_request_s Struct Reference

Defines MLME-SET.request primitive.

`#include <zb_mac.h>`

Data Fields

- **zb_mac_pib_attr_t pib_attr**
- **zb_uint8_t pib_index**
- **zb_uint8_t pib_length**

5.22.1 Detailed Description

Defines MLME-SET.request primitive.

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

- zb_mac.h

5.23 zb_nlde_data_req_s Struct Reference

Parameters for NLDE-DATA.request primitive.

`#include <zb_nwk.h>`

Data Fields

- **zb_uint16_t dst_addr**
- **zb_uint8_t radius**
- **zb_uint8_t addr_mode**
- **zb_uint8_t nonmember_radius**

- **zb_uint8_t discovery_route**
- **zb_uint8_t security_enable**
- **zb_uint8_t ndsu_handle**

5.23.1 Detailed Description

Parameters for NLDE-DATA.request primitive.

5.23.2 Field Documentation

5.23.2.1 zb_uint16_t zb_nlde_data_req_s::dst_addr

Destination address.

5.23.2.2 zb_uint8_t zb_nlde_data_req_s::radius

The distance, in hops, that a frame will be allowed to travel through the network.

5.23.2.3 zb_uint8_t zb_nlde_data_req_s::addr_mode

The type of destination address supplied by the DstAddr parameter -

See Also

zb_addr_mode_e

5.23.2.4 zb_uint8_t zb_nlde_data_req_s::nonmember_radius

The distance, in hops, that a multicast frame will be relayed by nodes not a member of the group. A value of 0x07 is treated as infinity.

5.23.2.5 zb_uint8_t zb_nlde_data_req_s::discovery_route

The DiscoverRoute parameter may be used to control route discovery operations for the transit of this frame (see sub-clause3.6.3.5): 0x00 = suppress route discovery 0x01 = enable route discovery

5.23.2.6 zb_uint8_t zb_nlde_data_req_s::security_enable

The SecurityEnable parameter may be used to enable NWK layer security processing for the current frame. If the nwkSecurityLevel attribute of the NIB has a value of 0, meaning no security, then this parameter will be ignored. Otherwise, a value of TRUE denotes that the security processing specified by the security level will be applied, and a value of FALSE denotes that no security processing will be applied.

5.23.2.7 zb_uint8_t zb_nlde_data_req_s::ndsu_handle

The handle associated with the NSDU to be transmitted by the NWK layer entity.

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

- zb_nwk.h

5.24 zb_nlme_get_confirm_s Struct Reference

Arguments of the NLME-GET.confirm routine.

```
#include <zb_nwk.h>
```

Data Fields

- **zb_nwk_status_t status**
- **zb_nib_attribute_t nib_attribute**
- **zb_uint16_t attribute_length**

5.24.1 Detailed Description

Arguments of the NLME-GET.confirm routine.

5.24.2 Field Documentation

5.24.2.1 zb_nwk_status_t zb_nlme_get_confirm_s::status

The result of the operation

5.24.2.2 zb_nib_attribute_t zb_nlme_get_confirm_s::nib_attribute

Attribute value,

See Also

zb_nib_attribute_t (p. 56)

5.24.2.3 zb_uint16_t zb_nlme_get_confirm_s::attribute_length

Length attribute value

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

- zb_nwk.h

5.25 zb_nlme_get_request_s Struct Reference

Arguments of the NLME-GET.request routine.

```
#include <zb_nwk.h>
```

Data Fields

- **zb_nib_attribute_t nib_attribute**

5.25.1 Detailed Description

Arguments of the NLME-GET.request routine.

5.25.2 Field Documentation

5.25.2.1 zb_nib_attribute_t zb_nlme_get_request_s::nib_attribute

Attribute value,

See Also

zb_nib_attribute_t (p. 56)

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

- zb_nwk.h

5.26 zb_nlme_send_status_s Struct Reference

Arguments of the NLME-SEND-STATUS.confirm routine.

```
#include <zb_nwk.h>
```

Data Fields

- **zb_uint16_t dest_addr**
- **zb_nlme_status_indication_t status**
- **zb_uint8_t ndsu_handle**

5.26.1 Detailed Description

Arguments of the NLME-SEND-STATUS.confirm routine.

5.26.2 Field Documentation

5.26.2.1 zb_uint16_t zb_nlme_send_status_s::dest_addr

address to send status information to

5.26.2.2 zb_nlme_status_indication_t zb_nlme_send_status_s::status

status information

See Also

zb_nlme_status_indication_t (p. 50)

5.26.2.3 zb_uint8_t zb_nlme_send_status_s::ndsu_handle

The handle associated with the NSDU to be transmitted by the NWK layer entity.

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

- zb_nwk.h

5.27 zb_nlme_set_confirm_s Struct Reference

Arguments of the NLME-SET.confirm routine.

```
#include <zb_nwk.h>
```

Data Fields

- **zb_nwk_status_t status**
- **zb_nib_attribute_t nib_attribute**

5.27.1 Detailed Description

Arguments of the NLME-SET.confirm routine.

5.27.2 Field Documentation

5.27.2.1 zb_nwk_status_t zb_nlme_set_confirm_s::status

The result of the operation

5.27.2.2 zb_nib_attribute_t zb_nlme_set_confirm_s::nib_attribute

Attribute value,

See Also

zb_nib_attribute_t (p. 56)

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

- zb_nwk.h

5.28 zb_nlme_set_request_s Struct Reference

Arguments of the NLME-SET.request routine.

```
#include <zb_nwk.h>
```

Data Fields

- **zb_nib_attribute_t nib_attribute**
- **zb_uint16_t attr_length**

5.28.1 Detailed Description

Arguments of the NLME-SET.request routine.

5.28.2 Field Documentation

5.28.2.1 zb_nib_attribute_t zb_nlme_set_request_s::nib_attribute

Attribute value,

See Also

zb_nib_attribute_t (p. 56)

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

- zb_nwk.h

5.29 zb_nlme_status_indication_s Struct Reference

Arguments of the NLME-STATUS.request routine.

```
#include <zb_nwk.h>
```

Data Fields

- **zb_nwk_command_status_t status**
- **zb_uint16_t network_addr**

5.29.1 Detailed Description

Arguments of the NLME-STATUS.request routine.

5.29.2 Field Documentation

5.29.2.1 zb_nwk_command_status_t zb_nlme_status_indication_s::status

Error code associated with the failure

5.29.2.2 zb_uint16_t zb_nlme_status_indication_s::network_addr

The network device address associated with the status information

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

- zb_nwk.h

5.30 ZB_PACKED_STRUCT Struct Reference

MAC PIB.

```
#include <zb_mac.h>
```

Data Fields

- **zb_uint16_t mac_ack_wait_duration**
- **zb_uint8_t mac_association_permit**
- **zb_uint8_t mac_auto_request**
- **zb_uint8_t mac_batt_life_ext**
- zb_mac_beacon_payload_t **mac_beacon_payload**
- **zb_uint8_t mac_beacon_payload_length**
- **zb_uint8_t mac_beacon_order**
- **zb_uint8_t mac_bsn**
- **zb_ieee_addr_t mac_coord_extended_address**
- **zb_uint16_t mac_coord_short_address**
- **zb_uint8_t mac_dsn**
- **zb_uint16_t mac_pan_id**
- **zb_uint8_t mac_rx_on_when_idle**
- **zb_uint16_t mac_short_address**
- **zb_uint16_t mac_superframe_order**
- **zb_uint8_t mac_max_frame_retries**
- **zb_uint8_t phy_current_page**
- **zb_uint8_t phy_current_channel**
- **zb_ieee_addr_t mac_extended_address**

5.30.1 Detailed Description

MAC PIB.

5.30.2 Field Documentation

5.30.2.1 zb_uint16_t ZB_PACKED_STRUCT::mac_ack_wait_duration

The maximum number of symbols to wait for an acknowledgment frame to arrive following a transmitted data frame. The commencement time is described in 7.5.6.4.2.

5.30.2.2 zb_uint8_t ZB_PACKED_STRUCT::mac_association_permit

Indication of whether a coordinator is currently allowing association. A value of TRUE indicates that association is permitted.

5.30.2.3 zb_uint8_t ZB_PACKED_STRUCT::mac_auto_request

Indication of whether a device automatically sends a data request command if its address is listed in the beacon frame. indication primitive (see 7.1.5.1.2).

5.30.2.4 zb_uint8_t ZB_PACKED_STRUCT::mac_batt_life_ext

Indication of whether BLE, through the reduction of coordinator receiver operation time during the CAP, is enabled. Also, see 7.5.1.4 for an explanation.

5.30.2.5 zb_mac_beacon_payload_t ZB_PACKED_STRUCT::mac_beacon_payload

The contents of the beacon payload.

5.30.2.6 zb_uint8_t ZB_PACKED_STRUCT::mac_beacon_payload_length

The length, in octets, of the beacon payload.

5.30.2.7 zb_uint8_t ZB_PACKED_STRUCT::mac_beacon_order

Specification of how often the coordinator transmits its beacon.

5.30.2.8 zb_uint8_t ZB_PACKED_STRUCT::mac_bsn

The sequence number added to the transmitted beacon frame.

5.30.2.9 zb_ieee_addr_t ZB_PACKED_STRUCT::mac_coord_extended_address

The 64-bit address of the coordinator through which the device is associated.

5.30.2.10 zb_uint16_t ZB_PACKED_STRUCT::mac_coord_short_address

The 16-bit short address assigned to the coordinator through which the device is associated.

5.30.2.11 zb_uint8_t ZB_PACKED_STRUCT::mac_dsn

The sequence number added to the transmitted data or MAC command frame.

5.30.2.12 zb_uint16_t ZB_PACKED_STRUCT::mac_pan_id

The 16-bit identifier of the PAN on which the device is operating. If this value is 0xffff, the device is not associated.

5.30.2.13 zb_uint8_t ZB_PACKED_STRUCT::mac_rx_on_when_idle

Indication of whether the MAC sublayer is to enable its receiver during idle periods.

5.30.2.14 zb_uint16_t ZB_PACKED_STRUCT::mac_short_address

The 16-bit address that the device uses to communicate in the PAN.

5.30.2.15 zb_uint16_t ZB_PACKED_STRUCT::mac_superframe_order

The length of the active portion of the outgoing superframe, including the beacon frame.

5.30.2.16 zb_uint8_t ZB_PACKED_STRUCT::mac_max_frame_retries

The maximum number of retries allowed after a transmission failure.

5.30.2.17 zb_ieee_addr_t ZB_PACKED_STRUCT::mac_extended_address

The 64-bit (IEEE) address assigned to the device.

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

- zb_mac.h

5.31 zb_sched_globals_s Struct Reference

Data structures for the delayed execution.

`#include <zb_scheduler.h>`

Public Member Functions

- **ZB_LIST_DEFINE** (**zb_tm_q_ent_t** ∗, tm_queue)
- **ZB_STK_DEFINE** (**zb_tm_q_ent_t** ∗, tm_freelist)
- **ZB_SL_LIST_DEFINE** (**zb_buf_q_ent_t** ∗, inbuf_queue)
- **ZB_SL_LIST_DEFINE** (**zb_buf_q_ent_t** ∗, outbuf_queue)
- **ZB_STK_DEFINE** (**zb_buf_q_ent_t** ∗, buf_freelist)

Data Fields

- zb_cb_q_t **cb_q**
- **zb_uint8_t mac_receive_pending**
- zb_mac_tx_q_t **mac_tx_q**
- **zb_tm_q_ent_t tm_buffer** [**ZB_SCHEDULER_Q_SIZE**]
- **zb_buf_q_ent_t delayed_buf** [**ZB_BUF_Q_SIZE**]

5.31.1 Detailed Description

Data structures for the delayed execution.

5.31.2 Member Function Documentation

5.31.2.1 zb_sched_globals_s::ZB_LIST_DEFINE (zb_tm_q_ent_t ∗ , tm_queue)

delayed callbacks queue

5.31.2.2 zb_sched_globals_s::ZB_STK_DEFINE (zb_tm_q_ent_t ∗ , tm_freelist)

freelist of the timer queue entries

5.31.3 Field Documentation

5.31.3.1 zb_cb_q_t zb_sched_globals_s::cb_q

immediate callbacks queue

5.31.3.2 zb_tm_q_ent_t zb_sched_globals_s::tm_buffer[ZB_SCHEDULER_Q_SIZE]

buffer for the timer queue entries

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

- zb_scheduler.h

5.32 zb_tm_q_ent_s Struct Reference

Delayed (scheduled to run after timeout) callbacks queue entry.

```
#include <zb_scheduler.h>
```

Public Member Functions

- **ZB_LIST_FIELD** (struct **zb_tm_q_ent_s** ∗, next)

Data Fields

- **zb_callback_t func**
- **zb_uint8_t param**
- **zb_time_t run_time**

5.32.1 Detailed Description

Delayed (scheduled to run after timeout) callbacks queue entry.

5.32.2 Field Documentation

5.32.2.1 zb_callback_t zb_tm_q_ent_s::func

function to call

5.32.2.2 zb_uint8_t zb_tm_q_ent_s::param

parameter to pass to 'func'

5.32.2.3 zb_time_t zb_tm_q_ent_s::run_time

time to run at

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

- zb_scheduler.h

5.33 zb_zdo_active_ep_req_s Struct Reference

Parameters of Active_desc_req primitive.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t nwk_addr**

5.33.1 Detailed Description

Parameters of Active_desc_req primitive.

To be put into buffer as data (means - after space alloc).

5.33.2 Field Documentation

5.33.2.1 zb_uint16_t zb_zdo_active_ep_req_s::nwk_addr

NWK address that is used for IEEE address mapping.

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

- zb_zdo.h

5.34 zb_zdo_bind_req_head_s Struct Reference

2.4.3.2.2 Bind_req request head send to the remote

```
#include <zb_zdo.h>
```

Data Fields

- **zb_ieee_addr_t src_address**
- **zb_uint8_t src_endp**
- **zb_uint16_t cluster_id**
- **zb_uint8_t dst_addr_mode**

5.34.1 Detailed Description

2.4.3.2.2 Bind_req request head send to the remote

5.34.2 Field Documentation

5.34.2.1 zb_ieee_addr_t zb_zdo_bind_req_head_s::src_address

The IEEE address for the source.

5.34.2.2 zb_uint8_t zb_zdo_bind_req_head_s::src_endp

The source endpoint for the binding entry.

5.34.2.3 zb_uint16_t zb_zdo_bind_req_head_s::cluster_id

The identifier of the cluster on the source device that is bound to the destination.

5.34.2.4 zb_uint8_t zb_zdo_bind_req_head_s::dst_addr_mode

The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 . 0xff = reserved

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

- zb_zdo.h

5.35 zb_zdo_bind_req_param_s Struct Reference

Parameters for 2.4.3.2.2 Bind_req API call.

`#include <zb_zdo.h>`

Data Fields

- **zb_ieee_addr_t src_address**
- **zb_uint8_t src_endp**
- **zb_uint16_t cluster_id**
- **zb_uint8_t dst_addr_mode**
- union **zb_addr_u dst_address**
- **zb_uint8_t dst_endp**
- **zb_uint16_t req_dst_addr**

5.35.1 Detailed Description

Parameters for 2.4.3.2.2 Bind_req API call.

5.35.2 Field Documentation

5.35.2.1 zb_ieee_addr_t zb_zdo_bind_req_param_s::src_address

The IEEE address for the source.

5.35.2.2 zb_uint8_t zb_zdo_bind_req_param_s::src_endp

The source endpoint for the binding entry.

5.35.2.3 zb_uint16_t zb_zdo_bind_req_param_s::cluster_id

The identifier of the cluster on the source device that is bound to the destination.

5.35.2.4 zb_uint8_t zb_zdo_bind_req_param_s::dst_addr_mode

The addressing mode for the destination address used in this command. This field can take one of the non-reserved values from the following list: 0x00 = reserved 0x01 = 16-bit group address for DstAddress and DstEndp not present 0x02 = reserved 0x03 = 64-bit extended address for DstAddress and DstEndp present 0x04 . 0xff = reserved

5.35.2.5 union zb_addr_u zb_zdo_bind_req_param_s::dst_address

The destination address for the binding entry.

5.35.2.6 zb_uint8_t zb_zdo_bind_req_param_s::dst_endp

This field shall be present only if the DstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

5.35.2.7 zb_uint16_t zb_zdo_bind_req_param_s::req_dst_addr

Destinition address of the request

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

- zb_zdo.h

5.36 zb_zdo_bind_req_tail_1_s Struct Reference

2.4.3.2.2 Bind_req request tail 1st variant send to the remote

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t dst_addr**

5.36.1 Detailed Description

2.4.3.2.2 Bind_req request tail 1st variant send to the remote

5.36.2 Field Documentation

5.36.2.1 zb_uint16_t zb_zdo_bind_req_tail_1_s::dst_addr

The destination address for the binding entry.

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

- zb_zdo.h

5.37 zb_zdo_bind_req_tail_2_s Struct Reference

2.4.3.2.2 Bind_req request tail 2nd variant send to the remote

```
#include <zb_zdo.h>
```

Data Fields

- **zb_ieee_addr_t dst_addr**
- **zb_uint8_t dst_endp**

5.37.1 Detailed Description

2.4.3.2.2 Bind_req request tail 2nd variant send to the remote

5.37.2 Field Documentation

5.37.2.1 zb_ieee_addr_t zb_zdo_bind_req_tail_2_s::dst_addr

The destination address for the binding entry.

5.37.2.2 zb_uint8_t zb_zdo_bind_req_tail_2_s::dst_endp

The destination address for the binding entry.

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

- zb_zdo.h

5.38 zb_zdo_bind_resp_s Struct Reference

Data Fields

- **zb_uint8_t status**

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

- zb_zdo.h

5.39 zb_zdo_configuration_attributes_e Struct Reference

Data Fields

- zb_af_node_desc_t **node_desc**
- zb_af_node_power_desc_t **node_power_desc**
- zb_af_simple_desc_7_8_t **zdo_simple_desc**
- zb_af_simple_desc_1_1_t ∗ **simple_desc_list** [ZB_MAX_EP_NUMBER]
- **zb_uint8_t simple_desc_number**
- **zb_uint8_t nwk_scan_attempts**
- **zb_uint16_t nwk_time_btwn_scans**
- **zb_uint8_t enddev_bind_timeout**
- **zb_time_t nwk_indirect_poll_rate**
- **zb_uint8_t permit_join_duration**

5.39.1 Field Documentation

5.39.1.1 zb_uint8_t zb_zdo_configuration_attributes_e::permit_join_duration

Permit join duration, 0x00 - disable join, 0xff - join is allowed forever

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

- zb_zdo_globals.h

5.40 zb_zdo_desc_resp_hdr_s Struct Reference

Header of Node_desc_resp primitive.

`#include <zb_zdo.h>`

Data Fields

- **zb_zdp_status_t status**
- **zb_uint16_t nwk_addr**

5.40.1 Detailed Description

Header of Node_desc_resp primitive.

5.40.2 Field Documentation

5.40.2.1 zb_zdp_status_t zb_zdo_desc_resp_hdr_s::status

The status of the Desc_req command

5.40.2.2 zb_uint16_t zb_zdo_desc_resp_hdr_s::nwk_addr

NWK address for the request

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

- zb_zdo.h

5.41 zb_zdo_end_device_bind_req_head_s Struct Reference

2.4.3.2.1 End_Device_Bind_req command head

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t binding_target**
- **zb_ieee_addr_t src_ieee_addr**
- **zb_uint8_t src_endp**
- **zb_uint16_t profile_id**
- **zb_uint8_t num_in_cluster**

5.41.1 Detailed Description

2.4.3.2.1 End_Device_Bind_req command head

5.41.2 Field Documentation

5.41.2.1 zb_uint16_t zb_zdo_end_device_bind_req_head_s::binding_target

The address of the target for the binding. This can be either the primary binding cache device or the short address of the local device.

5.41.2.2 zb_ieee_addr_t zb_zdo_end_device_bind_req_head_s::src_ieee_addr

The IEEE address of the device generating the request

5.41.2.3 zb_uint8_t zb_zdo_end_device_bind_req_head_s::src_endp

The endpoint on the device generating the request

5.41.2.4 zb_uint16_t zb_zdo_end_device_bind_req_head_s::profile_id

ProfileID which is to be matched between two End_Device_Bind_req received at the ZigBee Coordinator

5.41.2.5 zb_uint8_t zb_zdo_end_device_bind_req_head_s::num_in_cluster

The number of Input Clusters provided for end device binding within the InClusterList.

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

- zb_zdo.h

5.42 zb_zdo_end_device_bind_req_tail_s Struct Reference

2.4.3.2.1 End_Device_Bind_req command head

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint8_t num_out_cluster**

5.42.1 Detailed Description

2.4.3.2.1 End_Device_Bind_req command head

5.42.2 Field Documentation

5.42.2.1 zb_uint8_t zb_zdo_end_device_bind_req_tail_s::num_out_cluster

The number of Output Clusters provided for matching within OutClusterList

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

- zb_zdo.h

5.43 zb_zdo_end_device_bind_resp_s Struct Reference

Data Fields

- **zb_uint8_t status**

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

- zb_zdo.h

5.44 zb_zdo_ep_resp_s Struct Reference

Active EP response.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint8_t status**
- **zb_uint16_t nwk_addr**
- **zb_uint8_t ep_count**

5.44.1 Detailed Description

Active EP response.

5.44.2 Field Documentation

5.44.2.1 zb_uint8_t zb_zdo_ep_resp_s::status

The status of the Active_EP_req command.

5.44.2.2 zb_uint16_t zb_zdo_ep_resp_s::nwk_addr

NWK address for the request.

5.44.2.3 zb_uint8_t zb_zdo_ep_resp_s::ep_count

The count of active endpoints on the Remote Device.

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

- zb_zdo.h

5.45 zb_zdo_ieee_addr_req_s Struct Reference

Parameters of IEEE_addr_req primitive.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t nwk_addr**
- **zb_uint8_t request_type**
- **zb_uint8_t start_index**

5.45.1 Detailed Description

Parameters of IEEE_addr_req primitive.

To be put into buffer as data (means - after space alloc).

5.45.2 Field Documentation

5.45.2.1 zb_uint16_t zb_zdo_ieee_addr_req_s::nwk_addr

NWK address that is used for IEEE address mapping.

5.45.2.2 zb_uint8_t zb_zdo_ieee_addr_req_s::request_type

Request type for this command: 0x00 Single device response 0x01 Extended response

5.45.2.3 zb_uint8_t zb_zdo_ieee_addr_req_s::start_index

If the Request type for this command is Extended response, the StartIndex provides the starting index for the requested elements of the associated devices list

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

- zb_zdo.h

5.46 zb_zdo_match_desc_param_s Struct Reference

Parameters of match_desc_req primitive.

`#include <zb_zdo.h>`

Data Fields

- **zb_uint16_t nwk_addr**
- **zb_uint16_t profile_id**
- **zb_uint8_t num_in_clusters**
- **zb_uint8_t num_out_clusters**
- **zb_uint16_t cluster_list** [1]

5.46.1 Detailed Description

Parameters of match_desc_req primitive.

To be put into buffer as data (means - after space alloc).

5.46.2 Field Documentation

5.46.2.1 zb_uint16_t zb_zdo_match_desc_param_s::nwk_addr

NWK address that is used for IEEE address mapping.

5.46.2.2 zb_uint16_t zb_zdo_match_desc_param_s::profile_id

Profile ID to be matched at the destination.

5.46.2.3 zb_uint8_t zb_zdo_match_desc_param_s::num_in_clusters

The number of Input Clusters provided for matching within the InClusterList.

5.46.2.4 zb_uint8_t zb_zdo_match_desc_param_s::num_out_clusters

The number of Output Clusters provided for matching within OutClusterList.

5.46.2.5 zb_uint16_t zb_zdo_match_desc_param_s::cluster_list[1]

variable size: [num_in_clusters] + [num_out_clusters] List of Input ClusterIDs to be used for matching; the InClusterList is the desired list to be matched by the Remote Device (the elements of the InClusterList are the supported output clusters of the Local Device). List of Output ClusterIDs to be used for matching; the OutClusterList is the desired list to be matched by the Remote Device (the elements of the OutClusterList are the supported input clusters of the Local Device).

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

- zb_zdo.h

5.47 zb_zdo_match_desc_req_head_s Struct Reference

Match_desc_req head.

`#include <zb_zdo.h>`

Data Fields

- **zb_uint16_t nwk_addr**
- **zb_uint16_t profile_id**
- **zb_uint8_t num_in_clusters**

5.47.1 Detailed Description

Match_desc_req head.

5.47.2 Field Documentation

5.47.2.1 zb_uint16_t zb_zdo_match_desc_req_head_s::nwk_addr

NWK address that is used for IEEE address mapping.

5.47.2.2 zb_uint16_t zb_zdo_match_desc_req_head_s::profile_id

Profile ID to be matched at the destination.

5.47.2.3 zb_uint8_t zb_zdo_match_desc_req_head_s::num_in_clusters

The number of Input Clusters provided for matching within the InClusterList.

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

- zb_zdo.h

5.48 zb_zdo_match_desc_req_tail_s Struct Reference

Match_desc_req tail.

`#include <zb_zdo.h>`

Data Fields

- **zb_uint8_t num_out_clusters**

5.48.1 Detailed Description

Match_desc_req tail.

5.48.2 Field Documentation

5.48.2.1 zb_uint8_t zb_zdo_match_desc_req_tail_s::num_out_clusters

The number of Output Clusters provided for matching within OutClusterList.

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

- zb_zdo.h

5.49 zb_zdo_match_desc_resp_s Struct Reference

2.4.4.1.7 Match_Desc_rsp response structure

`#include <zb_zdo.h>`

Data Fields

- **zb_uint8_t status**
- **zb_uint16_t nwk_addr**
- **zb_uint8_t match_len**

5.49.1 Detailed Description

2.4.4.1.7 Match_Desc_rsp response structure

5.49.2 Field Documentation

5.49.2.1 zb_uint8_t zb_zdo_match_desc_resp_s::status

The status of the Match_Desc_req command.

5.49.2.2 zb_uint16_t zb_zdo_match_desc_resp_s::nwk_addr

NWK address for the request.

5.49.2.3 zb_uint8_t zb_zdo_match_desc_resp_s::match_len

The count of endpoints on the Remote Device that match the request criteria.

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

- zb_zdo.h

5.50 zb_zdo_mgmt_leave_param_s Struct Reference

Request for 2.4.3.3.5 Mgmt_Leave_req.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_ieee_addr_t device_address**
- **zb_uint16_t dst_addr**
- **zb_bitfield_t reserved**:6
- **zb_bitfield_t remove_children**:1
- **zb_bitfield_t rejoin**:1

5.50.1 Detailed Description

Request for 2.4.3.3.5 Mgmt_Leave_req.

Problem in the specification: in 2.4.3.3.5 Mgmt_Leave_req only one DeviceAddress exists. But, in such case it is impossible to satisfy 2.4.3.3.5.1: "The Mgmt_Leave_req is generated from a Local Device requesting that a Remote Device leave the network or to request that another device leave the network." Also, in the PRO TC document, 14.2-TP/NWK/BV-04 ZR-ZDO-APL RX Join/Leave is following note: "gZC sends Mgmt_Leave.request with DevAddr=all zero, DstAddr=ZR"

5.50.2 Field Documentation

5.50.2.1 zb_ieee_addr_t zb_zdo_mgmt_leave_param_s::device_address

64 bit IEEE address

5.50.2.2 zb_uint16_t zb_zdo_mgmt_leave_param_s::dst_addr

destinition address. Not defined in the spac - let's it be short address

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

- zb_zdo.h

5.51 zb_zdo_mgmt_leave_req_s Struct Reference

Request for 2.4.3.3.5 Mgmt_Leave_req.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_ieee_addr_t device_address**
- **zb_bitfield_t reserved**:6
- **zb_bitfield_t remove_children**:1
- **zb_bitfield_t rejoin**:1

5.51.1 Detailed Description

Request for 2.4.3.3.5 Mgmt_Leave_req.

5.51.2 Field Documentation

5.51.2.1 zb_ieee_addr_t zb_zdo_mgmt_leave_req_s::device_address

64 bit IEEE address

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

- zb_zdo.h

5.52 zb_zdo_mgmt_leave_res_s Struct Reference

Response for 2.4.4.3.5 Mgmt_Leave_rsp.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint8_t status**

5.52.1 Detailed Description

Response for 2.4.4.3.5 Mgmt_Leave_rsp.

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

- zb_zdo.h

5.53 zb_zdo_mgmt_lqi_param_s Struct Reference

Parameters for 2.4.3.3.2 Mgmt_Lqi_req.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint8_t start_index**
- **zb_uint16_t dst_addr**

5.53.1 Detailed Description

Parameters for 2.4.3.3.2 Mgmt_Lqi_req.

5.53.2 Field Documentation

5.53.2.1 zb_uint8_t zb_zdo_mgmt_lqi_param_s::start_index

Starting Index for the requested elements of the Neighbor Table

5.53.2.2 zb_uint16_t zb_zdo_mgmt_lqi_param_s::dst_addr

destinition address

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

- zb_zdo.h

5.54 zb_zdo_mgmt_lqi_req_s Struct Reference

Request for 2.4.3.3.2 Mgmt_Lqi_req.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint8_t start_index**

5.54.1 Detailed Description

Request for 2.4.3.3.2 Mgmt_Lqi_req.

5.54.2 Field Documentation

5.54.2.1 zb_uint8_t zb_zdo_mgmt_lqi_req_s::start_index

Starting Index for the requested elements of the Neighbor Table

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

- zb_zdo.h

5.55 zb_zdo_mgmt_lqi_resp_s Struct Reference

Response for 2.4.4.3.2 Mgmt_Lqi_rsp.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint8_t status**
- **zb_uint8_t neighbor_table_entries**
- **zb_uint8_t start_index**
- **zb_uint8_t neighbor_table_list_count**

5.55.1 Detailed Description

Response for 2.4.4.3.2 Mgmt_Lqi_rsp.

5.55.2 Field Documentation

5.55.2.1 zb_uint8_t zb_zdo_mgmt_lqi_resp_s::status

The status of the Mgmt_Lqi_req command.

5.55.2.2 zb_uint8_t zb_zdo_mgmt_lqi_resp_s::neighbor_table_entries

Total number of Neighbor Table entries within the Remote Device

5.55.2.3 zb_uint8_t zb_zdo_mgmt_lqi_resp_s::start_index

Starting index within the Neighbor Table to begin reporting for the NeighborTableList.

5.55.2.4 zb_uint8_t zb_zdo_mgmt_lqi_resp_s::neighbor_table_list_count

Number of Neighbor Table entries included within NeighborTableList

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

- zb_zdo.h

5.56 zb_zdo_mgmt_nwk_update_notify_hdr_s Struct Reference

Header parameters for mgmt_nwk_update_notify.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint8_t status**
- **zb_uint32_t scanned_channels**
- **zb_uint16_t total_transmissions**
- **zb_uint16_t transmission_failures**
- **zb_uint8_t scanned_channels_list_count**

5.56.1 Detailed Description

Header parameters for mgmt_nwk_update_notify.

5.56.2 Field Documentation

5.56.2.1 zb_uint8_t zb_zdo_mgmt_nwk_update_notify_hdr_s::status

The status of the Mgmt_NWK_Update_notify command.

5.56.2.2 zb_uint32_t zb_zdo_mgmt_nwk_update_notify_hdr_s::scanned_channels

List of channels scanned by the request

5.56.2.3 zb_uint16_t zb_zdo_mgmt_nwk_update_notify_hdr_s::total_transmissions

Count of the total transmissions reported by the device

5.56.2.4 zb_uint16_t zb_zdo_mgmt_nwk_update_notify_hdr_s::transmission_failures

Sum of the total transmission failures reported by the device

5.56.2.5 zb_uint8_t zb_zdo_mgmt_nwk_update_notify_hdr_s::scanned_channels_list_count

The list shall contain the number of records contained in the EnergyValues parameter.

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

- zb_zdo.h

5.57 zb_zdo_mgmt_nwk_update_notify_param_s Struct Reference

Parameters for mgmt_nwk_update_notify.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_zdo_mgmt_nwk_update_notify_hdr_t hdr**
- **zb_uint8_t energy_values** [ZB_MAC_SUPPORTED_CHANNELS]
- **zb_uint16_t dst_addr**
- **zb_uint8_t tsn**

5.57.1 Detailed Description

Parameters for mgmt_nwk_update_notify.

5.57.2 Field Documentation

5.57.2.1 zb_zdo_mgmt_nwk_update_notify_hdr_t zb_zdo_mgmt_nwk_update_notify_param_s::hdr

Fixed parameters set

5.57.2.2 zb_uint8_t zb_zdo_mgmt_nwk_update_notify_param_s::energy_values[ZB_MAC_SUPPORTED_CHANNELS]

ed scan values

5.57.2.3 zb_uint16_t zb_zdo_mgmt_nwk_update_notify_param_s::dst_addr

destinition address

5.57.2.4 zb_uint8_t zb_zdo_mgmt_nwk_update_notify_param_s::tsn

tsn value

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

- zb_zdo.h

5.58 zb_zdo_mgmt_nwk_update_req_hdr_s Struct Reference

Header of parameters for Mgmt_NWK_Update_req.

`#include <zb_zdo.h>`

Data Fields

- **zb_uint32_t scan_channels**
- **zb_uint8_t scan_duration**

5.58.1 Detailed Description

Header of parameters for Mgmt_NWK_Update_req.

5.58.2 Field Documentation

5.58.2.1 zb_uint32_t zb_zdo_mgmt_nwk_update_req_hdr_s::scan_channels

Channels bitmask

5.58.2.2 zb_uint8_t zb_zdo_mgmt_nwk_update_req_hdr_s::scan_duration

A value used to calculate the length of time to spend scanning each channel.

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

- zb_zdo.h

5.59 zb_zdo_mgmt_nwk_update_req_s Struct Reference

Parameters for Mgmt_NWK_Update_req.

`#include <zb_zdo.h>`

Data Fields

- **zb_zdo_mgmt_nwk_update_req_hdr_t hdr**
- **zb_uint8_t scan_count**
- **zb_uint8_t update_id**
- **zb_uint16_t manager_addr**
- **zb_uint16_t dst_addr**

5.59.1 Detailed Description

Parameters for Mgmt_NWK_Update_req.

5.59.2 Field Documentation

5.59.2.1 zb_zdo_mgmt_nwk_update_req_hdr_t zb_zdo_mgmt_nwk_update_req_s::hdr

Request header

5.59.2.2 zb_uint8_t zb_zdo_mgmt_nwk_update_req_s::scan_count

This field represents the number of energy scans to be conducted and reported

5.59.2.3 zb_uint8_t zb_zdo_mgmt_nwk_update_req_s::update_id

This value is set by the Network Channel Manager prior to sending the message. This field shall only be present of the ScanDuration is 0xfe or 0xff

5.59.2.4 zb_uint16_t zb_zdo_mgmt_nwk_update_req_s::manager_addr

This field shall be present only if the ScanDuration is set to 0xff, and, where present, indicates the NWK address for the device with the Network Manager bit set in its Node Descriptor.

5.59.2.5 zb_uint16_t zb_zdo_mgmt_nwk_update_req_s::dst_addr

Destinition address

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

- zb_zdo.h

5.60 zb_zdo_mgmt_permit_joining_req_param_s Struct Reference

Parameters for zb_zdo_mgmt_permit_joining_req.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t dest_addr**
- **zb_uint8_t permit_duration**
- **zb_uint8_t tc_significance**

5.60.1 Detailed Description

Parameters for zb_zdo_mgmt_permit_joining_req.

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

- zb_zdo.h

5.61 zb_zdo_mgmt_permit_joining_req_s Struct Reference

Parameters for 2.4.3.3.7 Mgmt_Permit_Joining_req.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint8_t permit_duration**
- **zb_uint8_t tc_significance**

5.61.1 Detailed Description

Parameters for 2.4.3.3.7 Mgmt_Permit_Joining_req.

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

- zb_zdo.h

5.62 zb_zdo_neighbor_table_record_s Struct Reference

NeighborTableList Record Format for mgmt_lqi_resp.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_ext_pan_id_t ext_pan_id**
- **zb_ieee_addr_t ext_addr**
- **zb_uint16_t network_addr**
- **zb_uint8_t type_flags**
- **zb_uint8_t permit_join**
- **zb_uint8_t depth**
- **zb_uint8_t lqi**

5.62.1 Detailed Description

NeighborTableList Record Format for mgmt_lqi_resp.

5.62.2 Field Documentation

5.62.2.1 zb_ext_pan_id_t zb_zdo_neighbor_table_record_s::ext_pan_id

The 64-bit extended PAN identifier of the neighboring device.

5.62.2.2 zb_ieee_addr_t zb_zdo_neighbor_table_record_s::ext_addr

64-bit IEEE address that is unique to every device.

5.62.2.3 zb_uint16_t zb_zdo_neighbor_table_record_s::network_addr

The 16-bit network address of the neighboring device

5.62.2.4 zb_uint8_t zb_zdo_neighbor_table_record_s::type_flags

device type, rx_on_when_idle, relationship

5.62.2.5 zb_uint8_t zb_zdo_neighbor_table_record_s::permit_join

An indication of whether the neighbor device is accepting join requests

5.62.2.6 zb_uint8_t zb_zdo_neighbor_table_record_s::depth

The tree depth of the neighbor device.

5.62.2.7 zb_uint8_t zb_zdo_neighbor_table_record_s::lqi

The estimated link quality for RF transmissions from this device

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

- zb_zdo.h

5.63 zb_zdo_node_desc_req_s Struct Reference

Parameters of Node_desc_req primitive.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t nwk_addr**

5.63.1 Detailed Description

Parameters of Node_desc_req primitive.

To be put into buffer as data (means - after space alloc).

5.63.2 Field Documentation

5.63.2.1 zb_uint16_t zb_zdo_node_desc_req_s::nwk_addr

NWK address that is used for IEEE address mapping.

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

- zb_zdo.h

5.64 zb_zdo_node_desc_resp_s Struct Reference

Parameters of Node_desc_resp primitive.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_zdo_desc_resp_hdr_t hdr**
- zb_af_node_desc_t **node_desc**

5.64.1 Detailed Description

Parameters of Node_desc_resp primitive.

5.64.2 Field Documentation

5.64.2.1 zb_zdo_desc_resp_hdr_t zb_zdo_node_desc_resp_s::hdr

header for response

5.64.2.2 zb_af_node_desc_t zb_zdo_node_desc_resp_s::node_desc

Node Descriptor

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

- zb_zdo.h

5.65 zb_zdo_nwk_addr_req_param_s Struct Reference

Parameters for nwk_addr_req command.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t dst_addr**
- **zb_ieee_addr_t ieee_addr**
- **zb_uint8_t request_type**
- **zb_uint8_t start_index**

5.65.1 Detailed Description

Parameters for nwk_addr_req command.

5.65.2 Field Documentation

5.65.2.1 zb_uint16_t zb_zdo_nwk_addr_req_param_s::dst_addr

Destinitions address

5.65.2.2 zb_ieee_addr_t zb_zdo_nwk_addr_req_param_s::ieee_addr

The IEEE address to be matched by the Remote Device

5.65.2.3 zb_uint8_t zb_zdo_nwk_addr_req_param_s::request_type

Request type for this command: 0x00 Single device response 0x01 Extended response

5.65.2.4 zb_uint8_t zb_zdo_nwk_addr_req_param_s::start_index

If the Request type for this command is Extended response, the StartIndex provides the starting index for the requested elements of the associated devices list

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

- zb_zdo.h

5.66 zb_zdo_nwk_addr_req_s Struct Reference

NWK_addr_req command primitive.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_ieee_addr_t ieee_addr**
- **zb_uint8_t request_type**
- **zb_uint8_t start_index**

5.66.1 Detailed Description

NWK_addr_req command primitive.

5.66.2 Field Documentation

5.66.2.1 zb_ieee_addr_t zb_zdo_nwk_addr_req_s::ieee_addr

The IEEE address to be matched by the Remote Device

5.66.2.2 zb_uint8_t zb_zdo_nwk_addr_req_s::request_type

Request type for this command: 0x00 Single device response 0x01 Extended response

5.66.2.3 zb_uint8_t zb_zdo_nwk_addr_req_s::start_index

If the Request type for this command is Extended response, the StartIndex provides the starting index for the requested elements of the associated devices list

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

- zb_zdo.h

5.67 zb_zdo_nwk_addr_resp_head_s Struct Reference

Data Fields

- **zb_uint8_t status**
- **zb_ieee_addr_t ieee_addr**
- **zb_uint16_t nwk_addr**

5.67.1 Field Documentation

5.67.1.1 zb_uint8_t zb_zdo_nwk_addr_resp_head_s::status

The status of the NWK_addr_req command.

5.67.1.2 zb_ieee_addr_t zb_zdo_nwk_addr_resp_head_s::ieee_addr

64-bit address for the Remote Device.

5.67.1.3 zb_uint16_t zb_zdo_nwk_addr_resp_head_s::nwk_addr

16-bit address for the Remote Device.

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

- zb_zdo.h

5.68 zb_zdo_power_desc_req_s Struct Reference

Parameters of Power_desc_req primitive.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t nwk_addr**

5.68.1 Detailed Description

Parameters of Power_desc_req primitive.

To be put into buffer as data (means - after space alloc).

5.68.2 Field Documentation

5.68.2.1 zb_uint16_t zb_zdo_power_desc_req_s::nwk_addr

NWK address that is used for IEEE address mapping.

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

- zb_zdo.h

5.69 zb_zdo_power_desc_resp_s Struct Reference

Parameters of Power_desc_resp primitive.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_zdo_desc_resp_hdr_t hdr**
- zb_af_node_power_desc_t **power_desc**

5.69.1 Detailed Description

Parameters of Power_desc_resp primitive.

5.69.2 Field Documentation

5.69.2.1 zb_zdo_desc_resp_hdr_t zb_zdo_power_desc_resp_s::hdr

header for response

5.69.2.2 zb_af_node_power_desc_t zb_zdo_power_desc_resp_s::power_desc

Power Descriptor

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

- zb_zdo.h

5.70 zb_zdo_simple_desc_req_s Struct Reference

Parameters of Power_desc_req primitive.

```
#include <zb_zdo.h>
```

Data Fields

- **zb_uint16_t nwk_addr**
- **zb_uint8_t endpoint**

5.70.1 Detailed Description

Parameters of Power_desc_req primitive.

To be put into buffer as data (means - after space alloc).

5.70.2 Field Documentation

5.70.2.1 zb_uint16_t zb_zdo_simple_desc_req_s::nwk_addr

NWK address that is used for IEEE address mapping.

5.70.2.2 zb_uint8_t zb_zdo_simple_desc_req_s::endpoint

The endpoint on the destination

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

- zb_zdo.h

5.71 zb_zdo_simple_desc_resp_hdr_s Struct Reference

Header of Node_desc_resp primitive.

`#include <zb_zdo.h>`

Data Fields

- **zb_zdp_status_t status**
- **zb_uint16_t nwk_addr**
- **zb_uint8_t length**

5.71.1 Detailed Description

Header of Node_desc_resp primitive.

5.71.2 Field Documentation

5.71.2.1 zb_zdp_status_t zb_zdo_simple_desc_resp_hdr_s::status

The status of the Desc_req command

5.71.2.2 zb_uint16_t zb_zdo_simple_desc_resp_hdr_s::nwk_addr

NWK address for the request

5.71.2.3 zb_uint8_t zb_zdo_simple_desc_resp_hdr_s::length

Length of the simple descriptor

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

- zb_zdo.h

5.72 zb_zdo_simple_desc_resp_s Struct Reference

Parameters of simple_desc_resp primitive.

`#include <zb_zdo.h>`

Data Fields

- **zb_zdo_simple_desc_resp_hdr_t hdr**
- zb_af_simple_desc_1_1_t **simple_desc**

5.72.1 Detailed Description

Parameters of simple_desc_resp primitive.

5.72.2 Field Documentation

5.72.2.1 zb_zdo_simple_desc_resp_hdr_t zb_zdo_simple_desc_resp_s::hdr

header for response

5.72.2.2 zb_af_simple_desc_1_1_t zb_zdo_simple_desc_resp_s::simple_desc

Simple Descriptor

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

- zb_zdo.h

5.73 zb_zdo_system_server_discovery_req_s Struct Reference

Request parameters for 2.4.3.1.13 System_Server_Discovery_req.

`#include <zb_zdo.h>`

Data Fields

- **zb_uint16_t server_mask**

5.73.1 Detailed Description

Request parameters for 2.4.3.1.13 System_Server_Discovery_req.

5.73.2 Field Documentation

5.73.2.1 zb_uint16_t zb_zdo_system_server_discovery_req_s::server_mask

Server mask for device discovery

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

- zb_zdo.h

5.74 zb_zdo_system_server_discovery_resp_s Struct Reference

Response parameters for 2.4.4.1.10 System_Server_Discovery_rsp.

`#include <zb_zdo.h>`

Data Fields

- **zb_uint8_t status**
- **zb_uint16_t server_mask**

5.74.1 Detailed Description

Response parameters for 2.4.4.1.10 System_Server_Discovery_rsp.

5.74.2 Field Documentation

5.74.2.1 zb_uint8_t zb_zdo_system_server_discovery_resp_s::status

Status of the operation

5.74.2.2 zb_uint16_t zb_zdo_system_server_discovery_resp_s::server_mask

Mask of the supported features

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

- zb_zdo.h

Index

AF functions visible to applications, 38
 zb_af_set_data_indication, 38
APS functions visible to applications, 39
 ZB_APS_ADDR_MODE_16_ENDP_PRESENT, 42
 ZB_APS_ADDR_MODE_16_GROUP_ENDP_NOT_PRESENT, 42
 ZB_APS_ADDR_MODE_64_ENDP_PRESENT, 42
 ZB_APS_ADDR_MODE_DST_ADDR_ENDP_NOT_PRESENT, 42
 ZB_APS_HDR_CUT, 41
 ZB_APS_HDR_CUT_P, 41
 ZB_APS_STATUS_INVALID_BINDING, 42
 ZB_APS_STATUS_INVALID_GROUP, 42
 ZB_APS_STATUS_INVALID_PARAMETER, 42
 ZB_APS_STATUS_NO_BOUND_DEVICE, 42
 ZB_APS_STATUS_NO_SHORT_ADDRESS, 43
 ZB_APS_STATUS_NOT_SUPPORTED, 43
 ZB_APS_STATUS_SECURED_LINK_KEY, 43
 ZB_APS_STATUS_SECURED_NWK_KEY, 43
 ZB_APS_STATUS_SECURITY_FAIL, 43
 ZB_APS_STATUS_SUCCESS, 42
 ZB_APS_STATUS_TABLE_FULL, 43
 ZB_APS_STATUS_UNSECURED, 43
 ZB_APS_STATUS_UNSUPPORTED_ATTRIBUTE, 43
 ZB_APSDE_TX_OPT_ACK_TX, 43
 ZB_APSDE_TX_OPT_FRAG_PERMITTED, 43
 ZB_APSDE_TX_OPT_SECURITY_ENABLED, 43
 ZB_APSDE_TX_OPT_USE_NWK_KEY, 43
 zb_aps_addr_mode_e, 42
 zb_aps_hdr_t, 42
 zb_aps_status_e, 42
 zb_apsde_data_indication_t, 42
 zb_apsde_data_req_t, 41
 zb_apsde_data_request, 40
 zb_apsde_tx_opt_e, 43
 zb_apsme_add_group_conf_t, 42
 zb_apsme_add_group_req_t, 42
 zb_apsme_binding_req_t, 41
APS Informational Base, 44
 ZB_APS_AIB_BINDING, 45
 ZB_APS_AIB_CHANNEL_MASK, 46
 ZB_APS_AIB_CHANNEL_TIMER, 46
 ZB_APS_AIB_DESIGNATED_COORD, 45
 ZB_APS_AIB_GROUP_TABLE, 46
 ZB_APS_AIB_INTERFRAME_DELAY, 46
 ZB_APS_AIB_LAST_CHANNEL_ENERGY, 46
 ZB_APS_AIB_LAST_CHANNEL_FAILURE_RATE, 46
 ZB_APS_AIB_NONMEMBER_RADIUS, 46
 ZB_APS_AIB_PERMISSION_CONFIG, 46
 ZB_APS_AIB_USE_EXT_PANID, 46
 ZB_APS_AIB_USE_INSECURE_JOIN, 46
 zb_aps_aib_attr_id_e, 45
 zb_aps_aib_attr_id_t, 45
 zb_apsme_get_confirm, 45
 zb_apsme_get_confirm_t, 45
 zb_apsme_get_request, 45
 zb_apsme_get_request_t, 45
 zb_apsme_set_confirm, 45
 zb_apsme_set_confirm_t, 45
 zb_apsme_set_request, 45
 zb_apsme_set_request_t, 45
addr_mode
 zb_apsde_data_req_s, 103
 zb_apsme_binding_req_s, 105
 zb_nlde_data_req_s, 114
aib_attr
 zb_apsme_get_confirm_s, 106
 zb_apsme_get_request_s, 107
 zb_apsme_set_confirm_s, 107
 zb_apsme_set_request_s, 108
aib_length
 zb_apsme_get_confirm_s, 106
 zb_apsme_set_request_s, 108
attribute_length
 zb_nlme_get_confirm_s, 115

Base typedefs, 76
 ZB_64BIT_ADDR_CMP, 79
 ZB_64BIT_ADDR_COPY, 79
 ZB_64BIT_ADDR_ZERO, 79
 ZB_ADDR_CMP, 79
 ZB_HTOLE16, 79
 ZB_INT8_C, 79
 ZB_IS_64BIT_ADDR_ZERO, 79
 ZB_LETOH16, 79
 ZB_SHORT_MIN, 79
 zb_64bit_addr_t, 81
 zb_bitfield_t, 80
 zb_bool_e, 81
 zb_bool_t, 80
 zb_char_t, 80
 zb_ext_pan_id_t, 81
 zb_ieee_addr_t, 81
 zb_int16_t, 80
 zb_int32_t, 80

zb_int8_t, 80
zb_int_t, 81
zb_long_t, 81
zb_put_next_htole16, 78
zb_sbitfield_t, 80
zb_short_t, 81
zb_uchar_t, 80
zb_uint16_t, 80
zb_uint32_t, 80
zb_uint8_t, 80
zb_uint_t, 81
zb_ulong_t, 81
zb_ushort_t, 81
zb_voidp_t, 81
binding_target
zb_zdo_end_device_bind_req_head_s, 127

cb_q
zb_sched_globals_s, 121
cluster_id
zb_zdo_bind_req_head_s, 123
zb_zdo_bind_req_param_s, 124
cluster_list
zb_end_device_bind_req_param_s, 111
zb_zdo_match_desc_param_s, 131
clusterid
zb_apsde_data_req_s, 102
zb_apsme_binding_req_s, 105
Compile-time configuration parameters, 65
N_SECUR_MATERIAL, 72
NO_NVRAM, 68
ZB_APS_COMMAND_RADIUS, 73
ZB_APS_DST_BINDING_TABLE_SIZE, 71
ZB_APS_DUP_CHECK_TIMEOUT, 71
ZB_APS_ENDPOINTS_IN_GROUP_TABLE, 71
ZB_APS_GROUP_TABLE_SIZE, 71
ZB_APS_GROUP_UP_Q_SIZE, 71
ZB_APS_POLL_AFTER_REQ_TMO, 71
ZB_APS_RETRANS_ACK_Q_SIZE, 71
ZB_APS_SRC_BINDING_TABLE_SIZE, 71
ZB_BUF_Q_SIZE, 70
ZB_CCM_KEY_SIZE, 73
ZB_CCM_L, 73
ZB_CCM_M, 74
ZB_CCM_NONCE_LEN, 74
ZB_COORDINATOR_ROLE, 69
ZB_DEFAULT_MAX_CHILDREN, 73
ZB_DEFAULT_PRMIT_JOINING_DURATION, 73
ZB_DEFAULT_SCAN_DURATION, 73
ZB_DEFAULT_SECURE_ALL_FRAMES, 74
ZB_IEEE_ADDR_TABLE_SIZE, 72
ZB_INIT_HAS_ARGS, 68
ZB_IO_BUF_SIZE, 70
ZB_IOBUF_POOL_SIZE, 70
ZB_LINUX_PIPE_TRANSPORT_TIMEOUT, 69
ZB_LITTLE_ENDIAN, 69
ZB_MAC_MAX_FRAME_RETRIES, 70
ZB_MAC_MAX_REQUESTS, 70
ZB_MAC_RESPONSE_WAIT_TIME, 70
ZB_MANUAL_ACK, 69
ZB_MAX_FRAME_TOTAL_WAIT_TIME, 70
ZB_N_APS_ACK_WAIT_DURATION, 71
ZB_N_APS_MAX_FRAME_ENTRIES, 71
ZB_N_APS_RETRANS_ENTRIES, 71
ZB_NEIGHBOR_TABLE_SIZE, 72
ZB_NS_BUILD, 69
ZB_NWK_DISTRIBUTED_ADDRESS_ASSIGN, 72
ZB_NWK_MAX_CHILDREN, 72
ZB_NWK_MAX_DEPTH, 72
ZB_NWK_MAX_ROUTERS, 72
ZB_NWK_REJOIN_REQUEST_TABLE_SIZE, 73
ZB_NWK_ROUTE_DISCOVERY_TABLE_SIZE, 72
ZB_NWK_ROUTING, 72
ZB_NWK_ROUTING_TABLE_SIZE, 72
ZB_NWK_TREE_ROUTING, 72
ZB_PANID_TABLE_SIZE, 72
ZB_PROTOCOL_VERSION, 70
ZB_SCHEDULER_Q_SIZE, 70
ZB_SECUR_NWK_COUNTER_LIMIT, 74
ZB_SECURITY, 69
ZB_SECURITY_LEVEL, 73
ZB_STACK_PROFILE, 70
ZB_STACK_PROFILE_2007, 70
ZB_STANDARD_SECURITY, 73
ZB_TC_AT_ZC, 73
ZB_TC_GENERATES_KEYS, 73
ZB_TRAFFIC_DUMP_ON, 69
ZB_TRANSPORT_LINUX_PIPES, 69
ZB_UDP_PORT_NS, 69
ZB_UDP_PORT_REAL, 69
ZB_WORD_SIZE_4, 69
ZB_ZCL_CLUSTER_NUM, 74
ZB_ZDO_15_MIN_TIMEOUT, 75
ZB_ZDO_1_MIN_TIMEOUT, 75
ZB_ZDO_APS_CHANEL_TIMER, 75
ZB_ZDO_CHANNEL_CHECK_TIMEOUT, 75
ZB_ZDO_ENDDEV_BIND_TIMEOUT, 75
ZB_ZDO_INDIRECT_POLL_TIMER, 74
ZB_ZDO_MAX_PARENT_THRESHOLD_RETRY, 74
ZB_ZDO_MAX_SCAN_DURATION, 74
ZB_ZDO_MIN_SCAN_DURATION, 74
ZB_ZDO_NEW_ACTIVE_CHANNEL, 74
ZB_ZDO_NEW_CHANNEL_MASK, 74
ZB_ZDO_NWK_SCAN_ATTEMPTS, 75
ZB_ZDO_NWK_TIME_BTWN_SCANS, 75
ZB_ZDO_PARENT_LINK_FAILURE_CNT, 75
ZB_ZDO_PENDING_LEAVE_SIZE, 75
ZDO_TRAN_TABLE_SIZE, 75

data_offset
zb_buf_hdr_s, 108
Debug trace, 96
TRACE_COMMON1, 100
TRACE_ERROR, 100
TRACE_FORMAT_64, 100

TRACE_MSG, 99
TRACE_SUBSYSTEM_COMMON, 100
depth
zb_zdo_neighbor_table_record_s, 141
dest_addr
zb_nlme_send_status_s, 116
device_address
zb_zdo_mgmt_leave_param_s, 133
zb_zdo_mgmt_leave_req_s, 134
discovery_route
zb_nlde_data_req_s, 114
dst_addr
zb_apsde_data_req_s, 102
zb_apsme_binding_req_s, 105
zb_end_device_bind_req_param_s, 111
zb_nlde_data_req_s, 114
zb_zdo_bind_req_tail_1_s, 125
zb_zdo_bind_req_tail_2_s, 126
zb_zdo_mgmt_leave_param_s, 133
zb_zdo_mgmt_lqi_param_s, 135
zb_zdo_mgmt_nwk_update_notify_param_s, 138
zb_zdo_mgmt_nwk_update_req_s, 139
zb_zdo_nwk_addr_req_param_s, 142
dst_addr_mode
zb_zdo_bind_req_head_s, 123
zb_zdo_bind_req_param_s, 124
dst_address
zb_zdo_bind_req_param_s, 124
dst_endp
zb_zdo_bind_req_param_s, 124
zb_zdo_bind_req_tail_2_s, 126
dst_endpoint
zb_apsde_data_req_s, 103
zb_apsme_binding_req_s, 105

encrypt_type
zb_buf_hdr_s, 109
endpoint
zb_apsme_add_group_conf_s, 104
zb_apsme_add_group_req_s, 104
zb_zdo_simple_desc_req_s, 145
energy_values
zb_zdo_mgmt_nwk_update_notify_param_s, 137
ep_count
zb_zdo_ep_resp_s, 129
ext_addr
zb_zdo_neighbor_table_record_s, 140
ext_pan_id
zb_zdo_neighbor_table_record_s, 140

func
zb_buf_q_ent_s, 109
zb_cb_q_ent_s, 110
zb_tm_q_ent_s, 122

group_address
zb_apsme_add_group_conf_s, 104
zb_apsme_add_group_req_s, 104

handle
zb_buf_hdr_s, 108
hdr
zb_zdo_mgmt_nwk_update_notify_param_s, 137
zb_zdo_mgmt_nwk_update_req_s, 139
zb_zdo_node_desc_resp_s, 142
zb_zdo_power_desc_resp_s, 145
zb_zdo_simple_desc_resp_s, 147
head_param
zb_end_device_bind_req_param_s, 111

ieee_addr
zb_zdo_nwk_addr_req_param_s, 142
zb_zdo_nwk_addr_req_s, 143
zb_zdo_nwk_addr_resp_head_s, 144
is_in_buf
zb_buf_hdr_s, 109

len
zb_buf_hdr_s, 108
length
zb_zdo_simple_desc_resp_hdr_s, 146
Low level API, 64
lqi
zb_zdo_neighbor_table_record_s, 141

MAC API, 57
MAC_BEACON_LOSS, 61
MAC_CHANNEL_ACCESS_FAILURE, 61
MAC_COUNTER_ERROR, 61
MAC_DENIED, 61
MAC_DISABLE_TRX_FAILURE, 61
MAC_FRAME_TOO_LONG, 61
MAC_IMPROPER_KEY_TYPE, 61
MAC_IMPROPER_SECURITY_LEVEL, 61
MAC_INVALID_ADDRESS, 61
MAC_INVALID_GTS, 61
MAC_INVALID_HANDLE, 61
MAC_INVALID_INDEX, 61
MAC_INVALID_PARAMETER, 61
MAC_LIMIT_REACHED, 61
MAC_NO_ACK, 61
MAC_NO_BEACON, 61
MAC_NO_DATA, 61
MAC_NO_SHORT_ADDRESS, 61
MAC_ON_TIME_TOO_LONG, 61
MAC_OUT_OF_CAP, 61
MAC_PAN_ID_CONFLICT, 61
MAC_PAST_TIME, 61
MAC_PIB, 59
MAC_READ_ONLY, 61
MAC_REALIGNMENT, 61
MAC_SCAN_IN_PROGRESS, 61
MAC_SECURITY_ERROR, 61
MAC_SUCCESS, 61
MAC_SUPERFRAME_OVERLAP, 61
MAC_TRACKING_OFF, 62
MAC_TRANSACTION_EXPIRED, 62
MAC_TRANSACTION_OVERFLOW, 62

MAC_TX_ACTIVE, 62
MAC_UNAVAILABLE_KEY, 62
MAC_UNSUPPORTED_ATTRIBUTE, 62
MAC_UNSUPPORTED_LEGACY, 62
MAC_UNSUPPORTED_SECURITY, 62
ZB_INC_MAC_BSN, 60
ZB_INC_MAC_DSN, 60
ZB_MAC_BSN, 60
ZB_MAC_DSN, 60
ZB_MLME_BUILD_GET_REQ, 60
ZB_PIB_BEACON_PAYLOAD, 60
ZB_PIB_COORD_SHORT_ADDRESS, 59
ZB_PIB_EXTENDED_ADDRESS, 59
ZB_PIB_RX_ON_WHEN_IDLE, 59
ZB_PIB_SHORT_ADDRESS, 59
ZB_PIB_SHORT_PAN_ID, 59
zb_mac_pib_attr_t, 62
zb_mac_status_e, 61
zb_mac_status_t, 60
zb_mlme_get_confirm, 59
zb_mlme_get_confirm_t, 60
zb_mlme_get_request, 59
zb_mlme_get_request_t, 60
zb_mlme_set_confirm, 59
zb_mlme_set_confirm_t, 61
zb_mlme_set_request, 59
zb_mlme_set_request_t, 61

MAC_BEACON_LOSS
MAC API, 61
MAC_CHANNEL_ACCESS_FAILURE
MAC API, 61
MAC_COUNTER_ERROR
MAC API, 61
MAC_DENIED
MAC API, 61
MAC_DISABLE_TRX_FAILURE
MAC API, 61
MAC_FRAME_TOO_LONG
MAC API, 61
MAC_IMPROPER_KEY_TYPE
MAC API, 61
MAC_IMPROPER_SECURITY_LEVEL
MAC API, 61
MAC_INVALID_ADDRESS
MAC API, 61
MAC_INVALID_GTS
MAC API, 61
MAC_INVALID_HANDLE
MAC API, 61
MAC_INVALID_INDEX
MAC API, 61
MAC_INVALID_PARAMETER
MAC API, 61
MAC_LIMIT_REACHED
MAC API, 61
MAC_NO_ACK
MAC API, 61
MAC_NO_BEACON
MAC API, 61
MAC_NO_DATA
MAC API, 61
MAC_NO_SHORT_ADDRESS
MAC API, 61
MAC_ON_TIME_TOO_LONG
MAC API, 61
MAC_OUT_OF_CAP
MAC API, 61
MAC_PAN_ID_CONFLICT
MAC API, 61
MAC_PAST_TIME
MAC API, 61
MAC_PIB
MAC API, 59
MAC_READ_ONLY
MAC API, 61
MAC_REALIGNMENT
MAC API, 61
MAC_SCAN_IN_PROGRESS
MAC API, 61
MAC_SECURITY_ERROR
MAC API, 61
MAC_SUCCESS
MAC API, 61
MAC_SUPERFRAME_OVERLAP
MAC API, 61
MAC_TRACKING_OFF
MAC API, 62
MAC_TRANSACTION_EXPIRED
MAC API, 62
MAC_TRANSACTION_OVERFLOW
MAC API, 62
MAC_TX_ACTIVE
MAC API, 62
MAC_UNAVAILABLE_KEY
MAC API, 62
MAC_UNSUPPORTED_ATTRIBUTE
MAC API, 62
MAC_UNSUPPORTED_LEGACY
MAC API, 62
MAC_UNSUPPORTED_SECURITY
MAC API, 62
mac_ack_wait_duration
ZB_PACKED_STRUCT, 119
mac_association_permit
ZB_PACKED_STRUCT, 119
mac_auto_request
ZB_PACKED_STRUCT, 119
mac_batt_life_ext
ZB_PACKED_STRUCT, 119
mac_beacon_order
ZB_PACKED_STRUCT, 120
mac_beacon_payload
ZB_PACKED_STRUCT, 119
mac_beacon_payload_length
ZB_PACKED_STRUCT, 119
mac_bsn

ZB_PACKED_STRUCT, 120
mac_coord_extended_address
ZB_PACKED_STRUCT, 120
mac_coord_short_address
ZB_PACKED_STRUCT, 120
mac_dsn
ZB_PACKED_STRUCT, 120
mac_extended_address
ZB_PACKED_STRUCT, 120
mac_max_frame_retries
ZB_PACKED_STRUCT, 120
mac_pan_id
ZB_PACKED_STRUCT, 120
mac_rx_on_when_idle
ZB_PACKED_STRUCT, 120
mac_short_address
ZB_PACKED_STRUCT, 120
mac_superframe_order
ZB_PACKED_STRUCT, 120
manager_addr
zb_zdo_mgmt_nwk_update_req_s, 139
match_len
zb_zdo_match_desc_resp_s, 132

N_SECUR_MATERIAL
Compile-time configuration parameters, 72
NO_NVRAM
Compile-time configuration parameters, 68
NWK functions visible to applications, 47
ZB_NWK_BROADCAST_ALL_DEVICES, 51
ZB_NWK_BROADCAST_LOW_POWER_ROUTE-R, 51
ZB_NWK_BROADCAST_ROUTER_COORDINA-TOR, 51
ZB_NWK_BROADCAST_RX_ON_WHEN_IDLE, 51
ZB_NWK_COMMAND_STATUS_ADDRESS_CO-NFLICT, 52
ZB_NWK_COMMAND_STATUS_BAD_FRAME_-COUNTER, 52
ZB_NWK_COMMAND_STATUS_BAD_KEY_SE-QUENCE_NUMBER, 52
ZB_NWK_COMMAND_STATUS_FRAME_SECU-RITY_FAILED, 50
ZB_NWK_COMMAND_STATUS_INDIRECT_TR-ANSACTION_EXPIRY, 52
ZB_NWK_COMMAND_STATUS_IS_SECURE, 50
ZB_NWK_COMMAND_STATUS_LOW_BATTER-Y_LEVEL, 52
ZB_NWK_COMMAND_STATUS_MANY_TO_ON-E_ROUTE_FAILURE, 52
ZB_NWK_COMMAND_STATUS_NETWORK_AD-DRESS_UPDATE, 52
ZB_NWK_COMMAND_STATUS_NO_INDIRECT-_CAPACITY, 52
ZB_NWK_COMMAND_STATUS_NO_ROUTE_A-VAILABLE, 52
ZB_NWK_COMMAND_STATUS_NO_ROUTING-_CAPACITY, 52
ZB_NWK_COMMAND_STATUS_NONE_TREE_-LINK_FAILURE, 52
ZB_NWK_COMMAND_STATUS_PAN_IDENTIFI-ER_UPDATE, 52
ZB_NWK_COMMAND_STATUS_PARENT_LINK-_FAILURE, 52
ZB_NWK_COMMAND_STATUS_SOURCE_ROU-TE_FAILURE, 52
ZB_NWK_COMMAND_STATUS_TARGET_ADD-RESS_UNALLOCATED, 52
ZB_NWK_COMMAND_STATUS_TARGET_DEVI-CE_UNAVAILABLE, 52
ZB_NWK_COMMAND_STATUS_TREE_LINK_F-AILURE, 52
ZB_NWK_COMMAND_STATUS_VALIDATE_RO-UTE, 52
ZB_NWK_COMMAND_STATUS_VERIFY_ADDR-ESS, 52
ZB_NWK_IS_ADDRESS_BROADCAST, 49
ZB_NWK_STATUS_ALREADY_PRESENT, 51
ZB_NWK_STATUS_BAD_CCM_OUTPUT, 51
ZB_NWK_STATUS_BT_TABLE_FULL, 52
ZB_NWK_STATUS_FRAME_NOT_BUFFERED, 52
ZB_NWK_STATUS_INVALID_PARAMETER, 51
ZB_NWK_STATUS_INVALID_REQUEST, 51
ZB_NWK_STATUS_MAX_FRM_COUNTER, 51
ZB_NWK_STATUS_NEIGHBOR_TABLE_FULL, 51
ZB_NWK_STATUS_NO_KEY, 51
ZB_NWK_STATUS_NO_NETWORKS, 51
ZB_NWK_STATUS_NO_ROUTING_CAPACITY, 51
ZB_NWK_STATUS_NOT_PERMITTED, 51
ZB_NWK_STATUS_ROUTE_DISCOVERY_FAIL-ED, 51
ZB_NWK_STATUS_ROUTE_ERROR, 51
ZB_NWK_STATUS_STARTUP_FAILURE, 51
ZB_NWK_STATUS_SUCCESS, 51
ZB_NWK_STATUS_SYNC_FAILURE, 51
ZB_NWK_STATUS_UNKNOWN_DEVICE, 51
ZB_NWK_STATUS_UNSUPPORTED_ATTRIBU-TE, 51
zb_nlde_data_req_t, 50
zb_nlde_data_request, 48
zb_nlme_send_status, 49
zb_nlme_send_status_t, 50
zb_nlme_status_indication_t, 50
zb_nwk_broadcast_address_e, 51
zb_nwk_broadcast_address_t, 50
zb_nwk_command_status_e, 52
zb_nwk_command_status_t, 50
zb_nwk_status_e, 51
zb_nwk_status_t, 50
NWK Informational Base, 53
zb_nib_attribute_e, 56
zb_nib_attribute_t, 56
zb_nlme_get_confirm, 55

zb_nlme_get_confirm_t, 56
zb_nlme_get_request, 54
zb_nlme_get_request_t, 56
zb_nlme_set_confirm, 55
zb_nlme_set_confirm_t, 56
zb_nlme_set_request, 55
zb_nlme_set_request_t, 56
ndsu_handle
zb_nlde_data_req_s, 114
zb_nlme_send_status_s, 116
neighbor_table_entries
zb_zdo_mgmt_lqi_resp_s, 136
neighbor_table_list_count
zb_zdo_mgmt_lqi_resp_s, 136
network_addr
zb_nlme_status_indication_s, 118
zb_zdo_neighbor_table_record_s, 141
nib_attribute
zb_nlme_get_confirm_s, 115
zb_nlme_get_request_s, 116
zb_nlme_set_confirm_s, 117
zb_nlme_set_request_s, 118
node_desc
zb_zdo_node_desc_resp_s, 142
nonmember_radius
zb_nlde_data_req_s, 114
num_in_cluster
zb_zdo_end_device_bind_req_head_s, 128
num_in_clusters
zb_zdo_match_desc_param_s, 130
zb_zdo_match_desc_req_head_s, 131
num_out_cluster
zb_zdo_end_device_bind_req_tail_s, 128
num_out_clusters
zb_zdo_match_desc_param_s, 130
zb_zdo_match_desc_req_tail_s, 132
nwk_addr
zb_zdo_active_ep_req_s, 123
zb_zdo_desc_resp_hdr_s, 127
zb_zdo_ep_resp_s, 129
zb_zdo_ieee_addr_req_s, 130
zb_zdo_match_desc_param_s, 130
zb_zdo_match_desc_req_head_s, 131
zb_zdo_match_desc_resp_s, 132
zb_zdo_node_desc_req_s, 141
zb_zdo_nwk_addr_resp_head_s, 144
zb_zdo_power_desc_req_s, 144
zb_zdo_simple_desc_req_s, 145
zb_zdo_simple_desc_resp_hdr_s, 146

Packet buffers pool, 82
ZB_BUF_ALLOC_LEFT, 86
ZB_BUF_ALLOC_RIGHT, 86
ZB_BUF_BEGIN, 85
ZB_BUF_COPY, 86
ZB_BUF_CUT_LEFT, 86
ZB_BUF_CUT_LEFT2, 86
ZB_BUF_CUT_RIGHT, 86
ZB_BUF_LEN, 85
ZB_BUF_OFFSET, 85
zb_buf_assign_param, 84
zb_buf_hdr_t, 87
zb_buf_initial_alloc, 83
zb_buf_reuse, 84
zb_buf_s_t, 87
zb_free_buf, 85
zb_get_buf_tail, 83
zb_get_in_buf, 84
zb_get_in_buf_delayed, 85
zb_get_out_buf, 84
zb_get_out_buf_delayed, 85
zb_init_buffers, 84
param
zb_cb_q_ent_s, 110
zb_tm_q_ent_s, 122
permit_join
zb_zdo_neighbor_table_record_s, 141
permit_join_duration
zb_zdo_configuration_attributes_e, 126
power_desc
zb_zdo_power_desc_resp_s, 145
profile_id
zb_zdo_end_device_bind_req_head_s, 128
zb_zdo_match_desc_param_s, 130
zb_zdo_match_desc_req_head_s, 131
profileid
zb_apsde_data_req_s, 102

radius
zb_apsde_data_req_s, 103
zb_nlde_data_req_s, 114
req_dst_addr
zb_zdo_bind_req_param_s, 125
request_type
zb_zdo_ieee_addr_req_s, 130
zb_zdo_nwk_addr_req_param_s, 143
zb_zdo_nwk_addr_req_s, 143
run_time
zb_tm_q_ent_s, 122

scan_channels
zb_zdo_mgmt_nwk_update_req_hdr_s, 138
scan_count
zb_zdo_mgmt_nwk_update_req_s, 139
scan_duration
zb_zdo_mgmt_nwk_update_req_hdr_s, 138
scanned_channels
zb_zdo_mgmt_nwk_update_notify_hdr_s, 137
scanned_channels_list_count
zb_zdo_mgmt_nwk_update_notify_hdr_s, 137
Scheduler, 88
ZB_ALARM_ALL_CB, 91
ZB_ALARM_ANY_PARAM, 91
ZB_RING_BUFFER_DECLARE, 89
ZB_SCHED_GLOBAL_LOCK, 91
ZB_SCHED_GLOBAL_LOCK_INT, 91
ZB_SCHED_GLOBAL_UNLOCK, 91
ZB_SCHED_GLOBAL_UNLOCK_INT, 91

ZB_SCHED_HAS_PENDING_CALLBACKS, 91
ZB_SCHED_WAIT_COND, 91
zb_callback_t, 92
zb_cb_q_ent_t, 92
zb_sched_globals_t, 92
zb_sched_init, 89
zb_sched_loop_iteration, 89
zb_schedule_alarm, 90
zb_schedule_alarm_cancel, 90
zb_schedule_callback, 89
zb_schedule_mac_cb, 90
zb_tm_q_ent_t, 92
secur_clear_preconfigured_key
Security subsystem API, 63
Security subsystem API, 63
secur_clear_preconfigured_key, 63
zb_secur_send_nwk_key_switch, 63
zb_secur_send_nwk_key_update_br, 63
zb_secur_setup_preconfigured_key, 63
security_enable
zb_nlde_data_req_s, 114
server_mask
zb_zdo_system_server_discovery_req_s, 147
zb_zdo_system_server_discovery_resp_s, 148
simple_desc
zb_zdo_simple_desc_resp_s, 147
src_addr
zb_apsme_binding_req_s, 105
src_address
zb_zdo_bind_req_head_s, 123
zb_zdo_bind_req_param_s, 124
src_endp
zb_zdo_bind_req_head_s, 123
zb_zdo_bind_req_param_s, 124
zb_zdo_end_device_bind_req_head_s, 127
src_endpoint
zb_apsde_data_req_s, 103
zb_apsme_binding_req_s, 105
src_ieee_addr
zb_zdo_end_device_bind_req_head_s, 127
Stack initialization API, 9
zb_init, 9
start_index
zb_zdo_ieee_addr_req_s, 130
zb_zdo_mgmt_lqi_param_s, 135
zb_zdo_mgmt_lqi_req_s, 135
zb_zdo_mgmt_lqi_resp_s, 136
zb_zdo_nwk_addr_req_param_s, 143
zb_zdo_nwk_addr_req_s, 143
status
zb_apsme_get_confirm_s, 106
zb_apsme_set_confirm_s, 107
zb_buf_hdr_s, 109
zb_nlme_get_confirm_s, 115
zb_nlme_send_status_s, 116
zb_nlme_set_confirm_s, 117
zb_nlme_status_indication_s, 118
zb_zdo_desc_resp_hdr_s, 127
zb_zdo_ep_resp_s, 129
zb_zdo_match_desc_resp_s, 132
zb_zdo_mgmt_lqi_resp_s, 136
zb_zdo_mgmt_nwk_update_notify_hdr_s, 137
zb_zdo_nwk_addr_resp_head_s, 144
zb_zdo_simple_desc_resp_hdr_s, 146
zb_zdo_system_server_discovery_resp_s, 148

TRACE_COMMON1
Debug trace, 100
TRACE_ERROR
Debug trace, 100
TRACE_FORMAT_64
Debug trace, 100
TRACE_MSG
Debug trace, 99
TRACE_SUBSYSTEM_COMMON
Debug trace, 100
tail_param
zb_end_device_bind_req_param_s, 111
Time, 93
ZB_BEACON_INTERVAL_USEC, 94
ZB_MILLISECONDS_TO_BEACON_INTERVAL, 94
ZB_TIME_ADD, 94
ZB_TIME_BEACON_INTERVAL_TO_MSEC, 94
ZB_TIME_GE, 94
ZB_TIME_ONE_SECOND, 94
ZB_TIME_SUBTRACT, 93
ZB_TIMER_GET, 93
ZB_TIMER_START, 95
zb_time_t, 95
tm_buffer
zb_sched_globals_s, 121
total_transmissions
zb_zdo_mgmt_nwk_update_notify_hdr_s, 137
transmission_failures
zb_zdo_mgmt_nwk_update_notify_hdr_s, 137
tsn
zb_zdo_mgmt_nwk_update_notify_param_s, 138
tx_options
zb_apsde_data_req_s, 103
type_flags
zb_zdo_neighbor_table_record_s, 141

update_id
zb_zdo_mgmt_nwk_update_req_s, 139
use_same_key
zb_buf_hdr_s, 109

ZB_64BIT_ADDR_CMP
Base typedefs, 79
ZB_64BIT_ADDR_COPY
Base typedefs, 79
ZB_64BIT_ADDR_ZERO
Base typedefs, 79
ZB_ADDR_CMP
Base typedefs, 79
ZB_ALARM_ALL_CB

Scheduler, 91
ZB_ALARM_ANY_PARAM
Scheduler, 91
ZB_APS_ADDR_MODE_16_ENDP_PRESENT
APS functions visible to applications, 42
ZB_APS_ADDR_MODE_16_GROUP_ENDP_NOT_P-RESENT
APS functions visible to applications, 42
ZB_APS_ADDR_MODE_64_ENDP_PRESENT
APS functions visible to applications, 42
ZB_APS_ADDR_MODE_DST_ADDR_ENDP_NOT_P-RESENT
APS functions visible to applications, 42
ZB_APS_AIB_BINDING
APS Informational Base, 45
ZB_APS_AIB_CHANNEL_MASK
APS Informational Base, 46
ZB_APS_AIB_CHANNEL_TIMER
APS Informational Base, 46
ZB_APS_AIB_DESIGNATED_COORD
APS Informational Base, 45
ZB_APS_AIB_GROUP_TABLE
APS Informational Base, 46
ZB_APS_AIB_INTERFRAME_DELAY
APS Informational Base, 46
ZB_APS_AIB_LAST_CHANNEL_ENERGY
APS Informational Base, 46
ZB_APS_AIB_LAST_CHANNEL_FAILURE_RATE
APS Informational Base, 46
ZB_APS_AIB_NONMEMBER_RADIUS
APS Informational Base, 46
ZB_APS_AIB_PERMISSION_CONFIG
APS Informational Base, 46
ZB_APS_AIB_USE_EXT_PANID
APS Informational Base, 46
ZB_APS_AIB_USE_INSECURE_JOIN
APS Informational Base, 46
ZB_APS_COMMAND_RADIUS
Compile-time configuration parameters, 73
ZB_APS_DST_BINDING_TABLE_SIZE
Compile-time configuration parameters, 71
ZB_APS_DUP_CHECK_TIMEOUT
Compile-time configuration parameters, 71
ZB_APS_ENDPOINTS_IN_GROUP_TABLE
Compile-time configuration parameters, 71
ZB_APS_GROUP_TABLE_SIZE
Compile-time configuration parameters, 71
ZB_APS_GROUP_UP_Q_SIZE
Compile-time configuration parameters, 71
ZB_APS_HDR_CUT
APS functions visible to applications, 41
ZB_APS_HDR_CUT_P
APS functions visible to applications, 41
ZB_APS_POLL_AFTER_REQ_TMO
Compile-time configuration parameters, 71
ZB_APS_RETRANS_ACK_Q_SIZE
Compile-time configuration parameters, 71
ZB_APS_SRC_BINDING_TABLE_SIZE
Compile-time configuration parameters, 71
ZB_APS_STATUS_INVALID_BINDING
APS functions visible to applications, 42
ZB_APS_STATUS_INVALID_GROUP
APS functions visible to applications, 42
ZB_APS_STATUS_INVALID_PARAMETER
APS functions visible to applications, 42
ZB_APS_STATUS_NO_BOUND_DEVICE
APS functions visible to applications, 42
ZB_APS_STATUS_NO_SHORT_ADDRESS
APS functions visible to applications, 43
ZB_APS_STATUS_NOT_SUPPORTED
APS functions visible to applications, 43
ZB_APS_STATUS_SECURED_LINK_KEY
APS functions visible to applications, 43
ZB_APS_STATUS_SECURED_NWK_KEY
APS functions visible to applications, 43
ZB_APS_STATUS_SECURITY_FAIL
APS functions visible to applications, 43
ZB_APS_STATUS_SUCCESS
APS functions visible to applications, 42
ZB_APS_STATUS_TABLE_FULL
APS functions visible to applications, 43
ZB_APS_STATUS_UNSECURED
APS functions visible to applications, 43
ZB_APS_STATUS_UNSUPPORTED_ATTRIBUTE
APS functions visible to applications, 43
ZB_APSDE_TX_OPT_ACK_TX
APS functions visible to applications, 43
ZB_APSDE_TX_OPT_FRAG_PERMITTED
APS functions visible to applications, 43
ZB_APSDE_TX_OPT_SECURITY_ENABLED
APS functions visible to applications, 43
ZB_APSDE_TX_OPT_USE_NWK_KEY
APS functions visible to applications, 43
ZB_BEACON_INTERVAL_USEC
Time, 94
ZB_BUF_ALLOC_LEFT
Packet buffers pool, 86
ZB_BUF_ALLOC_RIGHT
Packet buffers pool, 86
ZB_BUF_BEGIN
Packet buffers pool, 85
ZB_BUF_COPY
Packet buffers pool, 86
ZB_BUF_CUT_LEFT
Packet buffers pool, 86
ZB_BUF_CUT_LEFT2
Packet buffers pool, 86
ZB_BUF_CUT_RIGHT
Packet buffers pool, 86
ZB_BUF_LEN
Packet buffers pool, 85
ZB_BUF_OFFSET
Packet buffers pool, 85
ZB_BUF_Q_SIZE
Compile-time configuration parameters, 70
ZB_CCM_KEY_SIZE

Compile-time configuration parameters, 73
ZB_CCM_L
Compile-time configuration parameters, 73
ZB_CCM_M
Compile-time configuration parameters, 74
ZB_CCM_NONCE_LEN
Compile-time configuration parameters, 74
ZB_COORDINATOR_ROLE
Compile-time configuration parameters, 69
ZB_DEFAULT_MAX_CHILDREN
Compile-time configuration parameters, 73
ZB_DEFAULT_PRMIT_JOINING_DURATION
Compile-time configuration parameters, 73
ZB_DEFAULT_SCAN_DURATION
Compile-time configuration parameters, 73
ZB_DEFAULT_SECURE_ALL_FRAMES
Compile-time configuration parameters, 74
ZB_HTOLE16
Base typedefs, 79
ZB_IEEE_ADDR_TABLE_SIZE
Compile-time configuration parameters, 72
ZB_INC_MAC_BSN
MAC API, 60
ZB_INC_MAC_DSN
MAC API, 60
ZB_INIT_HAS_ARGS
Compile-time configuration parameters, 68
ZB_INT8_C
Base typedefs, 79
ZB_IO_BUF_SIZE
Compile-time configuration parameters, 70
ZB_IOBUF_POOL_SIZE
Compile-time configuration parameters, 70
ZB_IS_64BIT_ADDR_ZERO
Base typedefs, 79
ZB_LETOH16
Base typedefs, 79
ZB_LINUX_PIPE_TRANSPORT_TIMEOUT
Compile-time configuration parameters, 69
ZB_LIST_DEFINE
zb_sched_globals_s, 121
ZB_LITTLE_ENDIAN
Compile-time configuration parameters, 69
ZB_MAC_BSN
MAC API, 60
ZB_MAC_DSN
MAC API, 60
ZB_MAC_MAX_FRAME_RETRIES
Compile-time configuration parameters, 70
ZB_MAC_MAX_REQUESTS
Compile-time configuration parameters, 70
ZB_MAC_RESPONSE_WAIT_TIME
Compile-time configuration parameters, 70
ZB_MANUAL_ACK
Compile-time configuration parameters, 69
ZB_MAX_FRAME_TOTAL_WAIT_TIME
Compile-time configuration parameters, 70
ZB_MILLISECONDS_TO_BEACON_INTERVAL
Time, 94
ZB_MLME_BUILD_GET_REQ
MAC API, 60
ZB_N_APS_ACK_WAIT_DURATION
Compile-time configuration parameters, 71
ZB_N_APS_MAX_FRAME_ENTRIES
Compile-time configuration parameters, 71
ZB_N_APS_RETRANS_ENTRIES
Compile-time configuration parameters, 71
ZB_NEIGHBOR_TABLE_SIZE
Compile-time configuration parameters, 72
ZB_NS_BUILD
Compile-time configuration parameters, 69
ZB_NWK_BROADCAST_ALL_DEVICES
NWK functions visible to applications, 51
ZB_NWK_BROADCAST_LOW_POWER_ROUTER
NWK functions visible to applications, 51
ZB_NWK_BROADCAST_ROUTER_COORDINATOR
NWK functions visible to applications, 51
ZB_NWK_BROADCAST_RX_ON_WHEN_IDLE
NWK functions visible to applications, 51
ZB_NWK_COMMAND_STATUS_ADDRESS_CONFLI-CT
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_BAD_FRAME_COU-NTER
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_BAD_KEY_SEQUE-NCE_NUMBER
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_FRAME_SECURITY-_FAILED
NWK functions visible to applications, 50
ZB_NWK_COMMAND_STATUS_INDIRECT_TRANSA-CTION_EXPIRY
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_IS_SECURE
NWK functions visible to applications, 50
ZB_NWK_COMMAND_STATUS_LOW_BATTERY_LE-VEL
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_MANY_TO_ONE_R-OUTE_FAILURE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_NETWORK_ADDRE-SS_UPDATE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_NO_INDIRECT_CA-PACITY
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_NO_ROUTE_AVAIL-ABLE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_NO_ROUTING_CAP-ACITY
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_NONE_TREE_LINK-_FAILURE

NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_PAN_IDENTIFIER_-UPDATE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_PARENT_LINK_FAI-LURE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_SOURCE_ROUTE_-FAILURE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_TARGET_ADDRES-S_UNALLOCATED
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_TARGET_DEVICE_-UNAVAILABLE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_TREE_LINK_FAILU-RE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_VALIDATE_ROUTE
NWK functions visible to applications, 52
ZB_NWK_COMMAND_STATUS_VERIFY_ADDRESS
NWK functions visible to applications, 52
ZB_NWK_DISTRIBUTED_ADDRESS_ASSIGN
Compile-time configuration parameters, 72
ZB_NWK_IS_ADDRESS_BROADCAST
NWK functions visible to applications, 49
ZB_NWK_MAX_CHILDREN
Compile-time configuration parameters, 72
ZB_NWK_MAX_DEPTH
Compile-time configuration parameters, 72
ZB_NWK_MAX_ROUTERS
Compile-time configuration parameters, 72
ZB_NWK_REJOIN_REQUEST_TABLE_SIZE
Compile-time configuration parameters, 73
ZB_NWK_ROUTE_DISCOVERY_TABLE_SIZE
Compile-time configuration parameters, 72
ZB_NWK_ROUTING
Compile-time configuration parameters, 72
ZB_NWK_ROUTING_TABLE_SIZE
Compile-time configuration parameters, 72
ZB_NWK_STATUS_ALREADY_PRESENT
NWK functions visible to applications, 51
ZB_NWK_STATUS_BAD_CCM_OUTPUT
NWK functions visible to applications, 51
ZB_NWK_STATUS_BT_TABLE_FULL
NWK functions visible to applications, 52
ZB_NWK_STATUS_FRAME_NOT_BUFFERED
NWK functions visible to applications, 52
ZB_NWK_STATUS_INVALID_PARAMETER
NWK functions visible to applications, 51
ZB_NWK_STATUS_INVALID_REQUEST
NWK functions visible to applications, 51
ZB_NWK_STATUS_MAX_FRM_COUNTER
NWK functions visible to applications, 51
ZB_NWK_STATUS_NEIGHBOR_TABLE_FULL
NWK functions visible to applications, 51
ZB_NWK_STATUS_NO_KEY
NWK functions visible to applications, 51
ZB_NWK_STATUS_NO_NETWORKS
NWK functions visible to applications, 51
ZB_NWK_STATUS_NO_ROUTING_CAPACITY
NWK functions visible to applications, 51
ZB_NWK_STATUS_NOT_PERMITTED
NWK functions visible to applications, 51
ZB_NWK_STATUS_ROUTE_DISCOVERY_FAILED
NWK functions visible to applications, 51
ZB_NWK_STATUS_ROUTE_ERROR
NWK functions visible to applications, 51
ZB_NWK_STATUS_STARTUP_FAILURE
NWK functions visible to applications, 51
ZB_NWK_STATUS_SUCCESS
NWK functions visible to applications, 51
ZB_NWK_STATUS_SYNC_FAILURE
NWK functions visible to applications, 51
ZB_NWK_STATUS_UNKNOWN_DEVICE
NWK functions visible to applications, 51
ZB_NWK_STATUS_UNSUPPORTED_ATTRIBUTE
NWK functions visible to applications, 51
ZB_NWK_TREE_ROUTING
Compile-time configuration parameters, 72
ZB_PACKED_STRUCT, 118
mac_ack_wait_duration, 119
mac_association_permit, 119
mac_auto_request, 119
mac_batt_life_ext, 119
mac_beacon_order, 120
mac_beacon_payload, 119
mac_beacon_payload_length, 119
mac_bsn, 120
mac_coord_extended_address, 120
mac_coord_short_address, 120
mac_dsn, 120
mac_extended_address, 120
mac_max_frame_retries, 120
mac_pan_id, 120
mac_rx_on_when_idle, 120
mac_short_address, 120
mac_superframe_order, 120
ZB_PANID_TABLE_SIZE
Compile-time configuration parameters, 72
ZB_PIB_BEACON_PAYLOAD
MAC API, 60
ZB_PIB_COORD_SHORT_ADDRESS
MAC API, 59
ZB_PIB_EXTENDED_ADDRESS
MAC API, 59
ZB_PIB_RX_ON_WHEN_IDLE
MAC API, 59
ZB_PIB_SHORT_ADDRESS
MAC API, 59
ZB_PIB_SHORT_PAN_ID
MAC API, 59
ZB_PROTOCOL_VERSION
Compile-time configuration parameters, 70
ZB_RING_BUFFER_DECLARE

Scheduler, 89
ZB_SCHED_GLOBAL_LOCK
Scheduler, 91
ZB_SCHED_GLOBAL_LOCK_INT
Scheduler, 91
ZB_SCHED_GLOBAL_UNLOCK
Scheduler, 91
ZB_SCHED_GLOBAL_UNLOCK_INT
Scheduler, 91
ZB_SCHED_HAS_PENDING_CALLBACKS
Scheduler, 91
ZB_SCHED_WAIT_COND
Scheduler, 91
ZB_SCHEDULER_Q_SIZE
Compile-time configuration parameters, 70
ZB_SECUR_NWK_COUNTER_LIMIT
Compile-time configuration parameters, 74
ZB_SECURITY
Compile-time configuration parameters, 69
ZB_SECURITY_LEVEL
Compile-time configuration parameters, 73
ZB_SHORT_MIN
Base typedefs, 79
ZB_STACK_PROFILE
Compile-time configuration parameters, 70
ZB_STACK_PROFILE_2007
Compile-time configuration parameters, 70
ZB_STANDARD_SECURITY
Compile-time configuration parameters, 73
ZB_STK_DEFINE
zb_sched_globals_s, 121
ZB_TC_AT_ZC
Compile-time configuration parameters, 73
ZB_TC_GENERATES_KEYS
Compile-time configuration parameters, 73
ZB_TIME_ADD
Time, 94
ZB_TIME_BEACON_INTERVAL_TO_MSEC
Time, 94
ZB_TIME_GE
Time, 94
ZB_TIME_ONE_SECOND
Time, 94
ZB_TIME_SUBTRACT
Time, 93
ZB_TIMER_GET
Time, 93
ZB_TIMER_START
Time, 95
ZB_TRAFFIC_DUMP_ON
Compile-time configuration parameters, 69
ZB_TRANSPORT_LINUX_PIPES
Compile-time configuration parameters, 69
ZB_UDP_PORT_NS
Compile-time configuration parameters, 69
ZB_UDP_PORT_REAL
Compile-time configuration parameters, 69
ZB_WORD_SIZE_4
Compile-time configuration parameters, 69
ZB_ZCL_CLUSTER_NUM
Compile-time configuration parameters, 74
ZB_ZDO_15_MIN_TIMEOUT
Compile-time configuration parameters, 75
ZB_ZDO_1_MIN_TIMEOUT
Compile-time configuration parameters, 75
ZB_ZDO_APS_CHANEL_TIMER
Compile-time configuration parameters, 75
ZB_ZDO_CHANNEL_CHECK_TIMEOUT
Compile-time configuration parameters, 75
ZB_ZDO_ENDDEV_BIND_TIMEOUT
Compile-time configuration parameters, 75
ZB_ZDO_EXTENDED_DEVICE_RESP
ZDO discovery services, 25
ZB_ZDO_INDIRECT_POLL_TIMER
Compile-time configuration parameters, 74
ZB_ZDO_MAX_PARENT_THRESHOLD_RETRY
Compile-time configuration parameters, 74
ZB_ZDO_MAX_SCAN_DURATION
Compile-time configuration parameters, 74
ZB_ZDO_MIN_SCAN_DURATION
Compile-time configuration parameters, 74
ZB_ZDO_NEW_ACTIVE_CHANNEL
Compile-time configuration parameters, 74
ZB_ZDO_NEW_CHANNEL_MASK
Compile-time configuration parameters, 74
ZB_ZDO_NWK_SCAN_ATTEMPTS
Compile-time configuration parameters, 75
ZB_ZDO_NWK_TIME_BTWN_SCANS
Compile-time configuration parameters, 75
ZB_ZDO_PARENT_LINK_FAILURE_CNT
Compile-time configuration parameters, 75
ZB_ZDO_PENDING_LEAVE_SIZE
Compile-time configuration parameters, 75
ZB_ZDO_SINGLE_DEVICE_RESP
ZDO discovery services, 25
ZB_ZDP_STATUS_DEVICE_NOT_FOUND
ZDO base constants and definitions, 14
ZB_ZDP_STATUS_INSUFFICIENT_SPACE
ZDO base constants and definitions, 15
ZB_ZDP_STATUS_INV_REQUESTTYPE
ZDO base constants and definitions, 14
ZB_ZDP_STATUS_INVALID_EP
ZDO base constants and definitions, 14
ZB_ZDP_STATUS_NO_DESCRIPTOR
ZDO base constants and definitions, 15
ZB_ZDP_STATUS_NO_ENTRY
ZDO base constants and definitions, 15
ZB_ZDP_STATUS_NO_MATCH
ZDO base constants and definitions, 15
ZB_ZDP_STATUS_NOT_ACTIVE
ZDO base constants and definitions, 14
ZB_ZDP_STATUS_NOT_AUTHORIZED
ZDO base constants and definitions, 15
ZB_ZDP_STATUS_NOT_PERMITTED
ZDO base constants and definitions, 15
ZB_ZDP_STATUS_NOT_SUPPORTED

ZDO base constants and definitions, 14
ZB_ZDP_STATUS_SUCCESS
ZDO base constants and definitions, 14
ZB_ZDP_STATUS_TABLE_FULL
ZDO base constants and definitions, 15
ZB_ZDP_STATUS_TIMEOUT
ZDO base constants and definitions, 15
ZDO API, 12
ZDO base constants and definitions, 14
ZB_ZDP_STATUS_DEVICE_NOT_FOUND, 14
ZB_ZDP_STATUS_INSUFFICIENT_SPACE, 15
ZB_ZDP_STATUS_INV_REQUESTTYPE, 14
ZB_ZDP_STATUS_INVALID_EP, 14
ZB_ZDP_STATUS_NO_DESCRIPTOR, 15
ZB_ZDP_STATUS_NO_ENTRY, 15
ZB_ZDP_STATUS_NO_MATCH, 15
ZB_ZDP_STATUS_NOT_ACTIVE, 14
ZB_ZDP_STATUS_NOT_AUTHORIZED, 15
ZB_ZDP_STATUS_NOT_PERMITTED, 15
ZB_ZDP_STATUS_NOT_SUPPORTED, 14
ZB_ZDP_STATUS_SUCCESS, 14
ZB_ZDP_STATUS_TABLE_FULL, 15
ZB_ZDP_STATUS_TIMEOUT, 15
zb_zdp_status_e, 14
zb_zdp_status_t, 14
ZDO discovery services, 16
ZB_ZDO_EXTENDED_DEVICE_RESP, 25
ZB_ZDO_SINGLE_DEVICE_RESP, 25
zb_zdo_active_ep_req, 23
zb_zdo_active_ep_req_t, 26
zb_zdo_desc_resp_hdr_t, 26
zb_zdo_ep_resp_t, 26
zb_zdo_ieee_addr_req, 19
zb_zdo_ieee_addr_req_t, 25
zb_zdo_match_desc_param_t, 26
zb_zdo_match_desc_req, 23
zb_zdo_match_desc_req_head_t, 26
zb_zdo_match_desc_req_tail_t, 27
zb_zdo_match_desc_resp_t, 27
zb_zdo_node_desc_req, 20
zb_zdo_node_desc_req_t, 25
zb_zdo_node_desc_resp_t, 26
zb_zdo_nwk_addr_req, 18
zb_zdo_nwk_addr_req_param_t, 25
zb_zdo_nwk_addr_req_t, 25
zb_zdo_power_desc_req, 21
zb_zdo_power_desc_req_t, 26
zb_zdo_power_desc_resp_t, 26
zb_zdo_simple_desc_req, 22
zb_zdo_simple_desc_req_t, 26
zb_zdo_simple_desc_resp_hdr_t, 26
zb_zdo_simple_desc_resp_t, 26
zb_zdo_system_server_discovery_param_t, 27
zb_zdo_system_server_discovery_req, 24
zb_zdo_system_server_discovery_req_t, 27
zb_zdo_system_server_discovery_resp_t, 27
ZDO Informational Base, 13
ZDO init and main() structure, 10
zb_zdo_startup_complete, 10
zdo_dev_start, 10
zdo_main_loop, 10
ZDO management services, 28
zb_end_device_bind_req_param_t, 37
zb_zdo_add_group_req, 34
zb_zdo_bind_req, 32
zb_zdo_bind_req_head_t, 36
zb_zdo_bind_req_param_t, 36
zb_zdo_bind_req_tail_1_t, 36
zb_zdo_bind_req_tail_2_t, 36
zb_zdo_end_device_bind_req_head_t, 36
zb_zdo_end_device_bind_req_tail_t, 36
zb_zdo_mgmt_leave_param_t, 36
zb_zdo_mgmt_leave_req_t, 36
zb_zdo_mgmt_leave_res_t, 36
zb_zdo_mgmt_lqi_param_t, 35
zb_zdo_mgmt_lqi_req, 31
zb_zdo_mgmt_lqi_req_t, 35
zb_zdo_mgmt_lqi_resp_t, 36
zb_zdo_mgmt_nwk_update_notify_hdr_t, 35
zb_zdo_mgmt_nwk_update_notify_param_t, 35
zb_zdo_mgmt_nwk_update_req, 30
zb_zdo_mgmt_nwk_update_req_hdr_t, 35
zb_zdo_mgmt_nwk_update_req_t, 35
zb_zdo_mgmt_permit_joining_req_param_t, 37
zb_zdo_mgmt_permit_joining_req_t, 37
zb_zdo_neighbor_table_record_t, 36
zb_zdo_unbind_req, 33
zdo_mgmt_leave_req, 34
ZDO_TRAN_TABLE_SIZE
Compile-time configuration parameters, 75
zb_64bit_addr_t
Base typedefs, 81
zb_addr64_struct_s, 101
zb_addr_u, 101
zb_af_set_data_indication
AF functions visible to applications, 38
zb_aps_addr_mode_e
APS functions visible to applications, 42
zb_aps_aib_attr_id_e
APS Informational Base, 45
zb_aps_aib_attr_id_t
APS Informational Base, 45
zb_aps_hdr_s, 101
zb_aps_hdr_t
APS functions visible to applications, 42
zb_aps_status_e
APS functions visible to applications, 42
zb_apsde_data_indication_t
APS functions visible to applications, 42
zb_apsde_data_req_s, 102
addr_mode, 103
clusterid, 102
dst_addr, 102
dst_endpoint, 103
profileid, 102
radius, 103

src_endpoint, 103
tx_options, 103
zb_apsde_data_req_t
APS functions visible to applications, 41
zb_apsde_data_request
APS functions visible to applications, 40
zb_apsde_tx_opt_e
APS functions visible to applications, 43
zb_apsme_add_group_conf_s, 103
endpoint, 104
group_address, 104
zb_apsme_add_group_conf_t
APS functions visible to applications, 42
zb_apsme_add_group_req_s, 104
endpoint, 104
group_address, 104
zb_apsme_add_group_req_t
APS functions visible to applications, 42
zb_apsme_binding_req_s, 104
addr_mode, 105
clusterid, 105
dst_addr, 105
dst_endpoint, 105
src_addr, 105
src_endpoint, 105
zb_apsme_binding_req_t
APS functions visible to applications, 41
zb_apsme_get_confirm
APS Informational Base, 45
zb_apsme_get_confirm_s, 106
aib_attr, 106
aib_length, 106
status, 106
zb_apsme_get_confirm_t
APS Informational Base, 45
zb_apsme_get_request
APS Informational Base, 45
zb_apsme_get_request_s, 106
aib_attr, 107
zb_apsme_get_request_t
APS Informational Base, 45
zb_apsme_set_confirm
APS Informational Base, 45
zb_apsme_set_confirm_s, 107
aib_attr, 107
status, 107
zb_apsme_set_confirm_t
APS Informational Base, 45
zb_apsme_set_request
APS Informational Base, 45
zb_apsme_set_request_s, 107
aib_attr, 108
aib_length, 108
zb_apsme_set_request_t
APS Informational Base, 45
zb_bitfield_t
Base typedefs, 80
zb_bool_e
Base typedefs, 81
zb_bool_t
Base typedefs, 80
zb_buf_assign_param
Packet buffers pool, 84
zb_buf_hdr_s, 108
data_offset, 108
encrypt_type, 109
handle, 108
is_in_buf, 109
len, 108
status, 109
use_same_key, 109
zdo_cmd_no_resp, 109
zb_buf_hdr_t
Packet buffers pool, 87
zb_buf_initial_alloc
Packet buffers pool, 83
zb_buf_q_ent_s, 109
func, 109
zb_buf_reuse
Packet buffers pool, 84
zb_buf_s, 110
zb_buf_s_t
Packet buffers pool, 87
zb_callback_t
Scheduler, 92
zb_cb_q_ent_s, 110
func, 110
param, 110
zb_cb_q_ent_t
Scheduler, 92
zb_char_t
Base typedefs, 80
zb_end_device_bind_req_param_s, 111
cluster_list, 111
dst_addr, 111
head_param, 111
tail_param, 111
zb_end_device_bind_req_param_t
ZDO management services, 37
zb_ext_pan_id_t
Base typedefs, 81
zb_free_buf
Packet buffers pool, 85
zb_get_buf_tail
Packet buffers pool, 83
zb_get_in_buf
Packet buffers pool, 84
zb_get_in_buf_delayed
Packet buffers pool, 85
zb_get_out_buf
Packet buffers pool, 84
zb_get_out_buf_delayed
Packet buffers pool, 85
zb_ieee_addr_t
Base typedefs, 81
zb_init

Stack initialization API, 9
zb_init_buffers
Packet buffers pool, 84
zb_int16_t
Base typedefs, 80
zb_int32_t
Base typedefs, 80
zb_int8_t
Base typedefs, 80
zb_int_t
Base typedefs, 81
zb_long_t
Base typedefs, 81
zb_mac_cb_ent_s, 111
zb_mac_device_table_s, 112
zb_mac_pib_attr_t
MAC API, 62
zb_mac_status_e
MAC API, 61
zb_mac_status_t
MAC API, 60
zb_mlme_get_confirm
MAC API, 59
zb_mlme_get_confirm_s, 112
zb_mlme_get_confirm_t
MAC API, 60
zb_mlme_get_request
MAC API, 59
zb_mlme_get_request_s, 112
zb_mlme_get_request_t
MAC API, 60
zb_mlme_set_confirm
MAC API, 59
zb_mlme_set_confirm_s, 113
zb_mlme_set_confirm_t
MAC API, 61
zb_mlme_set_request
MAC API, 59
zb_mlme_set_request_s, 113
zb_mlme_set_request_t
MAC API, 61
zb_nib_attribute_e
NWK Informational Base, 56
zb_nib_attribute_t
NWK Informational Base, 56
zb_nlde_data_req_s, 113
addr_mode, 114
discovery_route, 114
dst_addr, 114
ndsu_handle, 114
nonmember_radius, 114
radius, 114
security_enable, 114
zb_nlde_data_req_t
NWK functions visible to applications, 50
zb_nlde_data_request
NWK functions visible to applications, 48
zb_nlme_get_confirm
NWK Informational Base, 55
zb_nlme_get_confirm_s, 115
attribute_length, 115
nib_attribute, 115
status, 115
zb_nlme_get_confirm_t
NWK Informational Base, 56
zb_nlme_get_request
NWK Informational Base, 54
zb_nlme_get_request_s, 115
nib_attribute, 116
zb_nlme_get_request_t
NWK Informational Base, 56
zb_nlme_send_status
NWK functions visible to applications, 49
zb_nlme_send_status_s, 116
dest_addr, 116
ndsu_handle, 116
status, 116
zb_nlme_send_status_t
NWK functions visible to applications, 50
zb_nlme_set_confirm
NWK Informational Base, 55
zb_nlme_set_confirm_s, 117
nib_attribute, 117
status, 117
zb_nlme_set_confirm_t
NWK Informational Base, 56
zb_nlme_set_request
NWK Informational Base, 55
zb_nlme_set_request_s, 117
nib_attribute, 118
zb_nlme_set_request_t
NWK Informational Base, 56
zb_nlme_status_indication_s, 118
network_addr, 118
status, 118
zb_nlme_status_indication_t
NWK functions visible to applications, 50
zb_nwk_broadcast_address_e
NWK functions visible to applications, 51
zb_nwk_broadcast_address_t
NWK functions visible to applications, 50
zb_nwk_command_status_e
NWK functions visible to applications, 52
zb_nwk_command_status_t
NWK functions visible to applications, 50
zb_nwk_status_e
NWK functions visible to applications, 51
zb_nwk_status_t
NWK functions visible to applications, 50
zb_put_next_htole16
Base typedefs, 78
zb_sbitfield_t
Base typedefs, 80
zb_sched_globals_s, 121
cb_q, 121
tm_buffer, 121

ZB_LIST_DEFINE, 121
ZB_STK_DEFINE, 121
zb_sched_globals_t
Scheduler, 92
zb_sched_init
Scheduler, 89
zb_sched_loop_iteration
Scheduler, 89
zb_schedule_alarm
Scheduler, 90
zb_schedule_alarm_cancel
Scheduler, 90
zb_schedule_callback
Scheduler, 89
zb_schedule_mac_cb
Scheduler, 90
zb_secur_send_nwk_key_switch
Security subsystem API, 63
zb_secur_send_nwk_key_update_br
Security subsystem API, 63
zb_secur_setup_preconfigured_key
Security subsystem API, 63
zb_short_t
Base typedefs, 81
zb_time_t
Time, 95
zb_tm_q_ent_s, 122
func, 122
param, 122
run_time, 122
zb_tm_q_ent_t
Scheduler, 92
zb_uchar_t
Base typedefs, 80
zb_uint16_t
Base typedefs, 80
zb_uint32_t
Base typedefs, 80
zb_uint8_t
Base typedefs, 80
zb_uint_t
Base typedefs, 81
zb_ulong_t
Base typedefs, 81
zb_ushort_t
Base typedefs, 81
zb_voidp_t
Base typedefs, 81
zb_zdo_active_ep_req
ZDO discovery services, 23
zb_zdo_active_ep_req_s, 122
nwk_addr, 123
zb_zdo_active_ep_req_t
ZDO discovery services, 26
zb_zdo_add_group_req
ZDO management services, 34
zb_zdo_bind_req
ZDO management services, 32
zb_zdo_bind_req_head_s, 123
cluster_id, 123
dst_addr_mode, 123
src_address, 123
src_endp, 123
zb_zdo_bind_req_head_t
ZDO management services, 36
zb_zdo_bind_req_param_s, 124
cluster_id, 124
dst_addr_mode, 124
dst_address, 124
dst_endp, 124
req_dst_addr, 125
src_address, 124
src_endp, 124
zb_zdo_bind_req_param_t
ZDO management services, 36
zb_zdo_bind_req_tail_1_s, 125
dst_addr, 125
zb_zdo_bind_req_tail_1_t
ZDO management services, 36
zb_zdo_bind_req_tail_2_s, 125
dst_addr, 126
dst_endp, 126
zb_zdo_bind_req_tail_2_t
ZDO management services, 36
zb_zdo_bind_resp_s, 126
zb_zdo_configuration_attributes_e, 126
permit_join_duration, 126
zb_zdo_desc_resp_hdr_s, 126
nwk_addr, 127
status, 127
zb_zdo_desc_resp_hdr_t
ZDO discovery services, 26
zb_zdo_end_device_bind_req_head_s, 127
binding_target, 127
num_in_cluster, 128
profile_id, 128
src_endp, 127
src_ieee_addr, 127
zb_zdo_end_device_bind_req_head_t
ZDO management services, 36
zb_zdo_end_device_bind_req_tail_s, 128
num_out_cluster, 128
zb_zdo_end_device_bind_req_tail_t
ZDO management services, 36
zb_zdo_end_device_bind_resp_s, 128
zb_zdo_ep_resp_s, 129
ep_count, 129
nwk_addr, 129
status, 129
zb_zdo_ep_resp_t
ZDO discovery services, 26
zb_zdo_ieee_addr_req
ZDO discovery services, 19
zb_zdo_ieee_addr_req_s, 129
nwk_addr, 130
request_type, 130

start_index, 130
zb_zdo_ieee_addr_req_t
ZDO discovery services, 25
zb_zdo_match_desc_param_s, 130
cluster_list, 131
num_in_clusters, 130
num_out_clusters, 130
nwk_addr, 130
profile_id, 130
zb_zdo_match_desc_param_t
ZDO discovery services, 26
zb_zdo_match_desc_req
ZDO discovery services, 23
zb_zdo_match_desc_req_head_s, 131
num_in_clusters, 131
nwk_addr, 131
profile_id, 131
zb_zdo_match_desc_req_head_t
ZDO discovery services, 26
zb_zdo_match_desc_req_tail_s, 132
num_out_clusters, 132
zb_zdo_match_desc_req_tail_t
ZDO discovery services, 27
zb_zdo_match_desc_resp_s, 132
match_len, 132
nwk_addr, 132
status, 132
zb_zdo_match_desc_resp_t
ZDO discovery services, 27
zb_zdo_mgmt_leave_param_s, 133
device_address, 133
dst_addr, 133
zb_zdo_mgmt_leave_param_t
ZDO management services, 36
zb_zdo_mgmt_leave_req_s, 133
device_address, 134
zb_zdo_mgmt_leave_req_t
ZDO management services, 36
zb_zdo_mgmt_leave_res_s, 134
zb_zdo_mgmt_leave_res_t
ZDO management services, 36
zb_zdo_mgmt_lqi_param_s, 134
dst_addr, 135
start_index, 135
zb_zdo_mgmt_lqi_param_t
ZDO management services, 35
zb_zdo_mgmt_lqi_req
ZDO management services, 31
zb_zdo_mgmt_lqi_req_s, 135
start_index, 135
zb_zdo_mgmt_lqi_req_t
ZDO management services, 35
zb_zdo_mgmt_lqi_resp_s, 135
neighbor_table_entries, 136
neighbor_table_list_count, 136
start_index, 136
status, 136
zb_zdo_mgmt_lqi_resp_t
ZDO management services, 36
zb_zdo_mgmt_nwk_update_notify_hdr_s, 136
scanned_channels, 137
scanned_channels_list_count, 137
status, 137
total_transmissions, 137
transmission_failures, 137
zb_zdo_mgmt_nwk_update_notify_hdr_t
ZDO management services, 35
zb_zdo_mgmt_nwk_update_notify_param_s, 137
dst_addr, 138
energy_values, 137
hdr, 137
tsn, 138
zb_zdo_mgmt_nwk_update_notify_param_t
ZDO management services, 35
zb_zdo_mgmt_nwk_update_req
ZDO management services, 30
zb_zdo_mgmt_nwk_update_req_hdr_s, 138
scan_channels, 138
scan_duration, 138
zb_zdo_mgmt_nwk_update_req_hdr_t
ZDO management services, 35
zb_zdo_mgmt_nwk_update_req_s, 138
dst_addr, 139
hdr, 139
manager_addr, 139
scan_count, 139
update_id, 139
zb_zdo_mgmt_nwk_update_req_t
ZDO management services, 35
zb_zdo_mgmt_permit_joining_req_param_s, 139
zb_zdo_mgmt_permit_joining_req_param_t
ZDO management services, 37
zb_zdo_mgmt_permit_joining_req_s, 140
zb_zdo_mgmt_permit_joining_req_t
ZDO management services, 37
zb_zdo_neighbor_table_record_s, 140
depth, 141
ext_addr, 140
ext_pan_id, 140
lqi, 141
network_addr, 141
permit_join, 141
type_flags, 141
zb_zdo_neighbor_table_record_t
ZDO management services, 36
zb_zdo_node_desc_req
ZDO discovery services, 20
zb_zdo_node_desc_req_s, 141
nwk_addr, 141
zb_zdo_node_desc_req_t
ZDO discovery services, 25
zb_zdo_node_desc_resp_s, 142
hdr, 142
node_desc, 142
zb_zdo_node_desc_resp_t
ZDO discovery services, 26

zb_zdo_nwk_addr_req
    ZDO discovery services, 18
zb_zdo_nwk_addr_req_param_s, 142
    dst_addr, 142
    ieee_addr, 142
    request_type, 143
    start_index, 143
zb_zdo_nwk_addr_req_param_t
    ZDO discovery services, 25
zb_zdo_nwk_addr_req_s, 143
    ieee_addr, 143
    request_type, 143
    start_index, 143
zb_zdo_nwk_addr_req_t
    ZDO discovery services, 25
zb_zdo_nwk_addr_resp_head_s, 144
    ieee_addr, 144
    nwk_addr, 144
    status, 144
zb_zdo_power_desc_req
    ZDO discovery services, 21
zb_zdo_power_desc_req_s, 144
    nwk_addr, 144
zb_zdo_power_desc_req_t
    ZDO discovery services, 26
zb_zdo_power_desc_resp_s, 145
    hdr, 145
    power_desc, 145
zb_zdo_power_desc_resp_t
    ZDO discovery services, 26
zb_zdo_simple_desc_req
    ZDO discovery services, 22
zb_zdo_simple_desc_req_s, 145
    endpoint, 145
    nwk_addr, 145
zb_zdo_simple_desc_req_t
    ZDO discovery services, 26
zb_zdo_simple_desc_resp_hdr_s, 146
    length, 146
    nwk_addr, 146
    status, 146
zb_zdo_simple_desc_resp_hdr_t
    ZDO discovery services, 26
zb_zdo_simple_desc_resp_s, 146
    hdr, 147
    simple_desc, 147
zb_zdo_simple_desc_resp_t
    ZDO discovery services, 26
zb_zdo_startup_complete
    ZDO init and main() structure, 10
zb_zdo_system_server_discovery_param_t
    ZDO discovery services, 27
zb_zdo_system_server_discovery_req
    ZDO discovery services, 24
zb_zdo_system_server_discovery_req_s, 147
    server_mask, 147
zb_zdo_system_server_discovery_req_t
    ZDO discovery services, 27
zb_zdo_system_server_discovery_resp_s, 147
    server_mask, 148
    status, 148
zb_zdo_system_server_discovery_resp_t
    ZDO discovery services, 27
zb_zdo_unbind_req
    ZDO management services, 33
zb_zdp_status_e
    ZDO base constants and definitions, 14
zb_zdp_status_t
    ZDO base constants and definitions, 14
zdo_cmd_no_resp
    zb_buf_hdr_s, 109
zdo_dev_start
    ZDO init and main() structure, 10
zdo_main_loop
    ZDO init and main() structure, 10
zdo_mgmt_leave_req
    ZDO management services, 34