# TF-M Reference Manual

Generated by Doxygen 1.8.17

**1 TF-M Reference Manual** 1

**2 Module Index** 3

2.1 Modules . . . 3

**3 Data Structure Index** 5

3.1 Data Structures . . . 5

**4 File Index** 9

4.1 File List . . . 9

**5 Module Documentation** 19

5.1 Set of functions implementing a thin shim . . . 19

5.1.1 Detailed Description . . . 20

5.1.2 Function Documentation . . . 20

5.1.2.1 tfm_crypto_aead_interface() . . . 20

5.1.2.2 tfm_crypto_asymmetric_encrypt_interface() . . . 20

5.1.2.3 tfm_crypto_asymmetric_sign_interface() . . . 21

5.1.2.4 tfm_crypto_cipher_interface() . . . 22

5.1.2.5 tfm_crypto_hash_interface() . . . 22

5.1.2.6 tfm_crypto_key_derivation_interface() . . . 23

5.1.2.7 tfm_crypto_key_management_interface() . . . 24

5.1.2.8 tfm_crypto_mac_interface() . . . 25

5.1.2.9 tfm_crypto_random_interface() . . . 25

5.2 Function that implement allocation and deallocation of . . . 27

5.2.1 Detailed Description . . . 27

5.2.2 Function Documentation . . . 27

5.2.2.1 tfm_crypto_init_alloc() . . . 27

5.2.2.2 tfm_crypto_operation_alloc() . . . 28

5.2.2.3 tfm_crypto_operation_lookup() . . . 28

5.2.2.4 tfm_crypto_operation_release() . . . 28

5.3 Set of functions implementing the abstractions of the underlying cryptographic . . . 30

5.3.1 Detailed Description . . . 30

5.3.2 Function Documentation . . . 30

5.3.2.1 mbedtls_psa_platform_get_builtin_key() . . . 30

5.3.2.2 tfm_crypto_core_library_init() . . . 31

5.3.2.3 tfm_crypto_library_get_info() . . . 31

5.3.2.4 tfm_crypto_library_get_library_key_id_set_owner() . . . 32

5.3.2.5 tfm_crypto_library_key_id_init() . . . 32

5.4 Tfm_builtin_key_loader . . . 34

5.4.1 Detailed Description . . . 34

5.4.2 Function Documentation . . . 34

5.4.2.1 tfm_builtin_key_loader_get_builtin_key() . . . 34

5.4.2.2 tfm_builtin_key_loader_get_key_buffer_size() . . . 35

5.4.2.3 tfm_builtin_key_loader_init() . . . 35

**6 Data Structure Documentation** **37**

6.1 _MTB_SRF_DATA_ALIGN Struct Reference . . . 37

6.1.1 Detailed Description . . . 38

6.1.2 Field Documentation . . . 38

6.1.2.1 call_type . . . 38

6.1.2.2 params . . . 38

6.1.2.3 reply . . . 38

6.1.2.4 semaphore_idx . . . 38

6.2 asset_desc_t Struct Reference . . . 39

6.2.1 Detailed Description . . . 39

6.2.2 Field Documentation . . . 39

6.2.2.1 "@12 . . . 39

6.2.2.2 attr . . . 39

6.2.2.3 dev . . . 39

6.2.2.4 dev_ref . . . 40

6.2.2.5 limit . . . 40

6.2.2.6 mem . . . 40

6.2.2.7 reserved . . . 40

6.2.2.8 start . . . 40

6.3 attest_boot_data Struct Reference . . . 41

6.3.1 Detailed Description . . . 41

6.3.2 Field Documentation . . . 41

6.3.2.1 data . . . 41

6.3.2.2 header . . . 42

6.4 attest_token_encode_ctx Struct Reference . . . 42

6.4.1 Detailed Description . . . 42

6.4.2 Field Documentation . . . 42

6.4.2.1 cbor_enc_ctx . . . 42

6.4.2.2 key_select . . . 43

6.4.2.3 opt_flags . . . 43

6.4.2.4 signer_ctx . . . 43

6.5 bi_list_node_t Struct Reference . . . 43

6.5.1 Detailed Description . . . 44

6.5.2 Field Documentation . . . 44

6.5.2.1 bnext . . . 44

6.5.2.2 bprev . . . 44

6.6 boot_data_access_policy Struct Reference . . . 44

6.6.1 Detailed Description . . . 44

6.6.2 Field Documentation . . . 45

6.6.2.1 major_type . . . 45

6.6.2.2 partition_id . . . 45
6.7 client_id_region_t Struct Reference . . . 45
6.7.1 Detailed Description . . . 45
6.7.2 Field Documentation . . . 45
6.7.2.1 base . . . 45
6.7.2.2 limit . . . 46
6.7.2.3 owner . . . 46
6.8 client_params_t Struct Reference . . . 46
6.8.1 Detailed Description . . . 46
6.8.2 Field Documentation . . . 47
6.8.2.1 ns_client_id_stateless . . . 47
6.8.2.2 p_invecs . . . 47
6.8.2.3 p_outvecs . . . 47
6.9 composite_addr_t Union Reference . . . 47
6.9.1 Detailed Description . . . 47
6.9.2 Field Documentation . . . 48
6.9.2.1 p_byte . . . 48
6.9.2.2 p_word . . . 48
6.9.2.3 uint_addr . . . 48
6.10 connection_t Struct Reference . . . 48
6.10.1 Detailed Description . . . 49
6.10.2 Field Documentation . . . 49
6.10.2.1 caller_outvec . . . 49
6.10.2.2 ctrl_param . . . 49
6.10.2.3 invec_accessed . . . 49
6.10.2.4 invec_base . . . 50
6.10.2.5 msg . . . 50
6.10.2.6 outvec_base . . . 50
6.10.2.7 outvec_written . . . 50
6.10.2.8 p_client . . . 50
6.10.2.9 service . . . 50
6.10.2.10 status . . . 51
6.11 context_ctrl_t Struct Reference . . . 51
6.11.1 Detailed Description . . . 51
6.11.2 Field Documentation . . . 51
6.11.2.1 exc_ret . . . 51
6.11.2.2 sp . . . 51
6.11.2.3 sp_base . . . 52
6.11.2.4 sp_limit . . . 52
6.12 context_flih_ret_t Struct Reference . . . 52
6.12.1 Detailed Description . . . 53
6.12.2 Field Documentation . . . 53

6.12.2.1 addi_ctx . . . 53
6.12.2.2 dummy . . . 53
6.12.2.3 exc_return . . . 53
6.12.2.4 psp . . . 53
6.12.2.5 psplim . . . 53
6.12.2.6 stack_seal . . . 54
6.12.2.7 state_ctx . . . 54
6.13 critical_section_t Struct Reference . . . 54
6.13.1 Detailed Description . . . 54
6.13.2 Field Documentation . . . 54
6.13.2.1 state . . . 54
6.14 formatted_buffer_t Struct Reference . . . 55
6.14.1 Detailed Description . . . 55
6.14.2 Field Documentation . . . 55
6.14.2.1 buf . . . 55
6.14.2.2 pos . . . 55
6.15 full_context_t Struct Reference . . . 55
6.15.1 Detailed Description . . . 56
6.15.2 Field Documentation . . . 56
6.15.2.1 addi_ctx . . . 56
6.15.2.2 stat_ctx . . . 56
6.16 fwu_image_info_data_s Struct Reference . . . 56
6.16.1 Detailed Description . . . 57
6.16.2 Field Documentation . . . 57
6.16.2.1 data . . . 57
6.16.2.2 header . . . 57
6.17 ifx_debug_policy_t Struct Reference . . . 57
6.17.1 Detailed Description . . . 57
6.17.2 Field Documentation . . . 58
6.17.2.1 access_cfg . . . 58
6.17.2.2 reserved . . . 58
6.17.2.3 syscpuss_ap_ctl . . . 58
6.18 ifx_driver_flash_obj_t Struct Reference . . . 58
6.18.1 Detailed Description . . . 59
6.18.2 Field Documentation . . . 59
6.18.2.1 flash_info . . . 59
6.18.2.2 start_address . . . 59
6.19 ifx_driver_rram_obj_t Struct Reference . . . 59
6.19.1 Detailed Description . . . 60
6.19.2 Field Documentation . . . 60
6.19.2.1 flash_info . . . 60
6.19.2.2 rram_base . . . 60

6.19.2.3 start_address . . . 61
6.20 ifx_driver_smif_mem_t Struct Reference . . . 61
6.20.1 Detailed Description . . . 61
6.20.2 Field Documentation . . . 61
6.20.2.1 block_config . . . 61
6.20.2.2 mem_config . . . 62
6.20.2.3 mem_config_dual_quad_pair . . . 62
6.20.2.4 smif_base . . . 62
6.20.2.5 smif_context . . . 62
6.21 ifx_driver_smif_obj_t Struct Reference . . . 63
6.21.1 Detailed Description . . . 63
6.21.2 Field Documentation . . . 63
6.21.2.1 flash_info . . . 64
6.21.2.2 memory . . . 64
6.21.2.3 offset . . . 64
6.21.2.4 size . . . 64
6.22 ifx_flash_driver_instance_t Struct Reference . . . 65
6.22.1 Detailed Description . . . 65
6.22.2 Field Documentation . . . 65
6.22.2.1 driver . . . 65
6.22.2.2 handler . . . 66
6.23 ifx_flash_driver_t Struct Reference . . . 66
6.23.1 Detailed Description . . . 66
6.23.2 Field Documentation . . . 66
6.23.2.1 EraseChip . . . 67
6.23.2.2 EraseSector . . . 67
6.23.2.3 GetCapabilities . . . 67
6.23.2.4 GetInfo . . . 67
6.23.2.5 GetStatus . . . 67
6.23.2.6 GetVersion . . . 67
6.23.2.7 Initialize . . . 68
6.23.2.8 PowerControl . . . 68
6.23.2.9 ProgramData . . . 68
6.23.2.10 ReadData . . . 68
6.23.2.11 Uninitialize . . . 68
6.24 ifx_memory_config_t Struct Reference . . . 68
6.24.1 Detailed Description . . . 69
6.24.2 Field Documentation . . . 69
6.24.2.1 mpc . . . 69
6.24.2.2 mpc_block_size . . . 69
6.24.2.3 s_address . . . 69
6.24.2.4 size . . . 70

6.25 ifx_mpc_cfg_t Struct Reference . . . 70
6.25.1 Detailed Description . . . 70
6.25.2 Field Documentation . . . 70
6.25.2.1 attr . . . 70
6.25.2.2 mem_offset . . . 70
6.25.2.3 mem_size . . . 71
6.26 ifx_mpc_ext_cache_cfg_info_t Struct Reference . . . 71
6.26.1 Detailed Description . . . 71
6.26.2 Field Documentation . . . 71
6.26.2.1 attr . . . 71
6.26.2.2 first_block_idx . . . 71
6.26.2.3 last_block_idx . . . 72
6.27 ifx_mpc_numbered_mmio_config_t Struct Reference . . . 72
6.27.1 Detailed Description . . . 72
6.27.2 Field Documentation . . . 72
6.27.2.1 ns_mask . . . 72
6.27.2.2 pc_mask . . . 73
6.28 ifx_mpc_policy_t Struct Reference . . . 73
6.28.1 Detailed Description . . . 73
6.28.2 Field Documentation . . . 73
6.28.2.1 mpc_struct . . . 74
6.28.2.2 n_flash_mpc . . . 74
6.28.2.3 n_ram_mpc . . . 74
6.28.2.4 unused . . . 74
6.29 ifx_mpc_raw_region_config_t Struct Reference . . . 74
6.29.1 Detailed Description . . . 75
6.29.2 Field Documentation . . . 75
6.29.2.1 mpc_base . . . 75
6.29.2.2 mpc_block_size . . . 75
6.29.2.3 ns_mask . . . 75
6.29.2.4 offset . . . 75
6.29.2.5 r_mask . . . 76
6.29.2.6 size . . . 76
6.29.2.7 w_mask . . . 76
6.30 ifx_mpc_region_config_t Struct Reference . . . 76
6.30.1 Detailed Description . . . 76
6.30.2 Field Documentation . . . 77
6.30.2.1 address . . . 77
6.30.2.2 ns_mask . . . 77
6.30.2.3 r_mask . . . 77
6.30.2.4 size . . . 77
6.30.2.5 w_mask . . . 77

6.31 ifx_msc_agc_resp_config_t Struct Reference . . . 78
6.31.1 Detailed Description . . . 78
6.31.2 Field Documentation . . . 78
6.31.2.1 bus_master . . . 78
6.31.2.2 gate_resp . . . 78
6.31.2.3 sec_resp . . . 78
6.32 ifx_msc_agc_resp_config_v1_t Struct Reference . . . 79
6.32.1 Detailed Description . . . 79
6.32.2 Field Documentation . . . 79
6.32.2.1 bus_master . . . 79
6.32.2.2 gate_resp . . . 79
6.32.2.3 sec_resp . . . 79
6.33 ifx_nv_counters Struct Reference . . . 80
6.33.1 Detailed Description . . . 80
6.33.2 Field Documentation . . . 80
6.33.2.1 bytes . . . 80
6.33.2.2 checksum . . . 80
6.33.2.3 counters . . . 81
6.33.2.4 init_value . . . 81
6.33.2.5 nv_cnt . . . 81
6.34 ifx_oem_policy_t Struct Reference . . . 81
6.34.1 Detailed Description . . . 82
6.34.2 Field Documentation . . . 82
6.34.2.1 boundary_scan . . . 82
6.34.2.2 dbg_policy . . . 82
6.34.2.3 debug_key . . . 82
6.34.2.4 padding . . . 82
6.34.2.5 reserved . . . 83
6.34.2.6 rma . . . 83
6.34.2.7 warm_boot . . . 83
6.35 ifx_partition_info_t Struct Reference . . . 83
6.35.1 Detailed Description . . . 84
6.35.2 Field Documentation . . . 84
6.35.2.1 asset_cnt . . . 84
6.35.2.2 ifx_domain . . . 84
6.35.2.3 ifx_ldinfo . . . 84
6.35.2.4 p_assets . . . 85
6.35.2.5 p_ldinfo . . . 85
6.35.2.6 p_peri_regions . . . 85
6.35.2.7 peri_region_count . . . 85
6.36 ifx_partition_load_info_t Struct Reference . . . 86
6.36.1 Detailed Description . . . 86

6.36.2 Field Documentation . . . 86
6.36.2.1 privileged . . . 86
6.36.2.2 protect_cfg_enabled . . . 87
6.37 ifx_plat_epc2_keys_t Struct Reference . . . 87
6.37.1 Detailed Description . . . 87
6.37.2 Field Documentation . . . 87
6.37.2.1 huk . . . 87
6.37.2.2 huk_checksum . . . 88
6.37.2.3 iak_private . . . 88
6.37.2.4 iak_private_checksum . . . 88
6.37.2.5 iak_public . . . 88
6.37.2.6 iak_public_checksum . . . 88
6.38 ifx_ppc_named_mmio_config_t Struct Reference . . . 88
6.38.1 Detailed Description . . . 89
6.38.2 Field Documentation . . . 89
6.38.2.1 allow_unpriv . . . 89
6.38.2.2 pc_mask . . . 89
6.38.2.3 sec_attr . . . 89
6.39 ifx_ppc_regions_config_t Struct Reference . . . 90
6.39.1 Detailed Description . . . 90
6.39.2 Field Documentation . . . 90
6.39.2.1 region_count . . . 90
6.39.2.2 regions . . . 90
6.40 ifx_ppcx_config_t Struct Reference . . . 90
6.40.1 Detailed Description . . . 91
6.40.2 Field Documentation . . . 91
6.40.2.1 allow_unpriv . . . 91
6.40.2.2 pc_mask . . . 91
6.40.2.3 region_count . . . 91
6.40.2.4 regions . . . 92
6.40.2.5 sec_attr . . . 92
6.41 ifx_protection_region_config_t Struct Reference . . . 92
6.41.1 Detailed Description . . . 92
6.41.2 Field Documentation . . . 92
6.41.2.1 mpu_regions_enable . . . 93
6.42 ifx_rram_counter_t Struct Reference . . . 93
6.42.1 Detailed Description . . . 93
6.42.2 Field Documentation . . . 93
6.42.2.1 bytes . . . 93
6.42.2.2 checksum . . . 93
6.42.2.3 data . . . 94
6.42.2.4 value . . . 94

6.43 irq_load_info_t Struct Reference . . . 94
6.43.1 Detailed Description . . . 94
6.43.2 Field Documentation . . . 94
6.43.2.1 client_id_base . . . 94
6.43.2.2 client_id_limit . . . 95
6.43.2.3 flih_func . . . 95
6.43.2.4 init . . . 95
6.43.2.5 pid . . . 95
6.43.2.6 signal . . . 95
6.43.2.7 source . . . 95
6.44 irq_t Struct Reference . . . 96
6.44.1 Detailed Description . . . 96
6.44.2 Field Documentation . . . 96
6.44.2.1 p_ildi . . . 96
6.44.2.2 p_pt . . . 96
6.45 its_block_meta_t Struct Reference . . . 97
6.45.1 Detailed Description . . . 97
6.45.2 Field Documentation . . . 97
6.45.2.1 data_start . . . 97
6.45.2.2 free_size . . . 97
6.45.2.3 phy_id . . . 98
6.46 its_file_meta_t Struct Reference . . . 98
6.46.1 Detailed Description . . . 98
6.46.2 Field Documentation . . . 98
6.46.2.1 cur_size . . . 99
6.46.2.2 data_idx . . . 99
6.46.2.3 flags . . . 99
6.46.2.4 id . . . 99
6.46.2.5 lblock . . . 99
6.46.2.6 max_size . . . 99
6.47 its_flash_config_t Struct Reference . . . 100
6.47.1 Detailed Description . . . 100
6.47.2 Field Documentation . . . 100
6.47.2.1 block_size . . . 100
6.47.2.2 context . . . 101
6.47.2.3 erase_val . . . 101
6.47.2.4 flash_area_addr . . . 101
6.47.2.5 num_blocks . . . 101
6.47.2.6 program_unit . . . 101
6.48 its_flash_fs_config_t Struct Reference . . . 102
6.48.1 Detailed Description . . . 102
6.48.2 Field Documentation . . . 102

6.48.2.1 flash_cfg . . . 102
6.48.2.2 max_file_size . . . 103
6.48.2.3 max_num_files . . . 103
6.48.2.4 ops . . . 103
6.49 its_flash_fs_ctx_t Struct Reference . . . 103
6.49.1 Detailed Description . . . 104
6.49.2 Field Documentation . . . 104
6.49.2.1 active_metablock . . . 104
6.49.2.2 cfg . . . 104
6.49.2.3 meta_block_header . . . 104
6.49.2.4 scratch_metablock . . . 105
6.50 its_flash_fs_file_info_t Struct Reference . . . 105
6.50.1 Detailed Description . . . 105
6.50.2 Field Documentation . . . 105
6.50.2.1 flags . . . 105
6.50.2.2 size_current . . . 106
6.50.2.3 size_max . . . 106
6.51 its_flash_nand_dev_t Struct Reference . . . 106
6.51.1 Detailed Description . . . 106
6.51.2 Field Documentation . . . 106
6.51.2.1 buf_block_id_0 . . . 107
6.51.2.2 buf_block_id_1 . . . 107
6.51.2.3 buf_size . . . 107
6.51.2.4 driver . . . 107
6.51.2.5 sector_size . . . 107
6.51.2.6 write_buf_0 . . . 107
6.51.2.7 write_buf_1 . . . 108
6.52 its_flash_ops_t Struct Reference . . . 108
6.52.1 Detailed Description . . . 108
6.52.2 Field Documentation . . . 108
6.52.2.1 erase . . . 108
6.52.2.2 flush . . . 109
6.52.2.3 init . . . 109
6.52.2.4 read . . . 110
6.52.2.5 write . . . 110
6.53 its_flash_ram_dev_t Struct Reference . . . 111
6.53.1 Detailed Description . . . 111
6.53.2 Field Documentation . . . 111
6.53.2.1 buffer . . . 112
6.53.2.2 size . . . 112
6.54 its_metadata_block_header_comp_t Struct Reference . . . 112
6.54.1 Detailed Description . . . 112

6.54.2 Field Documentation . . . 112
6.54.2.1 active_swap_count . . . 113
6.54.2.2 fs_version . . . 113
6.54.2.3 scratch_dblock . . . 113
6.55 its_metadata_block_header_t Struct Reference . . . 113
6.55.1 Detailed Description . . . 114
6.55.2 Field Documentation . . . 114
6.55.2.1 active_swap_count . . . 114
6.55.2.2 fs_version . . . 114
6.55.2.3 metadata_xor . . . 114
6.55.2.4 scratch_dblock . . . 115
6.56 mailbox_init_t Struct Reference . . . 115
6.56.1 Detailed Description . . . 115
6.56.2 Member Function Documentation . . . 116
6.56.2.1 __ALIGNED() . . . 116
6.56.3 Field Documentation . . . 116
6.56.3.1 slot_count . . . 116
6.56.3.2 slots . . . 116
6.57 mailbox_msg_t Struct Reference . . . 116
6.57.1 Detailed Description . . . 117
6.57.2 Field Documentation . . . 117
6.57.2.1 call_type . . . 117
6.57.2.2 client_id . . . 117
6.57.2.3 params . . . 117
6.58 mailbox_reply_t Struct Reference . . . 117
6.58.1 Detailed Description . . . 118
6.58.2 Field Documentation . . . 118
6.58.2.1 out_vec_len . . . 118
6.58.2.2 return_val . . . 118
6.59 mailbox_slot_t Struct Reference . . . 118
6.59.1 Detailed Description . . . 118
6.59.2 Member Function Documentation . . . 119
6.59.2.1 __ALIGNED() [1/2] . . . 119
6.59.2.2 __ALIGNED() [2/2] . . . 119
6.60 mailbox_status_t Struct Reference . . . 119
6.60.1 Detailed Description . . . 119
6.60.2 Field Documentation . . . 119
6.60.2.1 pend_slots . . . 119
6.60.2.2 replied_slots . . . 120
6.61 mpu_armv8m_region_cfg_t Struct Reference . . . 120
6.61.1 Detailed Description . . . 120
6.61.2 Field Documentation . . . 120

6.61.2.1 attr_access . . . 120
6.61.2.2 attr_exec . . . 120
6.61.2.3 attr_sh . . . 121
6.61.2.4 region_attridx . . . 121
6.61.2.5 region_base . . . 121
6.61.2.6 region_limit . . . 121
6.62 ns_mailbox_queue_t Struct Reference . . . 121
6.62.1 Detailed Description . . . 122
6.62.2 Member Function Documentation . . . 122
6.62.2.1 __ALIGNED() [1/2] . . . 122
6.62.2.2 __ALIGNED() [2/2] . . . 122
6.62.3 Field Documentation . . . 122
6.62.3.1 empty_slots . . . 122
6.62.3.2 is_full . . . 123
6.62.3.3 slots . . . 123
6.63 ns_mailbox_req_t Struct Reference . . . 123
6.63.1 Detailed Description . . . 124
6.63.2 Field Documentation . . . 124
6.63.2.1 call_type . . . 124
6.63.2.2 client_id . . . 124
6.63.2.3 owner . . . 124
6.63.2.4 params_ptr . . . 124
6.63.2.5 reply . . . 124
6.63.2.6 woken_flag . . . 125
6.64 ns_mailbox_slot_t Struct Reference . . . 125
6.64.1 Detailed Description . . . 125
6.64.2 Field Documentation . . . 125
6.64.2.1 is_woken . . . 126
6.64.2.2 owner . . . 126
6.64.2.3 reply . . . 126
6.65 ns_mailbox_stats_res_t Struct Reference . . . 126
6.65.1 Detailed Description . . . 126
6.65.2 Field Documentation . . . 126
6.65.2.1 avg_nr_slots . . . 127
6.65.2.2 avg_nr_slots_tenths . . . 127
6.66 partition_head_t Struct Reference . . . 127
6.66.1 Detailed Description . . . 127
6.66.2 Field Documentation . . . 128
6.66.2.1 next . . . 128
6.66.2.2 reserved . . . 128
6.67 partition_load_info_t Struct Reference . . . 128
6.67.1 Detailed Description . . . 128

6.67.2 Field Documentation . . . 129
6.67.2.1 client_id_base . . . 129
6.67.2.2 client_id_limit . . . 129
6.67.2.3 entry . . . 129
6.67.2.4 flags . . . 129
6.67.2.5 heap_size . . . 129
6.67.2.6 nassets . . . 130
6.67.2.7 ndeps . . . 130
6.67.2.8 nirqs . . . 130
6.67.2.9 nservices . . . 130
6.67.2.10 pid . . . 130
6.67.2.11 psa_ff_ver . . . 130
6.67.2.12 stack_size . . . 131
6.68 partition_t Struct Reference . . . 131
6.68.1 Detailed Description . . . 131
6.68.2 Field Documentation . . . 132
6.68.2.1 boundary . . . 132
6.68.2.2 next . . . 132
6.68.2.3 p_handles . . . 132
6.68.2.4 p_ldinf . . . 132
6.68.2.5 signals_allowed . . . 132
6.68.2.6 signals_asserted . . . 133
6.68.2.7 signals_waiting . . . 133
6.68.2.8 state . . . 133
6.69 partition_tfm_sp_idle_load_info_t Struct Reference . . . 133
6.69.1 Detailed Description . . . 134
6.69.2 Field Documentation . . . 134
6.69.2.1 heap_addr . . . 134
6.69.2.2 load_info . . . 134
6.69.2.3 stack_addr . . . 134
6.70 partition_tfm_sp_ns_agent_tz_load_info_t Struct Reference . . . 134
6.70.1 Detailed Description . . . 135
6.70.2 Field Documentation . . . 135
6.70.2.1 heap_addr . . . 135
6.70.2.2 load_info . . . 135
6.70.2.3 stack_addr . . . 135
6.71 ps_crypto_t Union Reference . . . 135
6.71.1 Detailed Description . . . 136
6.71.2 Field Documentation . . . 136
6.71.2.1 client_id . . . 136
6.71.2.2 iv . . . 136
6.71.2.3 ref . . . 136

6.71.2.4 tag . . . 137
6.71.2.5 uid . . . 137
6.72 ps_obj_header_t Struct Reference . . . 137
6.72.1 Detailed Description . . . 138
6.72.2 Field Documentation . . . 138
6.72.2.1 fid . . . 138
6.72.2.2 info . . . 138
6.72.2.3 version . . . 138
6.73 ps_obj_table_ctx_t Struct Reference . . . 139
6.73.1 Detailed Description . . . 139
6.73.2 Field Documentation . . . 139
6.73.2.1 active_table . . . 139
6.73.2.2 obj_table . . . 140
6.73.2.3 scratch_table . . . 140
6.74 ps_obj_table_entry_t Struct Reference . . . 140
6.74.1 Detailed Description . . . 140
6.74.2 Field Documentation . . . 140
6.74.2.1 client_id . . . 140
6.74.2.2 uid . . . 141
6.74.2.3 version . . . 141
6.75 ps_obj_table_info_t Struct Reference . . . 141
6.75.1 Detailed Description . . . 141
6.75.2 Field Documentation . . . 141
6.75.2.1 fid . . . 142
6.75.2.2 version . . . 142
6.76 ps_obj_table_init_ctx_t Struct Reference . . . 142
6.76.1 Detailed Description . . . 143
6.76.2 Field Documentation . . . 143
6.76.2.1 p_table . . . 143
6.76.2.2 table_state . . . 143
6.77 ps_obj_table_t Struct Reference . . . 143
6.77.1 Detailed Description . . . 144
6.77.2 Field Documentation . . . 144
6.77.2.1 obj_db . . . 144
6.77.2.2 swap_count . . . 144
6.77.2.3 version . . . 144
6.78 ps_object_info_t Struct Reference . . . 145
6.78.1 Detailed Description . . . 145
6.78.2 Field Documentation . . . 145
6.78.2.1 create_flags . . . 145
6.78.2.2 current_size . . . 145
6.78.2.3 max_size . . . 146

6.79 ps_object_t Struct Reference . . . 146
6.79.1 Detailed Description . . . 146
6.79.2 Field Documentation . . . 147
6.79.2.1 data . . . 147
6.79.2.2 header . . . 147
6.80 psa_client_params_t Struct Reference . . . 147
6.80.1 Detailed Description . . . 148
6.80.2 Field Documentation . . . 148
6.80.2.1 handle . . . 148
6.80.2.2 in_len . . . 148
6.80.2.3 in_vec . . . 149
6.80.2.4 out_len . . . 149
6.80.2.5 out_vec . . . 149
6.80.2.6 psa_call_params . . . 149
6.80.2.7 psa_close_params . . . 149
6.80.2.8 psa_connect_params . . . 149
6.80.2.9 psa_params . . . 149
6.80.2.10 psa_version_params . . . 150
6.80.2.11 sid . . . 150
6.80.2.12 type . . . 150
6.80.2.13 version . . . 150
6.81 psa_fwu_component_info_t Struct Reference . . . 150
6.81.1 Detailed Description . . . 151
6.81.2 Field Documentation . . . 151
6.81.2.1 error . . . 151
6.81.2.2 flags . . . 151
6.81.2.3 impl . . . 151
6.81.2.4 location . . . 152
6.81.2.5 max_size . . . 152
6.81.2.6 state . . . 152
6.81.2.7 version . . . 152
6.82 psa_fwu_image_version_t Struct Reference . . . 152
6.82.1 Detailed Description . . . 153
6.82.2 Field Documentation . . . 153
6.82.2.1 build . . . 153
6.82.2.2 major . . . 153
6.82.2.3 minor . . . 153
6.82.2.4 patch . . . 153
6.83 psa_fwu_impl_info_t Struct Reference . . . 154
6.83.1 Detailed Description . . . 154
6.83.2 Field Documentation . . . 154
6.83.2.1 candidate_digest . . . 154

6.84 psa_invec Struct Reference . . . 154
6.84.1 Detailed Description . . . 154
6.84.2 Field Documentation . . . 155
6.84.2.1 base . . . 155
6.84.2.2 len . . . 155
6.85 psa_msg_t Struct Reference . . . 155
6.85.1 Detailed Description . . . 155
6.85.2 Field Documentation . . . 156
6.85.2.1 client_id . . . 156
6.85.2.2 handle . . . 156
6.85.2.3 in_size . . . 156
6.85.2.4 out_size . . . 156
6.85.2.5 rhandle . . . 156
6.85.2.6 type . . . 157
6.86 psa_outvec Struct Reference . . . 157
6.86.1 Detailed Description . . . 157
6.86.2 Field Documentation . . . 157
6.86.2.1 base . . . 157
6.86.2.2 len . . . 157
6.87 psa_storage_info_t Struct Reference . . . 158
6.87.1 Detailed Description . . . 158
6.87.2 Field Documentation . . . 158
6.87.2.1 capacity . . . 158
6.87.2.2 flags . . . 158
6.87.2.3 size . . . 158
6.88 service_head_t Struct Reference . . . 159
6.88.1 Detailed Description . . . 159
6.88.2 Field Documentation . . . 159
6.88.2.1 next . . . 159
6.88.2.2 reserved . . . 160
6.89 service_load_info_t Struct Reference . . . 160
6.89.1 Detailed Description . . . 160
6.89.2 Field Documentation . . . 160
6.89.2.1 flags . . . 160
6.89.2.2 name_strid . . . 160
6.89.2.3 sfn . . . 161
6.89.2.4 sid . . . 161
6.89.2.5 version . . . 161
6.90 service_t Struct Reference . . . 161
6.90.1 Detailed Description . . . 162
6.90.2 Field Documentation . . . 162
6.90.2.1 next . . . 162

6.90.2.2 p_ldinf . . . 162
6.90.2.3 partition . . . 162
6.91 shared_data_tlv_entry Struct Reference . . . 162
6.91.1 Detailed Description . . . 163
6.91.2 Field Documentation . . . 163
6.91.2.1 tlv_len . . . 163
6.91.2.2 tlv_type . . . 163
6.92 shared_data_tlv_header Struct Reference . . . 163
6.92.1 Detailed Description . . . 163
6.92.2 Field Documentation . . . 163
6.92.2.1 tlv_magic . . . 163
6.92.2.2 tlv_tot_len . . . 164
6.93 tfm_additional_context_t Struct Reference . . . 164
6.93.1 Detailed Description . . . 164
6.93.2 Field Documentation . . . 164
6.93.2.1 callee . . . 164
6.93.2.2 integ_sign . . . 164
6.93.2.3 reserved . . . 164
6.94 tfm_boot_data Struct Reference . . . 164
6.94.1 Detailed Description . . . 165
6.94.2 Field Documentation . . . 165
6.94.2.1 data . . . 165
6.94.2.2 header . . . 165
6.95 tfm_builtin_key_t Struct Reference . . . 165
6.95.1 Detailed Description . . . 166
6.95.2 Field Documentation . . . 166
6.95.2.1 attr . . . 166
6.95.2.2 is_loaded . . . 166
6.95.2.3 key . . . 166
6.95.2.4 key_len . . . 166
6.96 tfm_crypto_aead_pack_input Struct Reference . . . 166
6.96.1 Detailed Description . . . 166
6.96.2 Field Documentation . . . 167
6.96.2.1 nonce . . . 167
6.96.2.2 nonce_length . . . 167
6.97 tfm_crypto_key_id_s Struct Reference . . . 167
6.97.1 Detailed Description . . . 167
6.97.2 Field Documentation . . . 167
6.97.2.1 key_id . . . 167
6.97.2.2 owner . . . 167
6.98 tfm_crypto_operation_s Struct Reference . . . 168
6.98.1 Detailed Description . . . 168

6.98.2 Field Documentation . . . 168
6.98.2.1 aead . . . 168
6.98.2.2 cipher . . . 168
6.98.2.3 hash . . . 168
6.98.2.4 in_use . . . 168
6.98.2.5 key_deriv . . . 169
6.98.2.6 mac . . . 169
6.98.2.7 operation . . . 169
6.98.2.8 owner . . . 169
6.98.2.9 type . . . 169
6.99 tfm_crypto_pack_iovec Struct Reference . . . 169
6.99.1 Detailed Description . . . 170
6.99.2 Field Documentation . . . 170
6.99.2.1 "@6 . . . 170
6.99.2.2 ad_length . . . 170
6.99.2.3 aead_in . . . 170
6.99.2.4 alg . . . 170
6.99.2.5 capacity . . . 170
6.99.2.6 function_id . . . 171
6.99.2.7 key_id . . . 171
6.99.2.8 op_handle . . . 171
6.99.2.9 plaintext_length . . . 171
6.99.2.10 step . . . 171
6.99.2.11 value . . . 171
6.100 tfm_fwu_ctx_s Struct Reference . . . 171
6.100.1 Detailed Description . . . 171
6.100.2 Field Documentation . . . 171
6.100.2.1 component_state . . . 172
6.100.2.2 error . . . 172
6.100.2.3 in_use . . . 172
6.101 tfm_fwu_mcuboot_ctx_s Struct Reference . . . 172
6.101.1 Detailed Description . . . 172
6.101.2 Field Documentation . . . 172
6.101.2.1 fap . . . 172
6.101.2.2 loaded_size . . . 172
6.102 tfm_ns_ctx_mgr_t Struct Reference . . . 172
6.102.1 Detailed Description . . . 173
6.102.2 Field Documentation . . . 173
6.102.2.1 active_ns_ctx_index . . . 173
6.102.2.2 ns_ctx_data . . . 173
6.102.2.3 ns_ctx_limit . . . 173
6.103 tfm_ns_ctx_t Struct Reference . . . 173

6.103.1 Detailed Description . . . 174
6.103.2 Field Documentation . . . 174
6.103.2.1 gid . . . 174
6.103.2.2 nsid . . . 174
6.103.2.3 ref_cnt . . . 174
6.103.2.4 tid . . . 174
6.104 tfm_original_iovec_t Struct Reference . . . 174
6.104.1 Detailed Description . . . 175
6.104.2 Field Documentation . . . 175
6.104.2.1 invec . . . 175
6.104.2.2 invec_count . . . 175
6.104.2.3 outvec . . . 175
6.104.2.4 outvec_count . . . 175
6.105 tfm_pool_chunk_t Struct Reference . . . 176
6.105.1 Detailed Description . . . 176
6.105.2 Field Documentation . . . 176
6.105.2.1 data . . . 176
6.105.2.2 magic . . . 176
6.105.2.3 next . . . 176
6.106 tfm_pool_instance_t Struct Reference . . . 176
6.106.1 Detailed Description . . . 177
6.106.2 Field Documentation . . . 177
6.106.2.1 chunks . . . 177
6.106.2.2 chunksz . . . 177
6.106.2.3 next . . . 177
6.106.2.4 pool_sz . . . 177
6.107 tfm_state_context_t Struct Reference . . . 178
6.107.1 Detailed Description . . . 178
6.107.2 Field Documentation . . . 178
6.107.2.1 lr . . . 178
6.107.2.2 r0 . . . 178
6.107.2.3 r1 . . . 178
6.107.2.4 r12 . . . 178
6.107.2.5 r2 . . . 178
6.107.2.6 r3 . . . 178
6.107.2.7 ra . . . 179
6.107.2.8 xpsr . . . 179
6.108 thread_t Struct Reference . . . 179
6.108.1 Detailed Description . . . 179
6.108.2 Field Documentation . . . 179
6.108.2.1 flags . . . 179
6.108.2.2 next . . . 180

6.108.2.3 p_context_ctrl . . . 180
6.108.2.4 priority . . . 180
6.108.2.5 state . . . 180
6.109 u64_in_u32_regs_t Union Reference . . . 180
6.109.1 Detailed Description . . . 180
6.109.2 Field Documentation . . . 180
6.109.2.1 r0 . . . 180
6.109.2.2 r1 . . . 180
6.109.2.3 u32_regs . . . 181
6.109.2.4 u64_val . . . 181
6.110 vectors Struct Reference . . . 181
6.110.1 Detailed Description . . . 181
6.110.2 Field Documentation . . . 181
6.110.2.1 in_use . . . 181
6.110.2.2 in_vec . . . 181
6.110.2.3 out_len . . . 182
6.110.2.4 out_vec . . . 182

**7 File Documentation** **183**
7.1 docs/doxygen/mainpage.dox File Reference . . . 183
7.2 interface/dir_interface.dox File Reference . . . 183
7.3 interface/include/crypto_keys/tfm_builtin_key_ids.h File Reference . . . 183
7.3.1 Enumeration Type Documentation . . . 183
7.3.1.1 tfm_builtin_key_id_t . . . 183
7.4 platform/ext/target/infineon/pse84/epc2/spe/services/crypto/tfm_builtin_key_ids.h File Reference . . . 184
7.4.1 Macro Definition Documentation . . . 184
7.4.1.1 MBEDTLS_PSA_KEY_ID_BUILTIN_MIN . . . 185
7.4.1.2 TFM_BUILTIN_KEY_ID_HUK . . . 185
7.4.1.3 TFM_BUILTIN_KEY_ID_IAK . . . 185
7.4.1.4 TFM_BUILTIN_KEY_ID_IAK_PUB . . . 185
7.4.2 Enumeration Type Documentation . . . 185
7.4.2.1 tfm_builtin_key_id_t . . . 185
7.5 interface/include/multi_core/tfm_mailbox.h File Reference . . . 185
7.5.1 Macro Definition Documentation . . . 187
7.5.1.1 MAILBOX_ALIGN . . . 187
7.5.1.2 MAILBOX_CACHE_LINE_SIZE . . . 187
7.5.1.3 MAILBOX_CALLBACK_REG_ERROR . . . 187
7.5.1.4 MAILBOX_CHAN_BUSY . . . 187
7.5.1.5 MAILBOX_GENERIC_ERROR . . . 187
7.5.1.6 MAILBOX_INIT_ERROR . . . 187
7.5.1.7 MAILBOX_INVAL_PARAMS . . . 187
7.5.1.8 MAILBOX_NO_PEND_EVENT . . . 188

7.5.1.9 MAILBOX_NO_PERMS . . . 188
7.5.1.10 MAILBOX_PSA_CALL . . . 188
7.5.1.11 MAILBOX_PSA_CLOSE . . . 188
7.5.1.12 MAILBOX_PSA_CONNECT . . . 188
7.5.1.13 MAILBOX_PSA_FRAMEWORK_VERSION . . . 188
7.5.1.14 MAILBOX_PSA_VERSION . . . 188
7.5.1.15 MAILBOX_QUEUE_FULL . . . 188
7.5.1.16 MAILBOX_SUCCESS . . . 188
7.5.2 Typedef Documentation . . . 188
7.5.2.1 mailbox_queue_status_t . . . 188
7.5.3 Function Documentation . . . 189
7.5.3.1 __ALIGNED() . . . 189
7.5.4 Variable Documentation . . . 189
7.5.4.1 __ALIGNED . . . 189
7.5.4.2 pend_slots . . . 189
7.5.4.3 replied_slots . . . 189
7.6 interface/include/multi_core/tfm_multi_core_api.h File Reference . . . 189
7.6.1 Function Documentation . . . 190
7.6.1.1 tfm_ns_wait_for_s_cpu_ready() . . . 190
7.6.1.2 tfm_platform_ns_wait_for_s_cpu_ready() . . . 191
7.7 interface/include/multi_core/tfm_ns_mailbox.h File Reference . . . 191
7.8 platform/ext/target/infineon/common/interface/include/multi_core/tfm_ns_mailbox.h File Reference . . . 191
7.8.1 Macro Definition Documentation . . . 193
7.8.1.1 DCACHE_IS_ENABLED . . . 193
7.8.1.2 MAILBOX_CLEAN_CACHE . . . 193
7.8.1.3 MAILBOX_DISABLE_CACHE . . . 193
7.8.1.4 MAILBOX_ENABLE_CACHE . . . 193
7.8.1.5 MAILBOX_INVALIDATE_CACHE . . . 193
7.8.1.6 tfm_ns_mailbox_client_call . . . 193
7.8.1.7 tfm_ns_mailbox_os_spin_lock . . . 193
7.8.1.8 tfm_ns_mailbox_os_spin_unlock . . . 193
7.8.1.9 tfm_ns_mailbox_os_wait_reply . . . 193
7.8.1.10 tfm_ns_mailbox_os_wake_task_isr . . . 194
7.8.1.11 tfm_ns_mailbox_thread_runner . . . 194
7.8.2 Function Documentation . . . 194
7.8.2.1 ifx_mtb_mailbox_client_call() . . . 194
7.8.2.2 tfm_ns_mailbox_client_call() . . . 194
7.8.2.3 tfm_ns_mailbox_hal_enter_critical() . . . 195
7.8.2.4 tfm_ns_mailbox_hal_enter_critical_isr() . . . 195
7.8.2.5 tfm_ns_mailbox_hal_exit_critical() . . . 196
7.8.2.6 tfm_ns_mailbox_hal_exit_critical_isr() . . . 196
7.8.2.7 tfm_ns_mailbox_hal_init() . . . 197

7.8.2.8 tfm_ns_mailbox_hal_notify_peer() . . . 197
7.8.2.9 tfm_ns_mailbox_init() . . . 198
7.8.2.10 tfm_ns_multi_core_boot() . . . 198
7.9 interface/include/multi_core/tfm_ns_mailbox_test.h File Reference . . . 199
7.9.1 Function Documentation . . . 200
7.9.1.1 tfm_ns_mailbox_stats_avg_slot() . . . 200
7.9.1.2 tfm_ns_mailbox_tx_stats_init() . . . 200
7.9.1.3 tfm_ns_mailbox_tx_stats_reinit() . . . 200
7.9.1.4 tfm_ns_mailbox_tx_stats_update() . . . 200
7.10 interface/include/os_wrapper/common.h File Reference . . . 201
7.10.1 Macro Definition Documentation . . . 201
7.10.1.1 OS_WRAPPER_DEFAULT_STACK_SIZE . . . 201
7.10.1.2 OS_WRAPPER_ERROR . . . 201
7.10.1.3 OS_WRAPPER_SUCCESS . . . 202
7.10.1.4 OS_WRAPPER_WAIT_FOREVER . . . 202
7.11 interface/include/os_wrapper/kernel.h File Reference . . . 202
7.11.1 Function Documentation . . . 203
7.11.1.1 os_wrapper_is_kernel_running() . . . 203
7.12 interface/include/os_wrapper/mutex.h File Reference . . . 203
7.12.1 Function Documentation . . . 204
7.12.1.1 os_wrapper_mutex_acquire() . . . 205
7.12.1.2 os_wrapper_mutex_create() . . . 205
7.12.1.3 os_wrapper_mutex_delete() . . . 205
7.12.1.4 os_wrapper_mutex_release() . . . 206
7.13 interface/include/psa/client.h File Reference . . . 206
7.13.1 Macro Definition Documentation . . . 207
7.13.1.1 IOVEC_LEN . . . 207
7.13.1.2 PSA_CALL_TYPE_MAX . . . 207
7.13.1.3 PSA_CALL_TYPE_MIN . . . 207
7.13.1.4 PSA_FRAMEWORK_VERSION . . . 208
7.13.1.5 PSA_HANDLE_IS_VALID . . . 208
7.13.1.6 PSA_HANDLE_TO_ERROR . . . 208
7.13.1.7 PSA_IPC_CALL . . . 208
7.13.1.8 PSA_MAX_IOVEC . . . 208
7.13.1.9 PSA_NULL_HANDLE . . . 208
7.13.1.10 PSA_VERSION_NONE . . . 208
7.13.2 Typedef Documentation . . . 208
7.13.2.1 psa_handle_t . . . 208
7.13.2.2 psa_invec . . . 209
7.13.2.3 psa_outvec . . . 209
7.13.3 Function Documentation . . . 209
7.13.3.1 psa_call() . . . 209

7.13.3.2 psa_close() . . . 211
7.13.3.3 psa_connect() . . . 212
7.13.3.4 psa_framework_version() . . . 213
7.13.3.5 psa_version() . . . 214
7.14 interface/include/psa/error.h File Reference . . . 214
7.14.1 Detailed Description . . . 216
7.14.2 Macro Definition Documentation . . . 216
7.14.2.1 PSA_ERROR_ALREADY_EXISTS . . . 216
7.14.2.2 PSA_ERROR_BAD_STATE . . . 216
7.14.2.3 PSA_ERROR_BUFFER_TOO_SMALL . . . 216
7.14.2.4 PSA_ERROR_COMMUNICATION_FAILURE . . . 216
7.14.2.5 PSA_ERROR_CONNECTION_BUSY . . . 216
7.14.2.6 PSA_ERROR_CONNECTION_REFUSED . . . 216
7.14.2.7 PSA_ERROR_CORRUPTION_DETECTED . . . 216
7.14.2.8 PSA_ERROR_DEPENDENCY_NEEDED . . . 216
7.14.2.9 PSA_ERROR_DOES_NOT_EXIST . . . 216
7.14.2.10 PSA_ERROR_FLASH_ABUSE . . . 217
7.14.2.11 PSA_ERROR_GENERIC_ERROR . . . 217
7.14.2.12 PSA_ERROR_HARDWARE_FAILURE . . . 217
7.14.2.13 PSA_ERROR_INSUFFICIENT_DATA . . . 217
7.14.2.14 PSA_ERROR_INSUFFICIENT_MEMORY . . . 217
7.14.2.15 PSA_ERROR_INSUFFICIENT_POWER . . . 217
7.14.2.16 PSA_ERROR_INSUFFICIENT_STORAGE . . . 217
7.14.2.17 PSA_ERROR_INVALID_ARGUMENT . . . 217
7.14.2.18 PSA_ERROR_INVALID_HANDLE . . . 217
7.14.2.19 PSA_ERROR_INVALID_SIGNATURE . . . 217
7.14.2.20 PSA_ERROR_NOT_PERMITTED . . . 218
7.14.2.21 PSA_ERROR_NOT_SUPPORTED . . . 218
7.14.2.22 PSA_ERROR_PROGRAMMER_ERROR . . . 218
7.14.2.23 PSA_ERROR_SERVICE_FAILURE . . . 218
7.14.2.24 PSA_ERROR_STORAGE_FAILURE . . . 218
7.14.2.25 PSA_SUCCESS . . . 218
7.14.3 Typedef Documentation . . . 218
7.14.3.1 psa_status_t . . . 218
7.15 interface/include/psa/internal_trusted_storage.h File Reference . . . 218
7.15.1 Macro Definition Documentation . . . 219
7.15.1.1 PSA_ITS_API_VERSION_MAJOR . . . 220
7.15.1.2 PSA_ITS_API_VERSION_MINOR . . . 220
7.15.2 Function Documentation . . . 220
7.15.2.1 psa_its_get() . . . 220
7.15.2.2 psa_its_get_info() . . . 221
7.15.2.3 psa_its_remove() . . . 222

7.15.2.4 psa_its_set() . . . 223
7.16 interface/include/psa/lifecycle.h File Reference . . . 224
7.16.1 Macro Definition Documentation . . . 225
7.16.1.1 PSA_LIFECYCLE_ASSEMBLY_AND_TEST . . . 225
7.16.1.2 PSA_LIFECYCLE_DECOMMISSIONED . . . 225
7.16.1.3 PSA_LIFECYCLE_IMP_STATE_MASK . . . 225
7.16.1.4 PSA_LIFECYCLE_NON_PSA_ROT_DEBUG . . . 225
7.16.1.5 PSA_LIFECYCLE_PSA_ROT_PROVISIONING . . . 225
7.16.1.6 PSA_LIFECYCLE_PSA_STATE_MASK . . . 225
7.16.1.7 PSA_LIFECYCLE_RECOVERABLE_PSA_ROT_DEBUG . . . 225
7.16.1.8 PSA_LIFECYCLE_SECURED . . . 225
7.16.1.9 PSA_LIFECYCLE_UNKNOWN . . . 225
7.16.2 Function Documentation . . . 225
7.16.2.1 psa_rot_lifecycle_state() . . . 226
7.17 interface/include/psa/protected_storage.h File Reference . . . 226
7.17.1 Macro Definition Documentation . . . 227
7.17.1.1 PSA_PS_API_VERSION_MAJOR . . . 227
7.17.1.2 PSA_PS_API_VERSION_MINOR . . . 227
7.17.2 Function Documentation . . . 227
7.17.2.1 psa_ps_create() . . . 227
7.17.2.2 psa_ps_get() . . . 228
7.17.2.3 psa_ps_get_info() . . . 229
7.17.2.4 psa_ps_get_support() . . . 230
7.17.2.5 psa_ps_remove() . . . 230
7.17.2.6 psa_ps_set() . . . 231
7.17.2.7 psa_ps_set_extended() . . . 232
7.18 interface/include/psa/service.h File Reference . . . 233
7.18.1 Macro Definition Documentation . . . 235
7.18.1.1 PSA_BLOCK . . . 235
7.18.1.2 PSA_DOORBELL . . . 235
7.18.1.3 PSA_FLIH_NO_SIGNAL . . . 235
7.18.1.4 PSA_FLIH_SIGNAL . . . 235
7.18.1.5 PSA_IPC_CONNECT . . . 235
7.18.1.6 PSA_IPC_DISCONNECT . . . 235
7.18.1.7 PSA_POLL . . . 235
7.18.1.8 PSA_WAIT_ANY . . . 236
7.18.2 Typedef Documentation . . . 236
7.18.2.1 psa_flih_result_t . . . 236
7.18.2.2 psa_irq_status_t . . . 236
7.18.2.3 psa_msg_t . . . 236
7.18.2.4 psa_signal_t . . . 236
7.18.3 Function Documentation . . . 236

7.18.3.1 psa_clear() . . . 236
7.18.3.2 psa_eoi() . . . 236
7.18.3.3 psa_get() . . . 237
7.18.3.4 psa_irq_disable() . . . 238
7.18.3.5 psa_irq_enable() . . . 238
7.18.3.6 psa_notify() . . . 238
7.18.3.7 psa_panic() . . . 239
7.18.3.8 psa_read() . . . 240
7.18.3.9 psa_reply() . . . 241
7.18.3.10 psa_reset_signal() . . . 242
7.18.3.11 psa_set_rhandle() . . . 242
7.18.3.12 psa_skip() . . . 242
7.18.3.13 psa_wait() . . . 243
7.18.3.14 psa_write() . . . 244
7.19 interface/include/psa/storage_common.h File Reference . . . 245
7.19.1 Macro Definition Documentation . . . 246
7.19.1.1 PSA_ERROR_DATA_CORRUPT . . . 246
7.19.1.2 PSA_ERROR_INVALID_SIGNATURE . . . 246
7.19.1.3 PSA_STORAGE_FLAG_NO_CONFIDENTIALITY . . . 246
7.19.1.4 PSA_STORAGE_FLAG_NO_REPLAY_PROTECTION . . . 246
7.19.1.5 PSA_STORAGE_FLAG_NONE . . . 247
7.19.1.6 PSA_STORAGE_FLAG_WRITE_ONCE . . . 247
7.19.1.7 PSA_STORAGE_SUPPORT_SET_EXTENDED . . . 247
7.19.2 Typedef Documentation . . . 247
7.19.2.1 psa_storage_create_flags_t . . . 247
7.19.2.2 psa_storage_uid_t . . . 247
7.20 interface/include/psa/update.h File Reference . . . 247
7.20.1 Macro Definition Documentation . . . 249
7.20.1.1 PSA_ERROR_DEPENDENCY_NEEDED . . . 250
7.20.1.2 PSA_ERROR_FLASH_ABUSE . . . 250
7.20.1.3 PSA_ERROR_INSUFFICIENT_POWER . . . 250
7.20.1.4 PSA_FWU_API_VERSION_MAJOR . . . 250
7.20.1.5 PSA_FWU_API_VERSION_MINOR . . . 250
7.20.1.6 PSA_FWU_CANDIDATE . . . 250
7.20.1.7 PSA_FWU_FAILED . . . 250
7.20.1.8 PSA_FWU_FLAG_ENCRYPTION . . . 250
7.20.1.9 PSA_FWU_FLAG_VOLATILE_STAGING . . . 250
7.20.1.10 PSA_FWU_MAX_WRITE_SIZE . . . 251
7.20.1.11 PSA_FWU_READY . . . 251
7.20.1.12 PSA_FWU_REJECTED . . . 251
7.20.1.13 PSA_FWU_STAGED . . . 251
7.20.1.14 PSA_FWU_TRIAL . . . 251

7.20.1.15 PSA_FWU_UPDATED . . . 251
7.20.1.16 PSA_FWU_WRITING . . . 251
7.20.1.17 PSA_SUCCESS_REBOOT . . . 251
7.20.1.18 PSA_SUCCESS_RESTART . . . 252
7.20.2 Typedef Documentation . . . 252
7.20.2.1 psa_fwu_component_info_t . . . 252
7.20.2.2 psa_fwu_component_t . . . 252
7.20.2.3 psa_fwu_image_version_t . . . 252
7.20.3 Function Documentation . . . 252
7.20.3.1 psa_fwu_accept() . . . 252
7.20.3.2 psa_fwu_cancel() . . . 253
7.20.3.3 psa_fwu_clean() . . . 254
7.20.3.4 psa_fwu_finish() . . . 254
7.20.3.5 psa_fwu_install() . . . 255
7.20.3.6 psa_fwu_query() . . . 255
7.20.3.7 psa_fwu_reject() . . . 256
7.20.3.8 psa_fwu_request_reboot() . . . 256
7.20.3.9 psa_fwu_start() . . . 257
7.20.3.10 psa_fwu_write() . . . 257
7.21 interface/include/tfm_attest_defs.h File Reference . . . 258
7.21.1 Macro Definition Documentation . . . 258
7.21.1.1 TFM_ATTEST_GET_TOKEN . . . 259
7.21.1.2 TFM_ATTEST_GET_TOKEN_SIZE . . . 259
7.22 interface/include/tfm_attest_iat_defs.h File Reference . . . 259
7.22.1 Macro Definition Documentation . . . 260
7.22.1.1 IAT_SW_COMPONENT_MEASUREMENT_DESC . . . 260
7.22.1.2 IAT_SW_COMPONENT_MEASUREMENT_TYPE . . . 260
7.22.1.3 IAT_SW_COMPONENT_MEASUREMENT_VALUE . . . 260
7.22.1.4 IAT_SW_COMPONENT_SIGNER_ID . . . 260
7.22.1.5 IAT_SW_COMPONENT_VERSION . . . 260
7.23 interface/include/tfm_crypto_defs.h File Reference . . . 260
7.23.1 Macro Definition Documentation . . . 263
7.23.1.1 AEAD_FUNCS . . . 263
7.23.1.2 ASYM_ENCRYPT_FUNCS . . . 263
7.23.1.3 ASYM_SIGN_FUNCS . . . 263
7.23.1.4 BASE__VALUE . . . 263
7.23.1.5 CIPHER_FUNCS . . . 263
7.23.1.6 HASH_FUNCS . . . 263
7.23.1.7 KEY_DERIVATION_FUNCS . . . 264
7.23.1.8 KEY_MANAGEMENT_FUNCS . . . 264
7.23.1.9 MAC_FUNCS . . . 264
7.23.1.10 RANDOM_FUNCS . . . 264

7.23.1.11 TFM_CRYPTO_GET_GROUP_ID . . . 264
7.23.1.12 TFM_CRYPTO_MAX_NONCE_LENGTH . . . 265
7.23.1.13 X . . . 265
7.23.2 Enumeration Type Documentation . . . 265
7.23.2.1 tfm_crypto_func_sid_t . . . 265
7.23.2.2 tfm_crypto_group_id_t . . . 267
7.24 interface/include/tfm_fwu_defs.h File Reference . . . 268
7.24.1 Macro Definition Documentation . . . 268
7.24.1.1 TFM_FWU_ACCEPT . . . 268
7.24.1.2 TFM_FWU_CANCEL . . . 268
7.24.1.3 TFM_FWU_CLEAN . . . 269
7.24.1.4 TFM_FWU_FINISH . . . 269
7.24.1.5 TFM_FWU_INSTALL . . . 269
7.24.1.6 TFM_FWU_QUERY . . . 269
7.24.1.7 TFM_FWU_REJECT . . . 269
7.24.1.8 TFM_FWU_REQUEST_REBOOT . . . 269
7.24.1.9 TFM_FWU_START . . . 269
7.24.1.10 TFM_FWU_WRITE . . . 269
7.25 interface/include/tfm_its_defs.h File Reference . . . 269
7.25.1 Macro Definition Documentation . . . 270
7.25.1.1 TFM_ITS_GET . . . 270
7.25.1.2 TFM_ITS_GET_INFO . . . 270
7.25.1.3 TFM_ITS_INVALID_UID . . . 270
7.25.1.4 TFM_ITS_REMOVE . . . 270
7.25.1.5 TFM_ITS_SET . . . 270
7.26 interface/include/tfm_ns_client_ext.h File Reference . . . 270
7.26.1 Macro Definition Documentation . . . 272
7.26.1.1 TFM_NS_CLIENT_ERR_INVALID_ACCESS . . . 272
7.26.1.2 TFM_NS_CLIENT_ERR_INVALID_NSID . . . 272
7.26.1.3 TFM_NS_CLIENT_ERR_INVALID_TOKEN . . . 272
7.26.1.4 TFM_NS_CLIENT_ERR_SUCCESS . . . 272
7.26.1.5 TFM_NS_CLIENT_INVALID_TOKEN . . . 272
7.26.2 Function Documentation . . . 272
7.26.2.1 tfm_nsce_acquire_ctx() . . . 272
7.26.2.2 tfm_nsce_init() . . . 273
7.26.2.3 tfm_nsce_load_ctx() . . . 273
7.26.2.4 tfm_nsce_release_ctx() . . . 274
7.26.2.5 tfm_nsce_save_ctx() . . . 275
7.27 interface/include/tfm_ns_interface.h File Reference . . . 275
7.27.1 Typedef Documentation . . . 277
7.27.1.1 veneer_fn . . . 277
7.27.2 Function Documentation . . . 277

7.27.2.1 tfm_ns_interface_dispatch() . . . 277
7.27.2.2 tfm_ns_interface_init() . . . 278
7.28 interface/include/tfm_platform_api.h File Reference . . . 278
7.28.1 Macro Definition Documentation . . . 280
7.28.1.1 TFM_PLATFORM_API_ID_IOCTL . . . 280
7.28.1.2 TFM_PLATFORM_API_ID_NV_INCREMENT . . . 280
7.28.1.3 TFM_PLATFORM_API_ID_NV_READ . . . 280
7.28.1.4 TFM_PLATFORM_API_ID_SYSTEM_RESET . . . 280
7.28.1.5 TFM_PLATFORM_API_VERSION_MAJOR . . . 280
7.28.1.6 TFM_PLATFORM_API_VERSION_MINOR . . . 280
7.28.2 Typedef Documentation . . . 280
7.28.2.1 tfm_platform_ioctl_req_t . . . 281
7.28.3 Enumeration Type Documentation . . . 281
7.28.3.1 tfm_platform_err_t . . . 281
7.28.4 Function Documentation . . . 281
7.28.4.1 tfm_platform_ioctl() . . . 281
7.28.4.2 tfm_platform_nv_counter_increment() . . . 282
7.28.4.3 tfm_platform_nv_counter_read() . . . 282
7.28.4.4 tfm_platform_system_reset() . . . 283
7.29 interface/include/tfm_ps_defs.h File Reference . . . 284
7.29.1 Macro Definition Documentation . . . 284
7.29.1.1 TFM_PS_GET . . . 284
7.29.1.2 TFM_PS_GET_INFO . . . 284
7.29.1.3 TFM_PS_GET_SUPPORT . . . 285
7.29.1.4 TFM_PS_INVALID_UID . . . 285
7.29.1.5 TFM_PS_REMOVE . . . 285
7.29.1.6 TFM_PS_SET . . . 285
7.30 interface/include/tfm_psa_call_pack.h File Reference . . . 285
7.30.1 Macro Definition Documentation . . . 286
7.30.1.1 IN_LEN_MASK . . . 286
7.30.1.2 IN_LEN_OFFSET . . . 286
7.30.1.3 NS_INVEC_BIT . . . 286
7.30.1.4 NS_INVEC_OFFSET . . . 286
7.30.1.5 NS_OUTVEC_BIT . . . 287
7.30.1.6 NS_OUTVEC_OFFSET . . . 287
7.30.1.7 NS_VEC_DESC_BIT . . . 287
7.30.1.8 OUT_LEN_MASK . . . 287
7.30.1.9 OUT_LEN_OFFSET . . . 287
7.30.1.10 PARAM_HAS_IOVEC . . . 287
7.30.1.11 PARAM_IS_NS_INVEC . . . 287
7.30.1.12 PARAM_IS_NS_OUTVEC . . . 287
7.30.1.13 PARAM_IS_NS_VEC . . . 287

7.30.1.14 PARAM_PACK . . . 288
7.30.1.15 PARAM_SET_NS_INVEC . . . 288
7.30.1.16 PARAM_SET_NS_OUTVEC . . . 288
7.30.1.17 PARAM_SET_NS_VEC . . . 288
7.30.1.18 PARAM_UNPACK_IN_LEN . . . 288
7.30.1.19 PARAM_UNPACK_OUT_LEN . . . 288
7.30.1.20 PARAM_UNPACK_TYPE . . . 288
7.30.1.21 TYPE_MASK . . . 288
7.30.2 Function Documentation . . . 289
7.30.2.1 tfm_psa_call_pack() . . . 289
7.31 interface/include/tfm_veneers.h File Reference . . . 289
7.31.1 Function Documentation . . . 290
7.31.1.1 tfm_psa_call_veneer() . . . 290
7.31.1.2 tfm_psa_close_veneer() . . . 291
7.31.1.3 tfm_psa_connect_veneer() . . . 291
7.31.1.4 tfm_psa_framework_version_veneer() . . . 291
7.31.1.5 tfm_psa_version_veneer() . . . 292
7.32 interface/src/multi_core/tfm_multi_core_ns_api.c File Reference . . . 293
7.32.1 Function Documentation . . . 294
7.32.1.1 tfm_ns_wait_for_s_cpu_ready() . . . 294
7.33 interface/src/multi_core/tfm_multi_core_psa_ns_api.c File Reference . . . 294
7.33.1 Macro Definition Documentation . . . 295
7.33.1.1 NON_SECURE_CLIENT_ID . . . 295
7.33.1.2 PSA_INTER_CORE_COMM_ERR . . . 295
7.33.2 Function Documentation . . . 296
7.33.2.1 psa_call() . . . 296
7.33.2.2 psa_close() . . . 297
7.33.2.3 psa_connect() . . . 297
7.33.2.4 psa_framework_version() . . . 298
7.33.2.5 psa_version() . . . 298
7.34 interface/src/multi_core/tfm_ns_mailbox.c File Reference . . . 299
7.34.1 Function Documentation . . . 299
7.34.1.1 tfm_ns_mailbox_client_call() . . . 299
7.34.1.2 tfm_ns_mailbox_init() . . . 300
7.35 interface/src/multi_core/tfm_ns_mailbox_thread.c File Reference . . . 300
7.35.1 Macro Definition Documentation . . . 301
7.35.1.1 NOT_WOKEN . . . 301
7.35.1.2 WOKEN_UP . . . 301
7.35.2 Function Documentation . . . 301
7.35.2.1 tfm_ns_mailbox_client_call() . . . 302
7.35.2.2 tfm_ns_mailbox_init() . . . 302
7.35.2.3 tfm_ns_mailbox_thread_runner() . . . 303

7.35.2.4 tfm_ns_mailbox_wake_reply_owner_isr() . . . 303
7.36 interface/src/os_wrapper/tfm_ns_interface_bare_metal.c File Reference . . . 303
7.36.1 Function Documentation . . . 303
7.36.1.1 tfm_ns_interface_dispatch() . . . 304
7.36.1.2 tfm_ns_interface_init() . . . 304
7.37 interface/src/os_wrapper/tfm_ns_interface_rtos.c File Reference . . . 304
7.37.1 Function Documentation . . . 305
7.37.1.1 tfm_ns_interface_dispatch() . . . 305
7.37.1.2 tfm_ns_interface_init() . . . 306
7.38 interface/src/tfm_attest_api.c File Reference . . . 307
7.38.1 Function Documentation . . . 307
7.38.1.1 psa_initial_attest_get_token() . . . 307
7.38.1.2 psa_initial_attest_get_token_size() . . . 308
7.39 interface/src/tfm_crypto_api.c File Reference . . . 308
7.39.1 Macro Definition Documentation . . . 311
7.39.1.1 API_DISPATCH . . . 311
7.39.1.2 API_DISPATCH_NO_OUTVEC . . . 311
7.39.1.3 TFM_CRYPTO_API . . . 312
7.39.2 Function Documentation . . . 312
7.39.2.1 psa_aead_abort() . . . 312
7.39.2.2 psa_aead_decrypt() . . . 312
7.39.2.3 psa_aead_decrypt_setup() . . . 313
7.39.2.4 psa_aead_encrypt() . . . 313
7.39.2.5 psa_aead_encrypt_setup() . . . 313
7.39.2.6 psa_aead_finish() . . . 313
7.39.2.7 psa_aead_generate_nonce() . . . 314
7.39.2.8 psa_aead_set_lengths() . . . 314
7.39.2.9 psa_aead_set_nonce() . . . 314
7.39.2.10 psa_aead_update() . . . 314
7.39.2.11 psa_aead_update_ad() . . . 315
7.39.2.12 psa_aead_verify() . . . 315
7.39.2.13 psa_asymmetric_decrypt() . . . 316
7.39.2.14 psa_asymmetric_encrypt() . . . 316
7.39.2.15 psa_cipher_abort() . . . 316
7.39.2.16 psa_cipher_decrypt() . . . 317
7.39.2.17 psa_cipher_decrypt_setup() . . . 317
7.39.2.18 psa_cipher_encrypt() . . . 317
7.39.2.19 psa_cipher_encrypt_setup() . . . 317
7.39.2.20 psa_cipher_finish() . . . 317
7.39.2.21 psa_cipher_generate_iv() . . . 318
7.39.2.22 psa_cipher_set_iv() . . . 318
7.39.2.23 psa_cipher_update() . . . 318

7.39.2.24 psa_close_key() . . . 318
7.39.2.25 psa_copy_key() . . . 318
7.39.2.26 psa_crypto_init() . . . 318
7.39.2.27 psa_destroy_key() . . . 318
7.39.2.28 psa_export_key() . . . 319
7.39.2.29 psa_export_public_key() . . . 319
7.39.2.30 psa_generate_key() . . . 319
7.39.2.31 psa_generate_random() . . . 319
7.39.2.32 psa_get_key_attributes() . . . 319
7.39.2.33 psa_hash_abort() . . . 319
7.39.2.34 psa_hash_clone() . . . 320
7.39.2.35 psa_hash_compare() . . . 320
7.39.2.36 psa_hash_compute() . . . 320
7.39.2.37 psa_hash_finish() . . . 320
7.39.2.38 psa_hash_setup() . . . 320
7.39.2.39 psa_hash_update() . . . 320
7.39.2.40 psa_hash_verify() . . . 321
7.39.2.41 psa_import_key() . . . 321
7.39.2.42 psa_key_derivation_abort() . . . 321
7.39.2.43 psa_key_derivation_get_capacity() . . . 321
7.39.2.44 psa_key_derivation_input_bytes() . . . 321
7.39.2.45 psa_key_derivation_input_integer() . . . 322
7.39.2.46 psa_key_derivation_input_key() . . . 322
7.39.2.47 psa_key_derivation_key_agreement() . . . 322
7.39.2.48 psa_key_derivation_output_bytes() . . . 322
7.39.2.49 psa_key_derivation_output_key() . . . 322
7.39.2.50 psa_key_derivation_set_capacity() . . . 323
7.39.2.51 psa_key_derivation_setup() . . . 323
7.39.2.52 psa_key_derivation_verify_bytes() . . . 323
7.39.2.53 psa_key_derivation_verify_key() . . . 323
7.39.2.54 psa_mac_abort() . . . 323
7.39.2.55 psa_mac_compute() . . . 323
7.39.2.56 psa_mac_sign_finish() . . . 324
7.39.2.57 psa_mac_sign_setup() . . . 324
7.39.2.58 psa_mac_update() . . . 324
7.39.2.59 psa_mac_verify() . . . 324
7.39.2.60 psa_mac_verify_finish() . . . 324
7.39.2.61 psa_mac_verify_setup() . . . 324
7.39.2.62 psa_open_key() . . . 325
7.39.2.63 psa_purge_key() . . . 325
7.39.2.64 psa_raw_key_agreement() . . . 325
7.39.2.65 psa_reset_key_attributes() . . . 325

7.39.2.66 psa_sign_hash() . . . 325
7.39.2.67 psa_sign_message() . . . 326
7.39.2.68 psa_verify_hash() . . . 326
7.39.2.69 psa_verify_message() . . . 326
7.40 interface/src/tfm_fwu_api.c File Reference . . . 326
7.40.1 Function Documentation . . . 327
7.40.1.1 psa_fwu_accept() . . . 327
7.40.1.2 psa_fwu_cancel() . . . 328
7.40.1.3 psa_fwu_clean() . . . 328
7.40.1.4 psa_fwu_finish() . . . 329
7.40.1.5 psa_fwu_install() . . . 329
7.40.1.6 psa_fwu_query() . . . 330
7.40.1.7 psa_fwu_reject() . . . 330
7.40.1.8 psa_fwu_request_reboot() . . . 331
7.40.1.9 psa_fwu_start() . . . 331
7.40.1.10 psa_fwu_write() . . . 332
7.41 interface/src/tfm_its_api.c File Reference . . . 332
7.41.1 Function Documentation . . . 333
7.41.1.1 psa_its_get() . . . 333
7.41.1.2 psa_its_get_info() . . . 334
7.41.1.3 psa_its_remove() . . . 335
7.41.1.4 psa_its_set() . . . 336
7.42 interface/src/tfm_platform_api.c File Reference . . . 337
7.42.1 Function Documentation . . . 338
7.42.1.1 tfm_platform_ioctl() . . . 338
7.42.1.2 tfm_platform_nv_counter_increment() . . . 338
7.42.1.3 tfm_platform_nv_counter_read() . . . 339
7.42.1.4 tfm_platform_system_reset() . . . 340
7.43 interface/src/tfm_ps_api.c File Reference . . . 340
7.43.1 Function Documentation . . . 341
7.43.1.1 psa_ps_create() . . . 341
7.43.1.2 psa_ps_get() . . . 342
7.43.1.3 psa_ps_get_info() . . . 343
7.43.1.4 psa_ps_get_support() . . . 344
7.43.1.5 psa_ps_remove() . . . 344
7.43.1.6 psa_ps_set() . . . 345
7.43.1.7 psa_ps_set_extended() . . . 346
7.44 interface/src/tfm_psa_call.c File Reference . . . 347
7.44.1 Function Documentation . . . 348
7.44.1.1 psa_call() . . . 348
7.45 interface/src/tfm_tz_psa_ns_api.c File Reference . . . 349
7.45.1 Function Documentation . . . 350

7.45.1.1 psa_call() . . . 351
7.45.1.2 psa_close() . . . 352
7.45.1.3 psa_connect() . . . 353
7.45.1.4 psa_framework_version() . . . 354
7.45.1.5 psa_version() . . . 354
7.46 platform/ext/target/infineon/common/device/include/device_definition.h File Reference . . . 355
7.47 platform/ext/target/infineon/common/device/include/ifx_startup.h File Reference . . . 356
7.47.1 Macro Definition Documentation . . . 357
7.47.1.1 DEFAULT_IRQ_HANDLER . . . 357
7.47.1.2 IFX_VECTOR_TABLE_ATTRIBUTE . . . 358
7.47.1.3 IS_POWER_OF_TWO . . . 358
7.47.2 Typedef Documentation . . . 358
7.47.2.1 VECTOR_TABLE_Type . . . 358
7.47.3 Variable Documentation . . . 358
7.47.3.1 __vector_table . . . 358
7.48 platform/ext/target/infineon/common/device/src/device_definition.c File Reference . . . 358
7.48.1 Detailed Description . . . 359
7.49 platform/ext/target/infineon/common/drivers/assets/ifx_assets_rram.c File Reference . . . 359
7.49.1 Function Documentation . . . 359
7.49.1.1 ifx_assets_rram_read_asset() . . . 359
7.49.1.2 ifx_assets_rram_read_block() . . . 360
7.50 platform/ext/target/infineon/common/drivers/assets/ifx_assets_rram.h File Reference . . . 360
7.50.1 Macro Definition Documentation . . . 361
7.50.1.1 IFX_ASSETS_RRAM_CHECKSUM_SIZE . . . 361
7.50.2 Function Documentation . . . 362
7.50.2.1 ifx_assets_rram_read_asset() . . . 362
7.50.2.2 ifx_assets_rram_read_block() . . . 362
7.51 platform/ext/target/infineon/common/drivers/flash/flash/ifx_driver_flash.c File Reference . . . 363
7.51.1 Macro Definition Documentation . . . 363
7.51.1.1 IFX_DRIVER_FLASH_EVENT . . . 363
7.51.1.2 IFX_DRIVER_FLASH_FUNCTION_ARGUMENTS . . . 363
7.51.1.3 IFX_DRIVER_FLASH_FUNCTION_PARAMETERS . . . 364
7.51.1.4 IFX_DRIVER_FLASH_INSTANCE . . . 364
7.51.1.5 IFX_DRIVER_FLASH_VERSION . . . 364
7.51.1.6 IFX_FDF_ARGS . . . 364
7.51.1.7 IFX_FDF_PARAMS . . . 364
7.51.1.8 IFX_UNUSED_FLASH_DRIVER_INSTANCE . . . 364
7.51.2 Variable Documentation . . . 364
7.51.2.1 ifx_driver_flash . . . 364
7.52 platform/ext/target/infineon/common/drivers/flash/flash/ifx_driver_flash.h File Reference . . . 365
7.52.1 Macro Definition Documentation . . . 366
7.52.1.1 IFX_DRIVER_FLASH_CREATE_INSTANCE . . . 366

7.52.1.2 IFX_DRIVER_FLASH_ERASE_VALUE . . . 366
7.52.1.3 IFX_DRIVER_FLASH_PROGRAM_UNIT . . . 366
7.52.1.4 IFX_DRIVER_FLASH_SECTOR_SIZE . . . 366
7.52.2 Variable Documentation . . . 366
7.52.2.1 ifx_driver_flash . . . 366
7.53 platform/ext/target/infineon/common/drivers/flash/ifx_driver_private.c File Reference . . . 366
7.53.1 Function Documentation . . . 367
7.53.1.1 ifx_flash_driver_initialize() . . . 367
7.53.1.2 ifx_flash_driver_validate_region() . . . 368
7.54 platform/ext/target/infineon/common/drivers/flash/ifx_driver_private.h File Reference . . . 368
7.54.1 Function Documentation . . . 369
7.54.1.1 ifx_flash_driver_initialize() . . . 369
7.54.1.2 ifx_flash_driver_validate_region() . . . 370
7.55 platform/ext/target/infineon/common/drivers/flash/ifx_flash_driver_api.h File Reference . . . 370
7.55.1 Macro Definition Documentation . . . 371
7.55.1.1 IFX_CREATE_CMSIS_FLASH_DRIVER . . . 372
7.55.2 Typedef Documentation . . . 372
7.55.2.1 ifx_flash_driver_event_t . . . 372
7.55.2.2 ifx_flash_driver_instance_t . . . 372
7.55.2.3 ifx_flash_driver_t . . . 372
7.55.3 Variable Documentation . . . 372
7.55.3.1 ifx_flash_driver_handler_t . . . 372
7.56 platform/ext/target/infineon/common/drivers/flash/rram/ifx_driver_rram.c File Reference . . . 372
7.56.1 Macro Definition Documentation . . . 373
7.56.1.1 IFX_DRIVER_RRAM_VERSION . . . 373
7.56.2 Function Documentation . . . 373
7.56.2.1 TFM_COVERITY_DEVIATE_LINE() . . . 373
7.56.3 Variable Documentation . . . 374
7.56.3.1 ifx_driver_rram . . . 374
7.57 platform/ext/target/infineon/common/drivers/flash/rram/ifx_driver_rram.h File Reference . . . 375
7.57.1 Macro Definition Documentation . . . 376
7.57.1.1 IFX_DRIVER_RRAM_CREATE_INSTANCE . . . 376
7.57.1.2 IFX_DRIVER_RRAM_ERASE_VALUE . . . 376
7.57.1.3 IFX_DRIVER_RRAM_PC_LOCK_RETRIES . . . 376
7.57.1.4 IFX_DRIVER_RRAM_PROGRAM_UNIT . . . 376
7.57.2 Typedef Documentation . . . 376
7.57.2.1 ifx_driver_rram_obj_t . . . 376
7.57.3 Variable Documentation . . . 377
7.57.3.1 ifx_driver_rram . . . 377
7.58 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif.c File Reference . . . 377
7.58.1 Macro Definition Documentation . . . 378
7.58.1.1 IFX_DRIVER_SMIF_VERSION . . . 378

7.58.1.2 IFX_SMIF_SHORT_POLLING_RETRIES . . . 378
7.58.2 Function Documentation . . . 378
7.58.2.1 ifx_driver_smif_erase_chip() . . . 378
7.58.2.2 ifx_driver_smif_erase_sector() . . . 379
7.58.2.3 ifx_driver_smif_get_capabilities() . . . 379
7.58.2.4 ifx_driver_smif_get_info() . . . 379
7.58.2.5 ifx_driver_smif_get_status() . . . 380
7.58.2.6 ifx_driver_smif_get_version() . . . 380
7.58.2.7 ifx_driver_smif_initialize() . . . 380
7.58.2.8 ifx_driver_smif_poll_block_busy() . . . 381
7.58.2.9 ifx_driver_smif_power_control() . . . 381
7.58.2.10 ifx_driver_smif_set_mode() . . . 381
7.58.2.11 ifx_driver_smif_uninitialize() . . . 382
7.58.2.12 ifx_driver_smif_validate_region() . . . 382
7.59 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif.h File Reference . . . 383
7.59.1 Macro Definition Documentation . . . 384
7.59.1.1 IFX_DRIVER_SMIF_MMIO_INSTANCE . . . 384
7.59.1.2 IFX_DRIVER_SMIF_XIP_INSTANCE . . . 385
7.59.2 Typedef Documentation . . . 385
7.59.2.1 ifx_driver_smif_mem_t . . . 385
7.59.2.2 ifx_driver_smif_obj_t . . . 385
7.59.3 Variable Documentation . . . 385
7.59.3.1 ifx_driver_smif_mmio . . . 385
7.59.3.2 ifx_driver_smif_xip . . . 385
7.60 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_mmio.c File Reference . . . 385
7.60.1 Variable Documentation . . . 386
7.60.1.1 ifx_driver_smif_mmio . . . 386
7.61 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_private.h File Reference . . . 386
7.61.1 Macro Definition Documentation . . . 388
7.61.1.1 IFX_DRIVER_SMIF_DATA_WIDTH . . . 388
7.61.2 Function Documentation . . . 388
7.61.2.1 ifx_driver_smif_erase_chip() . . . 388
7.61.2.2 ifx_driver_smif_erase_sector() . . . 388
7.61.2.3 ifx_driver_smif_get_capabilities() . . . 389
7.61.2.4 ifx_driver_smif_get_info() . . . 389
7.61.2.5 ifx_driver_smif_get_status() . . . 390
7.61.2.6 ifx_driver_smif_get_version() . . . 390
7.61.2.7 ifx_driver_smif_initialize() . . . 390
7.61.2.8 ifx_driver_smif_power_control() . . . 391
7.61.2.9 ifx_driver_smif_set_mode() . . . 391
7.61.2.10 ifx_driver_smif_uninitialize() . . . 391
7.61.2.11 ifx_driver_smif_validate_region() . . . 392

7.62 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_xip.c File Reference . . . . . 392
7.62.1 Variable Documentation . . . . . 393
7.62.1.1 ifx_driver_smif_xip . . . . . 393
7.63 platform/ext/target/infineon/common/drivers/protection/mpu_armv8m_drv.h File Reference . . . . . 393
7.63.1 Macro Definition Documentation . . . . . 395
7.63.1.1 HARDFAULT_NMI_ENABLE . . . . . 395
7.63.1.2 MPU_ARMV8M_MAIR_ATTR_CODE_IDX . . . . . 395
7.63.1.3 MPU_ARMV8M_MAIR_ATTR_CODE_VAL . . . . . 395
7.63.1.4 MPU_ARMV8M_MAIR_ATTR_DATA_IDX . . . . . 395
7.63.1.5 MPU_ARMV8M_MAIR_ATTR_DATA_VAL . . . . . 395
7.63.1.6 MPU_ARMV8M_MAIR_ATTR_DEVICE_IDX . . . . . 395
7.63.1.7 MPU_ARMV8M_MAIR_ATTR_DEVICE_VAL . . . . . 395
7.63.1.8 PRIVILEGED_DEFAULT_ENABLE . . . . . 396
7.63.2 Enumeration Type Documentation . . . . . 396
7.63.2.1 mpu_armv8m_attr_access_t . . . . . 396
7.63.2.2 mpu_armv8m_attr_exec_t . . . . . 396
7.63.2.3 mpu_armv8m_attr_shared_t . . . . . 396
7.64 platform/ext/target/infineon/common/drivers/stdio/uart_pdl_stdout.c File Reference . . . . . 396
7.64.1 Function Documentation . . . . . 397
7.64.1.1 stdio_init() . . . . . 397
7.64.1.2 stdio_output_string() . . . . . 397
7.64.1.3 stdio_uninit() . . . . . 398
7.65 platform/ext/target/infineon/common/drivers/stdio/uart_pdl_stdout.h File Reference . . . . . 398
7.65.1 Function Documentation . . . . . 399
7.65.1.1 stdio_output_string_raw() . . . . . 399
7.66 platform/ext/target/infineon/common/interface/include/ifx_platform_api.h File Reference . . . . . 399
7.66.1 Macro Definition Documentation . . . . . 401
7.66.1.1 IFX_PLATFORM_IOCTL_APP_MAX . . . . . 401
7.66.1.2 IFX_PLATFORM_IOCTL_APP_MIN . . . . . 401
7.66.2 Function Documentation . . . . . 401
7.66.2.1 ifx_platform_log_msg() . . . . . 401
7.67 platform/ext/target/infineon/common/interface/include/mtb_srf_ipc_custom_packet.h File Reference 402
7.67.1 Typedef Documentation . . . . . 403
7.67.1.1 mtb_srf_ipc_packet_t . . . . . 403
7.68 platform/ext/target/infineon/common/interface/include/multi_core/tfm_mailbox_config.h File Reference 403
7.69 platform/ext/target/infineon/common/interface/src/ifx_mtb_srf.c File Reference . . . . . 403
7.69.1 Function Documentation . . . . . 404
7.69.1.1 mtb_srf_request_submit() . . . . . 404
7.70 platform/ext/target/infineon/common/interface/src/ifx_mtb_srf_relay.c File Reference . . . . . 405
7.71 platform/ext/target/infineon/common/interface/src/ifx_platform_api.c File Reference . . . . . 405
7.71.1 Function Documentation . . . . . 406
7.71.1.1 ifx_platform_log_msg() . . . . . 406

7.72 platform/ext/target/infineon/common/interface/src/ifx_platform_private.h File Reference . . . . . 407
7.72.1 Macro Definition Documentation . . . . . 408
7.72.1.1 IFX_CUSTOM_PLATFORM_HAL_IOCTL . . . . . 408
7.72.1.2 IFX_PLATFORM_API_ID_MTB_SRF . . . . . 408
7.72.1.3 IFX_PLATFORM_IOCTL_LOG_MSG . . . . . 408
7.73 platform/ext/target/infineon/common/interface/src/multi_core/ifx_mtb_mailbox.c File Reference . . . 408
7.73.1 Macro Definition Documentation . . . . . 409
7.73.1.1 IFX_MTB_MAILBOX_CACHE_PRESENT . . . . . 409
7.73.2 Function Documentation . . . . . 409
7.73.2.1 ifx_mtb_mailbox_client_call() . . . . . 409
7.74 platform/ext/target/infineon/common/libs/ifx_se_rt_services_utils/spe/ifx_se_tfm_utils.h File Reference 410
7.74.1 Detailed Description . . . . . 410
7.74.2 Macro Definition Documentation . . . . . 411
7.74.2.1 IFX_SE_TFM_SYSCALL_CONTEXT . . . . . 411
7.75 platform/ext/target/infineon/common/nspe/mailbox/platform_ns_mailbox.c File Reference . . . . . 411
7.75.1 Function Documentation . . . . . 412
7.75.1.1 IFX_IPC_TFM_TO_NS_IPC_INTR() . . . . . 412
7.75.1.2 tfm_ns_mailbox_hal_enter_critical() . . . . . 412
7.75.1.3 tfm_ns_mailbox_hal_enter_critical_isr() . . . . . 412
7.75.1.4 tfm_ns_mailbox_hal_exit_critical() . . . . . 413
7.75.1.5 tfm_ns_mailbox_hal_exit_critical_isr() . . . . . 413
7.75.1.6 tfm_ns_mailbox_hal_init() . . . . . 414
7.75.1.7 tfm_ns_mailbox_hal_notify_peer() . . . . . 414
7.75.1.8 tfm_ns_multi_core_boot() . . . . . 415
7.75.1.9 tfm_platform_ns_wait_for_s_cpu_ready() . . . . . 415
7.76 platform/ext/target/infineon/common/nspe/mailbox/tfm_ns_mailbox_rtos_api.c File Reference . . . . 416
7.76.1 Macro Definition Documentation . . . . . 417
7.76.1.1 MAILBOX_THREAD_FLAG . . . . . 417
7.76.1.2 MAX_SEMAPHORE_COUNT . . . . . 417
7.76.2 Function Documentation . . . . . 417
7.76.2.1 tfm_ns_mailbox_os_get_task_handle() . . . . . 417
7.76.2.2 tfm_ns_mailbox_os_lock_acquire() . . . . . 417
7.76.2.3 tfm_ns_mailbox_os_lock_init() . . . . . 418
7.76.2.4 tfm_ns_mailbox_os_lock_release() . . . . . 418
7.76.2.5 tfm_ns_mailbox_os_spin_lock() . . . . . 418
7.76.2.6 tfm_ns_mailbox_os_spin_unlock() . . . . . 418
7.76.2.7 tfm_ns_mailbox_os_wait_reply() . . . . . 418
7.76.2.8 tfm_ns_mailbox_os_wake_task_isr() . . . . . 419
7.77 platform/ext/target/infineon/common/nspe/os_wrapper/os_wrapper_cyabs_rtos.c File Reference . . 419
7.78 platform/ext/target/infineon/common/nspe/os_wrapper/semaphore.h File Reference . . . . . 419
7.78.1 Function Documentation . . . . . 421
7.78.1.1 os_wrapper_semaphore_acquire() . . . . . 421

7.78.1.2 os_wrapper_semaphore_create() . . . 421
7.78.1.3 os_wrapper_semaphore_delete() . . . 421
7.78.1.4 os_wrapper_semaphore_release() . . . 423
7.79 platform/ext/target/infineon/common/nspe/test/ifx_fpu_ns.c File Reference . . . 423
7.79.1 Function Documentation . . . 424
7.79.1.1 TFM_FPU_NS_TEST_IRQ() . . . 424
7.79.1.2 tfm_test_ns_fpu_init() . . . 424
7.80 platform/ext/target/infineon/common/nspe/test/plat_test_ns.c File Reference . . . 425
7.81 platform/ext/target/infineon/common/shared/mailbox/ifx_platform_mailbox.h File Reference . . . 425
7.82 platform/ext/target/infineon/common/shared/mailbox/platform_multicore.c File Reference . . . 426
7.82.1 Macro Definition Documentation . . . 427
7.82.1.1 IPC_RX_CHAN . . . 427
7.82.1.2 IPC_RX_INT_MASK . . . 427
7.82.1.3 IPC_RX_INTR_STRUCT . . . 428
7.82.1.4 IPC_TX_CHAN . . . 428
7.82.1.5 IPC_TX_NOTIFY_MASK . . . 428
7.82.2 Function Documentation . . . 428
7.82.2.1 ifx_mailbox_fetch_msg_data() . . . 428
7.82.2.2 ifx_mailbox_fetch_msg_ptr() . . . 428
7.82.2.3 ifx_mailbox_send_msg_data() . . . 429
7.82.2.4 ifx_mailbox_send_msg_ptr() . . . 429
7.82.2.5 ifx_mailbox_wait_for_notify() . . . 430
7.83 platform/ext/target/infineon/common/shared/mailbox/platform_multicore.h File Reference . . . 430
7.83.1 Macro Definition Documentation . . . 431
7.83.1.1 IFX_IPC_SYNC_MAGIC . . . 431
7.83.1.2 IFX_NS_MAILBOX_INIT_ENABLE . . . 431
7.83.1.3 IFX_PLATFORM_MAILBOX_INIT_ERROR . . . 431
7.83.1.4 IFX_PLATFORM_MAILBOX_INVAL_PARAMS . . . 432
7.83.1.5 IFX_PLATFORM_MAILBOX_RX_ERROR . . . 432
7.83.1.6 IFX_PLATFORM_MAILBOX_SUCCESS . . . 432
7.83.1.7 IFX_PLATFORM_MAILBOX_TX_ERROR . . . 432
7.83.1.8 IFX_PSA_CLIENT_CALL_REPLY_MAGIC . . . 432
7.83.1.9 IFX_PSA_CLIENT_CALL_REQ_MAGIC . . . 432
7.83.1.10 IFX_S_MAILBOX_READY . . . 432
7.83.2 Function Documentation . . . 432
7.83.2.1 ifx_mailbox_fetch_msg_data() . . . 432
7.83.2.2 ifx_mailbox_fetch_msg_ptr() . . . 433
7.83.2.3 ifx_mailbox_send_msg_data() . . . 433
7.83.2.4 ifx_mailbox_send_msg_ptr() . . . 434
7.83.2.5 ifx_mailbox_wait_for_notify() . . . 434
7.84 platform/ext/target/infineon/common/spe/faults/faults.c File Reference . . . 434
7.84.1 Macro Definition Documentation . . . 435

7.84.1.1 SCB_SHCSR_ENABLED_FAULTS . . . 435
7.84.2 Function Documentation . . . 435
7.84.2.1 FIH_RET_TYPE() . . . 435
7.84.2.2 tfm_hal_platform_exception_info() . . . 438
7.85 platform/ext/target/infineon/common/spe/faults/faults.h File Reference . . . 438
7.85.1 Function Documentation . . . 439
7.85.1.1 FIH_RET_TYPE() . . . 439
7.86 platform/ext/target/infineon/common/spe/faults/faults_cat1b.c File Reference . . . 443
7.86.1 Function Documentation . . . 443
7.86.1.1 c_cpuss_interrupt_msc_IRQn_Handler() . . . 443
7.86.1.2 c_cpuss_interrupts_fault_0_IRQn_Handler() . . . 444
7.86.1.3 cpuss_interrupt_msc_IRQn_Handler() . . . 444
7.86.1.4 cpuss_interrupts_fault_0_IRQn_Handler() . . . 444
7.86.1.5 FIH_RET_TYPE() . . . 444
7.87 platform/ext/target/infineon/common/spe/faults/faults_cat1d.c File Reference . . . 447
7.87.1 Function Documentation . . . 448
7.87.1.1 c_m33syscpuss_interrupt_msc_IRQn_Handler() . . . 448
7.87.1.2 c_m33syscpuss_interrupts_fault_0_IRQn_Handler() . . . 448
7.87.1.3 FIH_RET_TYPE() . . . 448
7.87.1.4 m33syscpuss_interrupt_msc_IRQn_Handler() . . . 451
7.87.1.5 m33syscpuss_interrupts_fault_0_IRQn_Handler() . . . 451
7.88 platform/ext/target/infineon/common/spe/faults/faults_cat1e.c File Reference . . . 451
7.88.1 Macro Definition Documentation . . . 452
7.88.1.1 CY_SYSFAULT_NO_FAULT . . . 452
7.88.2 Function Documentation . . . 452
7.88.2.1 c_m33syscpuss_interrupt_msc_IRQn_Handler() . . . 452
7.88.2.2 c_m33syscpuss_interrupts_fault_0_IRQn_Handler() . . . 453
7.88.2.3 Cy_SysFault_ClearInterrupt() . . . 453
7.88.2.4 Cy_SysFault_ClearStatus() . . . 454
7.88.2.5 Cy_SysFault_GetErrorSource() . . . 454
7.88.2.6 Cy_SysFault_GetFaultData() . . . 454
7.88.2.7 Cy_SysFault_Init() . . . 455
7.88.2.8 Cy_SysFault_SetInterruptMask() . . . 456
7.88.2.9 Cy_SysFault_SetMaskByIdx() . . . 456
7.88.2.10 FIH_RET_TYPE() . . . 456
7.88.2.11 m33syscpuss_interrupt_msc_IRQn_Handler() . . . 459
7.88.2.12 m33syscpuss_interrupts_fault_0_IRQn_Handler() . . . 459
7.89 platform/ext/target/infineon/common/spe/faults/faults_dump.h File Reference . . . 459
7.89.1 Function Documentation . . . 460
7.89.1.1 ifx_faults_dump_default_fault() . . . 460
7.89.1.2 ifx_faults_dump_mpc_fault() . . . 461
7.89.1.3 ifx_faults_dump_peri_gpx_ahb_vio_fault() . . . 461

7.89.1.4 ifx_faults_dump_peri_gpx_timeout_vio_fault() . . . 462
7.89.1.5 ifx_faults_dump_peri_msx_ppc_vio_fault() . . . 462
7.89.1.6 ifx_faults_dump_peri_ppc_pc_mask_vio_fault() . . . 463
7.90 platform/ext/target/infineon/common/spe/fih/tfm_fih_mxcrypto.c File Reference . . . 463
7.90.1 Macro Definition Documentation . . . 464
7.90.1.1 PRNG_A . . . 464
7.90.1.2 PRNG_B . . . 464
7.90.1.3 PRNG_C . . . 464
7.90.2 Function Documentation . . . 464
7.90.2.1 tfm_fih_random_generate() . . . 464
7.90.2.2 tfm_fih_random_init() . . . 465
7.91 platform/ext/target/infineon/common/spe/otp/otp_flash.c File Reference . . . 465
7.91.1 Function Documentation . . . 465
7.91.1.1 tfm_plat_otp_get_size() . . . 465
7.91.1.2 tfm_plat_otp_init() . . . 465
7.91.1.3 tfm_plat_otp_read() . . . 466
7.91.1.4 tfm_plat_otp_write() . . . 466
7.92 platform/ext/target/infineon/common/spe/protection/partition_assets.h File Reference . . . 466
7.93 platform/ext/target/infineon/common/spe/protection/partition_info.h File Reference . . . 467
7.93.1 Function Documentation . . . 468
7.93.1.1 ifx_protection_get_partition_info() . . . 468
7.94 platform/ext/target/infineon/common/spe/protection/protection_mpc_api.h File Reference . . . 468
7.94.1 Detailed Description . . . 469
7.94.2 Function Documentation . . . 469
7.94.2.1 FIH_RET_TYPE() . . . 470
7.94.3 Variable Documentation . . . 470
7.94.3.1 access_type . . . 470
7.94.3.2 base . . . 470
7.94.3.3 size . . . 470
7.95 platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc.c File Reference . . . 470
7.95.1 Function Documentation . . . 471
7.95.1.1 FIH_CALL() . . . 471
7.95.1.2 FIH_RET() . . . 472
7.95.1.3 FIH_RET_TYPE() . . . 472
7.95.2 Variable Documentation . . . 475
7.95.2.1 address . . . 475
7.95.2.2 asset . . . 476
7.95.2.3 ifx_fixed_mpc_static_config . . . 476
7.95.2.4 ifx_fixed_mpc_static_config_count . . . 476
7.95.2.5 mpc_reg_cfg . . . 476
7.95.2.6 ns_mask . . . 476
7.95.2.7 r_mask . . . 476

7.95.2.8 size . . . 476
7.95.2.9 w_mask . . . 476
7.96 platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc.h File Reference . . . 476
7.96.1 Detailed Description . . . 477
7.96.2 Macro Definition Documentation . . . 478
7.96.2.1 IFX_MAX_FIXED_CONFIGS . . . 478
7.96.2.2 IFX_MPC_NAMED_MMIO_APP_ROT_CFG . . . 478
7.96.2.3 IFX_MPC_NAMED_MMIO_PSA_ROT_CFG . . . 478
7.96.2.4 IFX_MPC_NAMED_MMIO_SPM_ROT_CFG . . . 478
7.96.3 Function Documentation . . . 478
7.96.3.1 FIH_RET_TYPE() . . . 478
7.96.3.2 ifx_mpc_apply_configuration_with_mpc() . . . 497
7.96.4 Variable Documentation . . . 498
7.96.4.1 ifx_fixed_mpc_static_config . . . 498
7.96.4.2 ifx_fixed_mpc_static_config_count . . . 498
7.97 platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc_v1.c File Reference . 498
7.97.1 Macro Definition Documentation . . . 499
7.97.1.1 IFX_MPC_BLOCK_SIZE_TO_BYTES . . . 499
7.97.2 Function Documentation . . . 499
7.97.2.1 FIH_RET() . . . 499
7.97.2.2 FIH_RET_TYPE() . . . 500
7.97.2.3 for() . . . 503
7.97.2.4 if() [1/3] . . . 504
7.97.2.5 if() [2/3] . . . 504
7.97.2.6 if() [3/3] . . . 505
7.97.2.7 ifx_mpc_apply_configuration_with_mpc() . . . 506
7.97.3 Variable Documentation . . . 506
7.97.3.1 access_type . . . 506
7.97.3.2 base . . . 506
7.97.3.3 block_size . . . 507
7.97.3.4 expected_sec_attr . . . 507
7.97.3.5 first_block_idx . . . 507
7.97.3.6 is_secure . . . 507
7.97.3.7 last_block_idx . . . 507
7.97.3.8 mem_region_cfg . . . 507
7.97.3.9 offset . . . 507
7.97.3.10 pc . . . 507
7.97.3.11 size . . . 507
7.98 platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc_v2.c File Reference . 508
7.98.1 Macro Definition Documentation . . . 509
7.98.1.1 IFX_MPC_BLOCK_SIZE_TO_BYTES . . . 509
7.98.1.2 IFX_MPC_ROT_BLK_CFG_BLOCK_SIZE_Msk . . . 509

7.98.1.3 IFX_MPC_ROT_BLK_CFG_BLOCK_SIZE_Pos . . . 509
7.98.2 Function Documentation . . . 509
7.98.2.1 FIH_RET() . . . 509
7.98.2.2 FIH_RET_TYPE() . . . 510
7.98.2.3 for() . . . 513
7.98.2.4 if() [1/3] . . . 514
7.98.2.5 if() [2/3] . . . 514
7.98.2.6 if() [3/3] . . . 515
7.98.2.7 ifx_mpc_apply_configuration_with_mpc() . . . 515
7.98.3 Variable Documentation . . . 516
7.98.3.1 access_type . . . 516
7.98.3.2 base . . . 516
7.98.3.3 block_size . . . 516
7.98.3.4 expected_sec_attr . . . 516
7.98.3.5 first_block_idx . . . 516
7.98.3.6 is_secure . . . 516
7.98.3.7 last_block_idx . . . 516
7.98.3.8 last_offset . . . 516
7.98.3.9 mem_region_cfg . . . 517
7.98.3.10 offset . . . 517
7.98.3.11 pc . . . 517
7.98.3.12 size . . . 517
7.99 platform/ext/target/infineon/common/spe/protection/protection_mpc_sert.c File Reference . . . 517
7.99.1 Macro Definition Documentation . . . 518
7.99.1.1 IFX_MPC_EXT_BLOCK_SIZE_TO_BYTES . . . 518
7.99.1.2 IFX_SE_RT_RRAM_BLOCK_COUNT . . . 518
7.99.1.3 IFX_SE_RT_RRAM_SIZE . . . 518
7.99.2 Function Documentation . . . 518
7.99.2.1 FIH_RET() . . . 519
7.99.2.2 FIH_RET_TYPE() . . . 519
7.99.2.3 for() . . . 519
7.99.2.4 if() [1/4] . . . 519
7.99.2.5 if() [2/4] . . . 519
7.99.2.6 if() [3/4] . . . 519
7.99.2.7 if() [4/4] . . . 520
7.99.2.8 ifx_mpc_sert_apply_configuration() . . . 520
7.99.3 Variable Documentation . . . 520
7.99.3.1 access_type . . . 521
7.99.3.2 attr . . . 521
7.99.3.3 base . . . 521
7.99.3.4 first_block_idx . . . 521
7.99.3.5 is_secure . . . 521

7.99.3.6 last_block_idx . . . 521
7.99.3.7 mask . . . 521
7.99.3.8 memory_config . . . 521
7.99.3.9 offset . . . 521
7.99.3.10 pc . . . 522
7.99.3.11 size . . . 522
7.100 platform/ext/target/infineon/common/spe/protection/protection_mpc_sert.h File Reference . . . 522
7.100.1 Detailed Description . . . 523
7.100.2 Function Documentation . . . 523
7.100.2.1 FIH_RET_TYPE() . . . 523
7.100.2.2 ifx_mpc_sert_apply_configuration() . . . 524
7.100.3 Variable Documentation . . . 524
7.100.3.1 access_type . . . 524
7.100.3.2 base . . . 524
7.100.3.3 memory_config . . . 525
7.100.3.4 size . . . 525
7.101 platform/ext/target/infineon/common/spe/protection/protection_mpc_sw_policy.c File Reference . . . 525
7.101.1 Macro Definition Documentation . . . 526
7.101.1.1 CY_FLASH_BASE_S . . . 526
7.101.1.2 CY_SRAM0_BASE_S . . . 526
7.101.1.3 IFX_MPC_BLOCK_SIZE . . . 526
7.101.1.4 IFX_MPC_GET_PC_ATTR . . . 526
7.101.1.5 IFX_MPC_PC_ATTR_MSK . . . 526
7.101.1.6 IFX_MPC_PC_ATTR_NS_MSK . . . 527
7.101.1.7 IFX_MPC_PC_ATTR_R_MSK . . . 527
7.101.1.8 IFX_MPC_PC_ATTR_SIZE_IN_BITS . . . 527
7.101.1.9 IFX_MPC_PC_ATTR_W_MSK . . . 527
7.101.2 Function Documentation . . . 527
7.101.2.1 FIH_RET() . . . 527
7.101.2.2 FIH_RET_TYPE() . . . 527
7.101.2.3 for() . . . 530
7.101.2.4 if() [1/2] . . . 531
7.101.2.5 if() [2/2] . . . 531
7.101.2.6 TFM_COVERITY_DEVIATE_LINE() . . . 531
7.101.3 Variable Documentation . . . 531
7.101.3.1 access_type . . . 531
7.101.3.2 base . . . 531
7.101.3.3 base_s . . . 532
7.101.3.4 else . . . 532
7.101.3.5 mpc_base_addr . . . 532
7.101.3.6 mpc_end_idx . . . 532
7.101.3.7 mpc_policy . . . 532

7.101.3.8 size . . . 532
7.102 platform/ext/target/infineon/common/spe/protection/protection_mpc_sw_policy.h File Reference . . . 532
7.102.1 Detailed Description . . . 532
7.102.2 Macro Definition Documentation . . . 533
7.102.2.1 IFX_MPC_NAMED_MMIO_APP_ROT_CFG . . . 533
7.102.2.2 IFX_MPC_NAMED_MMIO_PSA_ROT_CFG . . . 533
7.102.2.3 IFX_MPC_NAMED_MMIO_SPM_ROT_CFG . . . 533
7.103 platform/ext/target/infineon/common/spe/protection/protection_mpu.c File Reference . . . 533
7.103.1 Function Documentation . . . 534
7.103.1.1 FIH_RET() . . . 534
7.103.1.2 FIH_RET_TYPE() . . . 534
7.103.1.3 if() . . . 537
7.103.1.4 ifx_mpu_config_enable_region() . . . 537
7.103.1.5 ifx_mpu_disable_asset_regions() . . . 537
7.103.2 Variable Documentation . . . 537
7.103.2.1 attr_access . . . 537
7.103.2.2 attr_exec . . . 538
7.103.2.3 attr_sh . . . 538
7.103.2.4 else . . . 538
7.103.2.5 p_asset . . . 538
7.103.2.6 region_base . . . 538
7.103.2.7 region_limit . . . 538
7.104 platform/ext/target/infineon/common/spe/protection/protection_mpu.h File Reference . . . 538
7.104.1 Function Documentation . . . 539
7.104.1.1 FIH_RET_TYPE() . . . 539
7.104.1.2 ifx_mpu_disable_asset_regions() . . . 540
7.104.2 Variable Documentation . . . 540
7.104.2.1 p_asset . . . 540
7.105 platform/ext/target/infineon/common/spe/protection/protection_msc.c File Reference . . . 540
7.105.1 Function Documentation . . . 541
7.105.1.1 FIH_RET_TYPE() . . . 541
7.106 platform/ext/target/infineon/common/spe/protection/protection_msc.h File Reference . . . 544
7.106.1 Function Documentation . . . 545
7.106.1.1 FIH_RET_TYPE() . . . 545
7.107 platform/ext/target/infineon/common/spe/protection/protection_pc.c File Reference . . . 564
7.107.1 Macro Definition Documentation . . . 565
7.107.1.1 IFX_RETURN_ON_EXCEPTION_BIT_MASK . . . 565
7.107.1.2 IFX_VTOR_ADDRESS . . . 565
7.107.2 Function Documentation . . . 565
7.107.2.1 FIH_RET_TYPE() . . . 565
7.107.2.2 ifx_arch_switch_protection_context() . . . 569
7.107.2.3 ifx_arch_switch_protection_context_from_irq() . . . 569

7.107.2.4 ifx_arch_switch_protection_context_thread() . . . 570
7.107.2.5 ifx_pc2_exception_handler() . . . 570
7.107.2.6 TFM_COVERITY_DEVIATE_LINE() . . . 570
7.107.3 Variable Documentation . . . 570
7.107.3.1 ifx_arch_active_protection_context . . . 570
7.107.3.2 ifx_pc2_vector_table . . . 570
7.108 platform/ext/target/infineon/common/spe/protection/protection_pc.h File Reference . . . 571
7.108.1 Function Documentation . . . 572
7.108.1.1 FIH_RET_TYPE() . . . 572
7.108.1.2 ifx_arch_switch_protection_context_thread() . . . 590
7.108.2 Variable Documentation . . . 591
7.108.2.1 ifx_arch_active_protection_context . . . 591
7.109 platform/ext/target/infineon/common/spe/protection/protection_ppc_api.h File Reference . . . 591
7.109.1 Detailed Description . . . 592
7.109.2 Function Documentation . . . 592
7.109.2.1 FIH_RET_TYPE() . . . 592
7.109.3 Variable Documentation . . . 593
7.109.3.1 asset . . . 593
7.110 platform/ext/target/infineon/common/spe/protection/protection_ppc_v1.c File Reference . . . 593
7.110.1 Function Documentation . . . 594
7.110.1.1 fih_delay() . . . 594
7.110.1.2 FIH_RET() . . . 595
7.110.1.3 if() [1/3] . . . 595
7.110.1.4 if() [2/3] . . . 595
7.110.1.5 if() [3/3] . . . 596
7.110.1.6 ifx_check_config_validity() . . . 596
7.110.2 Variable Documentation . . . 596
7.110.2.1 asset . . . 596
7.110.2.2 ppc_attribute . . . 596
7.110.2.3 ppc_pcmask . . . 596
7.111 platform/ext/target/infineon/common/spe/protection/protection_ppc_v1.h File Reference . . . 597
7.111.1 Detailed Description . . . 597
7.111.2 Macro Definition Documentation . . . 597
7.111.2.1 IFX_PPC_NAMED_MMIO_APP_ROT_CFG . . . 597
7.111.2.2 IFX_PPC_NAMED_MMIO_PSA_ROT_CFG . . . 598
7.111.2.3 IFX_PPC_NAMED_MMIO_SPM_ROT_CFG . . . 598
7.112 platform/ext/target/infineon/common/spe/protection/protection_ppc_v2.c File Reference . . . 598
7.112.1 Function Documentation . . . 598
7.112.1.1 fih_delay() . . . 599
7.112.1.2 FIH_RET() . . . 599
7.112.1.3 if() [1/3] . . . 599
7.112.1.4 if() [2/3] . . . 599

7.112.1.5 if() [3/3] . . . 600
7.112.1.6 ifx_check_config_validity() . . . 600
7.112.2 Variable Documentation . . . 600
7.112.2.1 asset . . . 600
7.112.2.2 ppc_attribute . . . 600
7.112.2.3 ppc_pcmask . . . 600
7.113 platform/ext/target/infineon/common/spe/protection/protection_ppc_v2.h File Reference . . . 601
7.113.1 Detailed Description . . . 601
7.113.2 Macro Definition Documentation . . . 601
7.113.2.1 IFX_PPC_NAMED_MMIO_APP_ROT_CFG . . . 601
7.113.2.2 IFX_PPC_NAMED_MMIO_PSA_ROT_CFG . . . 602
7.113.2.3 IFX_PPC_NAMED_MMIO_SPM_ROT_CFG . . . 602
7.114 platform/ext/target/infineon/common/spe/protection/protection_sau.c File Reference . . . 602
7.114.1 Macro Definition Documentation . . . 602
7.114.1.1 IFX_SAU_RESERVED_REGION_COUNT . . . 603
7.114.1.2 IFX_SAU_VENEER_REGION_COUNT . . . 603
7.114.2 Function Documentation . . . 603
7.114.2.1 FIH_RET_TYPE() . . . 603
7.115 platform/ext/target/infineon/common/spe/protection/protection_sau.h File Reference . . . 607
7.115.1 Function Documentation . . . 608
7.115.1.1 FIH_RET_TYPE() . . . 608
7.116 platform/ext/target/infineon/common/spe/protection/protection_shared_data.c File Reference . . . 627
7.116.1 Variable Documentation . . . 628
7.116.1.1 ifx_spm_state . . . 628
7.117 platform/ext/target/infineon/common/spe/protection/protection_shared_data.h File Reference . . . 628
7.117.1 Macro Definition Documentation . . . 629
7.117.1.1 IFX_IS_SPM_INITILIZING . . . 629
7.117.1.2 IFX_IS_SPM_RUNNING . . . 629
7.117.1.3 IFX_SPM_STATE_INITIALIZING . . . 629
7.117.1.4 IFX_SPM_STATE_RUNNING . . . 629
7.117.2 Variable Documentation . . . 629
7.117.2.1 ifx_spm_state . . . 629
7.118 platform/ext/target/infineon/common/spe/protection/protection_types.h File Reference . . . 630
7.118.1 Macro Definition Documentation . . . 631
7.118.1.1 IFX_GET_BOUNDARY_DOMAIN . . . 631
7.118.1.2 IFX_GET_PARTITION_PC . . . 631
7.118.1.3 IFX_IS_PARTITION_PRIVILEGED . . . 631
7.118.1.4 IFX_PROTECT_APP_ROT_DOMAIN_CFG . . . 631
7.118.1.5 IFX_PROTECT_APP_ROT_SINGLE_DOMAIN . . . 631
7.118.1.6 IFX_PROTECT_IS_DOMAIN_EQUAL . . . 632
7.118.1.7 IFX_PROTECT_IS_PRIVILEGED_DOMAIN . . . 632
7.118.1.8 IFX_PROTECT_IS_PRIVILEGED_DOMAIN_PRIVATE . . . 632

7.118.1.9 IFX_PROTECT_PSA_ROT_DOMAIN_CFG . . . 632
7.118.1.10 IFX_PROTECT_PSA_ROT_SINGLE_DOMAIN . . . 632
7.118.1.11 IFX_PROTECT_SPM_DOMAIN . . . 632
7.118.1.12 IFX_PROTECT_SPM_DOMAIN_CFG . . . 632
7.118.1.13 IFX_SPM_BOUNDARY . . . 632
7.118.2 Typedef Documentation . . . 632
7.118.2.1 ifx_partition_info_t . . . 633
7.118.2.2 ifx_ppc_regions_config_t . . . 633
7.119 platform/ext/target/infineon/common/spe/protection/protection_tz.c File Reference . . . 633
7.119.1 Function Documentation . . . 634
7.119.1.1 fih_delay() . . . 634
7.119.1.2 FIH_RET() [1/2] . . . 634
7.119.1.3 FIH_RET() [2/2] . . . 634
7.119.1.4 if() [1/8] . . . 634
7.119.1.5 if() [2/8] . . . 635
7.119.1.6 if() [3/8] . . . 635
7.119.1.7 if() [4/8] . . . 635
7.119.1.8 if() [5/8] . . . 635
7.119.1.9 if() [6/8] . . . 636
7.119.1.10 if() [7/8] . . . 636
7.119.1.11 if() [8/8] . . . 636
7.119.1.12 switch() [1/2] . . . 637
7.119.1.13 switch() [2/2] . . . 637
7.119.2 Variable Documentation . . . 637
7.119.2.1 access_type . . . 637
7.119.2.2 base . . . 637
7.119.2.3 else . . . 637
7.119.2.4 flags . . . 638
7.119.2.5 is_non_secure . . . 638
7.119.2.6 known . . . 638
7.119.2.7 known_secure_level . . . 638
7.119.2.8 pb . . . 638
7.119.2.9 pe . . . 638
7.119.2.10 perme . . . 638
7.119.2.11 singleCheck . . . 638
7.119.2.12 size . . . 638
7.120 platform/ext/target/infineon/common/spe/protection/protection_tz.h File Reference . . . 639
7.120.1 Function Documentation . . . 640
7.120.1.1 FIH_RET_TYPE() . . . 640
7.120.2 Variable Documentation . . . 640
7.120.2.1 access_type . . . 640
7.120.2.2 base . . . 640

7.120.2.3 size . . . 640
7.121 platform/ext/target/infineon/common/spe/protection/protection_utils.c File Reference . . . 640
7.121.1 Function Documentation . . . 641
7.121.1.1 ifx_find_memory_config() . . . 641
7.121.1.2 ifx_get_all_memory_configs() . . . 642
7.122 platform/ext/target/infineon/common/spe/protection/protection_utils.h File Reference . . . 642
7.123 platform/ext/target/infineon/common/spe/protection/tfm_hal_isolation.c File Reference . . . 643
7.123.1 Function Documentation . . . 644
7.123.1.1 __DMB() . . . 644
7.123.1.2 __DSB() . . . 645
7.123.1.3 __ISB() . . . 645
7.123.1.4 __set_CONTROL() . . . 645
7.123.1.5 FIH_CALL() [1/3] . . . 646
7.123.1.6 FIH_CALL() [2/3] . . . 646
7.123.1.7 FIH_CALL() [3/3] . . . 646
7.123.1.8 FIH_RET() [1/3] . . . 646
7.123.1.9 FIH_RET() [2/3] . . . 647
7.123.1.10 FIH_RET() [3/3] . . . 647
7.123.1.11 FIH_RET_TYPE() . . . 647
7.123.1.12 for() [1/2] . . . 647
7.123.1.13 for() [2/2] . . . 648
7.123.1.14 if() [1/11] . . . 648
7.123.1.15 if() [2/11] . . . 649
7.123.1.16 if() [3/11] . . . 649
7.123.1.17 if() [4/11] . . . 649
7.123.1.18 if() [5/11] . . . 650
7.123.1.19 if() [6/11] . . . 650
7.123.1.20 if() [7/11] . . . 650
7.123.1.21 if() [8/11] . . . 651
7.123.1.22 if() [9/11] . . . 651
7.123.1.23 if() [10/11] . . . 652
7.123.1.24 if() [11/11] . . . 652
7.123.2 Variable Documentation . . . 652
7.123.2.1 access_type . . . 652
7.123.2.2 asset . . . 652
7.123.2.3 asset_cnt . . . 653
7.123.2.4 base . . . 653
7.123.2.5 boundary . . . 653
7.123.2.6 cfg . . . 653
7.123.2.7 ctrl . . . 653
7.123.2.8 else . . . 653
7.123.2.9 p_assets . . . 653

7.123.2.10 p_boundary . . . 653
7.123.2.11 p_info . . . 654
7.123.2.12 p_ldinf . . . 654
7.123.2.13 result . . . 654
7.123.2.14 size . . . 654
7.123.2.15 w . . . 654
7.124 platform/ext/target/infineon/common/spe/protection/tfm_platform_arch_hooks.c File Reference . . . 654
7.124.1 Function Documentation . . . 654
7.124.1.1 ifx_arch_get_context_protection_context() . . . 655
7.124.1.2 ifx_arch_get_current_component_protection_context() . . . 655
7.125 platform/ext/target/infineon/common/spe/protection/tfm_platform_arch_hooks.h File Reference . . . 655
7.125.1 Macro Definition Documentation . . . 656
7.125.1.1 IFX_PLATFORM_EXIT_FROM_INTERRUPT_WITH_PC_RESTORE_HOOK . . . 657
7.125.1.2 PLATFORM_HARD_FAULT_HANDLER_HOOK . . . 657
7.125.1.3 PLATFORM_HARD_FAULT_HANDLER_HOOK_IAR_REQUIRED . . . 657
7.125.1.4 PLATFORM_HARD_FAULT_HANDLER_HOOK_VALIDATE_PC . . . 657
7.125.1.5 PLATFORM_INIT_SPM_FUNC_CONTEXT_HOOK . . . 657
7.125.1.6 PLATFORM_PENDSV_HANDLER_EXIT_HOOK . . . 657
7.125.1.7 PLATFORM_PENDSV_HANDLER_HOOK_IAR_REQUIRED . . . 658
7.125.1.8 PLATFORM_SVC_HANDLER_EXIT_HOOK . . . 658
7.125.1.9 PLATFORM_SVC_HANDLER_FROM_FLIH_FUNC_ENTER_HOOK . . . 658
7.125.1.10 PLATFORM_SVC_HANDLER_FROM_FLIH_FUNC_EXIT_HOOK . . . 658
7.125.1.11 PLATFORM_SVC_HANDLER_GET_MSP_HOOK . . . 658
7.125.1.12 PLATFORM_SVC_HANDLER_HOOK_IAR_REQUIRED . . . 658
7.125.1.13 PLATFORM_SVC_HANDLER_TO_FLIH_FUNC_ENTER_HOOK . . . 659
7.125.1.14 PLATFORM_SVC_HANDLER_TO_FLIH_FUNC_EXIT_HOOK . . . 659
7.125.1.15 PLATFORM_THREAD_MODE_SPM_RETURN_HOOK . . . 659
7.125.2 Function Documentation . . . 659
7.125.2.1 ifx_arch_get_context_protection_context() . . . 659
7.125.2.2 ifx_arch_get_current_component_protection_context() . . . 659
7.125.2.3 ifx_arch_switch_protection_context() . . . 660
7.125.2.4 ifx_arch_switch_protection_context_from_irq() . . . 660
7.126 platform/ext/target/infineon/common/spe/provisioning/provisioning.c File Reference . . . 660
7.126.1 Function Documentation . . . 661
7.126.1.1 ifx_get_security_lifecycle() . . . 661
7.126.1.2 tfm_plat_provisioning_check_for_dummy_keys() . . . 662
7.126.1.3 tfm_plat_provisioning_is_required() . . . 662
7.126.1.4 tfm_plat_provisioning_perform() . . . 662
7.127 platform/ext/target/infineon/common/spe/provisioning/provisioning.h File Reference . . . 662
7.127.1 Enumeration Type Documentation . . . 663
7.127.1.1 ifx_dap_state_t . . . 663
7.127.2 Function Documentation . . . 663

7.127.2.1 ifx_get_dap_state() . . . 663
7.127.2.2 ifx_get_security_lifecycle() . . . 664
7.128 platform/ext/target/infineon/common/spe/services/attestation/ifx_attest_hal.c File Reference . . . 664
7.128.1 Macro Definition Documentation . . . 665
7.128.1.1 BOOL_SEED_INITIALIZED . . . 665
7.128.1.2 BOOL_SEED_UNINITIALIZED . . . 665
7.128.1.3 IFX_ATTESTATION_KEY_ID . . . 665
7.128.1.4 IFX_ATTESTATION_PROFILE_DEFINITION . . . 665
7.128.1.5 IFX_ATTESTATION_VERIFICATION_SERVICE . . . 665
7.128.2 Function Documentation . . . 665
7.128.2.1 tfm_attest_hal_get_profile_definition() . . . 665
7.128.2.2 tfm_attest_hal_get_security_lifecycle() . . . 666
7.128.2.3 tfm_attest_hal_get_verification_service() . . . 666
7.128.2.4 tfm_plat_get_boot_seed() . . . 666
7.129 platform/ext/target/infineon/common/spe/services/attestation/ifx_attest_hal_se_rt.c File Reference 667
7.129.1 Function Documentation . . . 667
7.129.1.1 tfm_attest_hal_get_token() . . . 667
7.129.1.2 tfm_attest_hal_get_token_size() . . . 668
7.130 platform/ext/target/infineon/common/spe/services/attestation/ifx_plat_device_id.c File Reference . 668
7.130.1 Macro Definition Documentation . . . 669
7.130.1.1 IFX_ADD_TO_IMPLEMENTATION_ID . . . 669
7.130.1.2 IFX_ATTESTATION_HW_VERSION . . . 669
7.130.2 Function Documentation . . . 669
7.130.2.1 tfm_plat_get_cert_ref() . . . 669
7.130.2.2 tfm_plat_get_implementation_id() . . . 669
7.131 platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_flash.c File Reference . . . 670
7.131.1 Function Documentation . . . 670
7.131.1.1 tfm_plat_builtin_key_get_desc_table_ptr() . . . 670
7.131.1.2 tfm_plat_builtin_key_get_policy_table_ptr() . . . 671
7.132 platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_rram.c File Reference . . . 671
7.132.1 Macro Definition Documentation . . . 672
7.132.1.1 MAPPED_MAILBOX_NS_AGENT_DEFAULT_CLIENT_ID . . . 672
7.132.1.2 MAPPED_TZ_NS_AGENT_DEFAULT_CLIENT_ID . . . 672
7.132.1.3 NUMBER_OF_ELEMENTS_OF . . . 672
7.132.1.4 TFM_MBOX_NS_PARTITION_ID . . . 672
7.132.1.5 TFM_TZ_NS_PARTITION_ID . . . 672
7.132.2 Function Documentation . . . 672
7.132.2.1 tfm_plat_builtin_key_get_desc_table_ptr() . . . 672
7.132.2.2 tfm_plat_builtin_key_get_policy_table_ptr() . . . 672
7.133 platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_se_rt.c File Reference . . . 672
7.133.1 Macro Definition Documentation . . . 673
7.133.1.1 NUMBER_OF_ELEMENTS_OF . . . 673

7.133.2 Function Documentation . . . 673
7.133.2.1 tfm_plat_builtin_key_get_desc_table_ptr() . . . 673
7.134 platform/ext/target/infineon/common/spe/services/crypto/crypto_nv_seed.c File Reference . . . 673
7.134.1 Function Documentation . . . 674
7.134.1.1 tfm_plat_crypto_nv_seed_read() . . . 674
7.134.1.2 tfm_plat_crypto_nv_seed_write() . . . 674
7.134.1.3 tfm_plat_crypto_provision_entropy_seed() . . . 675
7.135 platform/ext/target/infineon/common/spe/services/crypto/crypto_rnd_se_rt.c File Reference . . . 675
7.136 platform/ext/target/infineon/common/spe/services/crypto/mbedtls_accel_configs/crypto_hw_↩cryptolite_config.h File Reference . . . 675
7.137 platform/ext/target/infineon/common/spe/services/crypto/mbedtls_accel_configs/crypto_hw_↩mxcrypto_config.h File Reference . . . 675
7.138 platform/ext/target/infineon/common/spe/services/its/drivers/its_flash_driver.c File Reference . . . 675
7.139 platform/ext/target/infineon/common/spe/services/its/drivers/its_rram_driver.c File Reference . . . 676
7.140 platform/ext/target/infineon/common/spe/services/its/its_hal.c File Reference . . . 676
7.140.1 Function Documentation . . . 676
7.140.1.1 tfm_hal_its_fs_info() . . . 676
7.141 platform/ext/target/infineon/common/spe/services/mailbox/platform_hal_multi_core_cm55.c File Reference . . . 677
7.141.1 Macro Definition Documentation . . . 677
7.141.1.1 IFX_IS_REGION_IN_DTCM_MEMORY . . . 677
7.141.1.2 IFX_IS_REGION_IN_DTCM_MEMORY_REMAPPED . . . 677
7.141.1.3 IFX_IS_REGION_IN_ITCM_MEMORY . . . 678
7.141.1.4 IFX_IS_REGION_IN_ITCM_MEMORY_REMAPPED . . . 678
7.142 platform/ext/target/infineon/common/spe/services/mailbox/platform_spe_mailbox.c File Reference . 678
7.142.1 Macro Definition Documentation . . . 679
7.142.1.1 MAILBOX_CLEAN_CACHE . . . 679
7.142.1.2 MAILBOX_INVALIDATE_CACHE . . . 679
7.142.2 Function Documentation . . . 679
7.142.2.1 tfm_mailbox_hal_deinit() . . . 679
7.142.2.2 tfm_mailbox_hal_enter_critical() . . . 679
7.142.2.3 tfm_mailbox_hal_exit_critical() . . . 679
7.142.2.4 tfm_mailbox_hal_init() . . . 679
7.142.2.5 tfm_mailbox_hal_notify_peer() . . . 680
7.143 platform/ext/target/infineon/common/spe/services/mailbox/tfm_hal_multi_core.c File Reference . . 680
7.143.1 Function Documentation . . . 681
7.143.1.1 tfm_hal_wait_for_ns_cpu_ready() . . . 681
7.144 platform/ext/target/infineon/common/spe/services/mailbox/tfm_interrupts.c File Reference . . . 681
7.144.1 Function Documentation . . . 681
7.144.1.1 IFX_IPC_NS_TO_TFM_IPC_INTR() . . . 681
7.144.1.2 mailbox_irq_init() . . . 681
7.145 platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_flash_driver.c File Reference . . . 682

7.145.1 Macro Definition Documentation . . . 682
7.145.1.1 IFX_TFM_NV_COUNTERS_ERASE_VALUE . . . 682
7.146 platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_flash_driver.h File Reference . . . 682
7.146.1 Macro Definition Documentation . . . 684
7.146.1.1 IFX_NV_COUNTERS_CMSIS_FLASH_INSTANCE . . . 684
7.146.1.2 IFX_TFM_NV_COUNTERS_PROGRAM_UNIT . . . 684
7.146.1.3 IFX_TFM_NV_COUNTERS_SECTOR_SIZE . . . 684
7.146.2 Variable Documentation . . . 684
7.146.2.1 ifx_nv_counters_cmsis_flash_instance . . . 684
7.147 platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_rram_driver.c File Reference . . . 684
7.147.1 Macro Definition Documentation . . . 684
7.147.1.1 IFX_TFM_NV_COUNTERS_ERASE_VALUE . . . 684
7.148 platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_rram_driver.h File Reference . . . 685
7.148.1 Macro Definition Documentation . . . 685
7.148.1.1 IFX_NV_COUNTERS_CMSIS_FLASH_INSTANCE . . . 686
7.148.1.2 IFX_TFM_NV_COUNTERS_PROGRAM_UNIT . . . 686
7.148.1.3 IFX_TFM_NV_COUNTERS_SECTOR_SIZE . . . 686
7.148.2 Variable Documentation . . . 686
7.148.2.1 ifx_nv_counters_cmsis_flash_instance . . . 686
7.149 platform/ext/target/infineon/common/spe/services/platform/nv_counters_flash.c File Reference . . . 686
7.149.1 Macro Definition Documentation . . . 687
7.149.1.1 IFX_CHECKSUM_SIZE . . . 687
7.149.1.2 IFX_INIT_VALUE_SIZE . . . 687
7.149.1.3 IFX_NUM_NV_COUNTERS . . . 687
7.149.1.4 IFX_NV_COUNTER_SIZE . . . 687
7.149.1.5 IFX_NV_COUNTERS_CRC_INIT . . . 688
7.149.1.6 IFX_NV_COUNTERS_INITIALIZED . . . 688
7.149.1.7 IFX_TFM_NV_COUNTERS_AREA_OFFSET . . . 688
7.149.1.8 IFX_TFM_NV_COUNTERS_BACKUP_OFFSET . . . 688
7.149.2 Typedef Documentation . . . 688
7.149.2.1 ifx_nv_counters_t . . . 688
7.149.3 Function Documentation . . . 688
7.149.3.1 tfm_plat_increment_nv_counter() . . . 688
7.149.3.2 tfm_plat_init_nv_counter() . . . 689
7.149.3.3 tfm_plat_read_nv_counter() . . . 689
7.149.3.4 tfm_plat_set_nv_counter() . . . 689
7.150 platform/ext/target/infineon/common/spe/services/platform/nv_counters_rram.c File Reference . . . 690
7.150.1 Macro Definition Documentation . . . 691
7.150.1.1 IFX_NVM_INIT_DONE_FLAG . . . 691
7.150.1.2 IFX_TFM_NV_COUNTERS_BACKUP_AREA_OFFSET . . . 691

7.150.1.3 IFX_TFM_NV_COUNTERS_BACKUP_AREA_SIZE . . . 691
7.150.1.4 IFX_TFM_NV_COUNTERS_BASE . . . 691
7.150.1.5 IFX_TFM_NV_COUNTERS_COUNTER_AREA_OFFSET . . . 691
7.150.1.6 IFX_TFM_NV_COUNTERS_EXPECTED_AREA_SIZE . . . 691
7.150.1.7 IFX_TFM_NV_COUNTERS_NVM_INIT_DONE_FLAG_OFFSET . . . 692
7.150.2 Typedef Documentation . . . 692
7.150.2.1 ifx_rram_counter_value_t . . . 692
7.150.3 Function Documentation . . . 692
7.150.3.1 ifx_rram_clear_counters_and_backups() . . . 692
7.150.3.2 tfm_plat_increment_nv_counter() . . . 692
7.150.3.3 tfm_plat_init_nv_counter() . . . 692
7.150.3.4 tfm_plat_read_nv_counter() . . . 693
7.150.3.5 tfm_plat_set_nv_counter() . . . 693
7.151 platform/ext/target/infineon/common/spe/services/platform/nv_counters_se_rt.c File Reference . . . 694
7.151.1 Macro Definition Documentation . . . 694
7.151.1.1 IFX_PLAT_NV_COUNTER_NS_0 . . . 695
7.151.1.2 IFX_PLAT_NV_COUNTER_PS_0 . . . 695
7.151.1.3 IFX_PLAT_RRAM_COUNTER_NUMBER . . . 695
7.151.2 Function Documentation . . . 695
7.151.2.1 tfm_plat_increment_nv_counter() . . . 695
7.151.2.2 tfm_plat_init_nv_counter() . . . 695
7.151.2.3 tfm_plat_read_nv_counter() . . . 695
7.152 platform/ext/target/infineon/common/spe/services/platform/tfm_platform_system.c File Reference . . . 695
7.152.1 Function Documentation . . . 696
7.152.1.1 ifx_mtb_srf_handler() . . . 696
7.152.1.2 tfm_platform_hal_additional_services() . . . 697
7.152.1.3 tfm_platform_hal_ioctl() . . . 697
7.152.1.4 tfm_platform_hal_system_reset() . . . 697
7.153 platform/ext/target/infineon/common/spe/services/ps/drivers/ps_rram_driver.c File Reference . . . 698
7.154 platform/ext/target/infineon/common/spe/services/ps/drivers/ps_smif_driver.c File Reference . . . 698
7.155 platform/ext/target/infineon/common/spe/services/ps/ps_hal.c File Reference . . . 699
7.155.1 Function Documentation . . . 699
7.155.1.1 tfm_hal_ps_fs_info() . . . 699
7.156 platform/ext/target/infineon/common/spe/services/ps/ps_keys_se_rt.c File Reference . . . 700
7.156.1 Function Documentation . . . 700
7.156.1.1 tfm_platform_ps_set_key() . . . 700
7.157 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_ipc.c File Reference . . . 701
7.157.1 Function Documentation . . . 702
7.157.1.1 ifx_se_ipc_service_spm_syscall() . . . 702
7.158 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_req_mngr.c File Reference . . . 702
7.158.1 Function Documentation . . . 703

7.158.1.1 ifx_se_ipc_service_entry() . . . 703
7.158.1.2 ifx_se_ipc_service_sfn() . . . 703
7.159 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_spm.c File Reference . . . 703
7.159.1 Function Documentation . . . 704
7.159.1.1 ifx_se_ipc_service_spm_init() . . . 704
7.159.1.2 ifx_se_ipc_service_spm_shutdown() . . . 704
7.160 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_spm.h File Reference . . . 704
7.160.1 Function Documentation . . . 705
7.160.1.1 ifx_se_ipc_service_spm_init() . . . 705
7.160.1.2 ifx_se_ipc_service_spm_shutdown() . . . 706
7.161 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall.h File Reference . . . 706
7.161.1 Function Documentation . . . 707
7.161.1.1 ifx_se_ipc_service_spm_syscall() . . . 707
7.162 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall_direct.c File Reference . . . 708
7.162.1 Function Documentation . . . 708
7.162.1.1 ifx_se_syscall() . . . 708
7.163 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall_partition.c File Reference . . . 708
7.163.1 Function Documentation . . . 709
7.163.1.1 ifx_se_syscall() . . . 709
7.163.1.2 tfm_core_panic() . . . 709
7.164 platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test.c File Reference . . . 711
7.165 platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test.h File Reference . . . 712
7.166 platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test_non_secure_timer.c File Reference . . . 713
7.167 platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test_secure_timer.c File Reference 713
7.167.1 Function Documentation . . . 714
7.167.1.1 tfm_plat_test_secure_timer_clear_intr() . . . 714
7.167.1.2 tfm_plat_test_secure_timer_start() . . . 714
7.167.1.3 tfm_plat_test_secure_timer_stop() . . . 714
7.168 platform/ext/target/infineon/common/spe/services/tfm_tests/tfm_interrupts_test_s.c File Reference 715
7.169 platform/ext/target/infineon/common/spe/svc/platform_svc_api.c File Reference . . . 715
7.169.1 Function Documentation . . . 716
7.169.1.1 ifx_call_platform_enable_systick() . . . 716
7.169.1.2 ifx_call_platform_system_reset() . . . 717
7.169.1.3 ifx_call_platform_uart_log() . . . 717
7.170 platform/ext/target/infineon/common/spe/svc/platform_svc_api.h File Reference . . . 718
7.170.1 Macro Definition Documentation . . . 719
7.170.1.1 IFX_SVC_PLATFORM_ENABLE_SYSTICK . . . 719
7.170.1.2 IFX_SVC_PLATFORM_SYSTEM_RESET . . . 719

7.170.1.3 IFX_SVC_PLATFORM_UART_LOG . . . 719
7.170.2 Function Documentation . . . 719
7.170.2.1 ifx_call_platform_enable_systick() . . . 719
7.170.2.2 ifx_call_platform_system_reset() . . . 720
7.170.2.3 ifx_call_platform_uart_log() . . . 720
7.171 platform/ext/target/infineon/common/spe/svc/platform_svc_handler.c File Reference . . . 721
7.171.1 Function Documentation . . . 721
7.171.1.1 platform_svc_handlers() . . . 721
7.172 platform/ext/target/infineon/common/spe/test/ifx_fpu_s.c File Reference . . . 722
7.172.1 Function Documentation . . . 722
7.172.1.1 TFM_FPU_S_TEST_IRQ() . . . 722
7.172.1.2 tfm_test_s_fpu_init() . . . 723
7.173 platform/ext/target/infineon/common/spe/test/ifx_fpu_s.h File Reference . . . 723
7.173.1 Function Documentation . . . 723
7.173.1.1 tfm_test_s_fpu_init() . . . 723
7.174 platform/ext/target/infineon/common/spe/v80m/target_cfg.c File Reference . . . 724
7.174.1 Macro Definition Documentation . . . 725
7.174.1.1 SCB_AIRCR_WRITE_MASK . . . 725
7.174.2 Function Documentation . . . 725
7.174.2.1 FIH_RET_TYPE() . . . 725
7.174.2.2 ifx_init_debug() . . . 729
7.174.2.3 ifx_init_system_control_block() . . . 729
7.175 platform/ext/target/infineon/common/spe/v80m/target_cfg.h File Reference . . . 730
7.175.1 Function Documentation . . . 731
7.175.1.1 FIH_RET_TYPE() . . . 731
7.175.1.2 ifx_init_debug() . . . 752
7.175.1.3 ifx_init_system_control_block() . . . 752
7.176 platform/ext/target/infineon/common/spe/v80m/tfm_hal_platform.c File Reference . . . 753
7.176.1 Function Documentation . . . 753
7.176.1.1 FIH_RET_TYPE() . . . 753
7.177 platform/ext/target/infineon/common/utils/ifx_boot_shared_data.c File Reference . . . 758
7.178 platform/ext/target/infineon/common/utils/ifx_boot_shared_data.h File Reference . . . 759
7.179 platform/ext/target/infineon/common/utils/ifx_fih.h File Reference . . . 759
7.179.1 Macro Definition Documentation . . . 760
7.179.1.1 IFX_FIH_BOOL . . . 760
7.179.1.2 IFX_FIH_FALSE . . . 760
7.179.1.3 ifx_fih_to_aapcs_fih . . . 761
7.179.1.4 IFX_FIH_TRUE . . . 761
7.179.2 Typedef Documentation . . . 761
7.179.2.1 ifx_aapcs_fih_int . . . 761
7.180 platform/ext/target/infineon/common/utils/ifx_interrupt_defs.h File Reference . . . 761
7.180.1 Macro Definition Documentation . . . 761

7.180.1.1 IFX_CONCAT . . . 761
7.180.1.2 IFX_IRQ_NAME_TO_HANDLER . . . 761
7.181 platform/ext/target/infineon/common/utils/ifx_regions.c File Reference . . . 762
7.181.1 Function Documentation . . . 762
7.181.1.1 ifx_is_region_inside_other() . . . 762
7.181.1.2 ifx_is_region_overlap_other() . . . 763
7.182 platform/ext/target/infineon/common/utils/ifx_regions.h File Reference . . . 763
7.182.1 Function Documentation . . . 764
7.182.1.1 ifx_is_region_inside_other() . . . 764
7.182.1.2 ifx_is_region_overlap_other() . . . 765
7.182.1.3 REGION_DECLARE() [1/9] . . . 765
7.182.1.4 REGION_DECLARE() [2/9] . . . 766
7.182.1.5 REGION_DECLARE() [3/9] . . . 766
7.182.1.6 REGION_DECLARE() [4/9] . . . 766
7.182.1.7 REGION_DECLARE() [5/9] . . . 766
7.182.1.8 REGION_DECLARE() [6/9] . . . 766
7.182.1.9 REGION_DECLARE() [7/9] . . . 766
7.182.1.10 REGION_DECLARE() [8/9] . . . 766
7.182.1.11 REGION_DECLARE() [9/9] . . . 766
7.183 platform/ext/target/infineon/common/utils/ifx_utils.h File Reference . . . 767
7.183.1 Macro Definition Documentation . . . 767
7.183.1.1 IFX_ALIGN_UP_TO . . . 767
7.183.1.2 IFX_ASSERT . . . 767
7.183.1.3 IFX_ROUND_DOWN_TO_MULTIPLE . . . 767
7.183.1.4 IFX_ROUND_UP_TO_MULTIPLE . . . 767
7.183.1.5 IFX_UNSIGNED . . . 768
7.183.1.6 IFX_UNSIGNED_TO_VALUE . . . 768
7.183.1.7 IFX_UNUSED . . . 768
7.184 platform/ext/target/infineon/common/utils/mxs22.h File Reference . . . 768
7.184.1 Macro Definition Documentation . . . 769
7.184.1.1 IFX_IDAU_SECURE_ADDRESS_MASK . . . 769
7.184.1.2 IFX_NS_ADDRESS_ALIAS . . . 769
7.184.1.3 IFX_NS_ADDRESS_ALIAS_T . . . 769
7.184.1.4 IFX_S_ADDRESS_ALIAS . . . 769
7.184.1.5 IFX_S_ADDRESS_ALIAS_T . . . 769
7.185 platform/ext/target/infineon/common/utils/se/ifx_se_crc32.c File Reference . . . 769
7.185.1 Detailed Description . . . 770
7.185.2 Function Documentation . . . 770
7.185.2.1 ifx_se_crc32d6_close() . . . 770
7.185.2.2 ifx_se_crc32d6_open() . . . 771
7.185.2.3 ifx_se_crc32d6a() . . . 771
7.185.2.4 ifx_se_crc32d6a_update() . . . 772

7.185.2.5 ifx_se_crc32d6b() . . . 773
7.185.2.6 ifx_se_crc32d6b_update() . . . 773
7.186 platform/ext/target/infineon/common/utils/se/ifx_se_crc32.h File Reference . . . 774
7.186.1 Detailed Description . . . 775
7.186.2 Macro Definition Documentation . . . 775
7.186.2.1 IFX_CRC32_BYTE_WIDTH . . . 775
7.186.2.2 IFX_CRC32_CALC . . . 775
7.186.2.3 IFX_CRC32_CALC_APPEND . . . 775
7.186.2.4 IFX_CRC32_CRC_SIZE . . . 776
7.186.2.5 IFX_CRC32_CRC_WIDTH . . . 776
7.186.2.6 IFX_CRC32_INIT . . . 776
7.186.2.7 IFX_CRC32_WORD_WIDTH . . . 776
7.186.3 Function Documentation . . . 776
7.186.3.1 ifx_se_crc32d6_close() . . . 776
7.186.3.2 ifx_se_crc32d6_open() . . . 776
7.186.3.3 ifx_se_crc32d6a() . . . 777
7.186.3.4 ifx_se_crc32d6a_update() . . . 778
7.186.3.5 ifx_se_crc32d6b() . . . 778
7.186.3.6 ifx_se_crc32d6b_update() . . . 779
7.187 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/device/include/project_memory_layout.h File Reference . . . 779
7.187.1 Macro Definition Documentation . . . 782
7.187.1.1 IFX_CM33_NS_DATA_ADDR . . . 782
7.187.1.2 IFX_CM33_NS_DATA_LOCATION . . . 782
7.187.1.3 IFX_CM33_NS_DATA_OFFSET . . . 782
7.187.1.4 IFX_CM33_NS_DATA_SIZE . . . 782
7.187.1.5 IFX_CM33_NS_IMAGE_EXECUTE_ADDR . . . 782
7.187.1.6 IFX_CM33_NS_IMAGE_EXECUTE_LOCATION . . . 782
7.187.1.7 IFX_CM33_NS_IMAGE_EXECUTE_OFFSET . . . 782
7.187.1.8 IFX_CM33_NS_IMAGE_EXECUTE_SIZE . . . 783
7.187.1.9 IFX_CM33_NS_SHARED_MEMORY_ADDR . . . 783
7.187.1.10 IFX_CM33_NS_SHARED_MEMORY_LOCATION . . . 783
7.187.1.11 IFX_CM33_NS_SHARED_MEMORY_OFFSET . . . 783
7.187.1.12 IFX_CM33_NS_SHARED_MEMORY_SIZE . . . 783
7.187.1.13 IFX_CM55_NS_DATA_ADDR . . . 783
7.187.1.14 IFX_CM55_NS_DATA_LOCATION . . . 783
7.187.1.15 IFX_CM55_NS_DATA_OFFSET . . . 783
7.187.1.16 IFX_CM55_NS_DATA_SIZE . . . 783
7.187.1.17 IFX_CM55_NS_IMAGE_EXECUTE_ADDR . . . 784
7.187.1.18 IFX_CM55_NS_IMAGE_EXECUTE_LOCATION . . . 784
7.187.1.19 IFX_CM55_NS_IMAGE_EXECUTE_OFFSET . . . 784
7.187.1.20 IFX_CM55_NS_IMAGE_EXECUTE_SIZE . . . 784

7.187.1.21 IFX_CM55_NS_SHARED_MEMORY_ADDR . . . . . 784
7.187.1.22 IFX_CM55_NS_SHARED_MEMORY_LOCATION . . . . . 784
7.187.1.23 IFX_CM55_NS_SHARED_MEMORY_OFFSET . . . . . 784
7.187.1.24 IFX_CM55_NS_SHARED_MEMORY_SIZE . . . . . 784
7.187.1.25 IFX_FF_TEST_DRIVER_PARTITION_MMIO_ADDR . . . . . 784
7.187.1.26 IFX_FF_TEST_DRIVER_PARTITION_MMIO_LOCATION . . . . . 784
7.187.1.27 IFX_FF_TEST_DRIVER_PARTITION_MMIO_OFFSET . . . . . 785
7.187.1.28 IFX_FF_TEST_DRIVER_PARTITION_MMIO_SIZE . . . . . 785
7.187.1.29 IFX_FF_TEST_SERVER_PARTITION_MMIO_ADDR . . . . . 785
7.187.1.30 IFX_FF_TEST_SERVER_PARTITION_MMIO_LOCATION . . . . . 785
7.187.1.31 IFX_FF_TEST_SERVER_PARTITION_MMIO_OFFSET . . . . . 785
7.187.1.32 IFX_FF_TEST_SERVER_PARTITION_MMIO_SIZE . . . . . 785
7.187.1.33 IFX_RRAM_CBUS_BASE . . . . . 785
7.187.1.34 IFX_RRAM_SBUS_BASE . . . . . 785
7.187.1.35 IFX_RRAM_SIZE . . . . . 785
7.187.1.36 IFX_SOCMEM_RAM_CBUS_BASE . . . . . 786
7.187.1.37 IFX_SOCMEM_RAM_SBUS_BASE . . . . . 786
7.187.1.38 IFX_SOCMEM_RAM_SIZE . . . . . 786
7.187.1.39 IFX_SRAM0_CBUS_BASE . . . . . 786
7.187.1.40 IFX_SRAM0_SBUS_BASE . . . . . 786
7.187.1.41 IFX_SRAM0_SIZE . . . . . 786
7.187.1.42 IFX_SRAM1_CBUS_BASE . . . . . 786
7.187.1.43 IFX_SRAM1_SBUS_BASE . . . . . 786
7.187.1.44 IFX_SRAM1_SIZE . . . . . 786
7.187.1.45 IFX_TEST_AROT_PARTITION_MEMORY_ADDR . . . . . 786
7.187.1.46 IFX_TEST_AROT_PARTITION_MEMORY_LOCATION . . . . . 787
7.187.1.47 IFX_TEST_AROT_PARTITION_MEMORY_OFFSET . . . . . 787
7.187.1.48 IFX_TEST_AROT_PARTITION_MEMORY_SIZE . . . . . 787
7.187.1.49 IFX_TEST_PROT_PARTITION_MEMORY_ADDR . . . . . 787
7.187.1.50 IFX_TEST_PROT_PARTITION_MEMORY_LOCATION . . . . . 787
7.187.1.51 IFX_TEST_PROT_PARTITION_MEMORY_OFFSET . . . . . 787
7.187.1.52 IFX_TEST_PROT_PARTITION_MEMORY_SIZE . . . . . 787
7.187.1.53 IFX_TFM_BOOT_SHARED_DATA_ADDR . . . . . 787
7.187.1.54 IFX_TFM_BOOT_SHARED_DATA_LOCATION . . . . . 787
7.187.1.55 IFX_TFM_BOOT_SHARED_DATA_OFFSET . . . . . 787
7.187.1.56 IFX_TFM_BOOT_SHARED_DATA_SIZE . . . . . 788
7.187.1.57 IFX_TFM_DATA_ADDR . . . . . 788
7.187.1.58 IFX_TFM_DATA_LOCATION . . . . . 788
7.187.1.59 IFX_TFM_DATA_OFFSET . . . . . 788
7.187.1.60 IFX_TFM_DATA_SIZE . . . . . 788
7.187.1.61 IFX_TFM_ITS_ADDR . . . . . 788
7.187.1.62 IFX_TFM_ITS_LOCATION . . . . . 788

7.187.1.63 IFX_TFM_ITS_OFFSET . . . 788
7.187.1.64 IFX_TFM_ITS_SIZE . . . 788
7.187.1.65 IFX_TFM_NV_COUNTERS_AREA_ADDR . . . 788
7.187.1.66 IFX_TFM_NV_COUNTERS_AREA_LOCATION . . . 789
7.187.1.67 IFX_TFM_NV_COUNTERS_AREA_OFFSET . . . 789
7.187.1.68 IFX_TFM_NV_COUNTERS_AREA_SIZE . . . 789
7.187.1.69 IFX_TFM_PS_ADDR . . . 789
7.187.1.70 IFX_TFM_PS_LOCATION . . . 789
7.187.1.71 IFX_TFM_PS_OFFSET . . . 789
7.187.1.72 IFX_TFM_PS_SIZE . . . 789
7.187.1.73 IFX_XIP_PORT0_CBUS_BASE . . . 789
7.187.1.74 IFX_XIP_PORT0_SBUS_BASE . . . 789
7.187.1.75 IFX_XIP_PORT0_SIZE . . . 789
7.187.1.76 IFX_XIP_PORT1_CBUS_BASE . . . 790
7.187.1.77 IFX_XIP_PORT1_SBUS_BASE . . . 790
7.187.1.78 IFX_XIP_PORT1_SIZE . . . 790
7.188 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/device/include/startup_↩ pse84.h File Reference . . . 790
7.189 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/device/source/startup_↩ pse84.c File Reference . . . 791
7.189.1 Function Documentation . . . 792
7.189.1.1 __PROGRAM_START() . . . 792
7.189.1.2 Default_Handler() . . . 792
7.189.1.3 Reset_Handler() . . . 792
7.189.2 Variable Documentation . . . 792
7.189.2.1 __INITIAL_SP . . . 793
7.189.2.2 __STACK_LIMIT . . . 793
7.189.2.3 IFX_VECTOR_TABLE_ATTRIBUTE . . . 793
7.190 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/ifx_peripherals_def.h File Reference . . . 793
7.190.1 Macro Definition Documentation . . . 794
7.190.1.1 IFX_IPC_NS_TO_TFM_IRQ_PRIORITY . . . 794
7.190.1.2 IFX_IPC_TFM_TO_NS_IRQ_PRIORITY . . . 794
7.190.1.3 IFX_IRQ_TEST_FLIH_INTERRUPT . . . 794
7.190.1.4 IFX_IRQ_TEST_FLIH_INTERRUPT_GPIO_PIN . . . 794
7.190.1.5 IFX_IRQ_TEST_FLIH_INTERRUPT_GPIO_PORT . . . 794
7.190.1.6 IFX_IRQ_TEST_NS_INTERRUPT . . . 794
7.190.1.7 IFX_IRQ_TEST_NS_INTERRUPT_GPIO_PIN . . . 794
7.190.1.8 IFX_IRQ_TEST_NS_INTERRUPT_GPIO_PORT . . . 795
7.190.1.9 IFX_MXCM55 . . . 795
7.190.1.10 TFM_FPU_NS_TEST_IRQ . . . 795
7.191 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/ifx_s_peripherals_def.h File Reference . . . 795

7.191.1 Macro Definition Documentation . . . 796
7.191.1.1 IFX_LCS_BASE . . . 796
7.191.1.2 IFX_NON_SECURE_SFLASH_BASE . . . 796
7.191.1.3 IFX_RRAMC0 . . . 796
7.191.1.4 IFX_SMIF_0_MEMORY_CONFIG . . . 796
7.191.1.5 IFX_SMIF_HW . . . 796
7.191.1.6 IFX_TFM_PS_SMIF_MEMORY_CONFIG . . . 796
7.191.1.7 TFM_FPU_S_TEST_IRQ . . . 797
7.191.2 Variable Documentation . . . 797
7.191.2.1 ifx_smif_0_memory_config . . . 797
7.192 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/ifx_spm_init.c File Reference . . . 797
7.192.1 Function Documentation . . . 797
7.192.1.1 ifx_init_spm_peripherals() . . . 797
7.193 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/ifx_spm_init.h File Reference . . . 798
7.193.1 Function Documentation . . . 798
7.193.1.1 ifx_init_spm_peripherals() . . . 798
7.194 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection_regions_cfg.c File Reference . . . 799
7.194.1 Variable Documentation . . . 799
7.194.1.1 ifx_fault_sources_fault_struct0 . . . 799
7.194.1.2 ifx_fault_sources_fault_struct0_count . . . 799
7.194.1.3 ifx_msc_agc_resp_config . . . 800
7.194.1.4 ifx_msc_agc_resp_config_count . . . 800
7.194.1.5 ifx_msc_agc_resp_config_v1 . . . 800
7.194.1.6 ifx_msc_agc_resp_config_v1_count . . . 800
7.194.1.7 ifx_ppcx_region_ptrs . . . 800
7.194.1.8 ifx_ppcx_region_ptrs_count . . . 800
7.194.1.9 ifx_ppcx_static_config . . . 800
7.194.1.10 ifx_ppcx_static_config_count . . . 801
7.194.1.11 ifx_secure_interrupts_config . . . 801
7.194.1.12 ifx_secure_interrupts_config_count . . . 801
7.195 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection_regions_cfg.h File Reference . . . 801
7.195.1 Macro Definition Documentation . . . 802
7.195.1.1 CY_SAU_REGION_CNT . . . 802
7.195.1.2 IFX_SMIF_XIP0_ENABLED . . . 802
7.195.1.3 IFX_SMIF_XIP1_ENABLED . . . 802
7.195.1.4 IFX_SOCMEM_ENABLED . . . 802
7.195.2 Typedef Documentation . . . 802
7.195.2.1 ifx_sau_config_t . . . 803
7.195.3 Variable Documentation . . . 803
7.195.3.1 ifx_fault_sources_fault_struct0 . . . 803
7.195.3.2 ifx_fault_sources_fault_struct0_count . . . 803

7.195.3.3 ifx_msc_agc_resp_config . . . 803
7.195.3.4 ifx_msc_agc_resp_config_count . . . 803
7.195.3.5 ifx_msc_agc_resp_config_v1 . . . 803
7.195.3.6 ifx_msc_agc_resp_config_v1_count . . . 803
7.195.3.7 ifx_ppcx_region_ptrs . . . 803
7.195.3.8 ifx_ppcx_region_ptrs_count . . . 803
7.195.3.9 ifx_ppcx_static_config . . . 804
7.195.3.10 ifx_ppcx_static_config_count . . . 804
7.195.3.11 ifx_secure_interrupts_config . . . 804
7.195.3.12 ifx_secure_interrupts_config_count . . . 804
7.196 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/shared_ro_data.c File Reference 804
7.196.1 Detailed Description . . . 804
7.197 platform/ext/target/infineon/pse84/config/ifx_platform_spe_types.h File Reference . . . 804
7.197.1 Detailed Description . . . 806
7.197.2 Macro Definition Documentation . . . 806
7.197.2.1 IFX_GET_PC . . . 806
7.197.2.2 IFX_MPC_IS_EXTERNAL . . . 806
7.197.2.3 IFX_PROTECTION_MPU_PERIPHERAL_REGION_LIMIT . . . 806
7.197.2.4 IFX_PROTECTION_MPU_PERIPHERAL_REGION_START . . . 806
7.197.2.5 VECTOR_FIXED_EXP_NR . . . 806
7.197.3 Typedef Documentation . . . 806
7.197.3.1 ifx_mem_domain_cfg_t . . . 806
7.197.3.2 ifx_mem_domain_region_cfg_t . . . 807
7.197.3.3 ifx_pc_mask_t . . . 807
7.197.4 Variable Documentation . . . 807
7.197.4.1 ifx_memory_cm33_config . . . 807
7.197.4.2 ifx_memory_cm33_config_count . . . 807
7.197.4.3 ifx_memory_cm55_config . . . 807
7.197.4.4 ifx_memory_cm55_config_count . . . 807
7.198 platform/ext/target/infineon/pse84/config/pse84_spe_config.h File Reference . . . 807
7.198.1 Detailed Description . . . 808
7.198.2 Macro Definition Documentation . . . 808
7.198.2.1 IFX_MPC_CM55_MPC . . . 808
7.198.2.2 IFX_MPC_CONFIGURED_BY_TFM . . . 808
7.198.2.3 IFX_MPC_DRIVER_HW_MPC_WITH_ROT . . . 809
7.198.2.4 IFX_MPC_DRIVER_HW_MPC_WITHOUT_ROT . . . 809
7.198.2.5 IFX_MPC_NOT_CONTROLLED_BY_TFM . . . 809
7.198.2.6 IFX_MSC_ACG_RESP_CONFIG . . . 809
7.198.2.7 IFX_MSC_ACG_RESP_CONFIG_V1 . . . 809
7.198.2.8 IFX_MSC_TFM_CORE_BUS_MASTER_ID . . . 809
7.198.2.9 IFX_PLATFORM_MPC_PRESENT . . . 809
7.198.2.10 IFX_PLATFORM_PPC_PRESENT . . . 809

7.198.2.11 IFX_REGION_MAX_MPC_COUNT . . . 809
7.198.2.12 IFX_SE_RT_RRAM_BLOCK_SIZE . . . 809
7.198.2.13 IFX_SE_RT_RRAM_MPC_DEFAULT_ATTRIBUTES . . . 810
7.199 platform/ext/target/infineon/pse84/config_tfm_target.h File Reference . . . 810
7.199.1 Macro Definition Documentation . . . 810
7.199.1.1 CRYPTO_STACK_SIZE . . . 810
7.199.1.2 IFX_MEMORY_CONFIGURATOR_MPC_CONFIG . . . 810
7.199.1.3 IFX_SE_IPC_SERVICE_STACK_SIZE . . . 811
7.199.1.4 ITS_BUF_SIZE . . . 811
7.199.1.5 PLATFORM_SERVICE_INPUT_BUFFER_SIZE . . . 811
7.199.1.6 PS_NUM_ASSETS . . . 811
7.199.1.7 PS_STACK_SIZE . . . 811
7.200 platform/ext/target/infineon/pse84/epc2/board/KIT_PSOCE84_EVK/config_bsp.h File Reference . . . 811
7.200.1 Macro Definition Documentation . . . 811
7.200.1.1 IFX_STARTUP_HEADER_FILE . . . 811
7.201 platform/ext/target/infineon/pse84/epc2/ifx_platform_config.h File Reference . . . 812
7.201.1 Macro Definition Documentation . . . 812
7.201.1.1 CONFIG_TFM_SCHEDULE_WHEN_NS_INTERRUPTED . . . 812
7.201.1.2 CONFIG_TFM_SECURE_THREAD_MASK_NS_INTERRUPT . . . 812
7.201.1.3 IFX_PSE84_EPC2 . . . 812
7.201.1.4 PS_DEFAULT_CRYPTO_CONFIG . . . 812
7.202 platform/ext/target/infineon/pse84/epc2/ifx_platform_spe_config.h File Reference . . . 813
7.202.1 Detailed Description . . . 813
7.202.2 Macro Definition Documentation . . . 814
7.202.2.1 IFX_APP_ROT_PPC_DYNAMIC_ISOLATION . . . 814
7.202.2.2 IFX_APP_ROT_PRIVILEGED . . . 814
7.202.2.3 IFX_MULTIPLE_IAK_KEY_TYPES . . . 814
7.202.2.4 IFX_NS_AGENT_TZ_PRIVILEGED . . . 814
7.202.2.5 IFX_PC_CM33_NSPE . . . 814
7.202.2.6 IFX_PC_CM33_NSPE_ID . . . 814
7.202.2.7 IFX_PC_CM55_NSPE . . . 814
7.202.2.8 IFX_PC_CM55_NSPE_ID . . . 814
7.202.2.9 IFX_PC_DEBUGGER . . . 814
7.202.2.10 IFX_PC_DEBUGGER_ID . . . 815
7.202.2.11 IFX_PC_DEFAULT . . . 815
7.202.2.12 IFX_PC_MAILBOX_NSPE . . . 815
7.202.2.13 IFX_PC_MAILBOX_NSPE_ID . . . 815
7.202.2.14 IFX_PC_NONE . . . 815
7.202.2.15 IFX_PC_SE_RT . . . 815
7.202.2.16 IFX_PC_SE_RT_ID . . . 815
7.202.2.17 IFX_PC_TFM_AROT . . . 815
7.202.2.18 IFX_PC_TFM_AROT_ID . . . 815

7.202.2.19 IFX_PC_TFM_PROT . . . 815
7.202.2.20 IFX_PC_TFM_PROT_ID . . . 816
7.202.2.21 IFX_PC_TFM_SPM . . . 816
7.202.2.22 IFX_PC_TFM_SPM_ID . . . 816
7.202.2.23 IFX_PC_TZ_NSPE . . . 816
7.202.2.24 IFX_PC_TZ_NSPE_ID . . . 816
7.202.2.25 IFX_PSA_ROT_PRIVILEGED . . . 816
7.203 platform/ext/target/infineon/pse84/epc2/spe/services/crypto/crypto_keys_rram.h File Reference . . . 816
7.203.1 Macro Definition Documentation . . . 817
7.203.1.1 IFX_CRYPTO_KEYS_HUK_ALGORITHM . . . 817
7.203.1.2 IFX_CRYPTO_KEYS_HUK_KEY_BITS . . . 818
7.203.1.3 IFX_CRYPTO_KEYS_HUK_KEY_LEN . . . 818
7.203.1.4 IFX_CRYPTO_KEYS_HUK_TYPE . . . 818
7.203.1.5 IFX_CRYPTO_KEYS_IAK_PRIVATE_ALGORITHM . . . 818
7.203.1.6 IFX_CRYPTO_KEYS_IAK_PRIVATE_KEY_BITS . . . 818
7.203.1.7 IFX_CRYPTO_KEYS_IAK_PRIVATE_KEY_LEN . . . 818
7.203.1.8 IFX_CRYPTO_KEYS_IAK_PRIVATE_TYPE . . . 818
7.203.1.9 IFX_CRYPTO_KEYS_IAK_PUBLIC_ALGORITHM . . . 818
7.203.1.10 IFX_CRYPTO_KEYS_IAK_PUBLIC_KEY_BITS . . . 818
7.203.1.11 IFX_CRYPTO_KEYS_IAK_PUBLIC_KEY_LEN . . . 818
7.203.1.12 IFX_CRYPTO_KEYS_IAK_PUBLIC_TYPE . . . 819
7.203.1.13 IFX_CRYPTO_KEYS_PTR . . . 819
7.204 platform/ext/target/infineon/pse84/epc2/spe/services/crypto/mbedtls_target_config_pse84.h File Reference . . . 819
7.204.1 Macro Definition Documentation . . . 819
7.204.1.1 MBEDTLS_PSA_ASSUME_EXCLUSIVE_BUFFERS . . . 819
7.205 platform/ext/target/infineon/pse84/epc2/spe/services/crypto/platform_builtin_key_loader_ids.h File Reference . . . 819
7.205.1 Macro Definition Documentation . . . 820
7.205.1.1 TFM_BUILTIN_KEY_SLOT_IAK . . . 820
7.205.1.2 TFM_BUILTIN_MAX_KEY_LEN . . . 820
7.205.2 Enumeration Type Documentation . . . 820
7.205.2.1 psa_drv_slot_number_t . . . 820
7.206 platform/ext/target/infineon/pse84/epc2/tests/ifx_isolation_test_data.c File Reference . . . 820
7.206.1 Variable Documentation . . . 821
7.206.1.1 num_of_cases . . . 821
7.206.1.2 test_cases . . . 821
7.207 platform/ext/target/infineon/pse84/epc2/tests/ifx_platform_tests_config.h File Reference . . . 821
7.207.1 Macro Definition Documentation . . . 821
7.207.1.1 IFX_MAILBOX_DTCM_ADDRESS_1 . . . 821
7.207.1.2 IFX_MAILBOX_DTCM_ADDRESS_2 . . . 821
7.207.1.3 IFX_MAILBOX_ITCM_ADDRESS_1 . . . 821
7.207.1.4 IFX_MAILBOX_ITCM_ADDRESS_2 . . . 821

7.207.1.5 IFX_PC_TZ_NSPE_ID . . . 822
7.208 platform/ext/target/infineon/pse84/shared/device/config/device_cfg.h File Reference . . . 822
7.208.1 Detailed Description . . . 822
7.208.2 Macro Definition Documentation . . . 822
7.208.2.1 IFX_IRQ_TEST_TIMER_S . . . 822
7.209 platform/ext/target/infineon/pse84/shared/device/include/cmsis.h File Reference . . . 822
7.210 platform/ext/target/infineon/pse84/shared/device/include/pse84_core_interrupts.h File Reference . . . 823
7.210.1 Macro Definition Documentation . . . 824
7.210.1.1 IFX_CORE_DEFINE_EXCEPTIONS_LIST . . . 824
7.210.1.2 IFX_CORE_DEFINE_INTERRUPTS_LIST . . . 824
7.210.1.3 IFX_CORE_EXCEPTIONS_LIST . . . 824
7.210.1.4 IFX_CORE_INTERRUPTS_LIST . . . 824
7.211 platform/ext/target/infineon/pse84/shared/partition/flash_layout.h File Reference . . . 824
7.211.1 Macro Definition Documentation . . . 825
7.211.1.1 FLASH_LAYOUT_H . . . 825
7.212 platform/ext/target/infineon/pse84/shared/partition/pse84_s_linker_alignments.h File Reference . . . 825
7.212.1 Macro Definition Documentation . . . 825
7.212.1.1 IFX_LINKER_CODE_ALIGNMENT . . . 826
7.212.1.2 IFX_LINKER_DATA_ALIGNMENT . . . 826
7.212.1.3 IFX_LINKER_RRAM_ALIGNMENT . . . 826
7.212.1.4 IFX_LINKER_SMIF_ALIGNMENT . . . 826
7.212.1.5 IFX_LINKER_SOCMEM_ALIGNMENT . . . 826
7.212.1.6 IFX_LINKER_SRAM_ALIGNMENT . . . 826
7.212.1.7 TFM_LINKER_APP_ROT_LINKER_DATA_ALIGNMENT . . . 826
7.212.1.8 TFM_LINKER_IFX_CODE_COVERAGE_ALIGNMENT . . . 826
7.212.1.9 TFM_LINKER_NS_AGENT_TZ_CODE_ALIGNMENT . . . 826
7.212.1.10 TFM_LINKER_PSA_ROT_LINKER_DATA_ALIGNMENT . . . 826
7.212.1.11 TFM_LINKER_SP_META_PTR_ALIGNMENT . . . 827
7.212.1.12 TFM_LINKER_UNPRIV_CODE_ALIGNMENT . . . 827
7.212.1.13 TFM_LINKER_VENEERS_ALIGNMENT . . . 827
7.213 platform/ext/target/infineon/pse84/shared/partition/region_defs.h File Reference . . . 827
7.213.1 Macro Definition Documentation . . . 828
7.213.1.1 BOOT_TFM_SHARED_DATA_BASE . . . 828
7.213.1.2 BOOT_TFM_SHARED_DATA_LIMIT . . . 828
7.213.1.3 BOOT_TFM_SHARED_DATA_SIZE . . . 828
7.213.1.4 IFX_CM33_NS_MSP_STACK_SIZE . . . 828
7.213.1.5 IFX_CM55_NS_MSP_STACK_SIZE . . . 828
7.213.1.6 IFX_OFF_CORE_NSPE_BOOT_ADDR . . . 828
7.213.1.7 S_CODE_LIMIT . . . 828
7.213.1.8 S_CODE_SIZE . . . 828
7.213.1.9 S_CODE_START . . . 829
7.213.1.10 S_CODE_VECTOR_TABLE_SIZE . . . 829

7.213.1.11 S_DATA_LIMIT . . . 829
7.213.1.12 S_DATA_SIZE . . . 829
7.213.1.13 S_DATA_START . . . 829
7.213.1.14 S_MSP_STACK_SIZE . . . 829
7.214 platform/ext/target/infineon/pse84/shared/platform_nv_counters_ids.h File Reference . . . 829
7.214.1 Enumeration Type Documentation . . . 830
7.214.1.1 tfm_nv_counter_t . . . 830
7.215 platform/ext/target/infineon/pse84/shared/tfm_peripherals_def.h File Reference . . . 830
7.215.1 Macro Definition Documentation . . . 831
7.215.1.1 DEFAULT_IRQ_PRIORITY . . . 831
7.215.1.2 IFX_IPC_MAILBOX_LOCK_CHAN . . . 831
7.215.1.3 IFX_IPC_NS_TO_TFM_CHAN . . . 831
7.215.1.4 IFX_IPC_NS_TO_TFM_INTR_MASK . . . 831
7.215.1.5 IFX_IPC_NS_TO_TFM_INTR_STRUCT . . . 832
7.215.1.6 IFX_IPC_NS_TO_TFM_IPC_INTR . . . 832
7.215.1.7 IFX_IPC_NS_TO_TFM_IRQ_PRIORITY . . . 832
7.215.1.8 IFX_IPC_NS_TO_TFM_NOTIFY_MASK . . . 832
7.215.1.9 IFX_IPC_TFM_TO_NS_CHAN . . . 832
7.215.1.10 IFX_IPC_TFM_TO_NS_INTR_MASK . . . 832
7.215.1.11 IFX_IPC_TFM_TO_NS_INTR_STRUCT . . . 832
7.215.1.12 IFX_IPC_TFM_TO_NS_IPC_INTR . . . 832
7.215.1.13 IFX_IPC_TFM_TO_NS_IRQ_PRIORITY . . . 832
7.215.1.14 IFX_IPC_TFM_TO_NS_NOTIFY_MASK . . . 832
7.215.1.15 IFX_TEST_PERIPHERAL_1 . . . 833
7.215.1.16 IFX_TEST_PERIPHERAL_1_BASE . . . 833
7.215.1.17 IFX_TEST_PERIPHERAL_1_SIZE . . . 833
7.215.1.18 IFX_TEST_PERIPHERAL_2 . . . 833
7.215.1.19 IFX_TEST_PERIPHERAL_2_BASE . . . 833
7.215.1.20 IFX_TEST_PERIPHERAL_2_SIZE . . . 833
7.215.1.21 MAILBOX_IRQ . . . 833
7.216 platform/ext/target/infineon/pse84/spe/platform_otp_ids.h File Reference . . . 833
7.216.1 Enumeration Type Documentation . . . 834
7.216.1.1 tfm_otp_element_id_t . . . 834
7.217 platform/ext/target/infineon/pse84/spe/protection/platform_partition_assets.h File Reference . . . 834
7.218 platform/ext/target/infineon/pse84/spe/protection/protection_data.c File Reference . . . 835
7.218.1 Function Documentation . . . 836
7.218.1.1 FIH_RET() . . . 836
7.218.1.2 FIH_RET_TYPE() . . . 836
7.218.1.3 if() . . . 836
7.218.2 Variable Documentation . . . 836
7.218.2.1 attr . . . 836
7.218.2.2 else . . . 836

7.218.2.3 ifx_memory_cm33_config . . . 836
7.218.2.4 ifx_memory_cm33_config_count . . . 836
7.218.2.5 ifx_memory_cm55_config . . . 837
7.218.2.6 ifx_memory_cm55_config_count . . . 837
7.218.2.7 limit . . . 837
7.218.2.8 p_asset . . . 837
7.218.2.9 start . . . 837
7.218.2.10 valid . . . 837
7.219 platform/ext/target/infineon/pse84/spe/provisioning/ifx_platform_provisioning.c File Reference . . . 837
7.219.1 Macro Definition Documentation . . . 838
7.219.1.1 IFX_DEBUG_POLICY_AP_ALLOW_CERT . . . 838
7.219.1.2 IFX_DEBUG_POLICY_AP_ALLOW_FW . . . 838
7.219.1.3 IFX_DEBUG_POLICY_AP_ALLOW_OPEN . . . 839
7.219.1.4 IFX_DEBUG_POLICY_AP_DISABLE . . . 839
7.219.1.5 IFX_DEBUG_POLICY_AP_ENABLE . . . 839
7.219.1.6 IFX_DEBUG_POLICY_CM33_AP_CTL_Msk . . . 839
7.219.1.7 IFX_DEBUG_POLICY_CM33_AP_CTL_Pos . . . 839
7.219.1.8 IFX_DEBUG_POLICY_CM55_AP_CTL_Msk . . . 839
7.219.1.9 IFX_DEBUG_POLICY_CM55_AP_CTL_Pos . . . 839
7.219.1.10 IFX_DEBUG_POLICY_SYS_AP_CTL_Msk . . . 839
7.219.1.11 IFX_DEBUG_POLICY_SYS_AP_CTL_Pos . . . 839
7.219.1.12 IFX_OEM_POLICY_ADDR . . . 839
7.219.1.13 IFX_OEM_POLICY_PUB_KEY_SIZE . . . 840
7.219.2 Function Documentation . . . 840
7.219.2.1 ifx_get_dap_state() . . . 840
7.220 platform/ext/target/infineon/pse84/tests/secure/s_test.c File Reference . . . 840
7.220.1 Function Documentation . . . 840
7.220.1.1 register_testsuite_extra_s_interface() . . . 841
7.221 secure_fw/include/array.h File Reference . . . 841
7.221.1 Macro Definition Documentation . . . 841
7.221.1.1 ARRAY_SIZE . . . 841
7.221.1.2 IOVEC_LEN . . . 841
7.222 secure_fw/include/async.h File Reference . . . 841
7.222.1 Macro Definition Documentation . . . 842
7.222.1.1 ASYNC_MSG_REPLY . . . 842
7.223 secure_fw/include/build_config_check.h File Reference . . . 842
7.224 secure_fw/include/compiler_ext_defs.h File Reference . . . 843
7.224.1 Macro Definition Documentation . . . 843
7.224.1.1 SYNTAX_UNIFIED . . . 843
7.225 secure_fw/include/config_tfm.h File Reference . . . 844
7.226 secure_fw/include/crt_impl_private.h File Reference . . . 844
7.226.1 Macro Definition Documentation . . . 845

7.226.1.1 ADDR_WORD_UNALIGNED . . . 845
7.227 secure_fw/include/security_defs.h File Reference . . . 845
7.227.1 Macro Definition Documentation . . . 845
7.227.1.1 STACK_SEAL_PATTERN . . . 845
7.228 secure_fw/partitions/crypto/config_crypto_check.h File Reference . . . 845
7.229 secure_fw/partitions/crypto/config_engine_buf.h File Reference . . . 847
7.230 secure_fw/partitions/crypto/crypto_aead.c File Reference . . . 847
7.231 secure_fw/partitions/crypto/crypto_alloc.c File Reference . . . 847
7.231.1 Macro Definition Documentation . . . 848
7.231.1.1 TFM_CRYPTO_INVALID_HANDLE . . . 848
7.231.2 Enumeration Type Documentation . . . 849
7.231.2.1 anonymous enum . . . 849
7.232 secure_fw/partitions/crypto/crypto_asymmetric.c File Reference . . . 849
7.233 secure_fw/partitions/crypto/crypto_check_config.h File Reference . . . 849
7.234 secure_fw/partitions/crypto/crypto_cipher.c File Reference . . . 850
7.235 secure_fw/partitions/crypto/crypto_hash.c File Reference . . . 851
7.236 secure_fw/partitions/crypto/crypto_init.c File Reference . . . 851
7.236.1 Macro Definition Documentation . . . 852
7.236.1.1 ALIGN . . . 852
7.236.1.2 TFM_CRYPTO_IOVEC_ALIGNMENT . . . 853
7.236.2 Function Documentation . . . 853
7.236.2.1 tfm_crypto_api_dispatcher() . . . 853
7.236.2.2 tfm_crypto_get_caller_id() . . . 854
7.236.2.3 tfm_crypto_init() . . . 855
7.236.2.4 tfm_crypto_sfn() . . . 855
7.237 secure_fw/partitions/crypto/crypto_key_derivation.c File Reference . . . 855
7.238 secure_fw/partitions/crypto/crypto_key_management.c File Reference . . . 855
7.239 secure_fw/partitions/crypto/crypto_library.c File Reference . . . 856
7.240 secure_fw/partitions/crypto/crypto_library.h File Reference . . . 857
7.240.1 Detailed Description . . . 858
7.240.2 Macro Definition Documentation . . . 858
7.240.2.1 CRYPTO_LIBRARY_GET_KEY_ID . . . 858
7.240.2.2 CRYPTO_LIBRARY_GET_OWNER . . . 858
7.240.3 Typedef Documentation . . . 858
7.240.3.1 tfm_crypto_library_key_id_t . . . 858
7.241 secure_fw/partitions/crypto/crypto_mac.c File Reference . . . 858
7.242 secure_fw/partitions/crypto/crypto_rng.c File Reference . . . 859
7.243 secure_fw/partitions/crypto/crypto_spe.h File Reference . . . 859
7.243.1 Detailed Description . . . 861
7.243.2 Macro Definition Documentation . . . 861
7.243.2.1 psa_aead_abort . . . 861
7.243.2.2 psa_aead_decrypt . . . 861

7.243.2.3 psa_aead_decrypt_setup . . . 861
7.243.2.4 psa_aead_encrypt . . . 862
7.243.2.5 psa_aead_encrypt_setup . . . 862
7.243.2.6 psa_aead_finish . . . 862
7.243.2.7 psa_aead_generate_nonce . . . 862
7.243.2.8 psa_aead_set_lengths . . . 862
7.243.2.9 psa_aead_set_nonce . . . 862
7.243.2.10 psa_aead_update . . . 862
7.243.2.11 psa_aead_update_ad . . . 862
7.243.2.12 psa_aead_verify . . . 862
7.243.2.13 psa_asymmetric_decrypt . . . 862
7.243.2.14 psa_asymmetric_encrypt . . . 863
7.243.2.15 psa_cipher_abort . . . 863
7.243.2.16 psa_cipher_decrypt . . . 863
7.243.2.17 psa_cipher_decrypt_setup . . . 863
7.243.2.18 psa_cipher_encrypt . . . 863
7.243.2.19 psa_cipher_encrypt_setup . . . 863
7.243.2.20 psa_cipher_finish . . . 863
7.243.2.21 psa_cipher_generate_iv . . . 863
7.243.2.22 psa_cipher_operation_init . . . 863
7.243.2.23 psa_cipher_set_iv . . . 863
7.243.2.24 psa_cipher_update . . . 864
7.243.2.25 psa_close_key . . . 864
7.243.2.26 psa_copy_key . . . 864
7.243.2.27 psa_crypto_init . . . 864
7.243.2.28 psa_destroy_key . . . 864
7.243.2.29 psa_export_key . . . 864
7.243.2.30 psa_export_public_key . . . 864
7.243.2.31 PSA_FUNCTION_NAME . . . 864
7.243.2.32 psa_generate_key . . . 864
7.243.2.33 psa_generate_random . . . 864
7.243.2.34 psa_get_key_attributes . . . 865
7.243.2.35 psa_hash_abort . . . 865
7.243.2.36 psa_hash_clone . . . 865
7.243.2.37 psa_hash_compare . . . 865
7.243.2.38 psa_hash_compute . . . 865
7.243.2.39 psa_hash_finish . . . 865
7.243.2.40 psa_hash_operation_init . . . 865
7.243.2.41 psa_hash_setup . . . 865
7.243.2.42 psa_hash_update . . . 865
7.243.2.43 psa_hash_verify . . . 865
7.243.2.44 psa_import_key . . . 866

7.243.2.45 psa_key_derivation_abort . . . 866
7.243.2.46 psa_key_derivation_get_capacity . . . 866
7.243.2.47 psa_key_derivation_input_bytes . . . 866
7.243.2.48 psa_key_derivation_input_integer . . . 866
7.243.2.49 psa_key_derivation_input_key . . . 866
7.243.2.50 psa_key_derivation_key_agreement . . . 866
7.243.2.51 psa_key_derivation_output_bytes . . . 866
7.243.2.52 psa_key_derivation_output_key . . . 866
7.243.2.53 psa_key_derivation_set_capacity . . . 866
7.243.2.54 psa_key_derivation_setup . . . 867
7.243.2.55 psa_mac_abort . . . 867
7.243.2.56 psa_mac_compute . . . 867
7.243.2.57 psa_mac_operation_init . . . 867
7.243.2.58 psa_mac_sign_finish . . . 867
7.243.2.59 psa_mac_sign_setup . . . 867
7.243.2.60 psa_mac_update . . . 867
7.243.2.61 psa_mac_verify . . . 867
7.243.2.62 psa_mac_verify_finish . . . 867
7.243.2.63 psa_mac_verify_setup . . . 867
7.243.2.64 psa_open_key . . . 868
7.243.2.65 psa_purge_key . . . 868
7.243.2.66 psa_raw_key_agreement . . . 868
7.243.2.67 psa_reset_key_attributes . . . 868
7.243.2.68 psa_sign_hash . . . 868
7.243.2.69 psa_sign_message . . . 868
7.243.2.70 psa_verify_hash . . . 868
7.243.2.71 psa_verify_message . . . 868
7.244 secure_fw/partitions/crypto/dir_crypto.dox File Reference . . . 868
7.245 secure_fw/partitions/crypto/psa_driver_api/tfm_builtin_key_loader.c File Reference . . . 868
7.245.1 Macro Definition Documentation . . . 869
7.245.1.1 NUMBER_OF_ELEMENTS_OF . . . 869
7.245.1.2 TFM_BUILTIN_MAX_KEY_LEN . . . 869
7.245.1.3 TFM_BUILTIN_MAX_KEYS . . . 869
7.246 secure_fw/partitions/crypto/psa_driver_api/tfm_builtin_key_loader.h File Reference . . . 870
7.246.1 Macro Definition Documentation . . . 871
7.246.1.1 PSA_CRYPTO_DRIVER_TFM_BUILTIN_KEY_LOADER . . . 871
7.246.1.2 TFM_BUILTIN_KEY_LOADER_KEY_LOCATION . . . 871
7.246.1.3 TFM_BUILTIN_KEY_LOADER_LIFETIME . . . 871
7.247 secure_fw/partitions/crypto/tfm_crypto_api.h File Reference . . . 871
7.247.1 Enumeration Type Documentation . . . 873
7.247.1.1 tfm_crypto_operation_type . . . 873
7.247.2 Function Documentation . . . 873

7.247.2.1 tfm_crypto_api_dispatcher() . . . 873
7.247.2.2 tfm_crypto_get_caller_id() . . . 874
7.247.2.3 tfm_crypto_init() . . . 875
7.248 secure_fw/partitions/crypto/tfm_crypto_key.h File Reference . . . 875
7.248.1 Macro Definition Documentation . . . 876
7.248.1.1 TFM_CRYPTO_KEY_ID_S_INIT . . . 876
7.249 secure_fw/partitions/crypto/tfm_mbedcrypto_alt.c File Reference . . . 876
7.250 secure_fw/partitions/crypto/tfm_mbedcrypto_include.h File Reference . . . 876
7.250.1 Macro Definition Documentation . . . 877
7.250.1.1 PSA_CRYPTO_SECURE . . . 877
7.251 secure_fw/partitions/dir_services.dox File Reference . . . 877
7.252 secure_fw/partitions/firmware_update/bootloader/mcuboot/tfm_mcuboot_fwu.c File Reference . . . 877
7.252.1 Macro Definition Documentation . . . 878
7.252.1.1 MAX_IMAGE_INFO_LENGTH . . . 879
7.252.2 Typedef Documentation . . . 879
7.252.2.1 fwu_image_info_data_t . . . 879
7.252.2.2 tfm_fwu_mcuboot_ctx_t . . . 879
7.252.3 Function Documentation . . . 879
7.252.3.1 fwu_bootloader_abort() . . . 879
7.252.3.2 fwu_bootloader_clean_component() . . . 879
7.252.3.3 fwu_bootloader_get_image_info() . . . 879
7.252.3.4 fwu_bootloader_init() . . . 880
7.252.3.5 fwu_bootloader_install_image() . . . 880
7.252.3.6 fwu_bootloader_load_image() . . . 881
7.252.3.7 fwu_bootloader_mark_image_accepted() . . . 881
7.252.3.8 fwu_bootloader_reject_staged_image() . . . 882
7.252.3.9 fwu_bootloader_reject_trial_image() . . . 882
7.252.3.10 fwu_bootloader_staging_area_init() . . . 882
7.253 secure_fw/partitions/firmware_update/bootloader/tfm_bootloader_fwu_abstraction.h File Reference 883
7.253.1 Function Documentation . . . 884
7.253.1.1 fwu_bootloader_clean_component() . . . 884
7.253.1.2 fwu_bootloader_get_image_info() . . . 884
7.253.1.3 fwu_bootloader_init() . . . 885
7.253.1.4 fwu_bootloader_install_image() . . . 885
7.253.1.5 fwu_bootloader_load_image() . . . 885
7.253.1.6 fwu_bootloader_mark_image_accepted() . . . 886
7.253.1.7 fwu_bootloader_reject_staged_image() . . . 886
7.253.1.8 fwu_bootloader_reject_trial_image() . . . 887
7.253.1.9 fwu_bootloader_staging_area_init() . . . 887
7.254 secure_fw/partitions/firmware_update/tfm_fwu_req_mngr.c File Reference . . . 887
7.254.1 Macro Definition Documentation . . . 888
7.254.1.1 COMPONENTS_ITER . . . 888

7.254.2 Typedef Documentation . . . 888
7.254.2.1 tfm_fwu_ctx_t . . . 888
7.254.3 Function Documentation . . . 888
7.254.3.1 tfm_firmware_update_service_sfn() . . . 888
7.254.3.2 tfm_fwu_entry() . . . 889
7.254.3.3 tfm_fwu_reject() . . . 889
7.255 secure_fw/partitions/idle_partition/idle_partition.c File Reference . . . 889
7.255.1 Function Documentation . . . 890
7.255.1.1 tfm_idle_thread() . . . 890
7.256 secure_fw/partitions/idle_partition/load_info_idle_sp.c File Reference . . . 890
7.256.1 Macro Definition Documentation . . . 891
7.256.1.1 IDLE_SP_STACK_SIZE . . . 891
7.256.2 Function Documentation . . . 891
7.256.2.1 tfm_idle_thread() . . . 891
7.256.3 Variable Documentation . . . 892
7.256.3.1 idle_sp_stack . . . 892
7.256.3.2 tfm_sp_idle_load . . . 892
7.257 secure_fw/partitions/initial_attestation/attest.h File Reference . . . 892
7.257.1 Enumeration Type Documentation . . . 894
7.257.1.1 psa_attest_err_t . . . 894
7.257.2 Function Documentation . . . 894
7.257.2.1 attest_get_boot_data() . . . 894
7.257.2.2 attest_get_caller_client_id() . . . 895
7.257.2.3 attest_init() . . . 895
7.257.2.4 initial_attest_get_token() . . . 896
7.257.2.5 initial_attest_get_token_size() . . . 896
7.258 secure_fw/partitions/initial_attestation/attest_asymmetric_key.c File Reference . . . 897
7.258.1 Macro Definition Documentation . . . 897
7.258.1.1 ATTEST_ECC_PUBLIC_KEY_SIZE . . . 898
7.258.1.2 ECC_P256_COORD_SIZE . . . 898
7.258.2 Function Documentation . . . 898
7.258.2.1 attest_get_instance_id() . . . 898
7.259 secure_fw/partitions/initial_attestation/attest_boot_data.c File Reference . . . 898
7.259.1 Macro Definition Documentation . . . 899
7.259.1.1 MAX_BOOT_STATUS . . . 899
7.259.2 Function Documentation . . . 899
7.259.2.1 attest_boot_data_init() . . . 899
7.259.2.2 attest_encode_sw_components_array() . . . 900
7.260 secure_fw/partitions/initial_attestation/attest_boot_data.h File Reference . . . 900
7.260.1 Function Documentation . . . 902
7.260.1.1 attest_boot_data_init() . . . 902
7.260.1.2 attest_encode_sw_components_array() . . . 902

7.261 secure_fw/partitions/initial_attestation/attest_core.c File Reference . . . 903
7.261.1 Macro Definition Documentation . . . 903
7.261.1.1 ARRAY_LENGTH . . . 904
7.261.2 Function Documentation . . . 904
7.261.2.1 attest_init() . . . 904
7.261.2.2 initial_attest_get_token() . . . 904
7.261.2.3 initial_attest_get_token_size() . . . 905
7.262 secure_fw/partitions/initial_attestation/attest_key.h File Reference . . . 905
7.262.1 Function Documentation . . . 906
7.262.1.1 attest_get_instance_id() . . . 906
7.263 secure_fw/partitions/initial_attestation/attest_symmetric_key.c File Reference . . . 906
7.263.1 Macro Definition Documentation . . . 907
7.263.1.1 INSTANCE_ID_HASH_ALG . . . 907
7.263.1.2 KID_BUF_LEN . . . 907
7.263.1.3 SYMMETRIC_IAK_MAX_SIZE . . . 907
7.263.2 Function Documentation . . . 907
7.263.2.1 attest_get_instance_id() . . . 907
7.264 secure_fw/partitions/initial_attestation/attest_token.h File Reference . . . 908
7.264.1 Detailed Description . . . 910
7.264.2 Macro Definition Documentation . . . 910
7.264.2.1 TOKEN_OPT_OMIT_CLAIMS . . . 910
7.264.2.2 TOKEN_OPT_SHORT_CIRCUIT_SIGN . . . 910
7.264.3 Enumeration Type Documentation . . . 910
7.264.3.1 attest_token_err_t . . . 910
7.264.4 Function Documentation . . . 911
7.264.4.1 attest_token_encode_add_bstr() . . . 911
7.264.4.2 attest_token_encode_add_cbor() . . . 911
7.264.4.3 attest_token_encode_add_integer() . . . 912
7.264.4.4 attest_token_encode_add_tstr() . . . 912
7.264.4.5 attest_token_encode_borrow_cbor_cntxt() . . . 912
7.264.4.6 attest_token_encode_finish() . . . 913
7.264.4.7 attest_token_encode_start() . . . 913
7.265 secure_fw/partitions/initial_attestation/attest_token_encode.c File Reference . . . 914
7.265.1 Detailed Description . . . 915
7.265.2 Function Documentation . . . 915
7.265.2.1 attest_token_encode_add_bstr() . . . 915
7.265.2.2 attest_token_encode_add_cbor() . . . 915
7.265.2.3 attest_token_encode_add_integer() . . . 915
7.265.2.4 attest_token_encode_add_tstr() . . . 916
7.265.2.5 attest_token_encode_borrow_cbor_cntxt() . . . 916
7.265.2.6 attest_token_encode_finish() . . . 916
7.265.2.7 attest_token_encode_start() . . . 917

7.266 secure_fw/partitions/initial_attestation/dir_initial_attestation.dox File Reference . . . 917
7.267 secure_fw/partitions/initial_attestation/tfm_attest.c File Reference . . . 917
7.267.1 Function Documentation . . . 918
7.267.1.1 attest_get_boot_data() . . . 918
7.267.1.2 attest_get_caller_client_id() . . . 919
7.267.2 Variable Documentation . . . 919
7.267.2.1 g_attest_caller_id . . . 919
7.268 secure_fw/partitions/initial_attestation/tfm_attest_req_mngr.c File Reference . . . 919
7.268.1 Macro Definition Documentation . . . 920
7.268.1.1 ECC_P256_PUBLIC_KEY_SIZE . . . 920
7.268.2 Typedef Documentation . . . 920
7.268.2.1 attest_func_t . . . 920
7.268.3 Function Documentation . . . 921
7.268.3.1 attest_partition_init() . . . 921
7.268.3.2 tfm_attestation_service_sfn() . . . 921
7.268.4 Variable Documentation . . . 921
7.268.4.1 g_attest_caller_id . . . 921
7.269 secure_fw/partitions/internal_trusted_storage/flash/its_flash.c File Reference . . . 921
7.270 secure_fw/partitions/internal_trusted_storage/flash/its_flash.h File Reference . . . 922
7.270.1 Macro Definition Documentation . . . 923
7.270.1.1 ITS_FLASH_ALIGNMENT . . . 923
7.270.1.2 ITS_FLASH_DEV . . . 923
7.270.1.3 ITS_FLASH_MAX_ALIGNMENT . . . 923
7.270.1.4 ITS_FLASH_OPS . . . 923
7.270.1.5 PS_FLASH_ALIGNMENT . . . 923
7.271 secure_fw/partitions/internal_trusted_storage/flash/its_flash_hal.h File Reference . . . 924
7.271.1 Detailed Description . . . 925
7.271.2 Macro Definition Documentation . . . 925
7.271.2.1 ITS_BLOCK_INVALID_ID . . . 925
7.272 secure_fw/partitions/internal_trusted_storage/flash/its_flash_nand.c File Reference . . . 925
7.272.1 Variable Documentation . . . 926
7.272.1.1 its_flash_ops_nand . . . 926
7.273 secure_fw/partitions/internal_trusted_storage/flash/its_flash_nand.h File Reference . . . 926
7.273.1 Detailed Description . . . 928
7.273.2 Variable Documentation . . . 928
7.273.2.1 its_flash_ops_nand . . . 928
7.274 secure_fw/partitions/internal_trusted_storage/flash/its_flash_nor.c File Reference . . . 928
7.274.1 Variable Documentation . . . 928
7.274.1.1 its_flash_ops_nor . . . 928
7.275 secure_fw/partitions/internal_trusted_storage/flash/its_flash_nor.h File Reference . . . 929
7.275.1 Detailed Description . . . 930
7.275.2 Variable Documentation . . . 930

7.275.2.1 its_flash_ops_nor . . . 930
7.276 secure_fw/partitions/internal_trusted_storage/flash/its_flash_ram.c File Reference . . . 930
7.276.1 Variable Documentation . . . 931
7.276.1.1 its_flash_fs_ops_ram . . . 931
7.277 secure_fw/partitions/internal_trusted_storage/flash/its_flash_ram.h File Reference . . . 931
7.277.1 Detailed Description . . . 932
7.277.2 Variable Documentation . . . 932
7.277.2.1 its_flash_fs_ops_ram . . . 932
7.278 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs.c File Reference . . . 932
7.278.1 Macro Definition Documentation . . . 933
7.278.1.1 ITS_FLASH_FS_FLAG_DELETE . . . 933
7.278.1.2 ITS_FLASH_FS_INTERNAL_FLAGS_MASK . . . 933
7.278.2 Function Documentation . . . 934
7.278.2.1 its_flash_fs_file_delete() . . . 934
7.278.2.2 its_flash_fs_file_get_info() . . . 934
7.278.2.3 its_flash_fs_file_read() . . . 935
7.278.2.4 its_flash_fs_file_write() . . . 936
7.278.2.5 its_flash_fs_init_ctx() . . . 936
7.278.2.6 its_flash_fs_prepare() . . . 937
7.278.2.7 its_flash_fs_wipe_all() . . . 938
7.279 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs.h File Reference . . . 938
7.279.1 Detailed Description . . . 940
7.279.2 Macro Definition Documentation . . . 940
7.279.2.1 ITS_FLASH_FS_FLAG_CREATE . . . 940
7.279.2.2 ITS_FLASH_FS_FLAG_TRUNCATE . . . 940
7.279.2.3 ITS_FLASH_FS_USER_FLAGS_MASK . . . 940
7.279.2.4 ITS_FLASH_FS_WRITE_FLAGS_MASK . . . 940
7.279.3 Typedef Documentation . . . 940
7.279.3.1 its_flash_fs_ctx_t . . . 941
7.279.4 Function Documentation . . . 941
7.279.4.1 its_flash_fs_file_delete() . . . 941
7.279.4.2 its_flash_fs_file_get_info() . . . 941
7.279.4.3 its_flash_fs_file_read() . . . 942
7.279.4.4 its_flash_fs_file_write() . . . 943
7.279.4.5 its_flash_fs_init_ctx() . . . 943
7.279.4.6 its_flash_fs_prepare() . . . 944
7.279.4.7 its_flash_fs_wipe_all() . . . 945
7.280 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_dblock.c File Reference . . . 945
7.280.1 Function Documentation . . . 946
7.280.1.1 its_flash_fs_dblock_compact_block() . . . 946
7.280.1.2 its_flash_fs_dblock_read_file() . . . 947
7.280.1.3 its_flash_fs_dblock_write_file() . . . 948

7.281 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_dblock.h File Reference . . . . . 949
7.281.1 Function Documentation . . . . . 950
7.281.1.1 its_flash_fs_dblock_compact_block() . . . . . 950
7.281.1.2 its_flash_fs_dblock_read_file() . . . . . 951
7.281.1.3 its_flash_fs_dblock_write_file() . . . . . 952
7.282 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.c File Reference . . . . . 953
7.282.1 Macro Definition Documentation . . . . . 954
7.282.1.1 ITS_BLOCK_META_HEADER_SIZE . . . . . 955
7.282.1.2 ITS_BLOCK_METADATA_SIZE . . . . . 955
7.282.1.3 ITS_FILE_METADATA_SIZE . . . . . 955
7.282.1.4 ITS_MAX_BLOCK_DATA_COPY . . . . . 955
7.282.1.5 ITS_METADATA_BLOCK0 . . . . . 955
7.282.1.6 ITS_METADATA_BLOCK1 . . . . . 955
7.282.1.7 ITS_OTHER_META_BLOCK . . . . . 955
7.282.2 Function Documentation . . . . . 955
7.282.2.1 its_flash_fs_block_to_block_move() . . . . . 955
7.282.2.2 its_flash_fs_mblock_cp_file_meta() . . . . . 956
7.282.2.3 its_flash_fs_mblock_cur_data_scratch_id() . . . . . 957
7.282.2.4 its_flash_fs_mblock_get_file_idx_flag() . . . . . 957
7.282.2.5 its_flash_fs_mblock_get_file_idx_meta() . . . . . 958
7.282.2.6 its_flash_fs_mblock_init() . . . . . 959
7.282.2.7 its_flash_fs_mblock_meta_update_finalize() . . . . . 960
7.282.2.8 its_flash_fs_mblock_migrate_lb0_data_to_scratch() . . . . . 960
7.282.2.9 its_flash_fs_mblock_read_block_metadata() . . . . . 961
7.282.2.10 its_flash_fs_mblock_read_block_metadata_comp() . . . . . 962
7.282.2.11 its_flash_fs_mblock_read_file_meta() . . . . . 962
7.282.2.12 its_flash_fs_mblock_reserve_file() . . . . . 963
7.282.2.13 its_flash_fs_mblock_reset_metablock() . . . . . 964
7.282.2.14 its_flash_fs_mblock_set_data_scratch() . . . . . 964
7.282.2.15 its_flash_fs_mblock_update_scratch_block_meta() . . . . . 965
7.282.2.16 its_flash_fs_mblock_update_scratch_file_meta() . . . . . 965
7.283 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.h File Reference . . . . . 966
7.283.1 Macro Definition Documentation . . . . . 968
7.283.1.1 _T1 . . . . . 968
7.283.1.2 _T1_COMP . . . . . 968
7.283.1.3 _T2 . . . . . 969
7.283.1.4 _T3 . . . . . 969
7.283.1.5 ITS_BACKWARD_SUPPORTED_VERSION . . . . . 969
7.283.1.6 ITS_LOGICAL_DBLOCK0 . . . . . 969
7.283.1.7 ITS_METADATA_INVALID_INDEX . . . . . 969
7.283.1.8 ITS_SUPPORTED_VERSION . . . . . 969
7.283.2 Function Documentation . . . . . 969

7.283.2.1 its_flash_fs_block_to_block_move() . . . 970
7.283.2.2 its_flash_fs_mblock_cp_file_meta() . . . 970
7.283.2.3 its_flash_fs_mblock_cur_data_scratch_id() . . . 971
7.283.2.4 its_flash_fs_mblock_get_file_idx_flag() . . . 971
7.283.2.5 its_flash_fs_mblock_get_file_idx_meta() . . . 972
7.283.2.6 its_flash_fs_mblock_init() . . . 973
7.283.2.7 its_flash_fs_mblock_meta_update_finalize() . . . 974
7.283.2.8 its_flash_fs_mblock_migrate_lb0_data_to_scratch() . . . 974
7.283.2.9 its_flash_fs_mblock_read_block_metadata() . . . 975
7.283.2.10 its_flash_fs_mblock_read_block_metadata_comp() . . . 976
7.283.2.11 its_flash_fs_mblock_read_file_meta() . . . 976
7.283.2.12 its_flash_fs_mblock_reserve_file() . . . 977
7.283.2.13 its_flash_fs_mblock_reset_metablock() . . . 978
7.283.2.14 its_flash_fs_mblock_set_data_scratch() . . . 978
7.283.2.15 its_flash_fs_mblock_update_scratch_block_meta() . . . 979
7.283.2.16 its_flash_fs_mblock_update_scratch_file_meta() . . . 979
7.284 secure_fw/partitions/internal_trusted_storage/its_crypto_interface.c File Reference . . . 980
7.284.1 Function Documentation . . . 981
7.284.1.1 tfm_its_crypt_file() . . . 981
7.285 secure_fw/partitions/internal_trusted_storage/its_crypto_interface.h File Reference . . . 981
7.285.1 Function Documentation . . . 982
7.285.1.1 tfm_its_crypt_file() . . . 982
7.286 secure_fw/partitions/internal_trusted_storage/its_utils.c File Reference . . . 983
7.286.1 Function Documentation . . . 983
7.286.1.1 its_utils_check_contained_in() . . . 983
7.286.1.2 its_utils_validate_fid() . . . 984
7.287 secure_fw/partitions/internal_trusted_storage/its_utils.h File Reference . . . 984
7.287.1 Macro Definition Documentation . . . 986
7.287.1.1 ITS_DATA_SIZE_FIELD_SIZE . . . 986
7.287.1.2 ITS_DEFAULT_EMPTY_BUFF_VAL . . . 986
7.287.1.3 ITS_FILE_ID_SIZE . . . 986
7.287.1.4 ITS_FLAG_SIZE . . . 986
7.287.1.5 ITS_UTILS_ALIGN . . . 986
7.287.1.6 ITS_UTILS_BOUND_CHECK . . . 986
7.287.1.7 ITS_UTILS_IS_ALIGNED . . . 987
7.287.1.8 ITS_UTILS_MAX . . . 987
7.287.1.9 ITS_UTILS_MIN . . . 987
7.287.2 Function Documentation . . . 987
7.287.2.1 its_utils_check_contained_in() . . . 988
7.287.2.2 its_utils_validate_fid() . . . 988
7.288 secure_fw/partitions/internal_trusted_storage/tfm_internal_trusted_storage.c File Reference . . . 989
7.288.1 Function Documentation . . . 990

7.288.1.1 tfm_its_get() . . . 990
7.288.1.2 tfm_its_get_info() . . . 991
7.288.1.3 tfm_its_init() . . . 992
7.288.1.4 tfm_its_remove() . . . 993
7.288.1.5 tfm_its_set() . . . 994
7.288.2 Variable Documentation . . . 995
7.288.2.1 p_psa_dest_data . . . 995
7.288.2.2 p_psa_src_data . . . 995
7.289 secure_fw/partitions/internal_trusted_storage/tfm_internal_trusted_storage.h File Reference . . . 995
7.289.1 Function Documentation . . . 996
7.289.1.1 tfm_its_get() . . . 996
7.289.1.2 tfm_its_get_info() . . . 997
7.289.1.3 tfm_its_init() . . . 998
7.289.1.4 tfm_its_remove() . . . 999
7.289.1.5 tfm_its_set() . . . 1000
7.290 secure_fw/partitions/internal_trusted_storage/tfm_its_req_mngr.c File Reference . . . 1001
7.290.1 Function Documentation . . . 1001
7.290.1.1 its_req_mngr_read() . . . 1001
7.290.1.2 its_req_mngr_write() . . . 1002
7.290.1.3 tfm_internal_trusted_storage_service_sfn() . . . 1002
7.290.1.4 tfm_its_entry() . . . 1002
7.291 secure_fw/partitions/internal_trusted_storage/tfm_its_req_mngr.h File Reference . . . 1003
7.291.1 Function Documentation . . . 1003
7.291.1.1 its_req_mngr_read() . . . 1004
7.291.1.2 its_req_mngr_write() . . . 1004
7.292 secure_fw/partitions/lib/runtime/crt_memcmp.c File Reference . . . 1004
7.292.1 Function Documentation . . . 1005
7.292.1.1 memcmp() . . . 1005
7.293 secure_fw/partitions/lib/runtime/crt_memmove.c File Reference . . . 1005
7.293.1 Macro Definition Documentation . . . 1006
7.293.1.1 memcpy_f . . . 1006
7.293.2 Function Documentation . . . 1006
7.293.2.1 memmove() . . . 1006
7.294 secure_fw/partitions/lib/runtime/crt_strnlen.c File Reference . . . 1006
7.294.1 Function Documentation . . . 1006
7.294.1.1 tfm_strnlen() . . . 1007
7.295 secure_fw/partitions/lib/runtime/include/service_api.h File Reference . . . 1007
7.295.1 Function Documentation . . . 1008
7.295.1.1 tfm_core_get_boot_data() . . . 1008
7.296 secure_fw/partitions/lib/runtime/include/tfm_sp_log.h File Reference . . . 1008
7.296.1 Macro Definition Documentation . . . 1008
7.296.1.1 LOG_DBGFMT . . . 1009

7.296.1.2 LOG_ERRFMT . . . 1009
7.296.1.3 LOG_INFFMT . . . 1009
7.296.1.4 TFM_PARTITION_LOG_LEVEL_DEBUG . . . 1009
7.296.1.5 TFM_PARTITION_LOG_LEVEL_ERROR . . . 1009
7.296.1.6 TFM_PARTITION_LOG_LEVEL_INFO . . . 1009
7.296.1.7 TFM_PARTITION_LOG_LEVEL_SILENCE . . . 1009
7.296.2 Function Documentation . . . 1009
7.296.2.1 printf() . . . 1009
7.297 secure_fw/partitions/lib/runtime/include/tfm_strnlen.h File Reference . . . 1010
7.297.1 Function Documentation . . . 1010
7.297.1.1 tfm_strnlen() . . . 1010
7.298 secure_fw/partitions/lib/runtime/psa_api_ipc.c File Reference . . . 1010
7.298.1 Function Documentation . . . 1011
7.298.1.1 psa_framework_version() . . . 1011
7.298.1.2 psa_get() . . . 1012
7.298.1.3 psa_panic() . . . 1012
7.298.1.4 psa_read() . . . 1013
7.298.1.5 psa_reply() . . . 1013
7.298.1.6 psa_rot_lifecycle_state() . . . 1014
7.298.1.7 psa_skip() . . . 1014
7.298.1.8 psa_version() . . . 1015
7.298.1.9 psa_wait() . . . 1015
7.298.1.10 psa_write() . . . 1016
7.298.1.11 tfm_psa_call_pack() . . . 1016
7.299 secure_fw/partitions/lib/runtime/service_api.c File Reference . . . 1016
7.299.1 Function Documentation . . . 1017
7.299.1.1 tfm_core_get_boot_data() . . . 1017
7.299.1.2 tfm_flih_func_return() . . . 1017
7.300 secure_fw/partitions/lib/runtime/sfn_common_thread.c File Reference . . . 1018
7.300.1 Function Documentation . . . 1018
7.300.1.1 common_sfn_thread() . . . 1018
7.301 secure_fw/partitions/lib/runtime/sprt_partition_metadata_indicator.c File Reference . . . 1019
7.301.1 Variable Documentation . . . 1019
7.301.1.1 p_partition_metadata . . . 1019
7.302 secure_fw/partitions/lib/runtime/sprt_partition_metadata_indicator.h File Reference . . . 1019
7.302.1 Macro Definition Documentation . . . 1020
7.302.1.1 PART_METADATA . . . 1020
7.302.2 Variable Documentation . . . 1020
7.302.2.1 p_partition_metadata . . . 1020
7.303 secure_fw/partitions/lib/runtime/tfm_sp_log_raw.c File Reference . . . 1021
7.303.1 Macro Definition Documentation . . . 1021
7.303.1.1 NUM_BUFF_SIZE . . . 1021

7.303.1.2 PRINT_BUFF_SIZE . . . 1021
7.303.2 Function Documentation . . . 1021
7.303.2.1 printf() . . . 1021
7.303.2.2 vprintf() . . . 1022
7.304 secure_fw/partitions/ns_agent_mailbox/ns_agent_mailbox.c File Reference . . . 1022
7.304.1 Enumeration Type Documentation . . . 1023
7.304.1.1 mailbox_state . . . 1023
7.304.2 Function Documentation . . . 1023
7.304.2.1 ns_agent_mailbox_entry() . . . 1023
7.305 secure_fw/partitions/ns_agent_mailbox/tfm_multi_core_client_id.c File Reference . . . 1023
7.305.1 Macro Definition Documentation . . . 1024
7.305.1.1 MAX_MAILBOX_NUMBER . . . 1024
7.305.2 Function Documentation . . . 1024
7.305.2.1 tfm_multi_core_hal_client_id_translate() . . . 1024
7.305.2.2 tfm_multi_core_register_client_id_range() . . . 1025
7.306 secure_fw/partitions/ns_agent_mailbox/tfm_spe_mailbox.c File Reference . . . 1025
7.306.1 Macro Definition Documentation . . . 1026
7.306.1.1 MAILBOX_CLEAN_CACHE . . . 1026
7.306.1.2 MAILBOX_INVALIDATE_CACHE . . . 1027
7.306.2 Function Documentation . . . 1027
7.306.2.1 check_mailbox_msg() . . . 1027
7.306.2.2 clear_nspe_queue_pend_status() . . . 1027
7.306.2.3 clear_spe_queue_empty_status() . . . 1027
7.306.2.4 get_nspe_queue_pend_status() . . . 1027
7.306.2.5 get_nspe_reply_addr() . . . 1027
7.306.2.6 get_spe_mailbox_msg_handle() . . . 1028
7.306.2.7 get_spe_mailbox_msg_idx() . . . 1028
7.306.2.8 get_spe_queue_empty_status() . . . 1028
7.306.2.9 set_nspe_queue_replied_status() . . . 1028
7.306.2.10 set_spe_queue_empty_status() . . . 1028
7.306.2.11 tfm_inter_core_comm_deinit() . . . 1028
7.306.2.12 tfm_inter_core_comm_disable() . . . 1028
7.306.2.13 tfm_inter_core_comm_enable() . . . 1029
7.306.2.14 tfm_inter_core_comm_init() . . . 1029
7.306.2.15 tfm_mailbox_handle_msg() . . . 1029
7.306.2.16 tfm_mailbox_reply_msg() . . . 1029
7.307 secure_fw/partitions/ns_agent_mailbox/tfm_spe_mailbox.h File Reference . . . 1030
7.307.1 Function Documentation . . . 1031
7.307.1.1 tfm_mailbox_handle_msg() . . . 1031
7.307.1.2 tfm_mailbox_reply_msg() . . . 1031
7.308 secure_fw/partitions/ns_agent_tz/load_info_ns_agent_tz.c File Reference . . . 1032
7.308.1 Macro Definition Documentation . . . 1032

7.308.1.1 TFM_SP_NS_AGENT_NDEPS . . . 1033
7.308.1.2 TFM_SP_NS_AGENT_NSERVS . . . 1033
7.308.2 Function Documentation . . . 1033
7.308.2.1 __aligned() . . . 1033
7.308.2.2 ns_agent_tz_main() . . . 1033
7.308.3 Variable Documentation . . . 1033
7.308.3.1 tfm_sp_ns_agent_tz_load . . . 1033
7.309 secure_fw/partitions/ns_agent_tz/ns_agent_tz.c File Reference . . . 1033
7.309.1 Function Documentation . . . 1034
7.309.1.1 ns_agent_tz_main() . . . 1034
7.310 secure_fw/partitions/ns_agent_tz/ns_agent_tz_v80m.c File Reference . . . 1034
7.310.1 Function Documentation . . . 1034
7.310.1.1 ns_agent_tz_main() . . . 1034
7.311 secure_fw/partitions/ns_agent_tz/psa_api_veneers.c File Reference . . . 1034
7.311.1 Function Documentation . . . 1035
7.311.1.1 tfm_psa_call_veneer() . . . 1035
7.311.1.2 tfm_psa_close_veneer() . . . 1036
7.311.1.3 tfm_psa_connect_veneer() . . . 1036
7.311.1.4 tfm_psa_framework_version_veneer() . . . 1036
7.311.1.5 tfm_psa_version_veneer() . . . 1037
7.312 secure_fw/partitions/ns_agent_tz/psa_api_veneers_v80m.c File Reference . . . 1037
7.312.1 Function Documentation . . . 1038
7.312.1.1 clear_caller_context() . . . 1038
7.312.1.2 tfm_psa_call_veneer() . . . 1038
7.312.1.3 tfm_psa_close_veneer() . . . 1040
7.312.1.4 tfm_psa_connect_veneer() . . . 1040
7.312.1.5 tfm_psa_framework_version_veneer() . . . 1041
7.312.1.6 tfm_psa_version_veneer() . . . 1041
7.312.2 Variable Documentation . . . 1042
7.312.2.1 ret_err . . . 1042
7.313 secure_fw/partitions/platform/dir_platform.dox File Reference . . . 1042
7.314 secure_fw/partitions/platform/platform_sp.c File Reference . . . 1042
7.314.1 Macro Definition Documentation . . . 1043
7.314.1.1 NV_COUNTER_ID_SIZE . . . 1043
7.314.1.2 NV_COUNTER_SIZE . . . 1043
7.314.2 Typedef Documentation . . . 1043
7.314.2.1 plat_func_t . . . 1043
7.314.3 Function Documentation . . . 1043
7.314.3.1 platform_sp_init() . . . 1044
7.314.3.2 platform_sp_system_reset() . . . 1044
7.314.3.3 tfm_platform_service_sfn() . . . 1044
7.315 secure_fw/partitions/platform/platform_sp.h File Reference . . . 1044

7.315.1 Function Documentation . . . 1045
7.315.1.1 platform_sp_init() . . . 1046
7.315.1.2 platform_sp_system_reset() . . . 1046
7.316 secure_fw/partitions/protected_storage/config_ps_check.h File Reference . . . 1046
7.317 secure_fw/partitions/protected_storage/crypto/ps_crypto_interface.c File Reference . . . 1047
7.317.1 Macro Definition Documentation . . . 1048
7.317.1.1 LABEL_LEN . . . 1048
7.317.1.2 ps_crypto_setkey . . . 1049
7.317.2 Typedef Documentation . . . 1049
7.317.2.1 PS_ERROR_NOT_AEAD_ALG . . . 1049
7.317.3 Function Documentation . . . 1049
7.317.3.1 ps_crypto_auth_and_decrypt() . . . 1049
7.317.3.2 ps_crypto_authenticate() . . . 1049
7.317.3.3 ps_crypto_encrypt_and_tag() . . . 1050
7.317.3.4 ps_crypto_generate_auth_tag() . . . 1050
7.317.3.5 ps_crypto_get_iv() . . . 1051
7.317.3.6 ps_crypto_init() . . . 1051
7.317.3.7 ps_crypto_set_iv() . . . 1051
7.317.3.8 ps_crypto_to_blocks() . . . 1052
7.318 secure_fw/partitions/protected_storage/crypto/ps_crypto_interface.h File Reference . . . 1052
7.318.1 Function Documentation . . . 1054
7.318.1.1 ps_crypto_auth_and_decrypt() . . . 1054
7.318.1.2 ps_crypto_authenticate() . . . 1054
7.318.1.3 ps_crypto_encrypt_and_tag() . . . 1055
7.318.1.4 ps_crypto_generate_auth_tag() . . . 1055
7.318.1.5 ps_crypto_get_iv() . . . 1056
7.318.1.6 ps_crypto_init() . . . 1056
7.318.1.7 ps_crypto_set_iv() . . . 1056
7.318.1.8 ps_crypto_to_blocks() . . . 1057
7.319 secure_fw/partitions/protected_storage/dir_protected_storage.dox File Reference . . . 1057
7.320 secure_fw/partitions/protected_storage/nv_counters/ps_nv_counters.c File Reference . . . 1057
7.320.1 Function Documentation . . . 1058
7.320.1.1 ps_increment_nv_counter() . . . 1058
7.320.1.2 ps_read_nv_counter() . . . 1059
7.321 secure_fw/partitions/protected_storage/nv_counters/ps_nv_counters.h File Reference . . . 1059
7.321.1 Macro Definition Documentation . . . 1061
7.321.1.1 PS_NV_COUNTER_SIZE . . . 1061
7.321.1.2 TFM_PS_NV_COUNTER_1 . . . 1061
7.321.1.3 TFM_PS_NV_COUNTER_2 . . . 1061
7.321.1.4 TFM_PS_NV_COUNTER_3 . . . 1061
7.321.2 Function Documentation . . . 1061
7.321.2.1 ps_increment_nv_counter() . . . 1061

7.321.2.2 ps_read_nv_counter() . . . 1062
7.322 secure_fw/partitions/protected_storage/ps_encrypted_object.c File Reference . . . 1062
7.322.1 Macro Definition Documentation . . . 1063
7.322.1.1 PS_CRYPTO_BUF_LEN . . . 1063
7.322.1.2 PS_ENCRYPT_SIZE . . . 1063
7.322.1.3 PS_MAX_ENCRYPTED_OBJ_SIZE . . . 1063
7.322.1.4 PS_OBJECT_START_POSITION . . . 1063
7.322.1.5 STORED_HEADER_DATA_SIZE . . . 1063
7.322.2 Function Documentation . . . 1064
7.322.2.1 ps_encrypted_object_blocks() . . . 1064
7.322.2.2 ps_encrypted_object_read() . . . 1064
7.322.2.3 ps_encrypted_object_write() . . . 1065
7.322.3 Variable Documentation . . . 1065
7.322.3.1 auth_data_t . . . 1065
7.323 secure_fw/partitions/protected_storage/ps_encrypted_object.h File Reference . . . 1065
7.323.1 Function Documentation . . . 1066
7.323.1.1 ps_encrypted_object_blocks() . . . 1067
7.323.1.2 ps_encrypted_object_read() . . . 1067
7.323.1.3 ps_encrypted_object_write() . . . 1068
7.324 secure_fw/partitions/protected_storage/ps_filesystem_interface.c File Reference . . . 1068
7.324.1 Function Documentation . . . 1069
7.324.1.1 psa_its_get() . . . 1069
7.324.1.2 psa_its_get_info() . . . 1071
7.324.1.3 psa_its_remove() . . . 1071
7.324.1.4 psa_its_set() . . . 1072
7.324.2 Variable Documentation . . . 1073
7.324.2.1 p_psa_dest_data . . . 1073
7.324.2.2 p_psa_src_data . . . 1073
7.325 secure_fw/partitions/protected_storage/ps_object_defs.h File Reference . . . 1073
7.325.1 Macro Definition Documentation . . . 1074
7.325.1.1 PS_MAX_NUM_OBJECTS . . . 1074
7.325.1.2 PS_MAX_OBJECT_DATA_SIZE . . . 1075
7.325.1.3 PS_MAX_OBJECT_SIZE . . . 1075
7.325.1.4 PS_OBJECT_BUF_SIZE . . . 1075
7.325.1.5 PS_OBJECT_HEADER_SIZE . . . 1075
7.326 secure_fw/partitions/protected_storage/ps_object_system.c File Reference . . . 1075
7.326.1 Macro Definition Documentation . . . 1076
7.326.1.1 PS_OBJECT_SIZE . . . 1076
7.326.1.2 PS_OBJECT_START_POSITION . . . 1076
7.326.2 Enumeration Type Documentation . . . 1076
7.326.2.1 read_type_t . . . 1076
7.326.3 Function Documentation . . . 1077

7.326.3.1 ps_init_empty_object() . . . 1077
7.326.3.2 ps_object_create() . . . 1077
7.326.3.3 ps_object_delete() . . . 1078
7.326.3.4 ps_object_get_info() . . . 1079
7.326.3.5 ps_object_read() . . . 1080
7.326.3.6 ps_object_write() . . . 1081
7.326.3.7 ps_system_prepare() . . . 1082
7.326.3.8 ps_system_wipe_all() . . . 1082
7.327 secure_fw/partitions/protected_storage/ps_object_system.h File Reference . . . 1083
7.327.1 Function Documentation . . . 1084
7.327.1.1 ps_object_create() . . . 1084
7.327.1.2 ps_object_delete() . . . 1085
7.327.1.3 ps_object_get_info() . . . 1086
7.327.1.4 ps_object_read() . . . 1087
7.327.1.5 ps_object_write() . . . 1088
7.327.1.6 ps_system_prepare() . . . 1089
7.327.1.7 ps_system_wipe_all() . . . 1089
7.328 secure_fw/partitions/protected_storage/ps_object_table.c File Reference . . . 1090
7.328.1 Macro Definition Documentation . . . 1092
7.328.1.1 PS_CRYPTO_ASSOCIATED_DATA . . . 1092
7.328.1.2 PS_CRYPTO_ASSOCIATED_DATA_LEN . . . 1092
7.328.1.3 PS_FLASH_DEFAULT_VAL . . . 1092
7.328.1.4 PS_INVALID_NVC_VALUE . . . 1092
7.328.1.5 PS_NON_AUTH_OBJ_TABLE_SIZE . . . 1092
7.328.1.6 PS_NUM_OBJ_TABLES . . . 1093
7.328.1.7 PS_OBJ_TABLE_ENTRIES . . . 1093
7.328.1.8 PS_OBJ_TABLE_IDX_0 . . . 1093
7.328.1.9 PS_OBJ_TABLE_IDX_1 . . . 1093
7.328.1.10 PS_OBJ_TABLE_SIZE . . . 1093
7.328.1.11 PS_OBJECT_FS_ID . . . 1093
7.328.1.12 PS_OBJECT_FS_ID_TO_IDX . . . 1093
7.328.1.13 PS_OBJECT_SYSTEM_VERSION . . . 1094
7.328.1.14 PS_OBJECT_TABLE_OBJECT_OFFSET . . . 1094
7.328.1.15 PS_OBJECTS_TABLE_ENTRY_SIZE . . . 1094
7.328.1.16 PS_TABLE_FS_ID . . . 1094
7.328.2 Typedef Documentation . . . 1094
7.328.2.1 OBJ_TABLE_NOT_FIT_IN_STATIC_OBJ_DATA_BUF . . . 1094
7.328.3 Enumeration Type Documentation . . . 1094
7.328.3.1 ps_obj_table_state . . . 1094
7.328.4 Function Documentation . . . 1095
7.328.4.1 ps_object_table_create() . . . 1095
7.328.4.2 ps_object_table_current_gen() . . . 1095

7.328.4.3 ps_object_table_delete_object() . . . 1096
7.328.4.4 ps_object_table_delete_old_table() . . . 1096
7.328.4.5 ps_object_table_fs_read_table() . . . 1096
7.328.4.6 ps_object_table_fs_write_table() . . . 1097
7.328.4.7 ps_object_table_get_free_fid() . . . 1097
7.328.4.8 ps_object_table_get_obj_tbl_info() . . . 1098
7.328.4.9 ps_object_table_init() . . . 1099
7.328.4.10 ps_object_table_obj_exist() . . . 1099
7.328.4.11 ps_object_table_set_gen() . . . 1099
7.328.4.12 ps_object_table_set_obj_tbl_info() . . . 1100
7.328.4.13 ps_object_table_validate_version() . . . 1100
7.328.4.14 ps_table_free_idx() . . . 1100
7.329 secure_fw/partitions/protected_storage/ps_object_table.h File Reference . . . 1101
7.329.1 Function Documentation . . . 1103
7.329.1.1 ps_object_table_create() . . . 1103
7.329.1.2 ps_object_table_delete_object() . . . 1103
7.329.1.3 ps_object_table_delete_old_table() . . . 1104
7.329.1.4 ps_object_table_get_free_fid() . . . 1104
7.329.1.5 ps_object_table_get_obj_tbl_info() . . . 1104
7.329.1.6 ps_object_table_init() . . . 1106
7.329.1.7 ps_object_table_obj_exist() . . . 1107
7.329.1.8 ps_object_table_set_obj_tbl_info() . . . 1107
7.330 secure_fw/partitions/protected_storage/ps_utils.c File Reference . . . 1107
7.330.1 Function Documentation . . . 1108
7.330.1.1 ps_utils_check_contained_in() . . . 1108
7.331 secure_fw/partitions/protected_storage/ps_utils.h File Reference . . . 1109
7.331.1 Macro Definition Documentation . . . 1110
7.331.1.1 PS_DEFAULT_EMPTY_BUFF_VAL . . . 1110
7.331.1.2 PS_INVALID_FID . . . 1110
7.331.1.3 PS_UTILS_BOUND_CHECK . . . 1110
7.331.1.4 PS_UTILS_MIN . . . 1110
7.331.2 Function Documentation . . . 1111
7.331.2.1 ps_utils_check_contained_in() . . . 1111
7.332 secure_fw/partitions/protected_storage/tfm_protected_storage.c File Reference . . . 1111
7.332.1 Function Documentation . . . 1112
7.332.1.1 tfm_ps_get() . . . 1112
7.332.1.2 tfm_ps_get_info() . . . 1113
7.332.1.3 tfm_ps_get_support() . . . 1114
7.332.1.4 tfm_ps_init() . . . 1114
7.332.1.5 tfm_ps_remove() . . . 1115
7.332.1.6 tfm_ps_set() . . . 1116
7.333 secure_fw/partitions/protected_storage/tfm_protected_storage.h File Reference . . . 1117

7.333.1 Function Documentation . . . 1118
7.333.1.1 tfm_ps_get() . . . 1118
7.333.1.2 tfm_ps_get_info() . . . 1119
7.333.1.3 tfm_ps_get_support() . . . 1120
7.333.1.4 tfm_ps_init() . . . 1120
7.333.1.5 tfm_ps_remove() . . . 1121
7.333.1.6 tfm_ps_set() . . . 1122
7.334 secure_fw/partitions/protected_storage/tfm_ps_req_mngr.c File Reference . . . 1123
7.334.1 Function Documentation . . . 1123
7.334.1.1 ps_req_mngr_read_asset_data() . . . 1124
7.334.1.2 ps_req_mngr_write_asset_data() . . . 1124
7.334.1.3 tfm_protected_storage_service_sfn() . . . 1125
7.334.1.4 tfm_ps_entry() . . . 1125
7.335 secure_fw/partitions/protected_storage/tfm_ps_req_mngr.h File Reference . . . 1125
7.335.1 Function Documentation . . . 1126
7.335.1.1 ps_req_mngr_read_asset_data() . . . 1126
7.335.1.2 ps_req_mngr_write_asset_data() . . . 1127
7.336 secure_fw/shared/crt_memcpy.c File Reference . . . 1127
7.336.1 Function Documentation . . . 1128
7.336.1.1 memcpy() . . . 1128
7.337 secure_fw/shared/crt_memset.c File Reference . . . 1128
7.337.1 Function Documentation . . . 1129
7.337.1.1 memset() . . . 1129
7.338 secure_fw/spm/core/arch/tfm_arch.c File Reference . . . 1130
7.338.1 Function Documentation . . . 1130
7.338.1.1 tfm_arch_free_msp_and_exc_ret() . . . 1130
7.338.1.2 tfm_arch_init_context() . . . 1130
7.338.1.3 tfm_arch_refresh_hardware_context() . . . 1131
7.339 secure_fw/spm/core/arch/tfm_arch_v6m_v7m.c File Reference . . . 1131
7.339.1 Function Documentation . . . 1132
7.339.1.1 SVC_Handler() . . . 1132
7.339.1.2 tfm_arch_config_extensions() . . . 1132
7.339.1.3 tfm_arch_set_secure_exception_priorities() . . . 1132
7.339.2 Variable Documentation . . . 1132
7.339.2.1 psp_limit . . . 1132
7.339.2.2 scheduler_lock . . . 1133
7.340 secure_fw/spm/core/arch/tfm_arch_v6m_v7m.h File Reference . . . 1133
7.340.1 Macro Definition Documentation . . . 1134
7.340.1.1 EXC_RETURN_MODE . . . 1134
7.340.1.2 EXC_RETURN_SPSEL . . . 1134
7.340.1.3 SCB_ICSR_ADDR . . . 1134
7.340.1.4 SCB_ICSR_PENDSVSET_BIT . . . 1134

7.340.2 Function Documentation . . . 1134
7.340.2.1 arch_seal_thread_stack() . . . 1134
7.340.2.2 arch_update_process_sp() . . . 1135
7.340.2.3 is_default_stacking_rules_apply() . . . 1135
7.340.2.4 is_return_secure_stack() . . . 1136
7.340.2.5 tfm_arch_check_msp_sealing() . . . 1136
7.340.2.6 tfm_arch_get_psplim() . . . 1137
7.340.2.7 tfm_arch_set_msplim() . . . 1137
7.340.2.8 tfm_arch_set_psplim() . . . 1137
7.340.3 Variable Documentation . . . 1137
7.340.3.1 psp_limit . . . 1138
7.341 secure_fw/spm/core/arch/tfm_arch_v8m_base.c File Reference . . . 1138
7.341.1 Function Documentation . . . 1138
7.341.1.1 SVC_Handler() . . . 1138
7.341.1.2 tfm_arch_config_extensions() . . . 1138
7.341.1.3 tfm_arch_set_secure_exception_priorities() . . . 1139
7.341.2 Variable Documentation . . . 1139
7.341.2.1 scheduler_lock . . . 1139
7.342 secure_fw/spm/core/arch/tfm_arch_v8m_main.c File Reference . . . 1139
7.342.1 Function Documentation . . . 1139
7.342.1.1 SVC_Handler() . . . 1140
7.342.1.2 tfm_arch_config_extensions() . . . 1140
7.342.1.3 tfm_arch_set_secure_exception_priorities() . . . 1140
7.342.2 Variable Documentation . . . 1140
7.342.2.1 scheduler_lock . . . 1140
7.343 secure_fw/spm/core/backend_ipc.c File Reference . . . 1140
7.343.1 Typedef Documentation . . . 1141
7.343.1.1 comp_init_fn_t . . . 1142
7.343.2 Function Documentation . . . 1142
7.343.2.1 ARCH_CLAIM_CTXCTRL_INSTANCE() . . . 1142
7.343.2.2 backend_abi_entering_spm() . . . 1142
7.343.2.3 backend_abi_leaving_spm() . . . 1142
7.343.2.4 backend_assert_signal() . . . 1142
7.343.2.5 backend_init_comp_assuredly() . . . 1143
7.343.2.6 backend_messaging() . . . 1143
7.343.2.7 backend_replying() . . . 1143
7.343.2.8 backend_system_run() . . . 1143
7.343.2.9 backend_wait_signals() . . . 1144
7.343.2.10 common_sfn_thread() . . . 1144
7.343.2.11 ipc_schedule() . . . 1144
7.343.3 Variable Documentation . . . 1145
7.343.3.1 comp_init_fns . . . 1145

7.343.3.2 p_spm_thread_context . . . 1145
7.343.3.3 partition_listhead . . . 1145
7.343.3.4 psa_api_svc . . . 1145
7.343.3.5 psa_api_thread_fn_call . . . 1145
7.344 secure_fw/spm/core/backend_sfn.c File Reference . . . 1146
7.344.1 Macro Definition Documentation . . . 1146
7.344.1.1 SFN_PARTITION_STATE_INITED . . . 1147
7.344.1.2 SFN_PARTITION_STATE_NOT_INITED . . . 1147
7.344.2 Function Documentation . . . 1147
7.344.2.1 backend_assert_signal() . . . 1147
7.344.2.2 backend_init_comp_assuredly() . . . 1147
7.344.2.3 backend_messaging() . . . 1148
7.344.2.4 backend_replying() . . . 1148
7.344.2.5 backend_system_run() . . . 1148
7.344.2.6 backend_wait_signals() . . . 1148
7.344.3 Variable Documentation . . . 1149
7.344.3.1 p_current_partition . . . 1149
7.344.3.2 partition_listhead . . . 1149
7.345 secure_fw/spm/core/internal_status_code.h File Reference . . . 1149
7.345.1 Macro Definition Documentation . . . 1150
7.345.1.1 SPM_ERROR_BAD_PARAMETERS . . . 1150
7.345.1.2 SPM_ERROR_GENERIC . . . 1150
7.345.1.3 SPM_ERROR_MEMORY_CHECK . . . 1150
7.345.1.4 SPM_ERROR_SHORT_BUFFER . . . 1150
7.345.1.5 SPM_ERROR_VERSION . . . 1150
7.345.1.6 SPM_SUCCESS . . . 1150
7.345.1.7 STATUS_NEED_SCHEDULE . . . 1151
7.346 secure_fw/spm/core/interrupt.c File Reference . . . 1151
7.346.1 Function Documentation . . . 1151
7.346.1.1 get_irq_info_for_signal() . . . 1152
7.346.1.2 spm_handle_interrupt() . . . 1152
7.346.1.3 tfm_flih_func_return() . . . 1153
7.346.1.4 tfm_flih_prepare_depriv_flih() . . . 1153
7.346.1.5 tfm_flih_return_to_isr() . . . 1154
7.346.2 Variable Documentation . . . 1154
7.346.2.1 spm_boundary . . . 1154
7.347 secure_fw/spm/core/interrupt.h File Reference . . . 1154
7.347.1 Function Documentation . . . 1155
7.347.1.1 get_irq_info_for_signal() . . . 1155
7.347.1.2 spm_handle_interrupt() . . . 1156
7.347.1.3 tfm_flih_prepare_depriv_flih() . . . 1156
7.347.1.4 tfm_flih_return_to_isr() . . . 1157

7.348 secure_fw/spm/core/mailbox_agent_api.c File Reference . . . 1157
7.348.1 Function Documentation . . . 1158
7.348.1.1 tfm_spm_agent_psa_call() . . . 1158
7.349 secure_fw/spm/core/main.c File Reference . . . 1158
7.349.1 Function Documentation . . . 1159
7.349.1.1 main() . . . 1159
7.349.2 Variable Documentation . . . 1159
7.349.2.1 spm_boundary . . . 1159
7.350 secure_fw/spm/core/memory_symbols.h File Reference . . . 1159
7.350.1 Macro Definition Documentation . . . 1160
7.350.1.1 PART_INFOLIST_END . . . 1160
7.350.1.2 PART_INFOLIST_START . . . 1161
7.350.1.3 PART_INFORAM_END . . . 1161
7.350.1.4 PART_INFORAM_START . . . 1161
7.350.1.5 SERV_INFORAM_END . . . 1161
7.350.1.6 SERV_INFORAM_START . . . 1161
7.350.1.7 SPM_BOOT_STACK_BOTTOM . . . 1161
7.350.1.8 SPM_BOOT_STACK_TOP . . . 1161
7.350.2 Function Documentation . . . 1161
7.350.2.1 REGION_DECLARE() [1/4] . . . 1161
7.350.2.2 REGION_DECLARE() [2/4] . . . 1161
7.350.2.3 REGION_DECLARE() [3/4] . . . 1162
7.350.2.4 REGION_DECLARE() [4/4] . . . 1162
7.351 secure_fw/spm/core/ns_agent_mailbox_interface.c File Reference . . . 1162
7.351.1 Function Documentation . . . 1162
7.351.1.1 ns_agent_boot_ns_core() . . . 1162
7.351.1.2 ns_agent_mailbox_disable() . . . 1163
7.351.1.3 ns_agent_mailbox_enable() . . . 1163
7.351.1.4 ns_agent_ns_core_shutdown() . . . 1164
7.352 secure_fw/spm/core/psa_api.c File Reference . . . 1164
7.352.1 Function Documentation . . . 1165
7.352.1.1 spm_handle_programmer_errors() . . . 1165
7.352.1.2 tfm_spm_get_lifecycle_state() . . . 1166
7.352.1.3 tfm_spm_partition_psa_panic() . . . 1167
7.352.1.4 tfm_spm_partition_psa_reply() . . . 1167
7.353 secure_fw/spm/core/psa_call_api.c File Reference . . . 1168
7.353.1 Function Documentation . . . 1168
7.353.1.1 spm_associate_call_params() . . . 1168
7.353.1.2 tfm_spm_client_psa_call() . . . 1169
7.354 secure_fw/spm/core/psa_connection_api.c File Reference . . . 1169
7.354.1 Function Documentation . . . 1170
7.354.1.1 spm_psa_close_client_id_associated() . . . 1170

7.354.1.2 spm_psa_connect_client_id_associated() . . . 1170
7.354.1.3 tfm_spm_client_psa_close() . . . 1171
7.354.1.4 tfm_spm_client_psa_connect() . . . 1171
7.354.1.5 tfm_spm_partition_psa_set_rhandle() . . . 1171
7.355 secure_fw/spm/core/psa_interface_sfn.c File Reference . . . 1171
7.355.1 Function Documentation . . . 1172
7.355.1.1 psa_framework_version() . . . 1172
7.355.1.2 psa_panic() . . . 1173
7.355.1.3 psa_read() . . . 1174
7.355.1.4 psa_skip() . . . 1175
7.355.1.5 psa_version() . . . 1176
7.355.1.6 psa_write() . . . 1177
7.355.1.7 tfm_psa_call_pack() . . . 1178
7.356 secure_fw/spm/core/psa_interface_svc.c File Reference . . . 1178
7.356.1 Function Documentation . . . 1179
7.356.1.1 psa_framework_version_svc() . . . 1179
7.356.1.2 psa_get_svc() . . . 1179
7.356.1.3 psa_panic_svc() . . . 1180
7.356.1.4 psa_read_svc() . . . 1180
7.356.1.5 psa_reply_svc() . . . 1180
7.356.1.6 psa_rot_lifecycle_state_svc() . . . 1180
7.356.1.7 psa_skip_svc() . . . 1180
7.356.1.8 psa_version_svc() . . . 1180
7.356.1.9 psa_wait_svc() . . . 1180
7.356.1.10 psa_write_svc() . . . 1180
7.356.1.11 tfm_psa_call_pack_svc() . . . 1181
7.356.2 Variable Documentation . . . 1181
7.356.2.1 psa_api_svc . . . 1181
7.357 secure_fw/spm/core/psa_interface_thread_fn_call.c File Reference . . . 1181
7.357.1 Macro Definition Documentation . . . 1182
7.357.1.1 TFM_THREAD_FN_CALL_ENTRY . . . 1182
7.357.2 Function Documentation . . . 1182
7.357.2.1 psa_framework_version_thread_fn_call() . . . 1182
7.357.2.2 psa_get_thread_fn_call() . . . 1183
7.357.2.3 psa_panic_thread_fn_call() . . . 1183
7.357.2.4 psa_read_thread_fn_call() . . . 1183
7.357.2.5 psa_reply_thread_fn_call() . . . 1183
7.357.2.6 psa_rot_lifecycle_state_thread_fn_call() . . . 1184
7.357.2.7 psa_skip_thread_fn_call() . . . 1184
7.357.2.8 psa_version_thread_fn_call() . . . 1184
7.357.2.9 psa_wait_thread_fn_call() . . . 1185
7.357.2.10 psa_write_thread_fn_call() . . . 1185

7.357.2.11 tfm_psa_call_pack_thread_fn_call() . . . 1185
7.357.3 Variable Documentation . . . 1185
7.357.3.1 psa_api_thread_fn_call . . . 1185
7.358 secure_fw/spm/core/psa_irq_api.c File Reference . . . 1186
7.358.1 Function Documentation . . . 1186
7.358.1.1 tfm_spm_partition_psa_irq_disable() . . . 1186
7.358.1.2 tfm_spm_partition_psa_irq_enable() . . . 1186
7.359 secure_fw/spm/core/psa_mmiovec_api.c File Reference . . . 1187
7.359.1 Function Documentation . . . 1187
7.359.1.1 tfm_spm_partition_psa_map_invec() . . . 1187
7.359.1.2 tfm_spm_partition_psa_map_outvec() . . . 1187
7.359.1.3 tfm_spm_partition_psa_unmap_invec() . . . 1188
7.359.1.4 tfm_spm_partition_psa_unmap_outvec() . . . 1188
7.360 secure_fw/spm/core/psa_read_write_skip_api.c File Reference . . . 1188
7.360.1 Function Documentation . . . 1188
7.360.1.1 tfm_spm_partition_psa_read() . . . 1188
7.360.1.2 tfm_spm_partition_psa_skip() . . . 1189
7.360.1.3 tfm_spm_partition_psa_write() . . . 1190
7.361 secure_fw/spm/core/psa_version_api.c File Reference . . . 1191
7.361.1 Function Documentation . . . 1191
7.361.1.1 tfm_spm_client_psa_framework_version() . . . 1192
7.361.1.2 tfm_spm_client_psa_version() . . . 1192
7.362 secure_fw/spm/core/rom_loader.c File Reference . . . 1193
7.362.1 Function Documentation . . . 1193
7.362.1.1 load_a_partition_assuredly() . . . 1193
7.362.1.2 load_irqs_assuredly() . . . 1194
7.362.1.3 load_services_assuredly() . . . 1194
7.363 secure_fw/spm/core/spm.h File Reference . . . 1194
7.363.1 Macro Definition Documentation . . . 1196
7.363.1.1 CLIENT_HANDLE_VALUE_MIN . . . 1196
7.363.1.2 GET_CTX_OWNER . . . 1196
7.363.1.3 GET_INDEX_FROM_STATIC_HANDLE . . . 1196
7.363.1.4 GET_THRD_OWNER . . . 1197
7.363.1.5 GET_VERSION_FROM_STATIC_HANDLE . . . 1197
7.363.1.6 IS_STATIC_HANDLE . . . 1197
7.363.1.7 PSA_TIMEOUT_MASK . . . 1197
7.363.1.8 SPM_INVALID_PARTITION_IDX . . . 1197
7.363.1.9 STATIC_HANDLE_IDX_BIT_WIDTH . . . 1197
7.363.1.10 STATIC_HANDLE_IDX_MASK . . . 1197
7.363.1.11 STATIC_HANDLE_INDICATOR_OFFSET . . . 1197
7.363.1.12 STATIC_HANDLE_NUM_LIMIT . . . 1197
7.363.1.13 STATIC_HANDLE_VER_BIT_WIDTH . . . 1198

7.363.1.14 STATIC_HANDLE_VER_MASK . . . 1198
7.363.1.15 STATIC_HANDLE_VER_OFFSET . . . 1198
7.363.1.16 TFM_CLIENT_ID_IS_NS . . . 1198
7.363.1.17 TFM_HANDLE_STATUS_ACTIVE . . . 1198
7.363.1.18 TFM_HANDLE_STATUS_IDLE . . . 1198
7.363.1.19 TFM_HANDLE_STATUS_TO_FREE . . . 1198
7.363.1.20 tfm_spm_is_rpc_msg . . . 1198
7.363.2 Function Documentation . . . 1198
7.363.2.1 connection_to_handle() . . . 1198
7.363.2.2 handle_to_connection() . . . 1199
7.363.2.3 ipc_schedule() . . . 1199
7.363.2.4 spm_allocate_connection() . . . 1200
7.363.2.5 spm_associate_call_params() . . . 1200
7.363.2.6 spm_free_connection() . . . 1200
7.363.2.7 spm_get_idle_connection() . . . 1201
7.363.2.8 spm_init_connection_space() . . . 1201
7.363.2.9 spm_init_idle_connection() . . . 1202
7.363.2.10 spm_msg_handle_to_connection() . . . 1203
7.363.2.11 spm_validate_connection() . . . 1203
7.363.2.12 tfm_spm_check_authorization() . . . 1204
7.363.2.13 tfm_spm_check_client_version() . . . 1205
7.363.2.14 tfm_spm_get_client_id() . . . 1205
7.363.2.15 tfm_spm_get_service_by_sid() . . . 1206
7.363.2.16 tfm_spm_init() . . . 1207
7.363.2.17 tfm_spm_is_ns_caller() . . . 1207
7.363.2.18 tfm_spm_partition_get_running_partition_id() . . . 1208
7.363.2.19 update_caller_outvec_len() . . . 1208
7.364 secure_fw/spm/core/spm_connection_pool.c File Reference . . . 1208
7.364.1 Macro Definition Documentation . . . 1209
7.364.1.1 CONVERSION_FACTOR_BITOFFSET . . . 1209
7.364.1.2 CONVERSION_FACTOR_VALUE . . . 1209
7.364.1.3 CONVERSION_FACTOR_VALUE_MAX . . . 1209
7.364.2 Function Documentation . . . 1209
7.364.2.1 connection_to_handle() . . . 1210
7.364.2.2 handle_to_connection() . . . 1210
7.364.2.3 spm_allocate_connection() . . . 1210
7.364.2.4 spm_free_connection() . . . 1210
7.364.2.5 spm_init_connection_space() . . . 1210
7.364.2.6 spm_validate_connection() . . . 1211
7.365 secure_fw/spm/core/spm_ipc.c File Reference . . . 1211
7.365.1 Function Documentation . . . 1212
7.365.1.1 spm_get_idle_connection() . . . 1213

7.365.1.2 spm_init_idle_connection() . . . 1213
7.365.1.3 spm_msg_handle_to_connection() . . . 1214
7.365.1.4 tfm_spm_check_authorization() . . . 1214
7.365.1.5 tfm_spm_check_client_version() . . . 1215
7.365.1.6 tfm_spm_get_client_id() . . . 1216
7.365.1.7 tfm_spm_get_service_by_sid() . . . 1216
7.365.1.8 tfm_spm_init() . . . 1217
7.365.1.9 tfm_spm_is_ns_caller() . . . 1217
7.365.1.10 tfm_spm_partition_get_running_partition_id() . . . 1218
7.365.1.11 update_caller_outvec_len() . . . 1219
7.365.2 Variable Documentation . . . 1219
7.365.2.1 stateless_services_ref_tbl . . . 1219
7.366 secure_fw/spm/core/spm_local_connection.c File Reference . . . 1219
7.366.1 Macro Definition Documentation . . . 1219
7.366.1.1 CONNECTION_SIZE . . . 1220
7.366.1.2 FIXED_STATIC_HANDLE . . . 1220
7.366.2 Function Documentation . . . 1220
7.366.2.1 connection_to_handle() . . . 1220
7.366.2.2 handle_to_connection() . . . 1220
7.366.2.3 spm_allocate_connection() . . . 1220
7.366.2.4 spm_free_connection() . . . 1221
7.366.2.5 spm_init_connection_space() . . . 1221
7.366.2.6 spm_validate_connection() . . . 1221
7.367 secure_fw/spm/core/spm_log.c File Reference . . . 1221
7.367.1 Macro Definition Documentation . . . 1222
7.367.1.1 MAX_DIGIT_BITS . . . 1222
7.367.2 Function Documentation . . . 1222
7.367.2.1 spm_log_msgval() . . . 1222
7.368 secure_fw/spm/core/stack_watermark.c File Reference . . . 1223
7.368.1 Macro Definition Documentation . . . 1223
7.368.1.1 SPMLOG . . . 1224
7.368.1.2 SPMLOG_VAL . . . 1224
7.368.1.3 STACK_WATERMARK_VAL . . . 1224
7.368.2 Function Documentation . . . 1224
7.368.2.1 for() . . . 1224
7.368.2.2 tfm_hal_output_spm_log() . . . 1224
7.368.2.3 UNI_LIST_FOREACH() . . . 1225
7.368.3 Variable Documentation . . . 1225
7.368.3.1 void . . . 1225
7.369 secure_fw/spm/core/stack_watermark.h File Reference . . . 1225
7.369.1 Macro Definition Documentation . . . 1226
7.369.1.1 dump_used_stacks . . . 1226

7.369.1.2 watermark_stack . . . 1226
7.370 secure_fw/spm/core/tfm_boot_data.c File Reference . . . 1226
7.370.1 Macro Definition Documentation . . . 1227
7.370.1.1 BOOT_DATA_INVALID . . . 1227
7.370.1.2 BOOT_DATA_VALID . . . 1227
7.370.2 Function Documentation . . . 1227
7.370.2.1 tfm_core_get_boot_data_handler() . . . 1227
7.370.2.2 tfm_core_validate_boot_data() . . . 1227
7.371 secure_fw/spm/core/tfm_boot_data.h File Reference . . . 1227
7.371.1 Function Documentation . . . 1228
7.371.1.1 tfm_core_get_boot_data_handler() . . . 1228
7.371.1.2 tfm_core_validate_boot_data() . . . 1229
7.372 secure_fw/spm/core/tfm_multi_core.h File Reference . . . 1229
7.372.1 Macro Definition Documentation . . . 1230
7.372.1.1 CLIENT_ID_OWNER_MAGIC . . . 1230
7.372.2 Function Documentation . . . 1230
7.372.2.1 tfm_inter_core_comm_deinit() . . . 1230
7.372.2.2 tfm_inter_core_comm_disable() . . . 1230
7.372.2.3 tfm_inter_core_comm_enable() . . . 1230
7.372.2.4 tfm_inter_core_comm_init() . . . 1231
7.372.2.5 tfm_multi_core_hal_client_id_translate() . . . 1231
7.372.2.6 tfm_multi_core_register_client_id_range() . . . 1231
7.373 secure_fw/spm/core/tfm_pools.c File Reference . . . 1232
7.373.1 Macro Definition Documentation . . . 1232
7.373.1.1 POOL_MAGIC_ALLOCATED . . . 1232
7.373.2 Function Documentation . . . 1233
7.373.2.1 is_valid_chunk_data_in_pool() . . . 1233
7.373.2.2 tfm_pool_alloc() . . . 1233
7.373.2.3 tfm_pool_free() . . . 1234
7.373.2.4 tfm_pool_init() . . . 1235
7.374 secure_fw/spm/core/tfm_pools.h File Reference . . . 1235
7.374.1 Macro Definition Documentation . . . 1237
7.374.1.1 POOL_BUFFER_SIZE . . . 1237
7.374.1.2 POOL_HEAD_SIZE . . . 1237
7.374.1.3 TFM_POOL_DECLARE . . . 1237
7.374.2 Function Documentation . . . 1237
7.374.2.1 is_valid_chunk_data_in_pool() . . . 1237
7.374.2.2 tfm_pool_alloc() . . . 1238
7.374.2.3 tfm_pool_free() . . . 1238
7.374.2.4 tfm_pool_init() . . . 1239
7.375 secure_fw/spm/core/tfm_rpc.c File Reference . . . 1240
7.375.1 Function Documentation . . . 1241

7.375.1.1 tfm_rpc_client_call_handler() . . . 1241
7.375.1.2 tfm_rpc_psa_call() . . . 1241
7.375.1.3 tfm_rpc_psa_framework_version() . . . 1241
7.375.1.4 tfm_rpc_psa_version() . . . 1241
7.375.1.5 tfm_rpc_register_ops() . . . 1242
7.375.1.6 tfm_rpc_unregister_ops() . . . 1242
7.376 secure_fw/spm/core/tfm_rpc.h File Reference . . . 1242
7.377 secure_fw/spm/core/tfm_svcalls.c File Reference . . . 1242
7.377.1 Macro Definition Documentation . . . 1243
7.377.1.1 INVALID_PSP_VALUE . . . 1243
7.377.1.2 set_ns_ctx_mgr . . . 1243
7.377.2 Function Documentation . . . 1243
7.377.2.1 spm_svc_handler() . . . 1243
7.377.2.2 tfm_svc_thread_mode_spm_return() . . . 1244
7.378 secure_fw/spm/core/tfm_svcalls.h File Reference . . . 1244
7.378.1 Function Documentation . . . 1245
7.378.1.1 spm_svc_handler() . . . 1245
7.378.1.2 tfm_svc_thread_mode_spm_return() . . . 1246
7.379 secure_fw/spm/core/thread.c File Reference . . . 1246
7.379.1 Macro Definition Documentation . . . 1246
7.379.1.1 LIST_HEAD . . . 1247
7.379.1.2 RNBL_HEAD . . . 1247
7.379.2 Function Documentation . . . 1247
7.379.2.1 thrd_next() . . . 1247
7.379.2.2 thrd_set_query_callback() . . . 1247
7.379.2.3 thrd_set_state() . . . 1248
7.379.2.4 thrd_start() . . . 1248
7.379.2.5 thrd_start_scheduler() . . . 1248
7.379.3 Variable Documentation . . . 1248
7.379.3.1 p_curr_thrd . . . 1248
7.380 secure_fw/spm/core/thread.h File Reference . . . 1248
7.380.1 Macro Definition Documentation . . . 1250
7.380.1.1 CURRENT_THREAD . . . 1250
7.380.1.2 THRD_ERR_GENERIC . . . 1250
7.380.1.3 THRD_GENERAL_EXIT . . . 1250
7.380.1.4 THRD_INIT . . . 1250
7.380.1.5 THRD_PRIOR_HIGH . . . 1250
7.380.1.6 THRD_PRIOR_HIGHEST . . . 1250
7.380.1.7 THRD_PRIOR_LOW . . . 1251
7.380.1.8 THRD_PRIOR_LOWEST . . . 1251
7.380.1.9 THRD_PRIOR_MEDIUM . . . 1251
7.380.1.10 THRD_SET_PRIORITY . . . 1251

7.380.1.11 THRD_STATE_BLOCK . . . 1251
7.380.1.12 THRD_STATE_CREATING . . . 1251
7.380.1.13 THRD_STATE_DETACH . . . 1251
7.380.1.14 THRD_STATE_INVALID . . . 1251
7.380.1.15 THRD_STATE_RET_VAL_AVAIL . . . 1251
7.380.1.16 THRD_STATE_RUNNABLE . . . 1251
7.380.1.17 THRD_SUCCESS . . . 1252
7.380.1.18 THRD_UPDATE_CUR_CTXCTRL . . . 1252
7.380.2 Typedef Documentation . . . 1252
7.380.2.1 thrd_fn_t . . . 1252
7.380.2.2 thrd_query_state_t . . . 1252
7.380.3 Function Documentation . . . 1252
7.380.3.1 thrd_next() . . . 1252
7.380.3.2 thrd_set_query_callback() . . . 1252
7.380.3.3 thrd_set_state() . . . 1253
7.380.3.4 thrd_start() . . . 1253
7.380.3.5 thrd_start_scheduler() . . . 1253
7.380.4 Variable Documentation . . . 1253
7.380.4.1 p_curr_thrd . . . 1253
7.381 platform/ext/target/infineon/common/nspe/os_wrapper/thread.h File Reference . . . 1254
7.381.1 Typedef Documentation . . . 1255
7.381.1.1 os_wrapper_thread_func . . . 1255
7.381.2 Function Documentation . . . 1255
7.381.2.1 os_wrapper_thread_exit() . . . 1255
7.381.2.2 os_wrapper_thread_get_handle() . . . 1255
7.381.2.3 os_wrapper_thread_get_priority() . . . 1255
7.381.2.4 os_wrapper_thread_new() . . . 1256
7.381.2.5 os_wrapper_thread_set_flag() . . . 1256
7.381.2.6 os_wrapper_thread_set_flag_isr() . . . 1256
7.381.2.7 os_wrapper_thread_wait_flag() . . . 1257
7.382 secure_fw/spm/core/utilities.c File Reference . . . 1257
7.382.1 Function Documentation . . . 1258
7.382.1.1 tfm_core_panic() . . . 1258
7.383 secure_fw/spm/include/aapcs_local.h File Reference . . . 1259
7.383.1 Macro Definition Documentation . . . 1260
7.383.1.1 AAPCS_DUAL_U32_AS_U64 . . . 1260
7.383.1.2 AAPCS_DUAL_U32_SET . . . 1260
7.383.1.3 AAPCS_DUAL_U32_SET_A0 . . . 1261
7.383.1.4 AAPCS_DUAL_U32_SET_A1 . . . 1261
7.383.1.5 AAPCS_DUAL_U32_T . . . 1261
7.384 secure_fw/spm/include/bitops.h File Reference . . . 1261
7.384.1 Macro Definition Documentation . . . 1262

7.384.1.1 IS_ONLY_ONE_BIT_IN_UINT32 . . . 1262
7.385 secure_fw/spm/include/boot/tfm_boot_status.h File Reference . . . 1262
7.385.1 Macro Definition Documentation . . . 1264
7.385.1.1 CLAIM_MASK . . . 1264
7.385.1.2 CLAIM_POS . . . 1264
7.385.1.3 GET_FWU_CLAIM . . . 1264
7.385.1.4 GET_FWU_MODULE . . . 1264
7.385.1.5 GET_IAS_CLAIM . . . 1264
7.385.1.6 GET_IAS_MEASUREMENT_CLAIM . . . 1264
7.385.1.7 GET_IAS_MODULE . . . 1265
7.385.1.8 GET_MAJOR . . . 1265
7.385.1.9 GET_MBS_CLAIM . . . 1265
7.385.1.10 GET_MBS_SLOT . . . 1265
7.385.1.11 GET_MINOR . . . 1265
7.385.1.12 MAJOR_MASK . . . 1265
7.385.1.13 MAJOR_POS . . . 1265
7.385.1.14 MASK_LEFT_SHIFT . . . 1265
7.385.1.15 MASK_RIGHT_SHIFT . . . 1265
7.385.1.16 MEASUREMENT_CLAIM_POS . . . 1266
7.385.1.17 MINOR_MASK . . . 1266
7.385.1.18 MINOR_POS . . . 1266
7.385.1.19 MODULE_MASK . . . 1266
7.385.1.20 MODULE_POS . . . 1266
7.385.1.21 SET_FWU_MINOR . . . 1266
7.385.1.22 SET_IAS_MINOR . . . 1266
7.385.1.23 SET_MBS_MINOR . . . 1267
7.385.1.24 SET_TLV_TYPE . . . 1267
7.385.1.25 SHARED_DATA_ENTRY_HEADER_SIZE . . . 1267
7.385.1.26 SHARED_DATA_ENTRY_SIZE . . . 1267
7.385.1.27 SHARED_DATA_HEADER_SIZE . . . 1267
7.385.1.28 SHARED_DATA_TLV_INFO_MAGIC . . . 1267
7.385.1.29 SLOT_ID_MASK . . . 1267
7.385.1.30 SLOT_ID_POS . . . 1267
7.385.1.31 SW_AROT . . . 1268
7.385.1.32 SW_BL2 . . . 1268
7.385.1.33 SW_BOOT_RECORD . . . 1268
7.385.1.34 SW_GENERAL . . . 1268
7.385.1.35 SW_MAX . . . 1268
7.385.1.36 SW_MEASURE_METADATA . . . 1268
7.385.1.37 SW_MEASURE_TYPE . . . 1269
7.385.1.38 SW_MEASURE_VALUE . . . 1269
7.385.1.39 SW_MEASURE_VALUE_NON_EXTENDABLE . . . 1269

7.385.1.40 SW_NSPE . . . 1269
7.385.1.41 SW_PROT . . . 1269
7.385.1.42 SW_S_NS . . . 1269
7.385.1.43 SW_SIGNER_ID . . . 1269
7.385.1.44 SW_SPE . . . 1269
7.385.1.45 SW_TYPE . . . 1269
7.385.1.46 SW_VERSION . . . 1269
7.385.1.47 TLV_MAJOR_CORE . . . 1270
7.385.1.48 TLV_MAJOR_FWU . . . 1270
7.385.1.49 TLV_MAJOR_IAS . . . 1270
7.385.1.50 TLV_MAJOR_INVALID . . . 1270
7.385.1.51 TLV_MAJOR_MBS . . . 1270
7.386 secure_fw/spm/include/config_spm.h File Reference . . . 1270
7.387 secure_fw/spm/include/critical_section.h File Reference . . . 1271
7.387.1 Macro Definition Documentation . . . 1271
7.387.1.1 CRITICAL_SECTION_ENTER . . . 1271
7.387.1.2 CRITICAL_SECTION_INIT . . . 1272
7.387.1.3 CRITICAL_SECTION_LEAVE . . . 1272
7.387.1.4 CRITICAL_SECTION_STATIC_INIT . . . 1272
7.388 secure_fw/spm/include/current.h File Reference . . . 1272
7.388.1 Macro Definition Documentation . . . 1272
7.388.1.1 GET_CURRENT_COMPONENT . . . 1272
7.388.1.2 SET_CURRENT_COMPONENT . . . 1273
7.389 secure_fw/spm/include/ffm/backend.h File Reference . . . 1273
7.389.1 Macro Definition Documentation . . . 1273
7.389.1.1 PARTITION_LIST_ADDR . . . 1274
7.389.1.2 SPM_THREAD_CONTEXT . . . 1274
7.389.2 Function Documentation . . . 1274
7.389.2.1 backend_assert_signal() . . . 1274
7.389.2.2 backend_init_comp_assuredly() . . . 1274
7.389.2.3 backend_messaging() . . . 1275
7.389.2.4 backend_replying() . . . 1275
7.389.2.5 backend_system_run() . . . 1276
7.389.2.6 backend_wait_signals() . . . 1276
7.389.3 Variable Documentation . . . 1276
7.389.3.1 p_spm_thread_context . . . 1277
7.389.3.2 partition_listhead . . . 1277
7.390 secure_fw/spm/include/ffm/backend_ipc.h File Reference . . . 1277
7.390.1 Macro Definition Documentation . . . 1277
7.390.1.1 BACKEND_SERVICE_SET . . . 1277
7.390.1.2 BACKEND_SPM_INIT . . . 1278
7.390.2 Function Documentation . . . 1278

7.390.2.1 backend_abi_entering_spm() . . . 1278
7.390.2.2 backend_abi_leaving_spm() . . . 1278
7.391 secure_fw/spm/include/ffm/backend_sfn.h File Reference . . . 1278
7.391.1 Macro Definition Documentation . . . 1278
7.391.1.1 BACKEND_SERVICE_SET . . . 1279
7.391.1.2 BACKEND_SPM_INIT . . . 1279
7.392 secure_fw/spm/include/ffm/mailbox_agent_api.h File Reference . . . 1279
7.392.1 Macro Definition Documentation . . . 1280
7.392.1.1 agent_psa_close . . . 1280
7.392.1.2 agent_psa_connect . . . 1280
7.392.2 Function Documentation . . . 1280
7.392.2.1 agent_psa_call() . . . 1280
7.393 secure_fw/spm/include/ffm/original_vector_api.h File Reference . . . 1281
7.393.1 Typedef Documentation . . . 1281
7.393.1.1 tfm_original_iovec_t . . . 1281
7.393.2 Function Documentation . . . 1281
7.393.2.1 original_iovec() . . . 1282
7.394 secure_fw/spm/include/ffm/psa_api.h File Reference . . . 1282
7.394.1 Macro Definition Documentation . . . 1283
7.394.1.1 tfm_spm_agent_psa_call . . . 1284
7.394.1.2 tfm_spm_agent_psa_close . . . 1284
7.394.1.3 tfm_spm_agent_psa_connect . . . 1284
7.394.1.4 tfm_spm_client_psa_close . . . 1284
7.394.1.5 tfm_spm_client_psa_connect . . . 1284
7.394.1.6 tfm_spm_partition_original_iovec . . . 1284
7.394.1.7 tfm_spm_partition_psa_clear . . . 1284
7.394.1.8 tfm_spm_partition_psa_eoi . . . 1284
7.394.1.9 tfm_spm_partition_psa_irq_disable . . . 1284
7.394.1.10 tfm_spm_partition_psa_irq_enable . . . 1284
7.394.1.11 tfm_spm_partition_psa_notify . . . 1285
7.394.1.12 tfm_spm_partition_psa_reset_signal . . . 1285
7.394.1.13 tfm_spm_partition_psa_set_rhandle . . . 1285
7.394.2 Function Documentation . . . 1285
7.394.2.1 spm_handle_programmer_errors() . . . 1285
7.394.2.2 tfm_spm_client_psa_call() . . . 1286
7.394.2.3 tfm_spm_client_psa_framework_version() . . . 1286
7.394.2.4 tfm_spm_client_psa_version() . . . 1287
7.394.2.5 tfm_spm_get_lifecycle_state() . . . 1288
7.394.2.6 tfm_spm_partition_psa_panic() . . . 1288
7.394.2.7 tfm_spm_partition_psa_read() . . . 1289
7.394.2.8 tfm_spm_partition_psa_reply() . . . 1290
7.394.2.9 tfm_spm_partition_psa_skip() . . . 1290

7.394.2.10 tfm_spm_partition_psa_write() . . . 1291
7.395 secure_fw/spm/include/interface/runtime_defs.h File Reference . . . 1292
7.395.1 Typedef Documentation . . . 1293
7.395.1.1 service_fn_t . . . 1293
7.395.1.2 sfn_init_fn_t . . . 1293
7.396 secure_fw/spm/include/interface/svc_num.h File Reference . . . 1293
7.396.1 Macro Definition Documentation . . . 1294
7.396.1.1 TFM_SVC_AGENT_PSA_CALL . . . 1294
7.396.1.2 TFM_SVC_AGENT_PSA_CLOSE . . . 1294
7.396.1.3 TFM_SVC_AGENT_PSA_CONNECT . . . 1294
7.396.1.4 TFM_SVC_FLIH_FUNC_RETURN . . . 1294
7.396.1.5 TFM_SVC_GET_BOOT_DATA . . . 1295
7.396.1.6 TFM_SVC_IS_HANDLER_MODE . . . 1295
7.396.1.7 TFM_SVC_IS_PLATFORM . . . 1295
7.396.1.8 TFM_SVC_IS_PSA_API . . . 1295
7.396.1.9 TFM_SVC_IS_SPM . . . 1295
7.396.1.10 TFM_SVC_NSID_SHM_INIT . . . 1295
7.396.1.11 TFM_SVC_NUM_HANDLER_MODE_MSK . . . 1295
7.396.1.12 TFM_SVC_NUM_HANDLER_MODE_POS . . . 1295
7.396.1.13 TFM_SVC_NUM_INDEX_MSK . . . 1296
7.396.1.14 TFM_SVC_NUM_PLATFORM_HANDLER . . . 1296
7.396.1.15 TFM_SVC_NUM_PLATFORM_MSK . . . 1296
7.396.1.16 TFM_SVC_NUM_PLATFORM_POS . . . 1296
7.396.1.17 TFM_SVC_NUM_PLATFORM_THREAD . . . 1296
7.396.1.18 TFM_SVC_NUM_PSA_API_MSK . . . 1296
7.396.1.19 TFM_SVC_NUM_PSA_API_POS . . . 1296
7.396.1.20 TFM_SVC_NUM_PSA_API_THREAD . . . 1296
7.396.1.21 TFM_SVC_NUM_SPM_HANDLER . . . 1297
7.396.1.22 TFM_SVC_NUM_SPM_THREAD . . . 1297
7.396.1.23 TFM_SVC_ORIGINAL_IOVEC . . . 1297
7.396.1.24 TFM_SVC_OUTPUT_UNPRIV_STRING . . . 1297
7.396.1.25 TFM_SVC_PREPARE_DEPRIV_FLIH . . . 1297
7.396.1.26 TFM_SVC_PSA_CALL . . . 1297
7.396.1.27 TFM_SVC_PSA_CLEAR . . . 1297
7.396.1.28 TFM_SVC_PSA_CLOSE . . . 1297
7.396.1.29 TFM_SVC_PSA_CONNECT . . . 1297
7.396.1.30 TFM_SVC_PSA_EOI . . . 1298
7.396.1.31 TFM_SVC_PSA_FRAMEWORK_VERSION . . . 1298
7.396.1.32 TFM_SVC_PSA_GET . . . 1298
7.396.1.33 TFM_SVC_PSA_IRQ_DISABLE . . . 1298
7.396.1.34 TFM_SVC_PSA_IRQ_ENABLE . . . 1298
7.396.1.35 TFM_SVC_PSA_LIFECYCLE . . . 1298

7.396.1.36 TFM_SVC_PSA_NOTIFY . . . 1298
7.396.1.37 TFM_SVC_PSA_PANIC . . . 1298
7.396.1.38 TFM_SVC_PSA_READ . . . 1298
7.396.1.39 TFM_SVC_PSA_REPLY . . . 1298
7.396.1.40 TFM_SVC_PSA_RESET_SIGNAL . . . 1299
7.396.1.41 TFM_SVC_PSA_SET_RHANDLE . . . 1299
7.396.1.42 TFM_SVC_PSA_SKIP . . . 1299
7.396.1.43 TFM_SVC_PSA_VERSION . . . 1299
7.396.1.44 TFM_SVC_PSA_WAIT . . . 1299
7.396.1.45 TFM_SVC_PSA_WRITE . . . 1299
7.396.1.46 TFM_SVC_SPM_INIT . . . 1299
7.396.1.47 TFM_SVC_THREAD_MODE_SPM_RETURN . . . 1299
7.397 secure_fw/spm/include/lists.h File Reference . . . 1299
7.397.1 Macro Definition Documentation . . . 1300
7.397.1.1 BI_LIST_FOR_EACH . . . 1300
7.397.1.2 BI_LIST_INIT_NODE . . . 1300
7.397.1.3 BI_LIST_INSERT_AFTER . . . 1300
7.397.1.4 BI_LIST_INSERT_BEFORE . . . 1301
7.397.1.5 BI_LIST_IS_EMPTY . . . 1301
7.397.1.6 BI_LIST_NEXT_NODE . . . 1301
7.397.1.7 BI_LIST_REMOVE_NODE . . . 1301
7.397.1.8 UNI_LISI_INIT_NODE . . . 1301
7.397.1.9 UNI_LIST_FOREACH . . . 1301
7.397.1.10 UNI_LIST_FOREACH_NODE_PNODE . . . 1302
7.397.1.11 UNI_LIST_FOREACH_NODE_PREV . . . 1302
7.397.1.12 UNI_LIST_INSERT_AFTER . . . 1302
7.397.1.13 UNI_LIST_IS_EMPTY . . . 1302
7.397.1.14 UNI_LIST_MOVE_AFTER . . . 1302
7.397.1.15 UNI_LIST_NEXT_NODE . . . 1303
7.397.1.16 UNI_LIST_REMOVE_NODE . . . 1303
7.397.1.17 UNI_LIST_REMOVE_NODE_BY_PNODE . . . 1303
7.398 secure_fw/spm/include/load/asset_defs.h File Reference . . . 1303
7.398.1 Macro Definition Documentation . . . 1304
7.398.1.1 ASSET_ATTR_MMIO . . . 1304
7.398.1.2 ASSET_ATTR_NAMED_MMIO . . . 1304
7.398.1.3 ASSET_ATTR_NUMBERED_MMIO . . . 1304
7.398.1.4 ASSET_ATTR_PPB . . . 1304
7.398.1.5 ASSET_ATTR_READ_ONLY . . . 1304
7.398.1.6 ASSET_ATTR_READ_WRITE . . . 1304
7.399 secure_fw/spm/include/load/interrupt_defs.h File Reference . . . 1305
7.400 secure_fw/spm/include/load/partition_defs.h File Reference . . . 1305
7.400.1 Macro Definition Documentation . . . 1307

7.400.1.1 ENTRY_TO_POSITION . . . 1307
7.400.1.2 INVALID_PARTITION_ID . . . 1307
7.400.1.3 IS_IPC_MODEL . . . 1307
7.400.1.4 IS_NS_AGENT . . . 1307
7.400.1.5 IS_NS_AGENT_MAILBOX . . . 1307
7.400.1.6 IS_NS_AGENT_TZ . . . 1307
7.400.1.7 IS_PSA_ROT . . . 1307
7.400.1.8 PARTITION_INFO_MAGIC . . . 1308
7.400.1.9 PARTITION_INFO_MAGIC_MASK . . . 1308
7.400.1.10 PARTITION_INFO_VERSION_MASK . . . 1308
7.400.1.11 PARTITION_MODEL_IPC . . . 1308
7.400.1.12 PARTITION_MODEL_PSA_ROT . . . 1308
7.400.1.13 PARTITION_NS_AGENT_MB . . . 1308
7.400.1.14 PARTITION_NS_AGENT_TZ . . . 1308
7.400.1.15 PARTITION_PRI_HIGH . . . 1308
7.400.1.16 PARTITION_PRI_HIGHEST . . . 1308
7.400.1.17 PARTITION_PRI_LOW . . . 1309
7.400.1.18 PARTITION_PRI_LOWEST . . . 1309
7.400.1.19 PARTITION_PRI_MASK . . . 1309
7.400.1.20 PARTITION_PRI_NORMAL . . . 1309
7.400.1.21 PARTITION_PRIORITY . . . 1309
7.400.1.22 PARTITION_TYPE_TO_INDEX . . . 1309
7.400.1.23 POSITION_TO_ENTRY . . . 1309
7.400.1.24 PTR_TO_REFERENCE . . . 1309
7.400.1.25 REFERENCE_TO_PTR . . . 1309
7.400.1.26 TFM_SP_IDLE . . . 1310
7.400.1.27 TFM_SP_TZ_AGENT . . . 1310
7.400.1.28 TO_THREAD_PRIORITY . . . 1310
7.401 secure_fw/spm/include/load/service_defs.h File Reference . . . 1310
7.401.1 Macro Definition Documentation . . . 1311
7.401.1.1 SERVICE_ENABLED_MM_IOVEC . . . 1311
7.401.1.2 SERVICE_FLAG_MM_IOVEC . . . 1311
7.401.1.3 SERVICE_FLAG_NS_ACCESSIBLE . . . 1311
7.401.1.4 SERVICE_FLAG_STATELESS . . . 1311
7.401.1.5 SERVICE_FLAG_STATELESS_HINDEX_MASK . . . 1311
7.401.1.6 SERVICE_FLAG_VERSION_POLICY_BIT . . . 1311
7.401.1.7 SERVICE_GET_STATELESS_HINDEX . . . 1312
7.401.1.8 SERVICE_GET_VERSION_POLICY . . . 1312
7.401.1.9 SERVICE_IS_NS_ACCESSIBLE . . . 1312
7.401.1.10 SERVICE_IS_STATELESS . . . 1312
7.401.1.11 SERVICE_VERSION_POLICY_RELAXED . . . 1312
7.401.1.12 SERVICE_VERSION_POLICY_STRICT . . . 1312

7.401.1.13 STRID_TO_STRING_PTR . . . 1312
7.401.1.14 STRING_PTR_TO_STRID . . . 1312
7.402 secure_fw/spm/include/load/spm_load_api.h File Reference . . . 1312
7.402.1 Macro Definition Documentation . . . 1313
7.402.1.1 LOAD_ALLOCED_STACK_ADDR . . . 1313
7.402.1.2 LOAD_INFO_ASSET . . . 1314
7.402.1.3 LOAD_INFO_DEPS . . . 1314
7.402.1.4 LOAD_INFO_EXT_LENGTH . . . 1314
7.402.1.5 LOAD_INFO_IRQ . . . 1314
7.402.1.6 LOAD_INFO_SERVICE . . . 1314
7.402.1.7 LOAD_INFSZ_BYTES . . . 1314
7.402.1.8 NO_MORE_PARTITION . . . 1314
7.402.2 Function Documentation . . . 1315
7.402.2.1 load_a_partition_assuredly() . . . 1315
7.402.2.2 load_irqs_assuredly() . . . 1315
7.402.2.3 load_services_assuredly() . . . 1315
7.403 secure_fw/spm/include/ns_agent_mailbox_defs.h File Reference . . . 1316
7.403.1 Macro Definition Documentation . . . 1316
7.403.1.1 BOOT_NS_CORE . . . 1316
7.403.1.2 DISABLE_MAILBOX . . . 1316
7.403.1.3 ENABLE_MAILBOX . . . 1316
7.403.1.4 NS_CORE_SHUTDOWN . . . 1316
7.404 secure_fw/spm/include/ns_agent_mailbox_interface.h File Reference . . . 1317
7.404.1 Function Documentation . . . 1318
7.404.1.1 ns_agent_boot_ns_core() . . . 1318
7.404.1.2 ns_agent_mailbox_disable() . . . 1318
7.404.1.3 ns_agent_mailbox_enable() . . . 1318
7.404.1.4 ns_agent_ns_core_shutdown() . . . 1319
7.405 secure_fw/spm/include/tfm_arch.h File Reference . . . 1319
7.405.1 Macro Definition Documentation . . . 1321
7.405.1.1 ARCH_CLAIM_CTXCTRL_INSTANCE . . . 1321
7.405.1.2 ARCH_CTXCTRL_ALLOCATE_STACK . . . 1321
7.405.1.3 ARCH_CTXCTRL_ALLOCATED_PTR . . . 1321
7.405.1.4 ARCH_CTXCTRL_EXCRET_PATTERN . . . 1321
7.405.1.5 ARCH_CTXCTRL_INIT . . . 1322
7.405.1.6 ARCH_FLUSH_FP_CONTEXT . . . 1322
7.405.1.7 ARCH_STATE_CTX_SET_R0 . . . 1322
7.405.1.8 PENDSV_PRIO_FOR_SCHED . . . 1322
7.405.1.9 SCHEDULER_ATTEMPTED . . . 1322
7.405.1.10 SCHEDULER_LOCKED . . . 1322
7.405.1.11 SCHEDULER_UNLOCKED . . . 1322
7.405.1.12 TFM_FPU_CONTEXT_SIZE . . . 1323

7.405.1.13 XPSR_T32 . . . 1323
7.405.2 Function Documentation . . . 1323
7.405.2.1 __restore_irq() . . . 1323
7.405.2.2 __save_disable_irq() . . . 1323
7.405.2.3 __set_CONTROL_nPRIV() . . . 1323
7.405.2.4 arch_acquire_sched_lock() . . . 1324
7.405.2.5 arch_attempt_schedule() . . . 1324
7.405.2.6 arch_clean_stack_and_launch() . . . 1324
7.405.2.7 arch_release_sched_lock() . . . 1325
7.405.2.8 tfm_arch_config_extensions() . . . 1325
7.405.2.9 tfm_arch_free_msp_and_exc_ret() . . . 1325
7.405.2.10 tfm_arch_init_context() . . . 1326
7.405.2.11 tfm_arch_is_priv() . . . 1326
7.405.2.12 tfm_arch_refresh_hardware_context() . . . 1326
7.405.2.13 tfm_arch_set_context_ret_code() . . . 1327
7.405.2.14 tfm_arch_set_secure_exception_priorities() . . . 1327
7.405.2.15 tfm_arch_thread_fn_call() . . . 1327
7.406 secure_fw/spm/include/tfm_arch_v8m.h File Reference . . . 1327
7.406.1 Macro Definition Documentation . . . 1329
7.406.1.1 EXC_RETURN_HANDLER . . . 1329
7.406.1.2 EXC_RETURN_RES1 . . . 1329
7.406.1.3 EXC_RETURN_THREAD_MSP . . . 1329
7.406.1.4 EXC_RETURN_THREAD_PSP . . . 1329
7.406.1.5 SCB_ICSR_ADDR . . . 1329
7.406.1.6 SCB_ICSR_PENDSVSET_BIT . . . 1329
7.406.1.7 TFM_NS_EXC_DISABLE . . . 1329
7.406.1.8 TFM_NS_EXC_ENABLE . . . 1329
7.406.2 Function Documentation . . . 1330
7.406.2.1 arch_seal_thread_stack() . . . 1330
7.406.2.2 arch_update_process_sp() . . . 1330
7.406.2.3 is_default_stacking_rules_apply() . . . 1330
7.406.2.4 is_return_secure_stack() . . . 1331
7.406.2.5 is_stack_alloc_fp_space() . . . 1331
7.406.2.6 tfm_arch_check_msp_sealing() . . . 1331
7.406.2.7 tfm_arch_get_psplim() . . . 1332
7.406.2.8 tfm_arch_set_msplim() . . . 1332
7.406.2.9 tfm_arch_set_psplim() . . . 1332
7.406.3 Variable Documentation . . . 1332
7.406.3.1 __STACK_SEAL . . . 1332
7.407 secure_fw/spm/include/tfm_core_trustzone.h File Reference . . . 1333
7.407.1 Macro Definition Documentation . . . 1333
7.407.1.1 TFM_ADDTIONAL_FP_CONTEXT_WORDS . . . 1333

7.407.1.2 TFM_BASIC_FP_CONTEXT_WORDS . . . 1333
7.407.1.3 TFM_STACK_SEAL_VALUE . . . 1334
7.407.1.4 TFM_STACK_SEAL_VALUE_64 . . . 1334
7.407.1.5 TFM_STACK_SEALED_SIZE . . . 1334
7.407.1.6 TFM_VENEER_LR_BIT0_MASK . . . 1334
7.408 secure_fw/spm/include/tfm_exc_num.h File Reference . . . 1334
7.408.1 Macro Definition Documentation . . . 1335
7.408.1.1 EXC_NUM_PENDSV . . . 1335
7.408.1.2 EXC_NUM_SVCALL . . . 1335
7.408.1.3 EXC_NUM_THREAD_MODE . . . 1335
7.408.2 Function Documentation . . . 1335
7.408.2.1 __get_active_exc_num() . . . 1335
7.409 secure_fw/spm/include/tfm_nspm.h File Reference . . . 1336
7.409.1 Macro Definition Documentation . . . 1337
7.409.1.1 __tz_c_veneer . . . 1337
7.409.1.2 TFM_NS_CLIENT_INVALID_ID . . . 1337
7.409.2 Function Documentation . . . 1337
7.409.2.1 tfm_nspm_ctx_init() . . . 1337
7.409.2.2 tfm_nspm_get_current_client_id() . . . 1338
7.409.2.3 tz_ns_agent_register_client_id_range() . . . 1338
7.410 secure_fw/spm/include/tfm_spm_log.h File Reference . . . 1339
7.410.1 Macro Definition Documentation . . . 1340
7.410.1.1 SPMLOG_DBGMSG . . . 1340
7.410.1.2 SPMLOG_DBGMSGVAL . . . 1340
7.410.1.3 SPMLOG_ERRMSG . . . 1340
7.410.1.4 SPMLOG_ERRMSGVAL . . . 1340
7.410.1.5 SPMLOG_INFMSG . . . 1340
7.410.1.6 SPMLOG_INFMSGVAL . . . 1340
7.410.1.7 TFM_SPM_LOG_LEVEL_DEBUG . . . 1340
7.410.1.8 TFM_SPM_LOG_LEVEL_ERROR . . . 1340
7.410.1.9 TFM_SPM_LOG_LEVEL_INFO . . . 1341
7.410.1.10 TFM_SPM_LOG_LEVEL_SILENCE . . . 1341
7.410.2 Function Documentation . . . 1341
7.410.2.1 spm_log_msgval() . . . 1341
7.411 secure_fw/spm/include/utilities.h File Reference . . . 1342
7.411.1 Macro Definition Documentation . . . 1342
7.411.1.1 ERROR_MSG . . . 1342
7.411.1.2 M2S . . . 1342
7.411.1.3 SPM_ASSERT . . . 1343
7.411.1.4 spm_memcpy . . . 1343
7.411.1.5 spm_memset . . . 1343
7.411.1.6 STRINGIFY_EXPAND . . . 1343

7.411.1.7 TO_CONTAINER . . . 1343
7.411.2 Function Documentation . . . 1343
7.411.2.1 tfm_core_panic() . . . 1343
7.412 secure_fw/spm/ns_client_ext/tfm_ns_client_ext.c File Reference . . . 1343
7.412.1 Macro Definition Documentation . . . 1344
7.412.1.1 IS_INVALID_TOKEN . . . 1344
7.412.1.2 MAKE_NS_CLIENT_TOKEN . . . 1344
7.412.1.3 NS_CLIENT_TOKEN_TO_CTX_IDX . . . 1345
7.412.1.4 NS_CLIENT_TOKEN_TO_GID . . . 1345
7.412.1.5 NS_CLIENT_TOKEN_TO_TID . . . 1345
7.412.2 Function Documentation . . . 1345
7.412.2.1 tfm_nsce_acquire_ctx() . . . 1345
7.412.2.2 tfm_nsce_init() . . . 1346
7.412.2.3 tfm_nsce_load_ctx() . . . 1346
7.412.2.4 tfm_nsce_release_ctx() . . . 1347
7.412.2.5 tfm_nsce_save_ctx() . . . 1348
7.412.3 Variable Documentation . . . 1348
7.412.3.1 ns_ctx_mgr . . . 1349
7.413 secure_fw/spm/ns_client_ext/tfm_ns_ctx.c File Reference . . . 1349
7.413.1 Macro Definition Documentation . . . 1349
7.413.1.1 TFM_NS_CONTEXT_MAX . . . 1349
7.413.1.2 TFM_NS_CONTEXT_MAX_TID . . . 1350
7.413.2 Function Documentation . . . 1350
7.413.2.1 acquire_ns_ctx() . . . 1350
7.413.2.2 load_ns_ctx() . . . 1350
7.413.2.3 release_ns_ctx() . . . 1350
7.413.2.4 save_ns_ctx() . . . 1351
7.413.3 Variable Documentation . . . 1351
7.413.3.1 ns_ctx_mgr . . . 1351
7.414 secure_fw/spm/ns_client_ext/tfm_ns_ctx.h File Reference . . . 1351
7.414.1 Function Documentation . . . 1352
7.414.1.1 acquire_ns_ctx() . . . 1352
7.414.1.2 get_nsid_from_active_ns_ctx() . . . 1353
7.414.1.3 init_ns_ctx() . . . 1353
7.414.1.4 load_ns_ctx() . . . 1354
7.414.1.5 release_ns_ctx() . . . 1354
7.414.1.6 save_ns_ctx() . . . 1355
7.415 secure_fw/spm/ns_client_ext/tfm_ns_ctx_s.c File Reference . . . 1355
7.415.1 Function Documentation . . . 1355
7.415.1.1 get_nsid_from_active_ns_ctx() . . . 1356
7.415.1.2 init_ns_ctx() . . . 1356
7.415.2 Variable Documentation . . . 1356

7.415.2.1 ns_ctx_mgr . . . . . 1356
7.416 secure_fw/spm/ns_client_ext/tfm_ns_ctx_shm.h File Reference . . . . . 1357
7.417 secure_fw/spm/ns_client_ext/tfm_spm_ns_ctx.c File Reference . . . . . 1357
7.417.1 Macro Definition Documentation . . . . . 1358
7.417.1.1 DEFAULT_NS_CLIENT_ID . . . . . 1358
7.417.2 Function Documentation . . . . . 1358
7.417.2.1 tfm_nspm_ctx_init() . . . . . 1358
7.417.2.2 tfm_nspm_get_current_client_id() . . . . . 1359
7.417.2.3 tz_ns_agent_register_client_id_range() . . . . . 1359

# Chapter 1

# TF-M Reference Manual

This document gives a reference for the TF-M core and services. This guide does not cover the non-secure application example, the boot-loader and the target platform implementations.

# Chapter 2

# Module Index

## 2.1 Modules

Here is a list of all modules:


Set of functions implementing a thin shim . . . . . 19
Function that implement allocation and deallocation of . . . . . 27
Set of functions implementing the abstractions of the underlying cryptographic . . . . . 30
Tfm_builtin_key_loader . . . . . 34

# Chapter 3

# Data Structure Index

## 3.1 Data Structures

Here are the data structures with brief descriptions:


_MTB_SRF_DATA_ALIGN
MTB SRF IPC request structure suitable for TFM needs. This structure overrides default mtb↩_srf_ipc_packet_t SRF structure . . . 37
asset_desc_t . . . 39
attest_boot_data
Contains the received boot status information from bootloader . . . 41
attest_token_encode_ctx . . . 42
bi_list_node_t . . . 43
boot_data_access_policy
Defines the access policy of secure partitions to data items in shared data area (between bootloader and runtime firmware) . . . 44
client_id_region_t . . . 45
client_params_t . . . 46
composite_addr_t . . . 47
connection_t . . . 48
context_ctrl_t . . . 51
context_flih_ret_t . . . 52
critical_section_t . . . 54
formatted_buffer_t . . . 55
full_context_t . . . 55
fwu_image_info_data_s . . . 56
ifx_debug_policy_t . . . 57
ifx_driver_flash_obj_t
Structure containing the FLASH driver instance parameters . . . 58
ifx_driver_rram_obj_t
Structure containing the RRAM driver instance parameters . . . 59
ifx_driver_smif_mem_t
Structure containing the SMIF driver memory configuration . . . 61
ifx_driver_smif_obj_t
Structure containing the SMIF driver instance data . . . 63
ifx_flash_driver_instance_t
Flash driver instance . . . 65
ifx_flash_driver_t
Access structure of the Flash Driver . . . 66
ifx_memory_config_t . . . 68

ifx_mpc_cfg_t . . . 70
ifx_mpc_ext_cache_cfg_info_t . . . 71
ifx_mpc_numbered_mmio_config_t . . . 72
ifx_mpc_policy_t . . . 73
ifx_mpc_raw_region_config_t
Configuration structure for MPC raw region . . . 74
ifx_mpc_region_config_t . . . 76
ifx_msc_agc_resp_config_t . . . 78
ifx_msc_agc_resp_config_v1_t . . . 79
ifx_nv_counters
Struct representing the NV counter data in flash . . . 80
ifx_oem_policy_t . . . 81
ifx_partition_info_t
Structure describing partition info . . . 83
ifx_partition_load_info_t
Structure used to store platform specific partition properties . . . 86
ifx_plat_epc2_keys_t . . . 87
ifx_ppc_named_mmio_config_t . . . 88
ifx_ppc_regions_config_t . . . 90
ifx_ppcx_config_t . . . 90
ifx_protection_region_config_t
Structure used to store platform specific protection properties . . . 92
ifx_rram_counter_t . . . 93
irq_load_info_t . . . 94
irq_t . . . 96
its_block_meta_t
Structure to store information about each physical flash memory block . . . 97
its_file_meta_t
Structure to store file metadata . . . 98
its_flash_config_t
Structure containing ITS flash driver instance configuration used by ITS flash FS layer . . . 100
its_flash_fs_config_t
Structure containing the flash filesystem configuration parameters . . . 102
its_flash_fs_ctx_t
Structure to store the ITS flash file system context . . . 103
its_flash_fs_file_info_t
Structure containing file information . . . 105
its_flash_nand_dev_t . . . 106
its_flash_ops_t
Structure containing the ITS flash driver operations . . . 108
its_flash_ram_dev_t
ITS RAM driver context . . . 111
its_metadata_block_header_comp_t
The struture of metadata block header in ITS_BACKWARD_SUPPORTED_VERSION . . . 112
its_metadata_block_header_t
Structure to store the metadata block header . . . 113
mailbox_init_t . . . 115
mailbox_msg_t . . . 116
mailbox_reply_t . . . 117
mailbox_slot_t . . . 118
mailbox_status_t . . . 119
mpu_armv8m_region_cfg_t . . . 120
ns_mailbox_queue_t . . . 121
ns_mailbox_req_t . . . 123
ns_mailbox_slot_t . . . 125
ns_mailbox_stats_res_t
The structure to hold the statistics result of NSPE mailbox . . . 126
partition_head_t . . . 127

partition_load_info_t . . . 128
partition_t . . . 131
partition_tfm_sp_idle_load_info_t . . . 133
partition_tfm_sp_ns_agent_tz_load_info_t . . . 134
ps_crypto_t . . . 135
ps_obj_header_t
Metadata attached as a header to object data before storage . . . 137
ps_obj_table_ctx_t
Object table context structure . . . 139
ps_obj_table_entry_t . . . 140
ps_obj_table_info_t
Object table information structure . . . 141
ps_obj_table_init_ctx_t . . . 142
ps_obj_table_t
Object table structure . . . 143
ps_object_info_t
Object information . . . 145
ps_object_t
The object to be written to the file system below. Made up of the object header and the object data 146
psa_client_params_t . . . 147
psa_fwu_component_info_t
Information about the firmware store for a firmware component . . . 150
psa_fwu_image_version_t
Version information about a firmware image . . . 152
psa_fwu_impl_info_t
The implementation-specific data in the component information structure . . . 154
psa_invec . . . 154
psa_msg_t . . . 155
psa_outvec . . . 157
psa_storage_info_t . . . 158
service_head_t . . . 159
service_load_info_t . . . 160
service_t . . . 161
shared_data_tlv_entry . . . 162
shared_data_tlv_header . . . 163
tfm_additional_context_t . . . 164
tfm_boot_data
Store the data for the runtime SW . . . 164
tfm_builtin_key_t
A structure which describes a builtin key slot . . . 165
tfm_crypto_aead_pack_input
This type is used to overcome a limitation in the number of maximum IOVECs that can be used especially in psa_aead_encrypt and psa_aead_decrypt. By using this type we pack the nonce and the actual nonce_length at part of the same structure . . . 166
tfm_crypto_key_id_s
The type which describes a key identifier to the Crypto service. The key identifiers must clearly provide a dedicated indication of the entity owner which owns the key . . . 167
tfm_crypto_operation_s
A type describing the context stored in Secure memory by the TF-M Crypto service to support multipart calls on secure side . . . 168
tfm_crypto_pack_iovec
Structure used to pack non-pointer types in a call to PSA Crypto APIs . . . 169
tfm_fwu_ctx_s . . . 171
tfm_fwu_mcuboot_ctx_s . . . 172
tfm_ns_ctx_mgr_t . . . 172
tfm_ns_ctx_t . . . 173
tfm_original_iovec_t . . . 174
tfm_pool_chunk_t . . . 176

tfm_pool_instance_t . . . . . 176
tfm_state_context_t . . . . . 178
thread_t . . . . . 179
u64_in_u32_regs_t . . . . . 180
vectors . . . . . 181

# Chapter 4

# File Index

## 4.1 File List

Here is a list of all files with brief descriptions:


interface/include/tfm_attest_defs.h . . . 258
interface/include/tfm_attest_iat_defs.h . . . 259
interface/include/tfm_crypto_defs.h . . . 260
interface/include/tfm_fwu_defs.h . . . 268
interface/include/tfm_its_defs.h . . . 269
interface/include/tfm_ns_client_ext.h . . . 270
interface/include/tfm_ns_interface.h . . . 275
interface/include/tfm_platform_api.h . . . 278
interface/include/tfm_ps_defs.h . . . 284
interface/include/tfm_psa_call_pack.h . . . 285
interface/include/tfm_veneers.h . . . 289
interface/include/crypto_keys/tfm_builtin_key_ids.h . . . 183
interface/include/multi_core/tfm_mailbox.h . . . 185
interface/include/multi_core/tfm_multi_core_api.h . . . 189
interface/include/multi_core/tfm_ns_mailbox.h . . . 191
interface/include/multi_core/tfm_ns_mailbox_test.h . . . 199
interface/include/os_wrapper/common.h . . . 201
interface/include/os_wrapper/kernel.h . . . 202
interface/include/os_wrapper/mutex.h . . . 203
interface/include/psa/client.h . . . 206
interface/include/psa/error.h
Standard error codes for the SPM and RoT Services . . . 214
interface/include/psa/internal_trusted_storage.h . . . 218
interface/include/psa/lifecycle.h . . . 224
interface/include/psa/protected_storage.h . . . 226
interface/include/psa/service.h . . . 233
interface/include/psa/storage_common.h . . . 245
interface/include/psa/update.h . . . 247
interface/src/tfm_attest_api.c . . . 307
interface/src/tfm_crypto_api.c . . . 308
interface/src/tfm_fwu_api.c . . . 326
interface/src/tfm_its_api.c . . . 332
interface/src/tfm_platform_api.c . . . 337
interface/src/tfm_ps_api.c . . . 340
interface/src/tfm_psa_call.c . . . 347

interface/src/tfm_tz_psa_ns_api.c . . . 349
interface/src/multi_core/tfm_multi_core_ns_api.c . . . 293
interface/src/multi_core/tfm_multi_core_psa_ns_api.c . . . 294
interface/src/multi_core/tfm_ns_mailbox.c . . . 299
interface/src/multi_core/tfm_ns_mailbox_thread.c . . . 300
interface/src/os_wrapper/tfm_ns_interface_bare_metal.c . . . 303
interface/src/os_wrapper/tfm_ns_interface_rtos.c . . . 304
platform/ext/target/infineon/common/device/include/device_definition.h . . . 355
platform/ext/target/infineon/common/device/include/ifx_startup.h . . . 356
platform/ext/target/infineon/common/device/src/device_definition.c
This file defines exports the structures based on the peripheral definitions from device_cfg.h. This retarget file is meant to be used as a helper for baremetal applications and/or as an example of how to configure the generic driver structures . . . 358
platform/ext/target/infineon/common/drivers/assets/ifx_assets_rram.c . . . 359
platform/ext/target/infineon/common/drivers/assets/ifx_assets_rram.h . . . 360
platform/ext/target/infineon/common/drivers/flash/ifx_driver_private.c . . . 366
platform/ext/target/infineon/common/drivers/flash/ifx_driver_private.h . . . 368
platform/ext/target/infineon/common/drivers/flash/ifx_flash_driver_api.h . . . 370
platform/ext/target/infineon/common/drivers/flash/flash/ifx_driver_flash.c . . . 363
platform/ext/target/infineon/common/drivers/flash/flash/ifx_driver_flash.h . . . 365
platform/ext/target/infineon/common/drivers/flash/rram/ifx_driver_rram.c . . . 372
platform/ext/target/infineon/common/drivers/flash/rram/ifx_driver_rram.h . . . 375
platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif.c . . . 377
platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif.h . . . 383
platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_mmio.c . . . 385
platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_private.h . . . 386
platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_xip.c . . . 392
platform/ext/target/infineon/common/drivers/protection/mpu_armv8m_drv.h . . . 393
platform/ext/target/infineon/common/drivers/stdio/uart_pdl_stdout.c . . . 396
platform/ext/target/infineon/common/drivers/stdio/uart_pdl_stdout.h . . . 398
platform/ext/target/infineon/common/interface/include/ifx_platform_api.h . . . 399
platform/ext/target/infineon/common/interface/include/mtb_srf_ipc_custom_packet.h . . . 402
platform/ext/target/infineon/common/interface/include/multi_core/tfm_mailbox_config.h . . . 403
platform/ext/target/infineon/common/interface/include/multi_core/tfm_ns_mailbox.h . . . 191
platform/ext/target/infineon/common/interface/src/ifx_mtb_srf.c . . . 403
platform/ext/target/infineon/common/interface/src/ifx_mtb_srf_relay.c . . . 405
platform/ext/target/infineon/common/interface/src/ifx_platform_api.c . . . 405
platform/ext/target/infineon/common/interface/src/ifx_platform_private.h . . . 407
platform/ext/target/infineon/common/interface/src/multi_core/ifx_mtb_mailbox.c . . . 408
platform/ext/target/infineon/common/libs/ifx_se_rt_services_utils/spe/ifx_se_tfm_utils.h
This file is a wrapper for ifx-se-rt-services-utils library for TF-M. It adds additional stuff needed to use this library from within secure partitions . . . 410
platform/ext/target/infineon/common/nspe/mailbox/platform_ns_mailbox.c . . . 411
platform/ext/target/infineon/common/nspe/mailbox/tfm_ns_mailbox_rtos_api.c . . . 416
platform/ext/target/infineon/common/nspe/os_wrapper/os_wrapper_cyabs_rtos.c . . . 419
platform/ext/target/infineon/common/nspe/os_wrapper/semaphore.h . . . 419
platform/ext/target/infineon/common/nspe/os_wrapper/thread.h . . . 1254
platform/ext/target/infineon/common/nspe/test/ifx_fpu_ns.c . . . 423
platform/ext/target/infineon/common/nspe/test/plat_test_ns.c . . . 425
platform/ext/target/infineon/common/shared/mailbox/ifx_platform_mailbox.h . . . 425
platform/ext/target/infineon/common/shared/mailbox/platform_multicore.c . . . 426
platform/ext/target/infineon/common/shared/mailbox/platform_multicore.h . . . 430
platform/ext/target/infineon/common/spe/faults/faults.c . . . 434
platform/ext/target/infineon/common/spe/faults/faults.h . . . 438
platform/ext/target/infineon/common/spe/faults/faults_cat1b.c . . . 443
platform/ext/target/infineon/common/spe/faults/faults_cat1d.c . . . 447
platform/ext/target/infineon/common/spe/faults/faults_cat1e.c . . . 451
platform/ext/target/infineon/common/spe/faults/faults_dump.h . . . 459

platform/ext/target/infineon/common/spe/fih/tfm_fih_mxcrypto.c . . . 463
platform/ext/target/infineon/common/spe/otp/otp_flash.c . . . 465
platform/ext/target/infineon/common/spe/protection/partition_assets.h . . . 466
platform/ext/target/infineon/common/spe/protection/partition_info.h . . . 467
platform/ext/target/infineon/common/spe/protection/protection_mpc_api.h
This file contains MPC driver API declaration . . . 468
platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc.c . . . 470
platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc.h
This file contains types/API used by HW MPC driver, see cy_mpc.h in PDL . . . 476
platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc_v1.c . . . 498
platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc_v2.c . . . 508
platform/ext/target/infineon/common/spe/protection/protection_mpc_sert.c . . . 517
platform/ext/target/infineon/common/spe/protection/protection_mpc_sert.h
This file contains types/API used by HW MPC driver, to protect memory via SE RT Basic/Full . 522
platform/ext/target/infineon/common/spe/protection/protection_mpc_sw_policy.c . . . 525
platform/ext/target/infineon/common/spe/protection/protection_mpc_sw_policy.h
This file contains types/API used by MPC SW Policy driver . . . 532
platform/ext/target/infineon/common/spe/protection/protection_mpu.c . . . 533
platform/ext/target/infineon/common/spe/protection/protection_mpu.h . . . 538
platform/ext/target/infineon/common/spe/protection/protection_msc.c . . . 540
platform/ext/target/infineon/common/spe/protection/protection_msc.h . . . 544
platform/ext/target/infineon/common/spe/protection/protection_pc.c . . . 564
platform/ext/target/infineon/common/spe/protection/protection_pc.h . . . 571
platform/ext/target/infineon/common/spe/protection/protection_ppc_api.h
This file contains PPC driver API declaration . . . 591
platform/ext/target/infineon/common/spe/protection/protection_ppc_v1.c . . . 593
platform/ext/target/infineon/common/spe/protection/protection_ppc_v1.h
This file contains types/API used by PPC driver . . . 597
platform/ext/target/infineon/common/spe/protection/protection_ppc_v2.c . . . 598
platform/ext/target/infineon/common/spe/protection/protection_ppc_v2.h
This file contains types/API used by PPC driver . . . 601
platform/ext/target/infineon/common/spe/protection/protection_sau.c . . . 602
platform/ext/target/infineon/common/spe/protection/protection_sau.h . . . 607
platform/ext/target/infineon/common/spe/protection/protection_shared_data.c . . . 627
platform/ext/target/infineon/common/spe/protection/protection_shared_data.h . . . 628
platform/ext/target/infineon/common/spe/protection/protection_types.h . . . 630
platform/ext/target/infineon/common/spe/protection/protection_tz.c . . . 633
platform/ext/target/infineon/common/spe/protection/protection_tz.h . . . 639
platform/ext/target/infineon/common/spe/protection/protection_utils.c . . . 640
platform/ext/target/infineon/common/spe/protection/protection_utils.h . . . 642
platform/ext/target/infineon/common/spe/protection/tfm_hal_isolation.c . . . 643
platform/ext/target/infineon/common/spe/protection/tfm_platform_arch_hooks.c . . . 654
platform/ext/target/infineon/common/spe/protection/tfm_platform_arch_hooks.h . . . 655
platform/ext/target/infineon/common/spe/provisioning/provisioning.c . . . 660
platform/ext/target/infineon/common/spe/provisioning/provisioning.h . . . 662
platform/ext/target/infineon/common/spe/services/attestation/ifx_attest_hal.c . . . 664
platform/ext/target/infineon/common/spe/services/attestation/ifx_attest_hal_se_rt.c . . . 667
platform/ext/target/infineon/common/spe/services/attestation/ifx_plat_device_id.c . . . 668
platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_flash.c . . . 670
platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_rram.c . . . 671
platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_se_rt.c . . . 672
platform/ext/target/infineon/common/spe/services/crypto/crypto_nv_seed.c . . . 673
platform/ext/target/infineon/common/spe/services/crypto/crypto_rnd_se_rt.c . . . 675
platform/ext/target/infineon/common/spe/services/crypto/mbedtls_accel_configs/crypto_hw_cryptolite_config.h
675
platform/ext/target/infineon/common/spe/services/crypto/mbedtls_accel_configs/crypto_hw_mxcrypto_config.h
675
platform/ext/target/infineon/common/spe/services/its/its_hal.c . . . 676

platform/ext/target/infineon/common/spe/services/its/drivers/its_flash_driver.c . . . 675
platform/ext/target/infineon/common/spe/services/its/drivers/its_rram_driver.c . . . 676
platform/ext/target/infineon/common/spe/services/mailbox/platform_hal_multi_core_cm55.c . . . 677
platform/ext/target/infineon/common/spe/services/mailbox/platform_spe_mailbox.c . . . 678
platform/ext/target/infineon/common/spe/services/mailbox/tfm_hal_multi_core.c . . . 680
platform/ext/target/infineon/common/spe/services/mailbox/tfm_interrupts.c . . . 681
platform/ext/target/infineon/common/spe/services/platform/nv_counters_flash.c . . . 686
platform/ext/target/infineon/common/spe/services/platform/nv_counters_rram.c . . . 690
platform/ext/target/infineon/common/spe/services/platform/nv_counters_se_rt.c . . . 694
platform/ext/target/infineon/common/spe/services/platform/tfm_platform_system.c . . . 695
platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_flash_driver.c . . . 682
platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_flash_driver.h . . . 682
platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_rram_driver.c . . . 684
platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_rram_driver.h . . . 685
platform/ext/target/infineon/common/spe/services/ps/ps_hal.c . . . 699
platform/ext/target/infineon/common/spe/services/ps/ps_keys_se_rt.c . . . 700
platform/ext/target/infineon/common/spe/services/ps/drivers/ps_rram_driver.c . . . 698
platform/ext/target/infineon/common/spe/services/ps/drivers/ps_smif_driver.c . . . 698
platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_ipc.c . . . 701
platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_req_mngr.c . . . 702
platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_spm.c . . . 703
platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_spm.h . . . 704
platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall.h . . . 706
platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall_direct.c . . . 708
platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall_partition.c 708
platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test.c . . . 711
platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test.h . . . 712
platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test_non_secure_timer.c . . . 713
platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test_secure_timer.c . . . 713
platform/ext/target/infineon/common/spe/services/tfm_tests/tfm_interrupts_test_s.c . . . 715
platform/ext/target/infineon/common/spe/svc/platform_svc_api.c . . . 715
platform/ext/target/infineon/common/spe/svc/platform_svc_api.h . . . 718
platform/ext/target/infineon/common/spe/svc/platform_svc_handler.c . . . 721
platform/ext/target/infineon/common/spe/test/ifx_fpu_s.c . . . 722
platform/ext/target/infineon/common/spe/test/ifx_fpu_s.h . . . 723
platform/ext/target/infineon/common/spe/v80m/target_cfg.c . . . 724
platform/ext/target/infineon/common/spe/v80m/target_cfg.h . . . 730
platform/ext/target/infineon/common/spe/v80m/tfm_hal_platform.c . . . 753
platform/ext/target/infineon/common/utils/ifx_boot_shared_data.c . . . 758
platform/ext/target/infineon/common/utils/ifx_boot_shared_data.h . . . 759
platform/ext/target/infineon/common/utils/ifx_fih.h . . . 759
platform/ext/target/infineon/common/utils/ifx_interrupt_defs.h . . . 761
platform/ext/target/infineon/common/utils/ifx_regions.c . . . 762
platform/ext/target/infineon/common/utils/ifx_regions.h . . . 763
platform/ext/target/infineon/common/utils/ifx_utils.h . . . 767
platform/ext/target/infineon/common/utils/mxs22.h . . . 768
platform/ext/target/infineon/common/utils/se/ifx_se_crc32.c
Fast high-distance 32-bit CRC for data integrity protection . . . 769
platform/ext/target/infineon/common/utils/se/ifx_se_crc32.h
Fast high-distance 32-bit CRC for data integrity protection . . . 774
platform/ext/target/infineon/pse84/config_tfm_target.h . . . 810
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/ifx_peripherals_def.h . . . 793
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/device/include/project_memory_layout.h
779
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/device/include/startup_pse84.h . . . 790
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/device/source/startup_pse84.c . . . 791
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/ifx_s_peripherals_def.h . . . 795
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/ifx_spm_init.c . . . 797

platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/ifx_spm_init.h . . . 798
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection_regions_cfg.c . . . 799
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection_regions_cfg.h . . . 801
platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/shared_ro_data.c
This file contains shared data that can be read/write within SPM and read-only from any secure partition . . . 804
platform/ext/target/infineon/pse84/config/ifx_platform_spe_types.h
This file contains platform specific types and data declaration used to build secure image . . . 804
platform/ext/target/infineon/pse84/config/pse84_spe_config.h
This file contains PSE84 family specific configuration used to build secure image . . . 807
platform/ext/target/infineon/pse84/epc2/ifx_platform_config.h . . . 812
platform/ext/target/infineon/pse84/epc2/ifx_platform_spe_config.h
This file contains platform specific configuration used to build secure image . . . 813
platform/ext/target/infineon/pse84/epc2/board/KIT_PSOCE84_EVK/config_bsp.h . . . 811
platform/ext/target/infineon/pse84/epc2/spe/services/crypto/crypto_keys_rram.h . . . 816
platform/ext/target/infineon/pse84/epc2/spe/services/crypto/mbedtls_target_config_pse84.h . . . 819
platform/ext/target/infineon/pse84/epc2/spe/services/crypto/platform_builtin_key_loader_ids.h . . . 819
platform/ext/target/infineon/pse84/epc2/spe/services/crypto/tfm_builtin_key_ids.h . . . 184
platform/ext/target/infineon/pse84/epc2/tests/ifx_isolation_test_data.c . . . 820
platform/ext/target/infineon/pse84/epc2/tests/ifx_platform_tests_config.h . . . 821
platform/ext/target/infineon/pse84/shared/platform_nv_counters_ids.h . . . 829
platform/ext/target/infineon/pse84/shared/tfm_peripherals_def.h . . . 830
platform/ext/target/infineon/pse84/shared/device/config/device_cfg.h
Configuration file native driver re-targeting . . . 822
platform/ext/target/infineon/pse84/shared/device/include/cmsis.h . . . 822
platform/ext/target/infineon/pse84/shared/device/include/pse84_core_interrupts.h . . . 823
platform/ext/target/infineon/pse84/shared/partition/flash_layout.h . . . 824
platform/ext/target/infineon/pse84/shared/partition/pse84_s_linker_alignments.h . . . 825
platform/ext/target/infineon/pse84/shared/partition/region_defs.h . . . 827
platform/ext/target/infineon/pse84/spe/platform_otp_ids.h . . . 833
platform/ext/target/infineon/pse84/spe/protection/platform_partition_assets.h . . . 834
platform/ext/target/infineon/pse84/spe/protection/protection_data.c . . . 835
platform/ext/target/infineon/pse84/spe/provisioning/ifx_platform_provisioning.c . . . 837
platform/ext/target/infineon/pse84/tests/secure/s_test.c . . . 840
secure_fw/include/array.h . . . 841
secure_fw/include/async.h . . . 841
secure_fw/include/build_config_check.h . . . 842
secure_fw/include/compiler_ext_defs.h . . . 843
secure_fw/include/config_tfm.h . . . 844
secure_fw/include/crt_impl_private.h . . . 844
secure_fw/include/security_defs.h . . . 845
secure_fw/partitions/crypto/config_crypto_check.h . . . 845
secure_fw/partitions/crypto/config_engine_buf.h . . . 847
secure_fw/partitions/crypto/crypto_aead.c . . . 847
secure_fw/partitions/crypto/crypto_alloc.c . . . 847
secure_fw/partitions/crypto/crypto_asymmetric.c . . . 849
secure_fw/partitions/crypto/crypto_check_config.h . . . 849
secure_fw/partitions/crypto/crypto_cipher.c . . . 850
secure_fw/partitions/crypto/crypto_hash.c . . . 851
secure_fw/partitions/crypto/crypto_init.c . . . 851
secure_fw/partitions/crypto/crypto_key_derivation.c . . . 855
secure_fw/partitions/crypto/crypto_key_management.c . . . 855
secure_fw/partitions/crypto/crypto_library.c . . . 856
secure_fw/partitions/crypto/crypto_library.h
This file contains some abstractions required to interface the TF-M Crypto service to an underlying cryptographic library that implements the PSA Crypto API. The TF-M Crypto service uses this library to provide a PSA Crypto core layer implementation and a software or hardware based implementation of crypto algorithms . . . 857

secure_fw/partitions/crypto/crypto_mac.c . . . 858
secure_fw/partitions/crypto/crypto_rng.c . . . 859
secure_fw/partitions/crypto/crypto_spe.h
When Mbed Crypto is built with the MBEDTLS_PSA_CRYPTO_SPM option enabled, this header is included by all .c files in Mbed Crypto that use PSA Crypto function names. This avoids duplication of symbols between TF-M and Mbed Crypto . . . 859
secure_fw/partitions/crypto/tfm_crypto_api.h . . . 871
secure_fw/partitions/crypto/tfm_crypto_key.h . . . 875
secure_fw/partitions/crypto/tfm_mbedcrypto_alt.c . . . 876
secure_fw/partitions/crypto/tfm_mbedcrypto_include.h . . . 876
secure_fw/partitions/crypto/psa_driver_api/tfm_builtin_key_loader.c . . . 868
secure_fw/partitions/crypto/psa_driver_api/tfm_builtin_key_loader.h . . . 870
secure_fw/partitions/firmware_update/tfm_fwu_req_mngr.c . . . 887
secure_fw/partitions/firmware_update/bootloader/tfm_bootloader_fwu_abstraction.h . . . 883
secure_fw/partitions/firmware_update/bootloader/mcuboot/tfm_mcuboot_fwu.c . . . 877
secure_fw/partitions/idle_partition/idle_partition.c . . . 889
secure_fw/partitions/idle_partition/load_info_idle_sp.c . . . 890
secure_fw/partitions/initial_attestation/attest.h . . . 892
secure_fw/partitions/initial_attestation/attest_asymmetric_key.c . . . 897
secure_fw/partitions/initial_attestation/attest_boot_data.c . . . 898
secure_fw/partitions/initial_attestation/attest_boot_data.h . . . 900
secure_fw/partitions/initial_attestation/attest_core.c . . . 903
secure_fw/partitions/initial_attestation/attest_key.h . . . 905
secure_fw/partitions/initial_attestation/attest_symmetric_key.c . . . 906
secure_fw/partitions/initial_attestation/attest_token.h
Attestation Token Creation Interface . . . 908
secure_fw/partitions/initial_attestation/attest_token_encode.c
Attestation token creation implementation . . . 914
secure_fw/partitions/initial_attestation/tfm_attest.c . . . 917
secure_fw/partitions/initial_attestation/tfm_attest_req_mngr.c . . . 919
secure_fw/partitions/internal_trusted_storage/its_crypto_interface.c . . . 980
secure_fw/partitions/internal_trusted_storage/its_crypto_interface.h . . . 981
secure_fw/partitions/internal_trusted_storage/its_utils.c . . . 983
secure_fw/partitions/internal_trusted_storage/its_utils.h . . . 984
secure_fw/partitions/internal_trusted_storage/tfm_internal_trusted_storage.c . . . 989
secure_fw/partitions/internal_trusted_storage/tfm_internal_trusted_storage.h . . . 995
secure_fw/partitions/internal_trusted_storage/tfm_its_req_mngr.c . . . 1001
secure_fw/partitions/internal_trusted_storage/tfm_its_req_mngr.h . . . 1003
secure_fw/partitions/internal_trusted_storage/flash/its_flash.c . . . 921
secure_fw/partitions/internal_trusted_storage/flash/its_flash.h . . . 922
secure_fw/partitions/internal_trusted_storage/flash/its_flash_hal.h
Internal Trusted Storage file system driver abstraction APIs. The purpose of this abstraction is to provide interface between ITS file system and implementation of storage back-end . . . 924
secure_fw/partitions/internal_trusted_storage/flash/its_flash_nand.c . . . 925
secure_fw/partitions/internal_trusted_storage/flash/its_flash_nand.h
Implementations of the flash interface functions for a NAND flash device. See its_flash_ops_t for full documentation of functions . . . 926
secure_fw/partitions/internal_trusted_storage/flash/its_flash_nor.c . . . 928
secure_fw/partitions/internal_trusted_storage/flash/its_flash_nor.h
Implementations of the flash interface functions for a NOR flash device. See its_flash_ops_t for full documentation of functions . . . 929
secure_fw/partitions/internal_trusted_storage/flash/its_flash_ram.c . . . 930
secure_fw/partitions/internal_trusted_storage/flash/its_flash_ram.h
Implementations of the flash interface functions for an emulated flash device using RAM. See its_flash_ops_t for full documentation of functions . . . 931
secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs.c . . . 932

secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs.h
Internal Trusted Storage service filesystem abstraction APIs. The purpose of this abstraction is to have the ability to plug-in other filesystems or filesystem proxies (supplicant) . . . 938
secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_dblock.c . . . 945
secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_dblock.h . . . 949
secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.c . . . 953
secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.h . . . 966
secure_fw/partitions/lib/runtime/crt_memcmp.c . . . 1004
secure_fw/partitions/lib/runtime/crt_memmove.c . . . 1005
secure_fw/partitions/lib/runtime/crt_strnlen.c . . . 1006
secure_fw/partitions/lib/runtime/psa_api_ipc.c . . . 1010
secure_fw/partitions/lib/runtime/service_api.c . . . 1016
secure_fw/partitions/lib/runtime/sfn_common_thread.c . . . 1018
secure_fw/partitions/lib/runtime/sprt_partition_metadata_indicator.c . . . 1019
secure_fw/partitions/lib/runtime/sprt_partition_metadata_indicator.h . . . 1019
secure_fw/partitions/lib/runtime/tfm_sp_log_raw.c . . . 1021
secure_fw/partitions/lib/runtime/include/service_api.h . . . 1007
secure_fw/partitions/lib/runtime/include/tfm_sp_log.h . . . 1008
secure_fw/partitions/lib/runtime/include/tfm_strnlen.h . . . 1010
secure_fw/partitions/ns_agent_mailbox/ns_agent_mailbox.c . . . 1022
secure_fw/partitions/ns_agent_mailbox/tfm_multi_core_client_id.c . . . 1023
secure_fw/partitions/ns_agent_mailbox/tfm_spe_mailbox.c . . . 1025
secure_fw/partitions/ns_agent_mailbox/tfm_spe_mailbox.h . . . 1030
secure_fw/partitions/ns_agent_tz/load_info_ns_agent_tz.c . . . 1032
secure_fw/partitions/ns_agent_tz/ns_agent_tz.c . . . 1033
secure_fw/partitions/ns_agent_tz/ns_agent_tz_v80m.c . . . 1034
secure_fw/partitions/ns_agent_tz/psa_api_veneers.c . . . 1034
secure_fw/partitions/ns_agent_tz/psa_api_veneers_v80m.c . . . 1037
secure_fw/partitions/platform/platform_sp.c . . . 1042
secure_fw/partitions/platform/platform_sp.h . . . 1044
secure_fw/partitions/protected_storage/config_ps_check.h . . . 1046
secure_fw/partitions/protected_storage/ps_encrypted_object.c . . . 1062
secure_fw/partitions/protected_storage/ps_encrypted_object.h . . . 1065
secure_fw/partitions/protected_storage/ps_filesystem_interface.c . . . 1068
secure_fw/partitions/protected_storage/ps_object_defs.h . . . 1073
secure_fw/partitions/protected_storage/ps_object_system.c . . . 1075
secure_fw/partitions/protected_storage/ps_object_system.h . . . 1083
secure_fw/partitions/protected_storage/ps_object_table.c . . . 1090
secure_fw/partitions/protected_storage/ps_object_table.h . . . 1101
secure_fw/partitions/protected_storage/ps_utils.c . . . 1107
secure_fw/partitions/protected_storage/ps_utils.h . . . 1109
secure_fw/partitions/protected_storage/tfm_protected_storage.c . . . 1111
secure_fw/partitions/protected_storage/tfm_protected_storage.h . . . 1117
secure_fw/partitions/protected_storage/tfm_ps_req_mngr.c . . . 1123
secure_fw/partitions/protected_storage/tfm_ps_req_mngr.h . . . 1125
secure_fw/partitions/protected_storage/crypto/ps_crypto_interface.c . . . 1047
secure_fw/partitions/protected_storage/crypto/ps_crypto_interface.h . . . 1052
secure_fw/partitions/protected_storage/nv_counters/ps_nv_counters.c . . . 1057
secure_fw/partitions/protected_storage/nv_counters/ps_nv_counters.h . . . 1059
secure_fw/shared/crt_memcpy.c . . . 1127
secure_fw/shared/crt_memset.c . . . 1128
secure_fw/spm/core/backend_ipc.c . . . 1140
secure_fw/spm/core/backend_sfn.c . . . 1146
secure_fw/spm/core/internal_status_code.h . . . 1149
secure_fw/spm/core/interrupt.c . . . 1151
secure_fw/spm/core/interrupt.h . . . 1154
secure_fw/spm/core/mailbox_agent_api.c . . . 1157
secure_fw/spm/core/main.c . . . 1158

secure_fw/spm/core/memory_symbols.h . . . 1159
secure_fw/spm/core/ns_agent_mailbox_interface.c . . . 1162
secure_fw/spm/core/psa_api.c . . . 1164
secure_fw/spm/core/psa_call_api.c . . . 1168
secure_fw/spm/core/psa_connection_api.c . . . 1169
secure_fw/spm/core/psa_interface_sfn.c . . . 1171
secure_fw/spm/core/psa_interface_svc.c . . . 1178
secure_fw/spm/core/psa_interface_thread_fn_call.c . . . 1181
secure_fw/spm/core/psa_irq_api.c . . . 1186
secure_fw/spm/core/psa_mmiovec_api.c . . . 1187
secure_fw/spm/core/psa_read_write_skip_api.c . . . 1188
secure_fw/spm/core/psa_version_api.c . . . 1191
secure_fw/spm/core/rom_loader.c . . . 1193
secure_fw/spm/core/spm.h . . . 1194
secure_fw/spm/core/spm_connection_pool.c . . . 1208
secure_fw/spm/core/spm_ipc.c . . . 1211
secure_fw/spm/core/spm_local_connection.c . . . 1219
secure_fw/spm/core/spm_log.c . . . 1221
secure_fw/spm/core/stack_watermark.c . . . 1223
secure_fw/spm/core/stack_watermark.h . . . 1225
secure_fw/spm/core/tfm_boot_data.c . . . 1226
secure_fw/spm/core/tfm_boot_data.h . . . 1227
secure_fw/spm/core/tfm_multi_core.h . . . 1229
secure_fw/spm/core/tfm_pools.c . . . 1232
secure_fw/spm/core/tfm_pools.h . . . 1235
secure_fw/spm/core/tfm_rpc.c . . . 1240
secure_fw/spm/core/tfm_rpc.h . . . 1242
secure_fw/spm/core/tfm_svcalls.c . . . 1242
secure_fw/spm/core/tfm_svcalls.h . . . 1244
secure_fw/spm/core/thread.c . . . 1246
secure_fw/spm/core/thread.h . . . 1248
secure_fw/spm/core/utilities.c . . . 1257
secure_fw/spm/core/arch/tfm_arch.c . . . 1130
secure_fw/spm/core/arch/tfm_arch_v6m_v7m.c . . . 1131
secure_fw/spm/core/arch/tfm_arch_v6m_v7m.h . . . 1133
secure_fw/spm/core/arch/tfm_arch_v8m_base.c . . . 1138
secure_fw/spm/core/arch/tfm_arch_v8m_main.c . . . 1139
secure_fw/spm/include/aapcs_local.h . . . 1259
secure_fw/spm/include/bitops.h . . . 1261
secure_fw/spm/include/config_spm.h . . . 1270
secure_fw/spm/include/critical_section.h . . . 1271
secure_fw/spm/include/current.h . . . 1272
secure_fw/spm/include/lists.h . . . 1299
secure_fw/spm/include/ns_agent_mailbox_defs.h . . . 1316
secure_fw/spm/include/ns_agent_mailbox_interface.h . . . 1317
secure_fw/spm/include/tfm_arch.h . . . 1319
secure_fw/spm/include/tfm_arch_v8m.h . . . 1327
secure_fw/spm/include/tfm_core_trustzone.h . . . 1333
secure_fw/spm/include/tfm_exc_num.h . . . 1334
secure_fw/spm/include/tfm_nspm.h . . . 1336
secure_fw/spm/include/tfm_spm_log.h . . . 1339
secure_fw/spm/include/utilities.h . . . 1342
secure_fw/spm/include/boot/tfm_boot_status.h . . . 1262
secure_fw/spm/include/ffm/backend.h . . . 1273
secure_fw/spm/include/ffm/backend_ipc.h . . . 1277
secure_fw/spm/include/ffm/backend_sfn.h . . . 1278
secure_fw/spm/include/ffm/mailbox_agent_api.h . . . 1279
secure_fw/spm/include/ffm/original_vector_api.h . . . 1281

secure_fw/spm/include/ffm/psa_api.h . . . 1282
secure_fw/spm/include/interface/runtime_defs.h . . . 1292
secure_fw/spm/include/interface/svc_num.h . . . 1293
secure_fw/spm/include/load/asset_defs.h . . . 1303
secure_fw/spm/include/load/interrupt_defs.h . . . 1305
secure_fw/spm/include/load/partition_defs.h . . . 1305
secure_fw/spm/include/load/service_defs.h . . . 1310
secure_fw/spm/include/load/spm_load_api.h . . . 1312
secure_fw/spm/ns_client_ext/tfm_ns_client_ext.c . . . 1343
secure_fw/spm/ns_client_ext/tfm_ns_ctx.c . . . 1349
secure_fw/spm/ns_client_ext/tfm_ns_ctx.h . . . 1351
secure_fw/spm/ns_client_ext/tfm_ns_ctx_s.c . . . 1355
secure_fw/spm/ns_client_ext/tfm_ns_ctx_shm.h . . . 1357
secure_fw/spm/ns_client_ext/tfm_spm_ns_ctx.c . . . 1357

# Chapter 5

# Module Documentation

## 5.1 Set of functions implementing a thin shim

layer between the TF-M Crypto service frontend and the underlying library which implements the PSA Crypto APIs (i.e. mbed TLS)

### Functions

- psa_status_t tfm_crypto_aead_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the AEAD module.*
- psa_status_t tfm_crypto_asymmetric_sign_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Asymmetric signing module.*
- psa_status_t tfm_crypto_asymmetric_encrypt_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Asymmetric encryption module.*
- psa_status_t tfm_crypto_cipher_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Cipher module.*
- psa_status_t tfm_crypto_hash_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ])

  *This function acts as interface for the Hash module.*
- psa_status_t tfm_crypto_key_derivation_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Key derivation module.*
- psa_status_t tfm_crypto_key_management_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Key management module.*
- psa_status_t tfm_crypto_mac_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the MAC module.*
- psa_status_t tfm_crypto_random_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ])

  *This function acts as interface for the Random module.*

### 5.1.1 Detailed Description

layer between the TF-M Crypto service frontend and the underlying library which implements the PSA Crypto APIs (i.e. mbed TLS)

### 5.1.2 Function Documentation

#### 5.1.2.1 tfm_crypto_aead_interface()

```
psa_status_t tfm_crypto_aead_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[],
            struct tfm_crypto_key_id_s * encoded_key )
```

This function acts as interface for the AEAD module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |
| in | *encoded_key* | Key encoded with partition_id and key_id |

**Returns**

Return values as described in psa_status_t

Definition at line 261 of file crypto_aead.c.

Here is the caller graph for this function:

#### 5.1.2.2 tfm_crypto_asymmetric_encrypt_interface()

```
psa_status_t tfm_crypto_asymmetric_encrypt_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[],
            struct tfm_crypto_key_id_s * encoded_key )
```

This function acts as interface for the Asymmetric encryption module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |
| in | *encoded_key* | Key encoded with partition_id and key_id |

**Returns**

Return values as described in psa_status_t

Definition at line 160 of file crypto_asymmetric.c.

Here is the caller graph for this function:

#### 5.1.2.3 tfm_crypto_asymmetric_sign_interface()

```
psa_status_t tfm_crypto_asymmetric_sign_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[],
            struct tfm_crypto_key_id_s * encoded_key )
```

This function acts as interface for the Asymmetric signing module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |
| in | *encoded_key* | Key encoded with partition_id and key_id |

**Returns**

Return values as described in psa_status_t

Definition at line 94 of file crypto_asymmetric.c.

Here is the caller graph for this function:

#### 5.1.2.4 tfm_crypto_cipher_interface()

```
psa_status_t tfm_crypto_cipher_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[],
            struct tfm_crypto_key_id_s * encoded_key )
```

This function acts as interface for the Cipher module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |
| in | *encoded_key* | Key encoded with partition_id and key_id |

**Returns**

Return values as described in psa_status_t

Definition at line 201 of file crypto_cipher.c.

Here is the caller graph for this function:

#### 5.1.2.5 tfm_crypto_hash_interface()

```
psa_status_t tfm_crypto_hash_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[] )
```

This function acts as interface for the Hash module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |

**Returns**

Return values as described in psa_status_t

Definition at line 194 of file crypto_hash.c.

Here is the caller graph for this function:

#### 5.1.2.6 tfm_crypto_key_derivation_interface()

```
psa_status_t tfm_crypto_key_derivation_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[],
            struct tfm_crypto_key_id_s * encoded_key )
```

This function acts as interface for the Key derivation module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |
| in | *encoded_key* | Key encoded with partition_id and key_id |

**Returns**

Return values as described in psa_status_t

Definition at line 196 of file crypto_key_derivation.c.

Here is the caller graph for this function:

#### 5.1.2.7 tfm_crypto_key_management_interface()

```
psa_status_t tfm_crypto_key_management_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[],
            struct tfm_crypto_key_id_s * encoded_key )
```

This function acts as interface for the Key management module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |
| in | *encoded_key* | Key encoded with partition_id and key_id |

**Returns**

Return values as described in psa_status_t

Definition at line 180 of file crypto_key_management.c.

Here is the caller graph for this function:

#### 5.1.2.8 tfm_crypto_mac_interface()

```
psa_status_t tfm_crypto_mac_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[],
            struct tfm_crypto_key_id_s * encoded_key )
```

This function acts as interface for the MAC module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |
| in | *encoded_key* | Key encoded with partition_id and key_id |

**Returns**

Return values as described in psa_status_t

Definition at line 187 of file crypto_mac.c.

Here is the caller graph for this function:

#### 5.1.2.9 tfm_crypto_random_interface()

```
psa_status_t tfm_crypto_random_interface (
            psa_invec in_vec[],
            psa_outvec out_vec[] )
```

This function acts as interface for the Random module.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| out | *out_vec* | Array of outvec parameters |

**Returns**

Return values as described in psa_status_t

Definition at line 24 of file crypto_rng.c.

Here is the call graph for this function:

Here is the caller graph for this function:

## 5.2 Function that implement allocation and deallocation of

contexts to be stored in the secure world for multipart operations

### Functions

- psa_status_t tfm_crypto_init_alloc (void)

  *Initialise the Alloc module.*
- psa_status_t tfm_crypto_operation_alloc (enum tfm_crypto_operation_type type, uint32_t *handle, void **ctx)

  *Allocate an operation context in the backend.*
- psa_status_t tfm_crypto_operation_release (uint32_t *handle)

  *Release an operation context in the backend.*
- psa_status_t tfm_crypto_operation_lookup (enum tfm_crypto_operation_type type, uint32_t handle, void **ctx)

  *Look up an operation context in the backend for the corresponding frontend operation.*

### 5.2.1 Detailed Description

contexts to be stored in the secure world for multipart operations

### 5.2.2 Function Documentation

#### 5.2.2.1 tfm_crypto_init_alloc()

```
psa_status_t tfm_crypto_init_alloc (
            void  )
```

Initialise the Alloc module.

**Returns**

Return values as described in psa_status_t

Definition at line 74 of file crypto_alloc.c.

Here is the call graph for this function:

#### 5.2.2.2 tfm_crypto_operation_alloc()

```
psa_status_t tfm_crypto_operation_alloc (
            enum tfm_crypto_operation_type type,
            uint32_t * handle,
            void ** ctx )
```

Allocate an operation context in the backend.

**Parameters**

| | | |
|---|---|---|
| in | *type* | Type of the operation context to allocate |
| out | *handle* | Pointer to hold the allocated handle |
| | *[out* | ctx Double pointer to the corresponding context |

**Returns**

Return values as described in psa_status_t

Definition at line 81 of file crypto_alloc.c.

#### 5.2.2.3 tfm_crypto_operation_lookup()

```
psa_status_t tfm_crypto_operation_lookup (
            enum tfm_crypto_operation_type type,
            uint32_t handle,
            void ** ctx )
```

Look up an operation context in the backend for the corresponding frontend operation.

**Parameters**

| | | |
|---|---|---|
| in | *type* | Type of the operation context to look up |
| in | *handle* | Handle of the context to lookup |
| out | *ctx* | Double pointer to the corresponding context |

**Returns**

Return values as described in psa_status_t

Definition at line 152 of file crypto_alloc.c.

#### 5.2.2.4 tfm_crypto_operation_release()

```
psa_status_t tfm_crypto_operation_release (
            uint32_t * handle )
```

Release an operation context in the backend.

**Parameters**

| | |
|---|---|
| *[in/out]* | handle Pointer to the handle of the context to release |

**Returns**

Return values as described in psa_status_t

Definition at line 119 of file crypto_alloc.c.

## 5.3 Set of functions implementing the abstractions of the underlying cryptographic

library that implements the PSA Crypto APIs to provide the PSA Crypto core functionality to the TF-M Crypto service. Currently it supports only an mbed TLS based abstraction.

### Functions

- tfm_crypto_library_key_id_t tfm_crypto_library_key_id_init (int32_t owner, psa_key_id_t key_id)

  *Function used to initialise an object of tfm_crypto_library_key_id_t to a (owner, key_id) pair.*
- char ∗ tfm_crypto_library_get_info (void)

  *This function is used to retrieve a string describing the library used in the backend to provide information to the crypto service and the user.*
- psa_status_t tfm_crypto_core_library_init (void)

  *This function is used to perform the necessary steps to initialise the underlying library that provides the implementation of the PSA Crypto core to the TF-M Crypto service.*
- void tfm_crypto_library_get_library_key_id_set_owner (int32_t owner, psa_key_attributes_t ∗attr)

  *Allows to set the owner of a library key embedded into the key attributes structure.*
- psa_status_t mbedtls_psa_platform_get_builtin_key (mbedtls_svc_key_id_t key_id, psa_key_lifetime_↩ t ∗lifetime, psa_drv_slot_number_t ∗slot_number)

  *This function is required by mbed TLS to enable support for platform builtin keys in the PSA Crypto core layer implemented by mbed TLS. This function is not standardized by the API hence this layer directly provides the symbol required by the library.*

### 5.3.1 Detailed Description

library that implements the PSA Crypto APIs to provide the PSA Crypto core functionality to the TF-M Crypto service. Currently it supports only an mbed TLS based abstraction.

### 5.3.2 Function Documentation

#### 5.3.2.1 mbedtls_psa_platform_get_builtin_key()

```
psa_status_t mbedtls_psa_platform_get_builtin_key (
            mbedtls_svc_key_id_t key_id,
            psa_key_lifetime_t * lifetime,
            psa_drv_slot_number_t * slot_number )
```

This function is required by mbed TLS to enable support for platform builtin keys in the PSA Crypto core layer implemented by mbed TLS. This function is not standardized by the API hence this layer directly provides the symbol required by the library.

**Note**

It maps builtin key IDs to cryptographic drivers and slots. The actual data is deferred to a platform function, as different platforms may have different key storage capabilities.

Definition at line 111 of file crypto_library.c.

Here is the call graph for this function:

#### 5.3.2.2 tfm_crypto_core_library_init()

```
psa_status_t tfm_crypto_core_library_init (
            void  )
```

This function is used to perform the necessary steps to initialise the underlying library that provides the implementation of the PSA Crypto core to the TF-M Crypto service.

**Returns**

PSA_SUCCESS on successful initialisation

Definition at line 80 of file crypto_library.c.

#### 5.3.2.3 tfm_crypto_library_get_info()

```
char* tfm_crypto_library_get_info (
            void  )
```

This function is used to retrieve a string describing the library used in the backend to provide information to the crypto service and the user.

**Returns**

A NULL terminated string describing the backend library

Definition at line 74 of file crypto_library.c.

Here is the call graph for this function:

#### 5.3.2.4 tfm_crypto_library_get_library_key_id_set_owner()

```
void tfm_crypto_library_get_library_key_id_set_owner (
            int32_t owner,
            psa_key_attributes_t * attr )
```

Allows to set the owner of a library key embedded into the key attributes structure.

**Parameters**

| | | |
|---|---|---|
| in | *owner* | The owner value to be written into the key attributes structure |
| out | *attr* | Pointer to the key attributes into which we want to e |

Definition at line 96 of file crypto_library.c.

#### 5.3.2.5 tfm_crypto_library_key_id_init()

```
tfm_crypto_library_key_id_t tfm_crypto_library_key_id_init (
            int32_t owner,
            psa_key_id_t key_id )
```

Function used to initialise an object of tfm_crypto_library_key_id_t to a (owner, key_id) pair.

**Parameters**

| | | |
|---|---|---|
| in | *owner* | Owner of the key |
| in | *key↩ _id* | key ID associated to the key of type psa_key_id_t |

**Returns**

An object of type tfm_crypto_library_key_id_t

Definition at line 69 of file crypto_library.c.

Here is the caller graph for this function:

## 5.4 Tfm_builtin_key_loader

### Functions

- psa_status_t tfm_builtin_key_loader_init (void)

  *This is the initialisation function for the tfm_builtin_key_laoder driver, to be called from the PSA core initialisation subsystem. It discovers the keys available in the underlying hardware platform and loads them in memory visible to the PSA Crypto subsystem to be used to the normal APIs.*
- psa_status_t tfm_builtin_key_loader_get_key_buffer_size (tfm_crypto_library_key_id_t key_id, size_t *len)

  *Returns the length of a key from the builtin driver.*
- psa_status_t tfm_builtin_key_loader_get_builtin_key (psa_drv_slot_number_t slot_number, psa_key_attributes_t *attributes, uint8_t *key_buffer, size_t key_buffer_size, size_t *key_buffer_length)

  *Returns the attributes and key material of a key from the builtin driver to be used by the PSA Crypto core.*

### 5.4.1 Detailed Description

### 5.4.2 Function Documentation

#### 5.4.2.1 tfm_builtin_key_loader_get_builtin_key()

```
psa_status_t tfm_builtin_key_loader_get_builtin_key (
            psa_drv_slot_number_t slot_number,
            psa_key_attributes_t * attributes,
            uint8_t * key_buffer,
            size_t key_buffer_size,
            size_t * key_buffer_length )
```

Returns the attributes and key material of a key from the builtin driver to be used by the PSA Crypto core.

**Note**

This function is called by the psa crypto driver wrapper.

**Parameters**

| | | |
|---|---|---|
| in | *slot_number* | The slot of the key |
| out | *attributes* | The attributes of the key. |
| out | *key_buffer* | The buffer to output the key material into. |
| in | *key_buffer_size* | The size of the key material buffer. |
| out | *key_buffer_length* | The length of the key material returned. |

**Returns**

Returns error code specified in psa_status_t

Definition at line 280 of file tfm_builtin_key_loader.c.

#### 5.4.2.2 tfm_builtin_key_loader_get_key_buffer_size()

```
psa_status_t tfm_builtin_key_loader_get_key_buffer_size (
            mbedtls_svc_key_id_t key_id,
            size_t * len )
```

Returns the length of a key from the builtin driver.

**Note**

This function is called by the psa crypto driver wrapper.

**Parameters**

| | | |
|---|---|---|
| in | *key↩ _id* | The ID of the key to return the length of. The type of this must match the expected type of the underlying library that provides the key management for the PSA Crypto core, and must support encoding the owner in addition to the key_id. |
| out | *len* | The length of the key. |

**Returns**

Returns error code specified in psa_status_t

Definition at line 255 of file tfm_builtin_key_loader.c.

#### 5.4.2.3 tfm_builtin_key_loader_init()

```
psa_status_t tfm_builtin_key_loader_init (
            void  )
```

This is the initialisation function for the tfm_builtin_key_laoder driver, to be called from the PSA core initialisation subsystem. It discovers the keys available in the underlying hardware platform and loads them in memory visible to the PSA Crypto subsystem to be used to the normal APIs.

**Returns**

Returns error code specified in psa_status_t

Definition at line 206 of file tfm_builtin_key_loader.c.

Here is the call graph for this function:

# Chapter 6

# Data Structure Documentation

## 6.1 _MTB_SRF_DATA_ALIGN Struct Reference

MTB SRF IPC request structure suitable for TFM needs. This structure overrides default mtb_srf_ipc_packet_t SRF structure.

```
#include <platform/ext/target/infineon/common/interface/include/mtb_srf_↩
ipc_custom_packet.h>
```

Collaboration diagram for _MTB_SRF_DATA_ALIGN:

### Data Fields

- uint32_t call_type
- struct psa_client_params_t params
- struct mailbox_reply_t reply
- uint16_t semaphore_idx

### 6.1.1 Detailed Description

MTB SRF IPC request structure suitable for TFM needs. This structure overrides default mtb_srf_ipc_packet_t SRF structure.

This structure is designed to be allocated in shared memory for inter-process communication (IPC). All members are stored as values (not pointers) to ensure the entire request is self-contained and can be safely copied into and out of shared memory. Parameters required for the PSA client call are copied directly into this structure, and the structure itself is allocated in the shared memory region. This design ensures that all data needed for the request is accessible to both communicating parties without relying on process-specific pointers/memory.

Definition at line 32 of file mtb_srf_ipc_custom_packet.h.

### 6.1.2 Field Documentation

#### 6.1.2.1 call_type

```
uint32_t call_type
```

Definition at line 34 of file mtb_srf_ipc_custom_packet.h.

#### 6.1.2.2 params

```
struct psa_client_params_t params
```

Definition at line 35 of file mtb_srf_ipc_custom_packet.h.

#### 6.1.2.3 reply

```
struct mailbox_reply_t reply
```

Definition at line 37 of file mtb_srf_ipc_custom_packet.h.

#### 6.1.2.4 semaphore_idx

```
uint16_t semaphore_idx
```

Definition at line 40 of file mtb_srf_ipc_custom_packet.h.

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

- platform/ext/target/infineon/common/interface/include/mtb_srf_ipc_custom_packet.h

## 6.2 asset_desc_t Struct Reference

```
#include <secure_fw/spm/include/load/asset_defs.h>
```

### Data Fields

- union {
  struct {
    uintptr_t start
    uintptr_t limit
  } mem
  struct {
    uintptr_t dev_ref
    uintptr_t reserved
  } dev
  };
- uint32_t attr

### 6.2.1 Detailed Description

Definition at line 22 of file asset_defs.h.

### 6.2.2 Field Documentation

#### 6.2.2.1 "@12

```
union { ... }
```

#### 6.2.2.2 attr

```
uint32_t attr
```

Definition at line 33 of file asset_defs.h.

#### 6.2.2.3 dev

```
struct { ... } dev
```

#### 6.2.2.4 dev_ref

```
uintptr_t dev_ref
```

Definition at line 29 of file asset_defs.h.

#### 6.2.2.5 limit

```
uintptr_t limit
```

Definition at line 26 of file asset_defs.h.

#### 6.2.2.6 mem

```
struct { ...  } mem
```

#### 6.2.2.7 reserved

```
uintptr_t reserved
```

Definition at line 30 of file asset_defs.h.

#### 6.2.2.8 start

```
uintptr_t start
```

Definition at line 25 of file asset_defs.h.

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

- secure_fw/spm/include/load/asset_defs.h

## 6.3 attest_boot_data Struct Reference

Contains the received boot status information from bootloader.

Collaboration diagram for attest_boot_data:

### Data Fields

- struct shared_data_tlv_header header
- uint8_t data [512]

### 6.3.1 Detailed Description

Contains the received boot status information from bootloader.

This is a redefinition of tfm_boot_data to allocate the appropriate, service dependent size of boot_data.

Definition at line 33 of file attest_boot_data.c.

### 6.3.2 Field Documentation

#### 6.3.2.1 data

```
uint8_t data[512]
```

Definition at line 35 of file attest_boot_data.c.

#### 6.3.2.2 header

```
struct shared_data_tlv_header header
```

Definition at line 34 of file attest_boot_data.c.

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

- secure_fw/partitions/initial_attestation/attest_boot_data.c

## 6.4 attest_token_encode_ctx Struct Reference

```
#include <secure_fw/partitions/initial_attestation/attest_token.h>
```

### Data Fields

- QCBOREncodeContext cbor_enc_ctx
- uint32_t opt_flags
- int32_t key_select
- struct t_cose_sign1_sign_ctx signer_ctx

### 6.4.1 Detailed Description

The context for creating an attestation token. The caller of attest_token_encode must create one of these and pass it to the functions here. It is small enough that it can go on the stack. It is most of the memory needed to create a token except the output buffer and any memory requirements for the cryptographic operations.

The structure is opaque for the caller.

This is roughly 148 + 8 + 32 = 188 bytes

Definition at line 125 of file attest_token.h.

### 6.4.2 Field Documentation

#### 6.4.2.1 cbor_enc_ctx

```
QCBOREncodeContext cbor_enc_ctx
```

Definition at line 127 of file attest_token.h.

#### 6.4.2.2 key_select

```
int32_t key_select
```

Definition at line 129 of file attest_token.h.

#### 6.4.2.3 opt_flags

```
uint32_t opt_flags
```

Definition at line 128 of file attest_token.h.

#### 6.4.2.4 signer_ctx

```
struct t_cose_sign1_sign_ctx signer_ctx
```

Definition at line 133 of file attest_token.h.

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

- secure_fw/partitions/initial_attestation/attest_token.h

## 6.5 bi_list_node_t Struct Reference

```
#include <secure_fw/spm/include/lists.h>
```

Collaboration diagram for bi_list_node_t:

### Data Fields

- struct bi_list_node_t * bprev
- struct bi_list_node_t * bnext

### 6.5.1 Detailed Description

Definition at line 12 of file lists.h.

### 6.5.2 Field Documentation

#### 6.5.2.1 bnext

```
struct bi_list_node_t* bnext
```

Definition at line 14 of file lists.h.

#### 6.5.2.2 bprev

```
struct bi_list_node_t* bprev
```

Definition at line 13 of file lists.h.

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

- secure_fw/spm/include/lists.h

## 6.6 boot_data_access_policy Struct Reference

Defines the access policy of secure partitions to data items in shared data area (between bootloader and runtime firmware).

### Data Fields

- int32_t partition_id
- uint32_t major_type

### 6.6.1 Detailed Description

Defines the access policy of secure partitions to data items in shared data area (between bootloader and runtime firmware).

Definition at line 55 of file tfm_boot_data.c.

### 6.6.2 Field Documentation

#### 6.6.2.1 major_type

```
uint32_t major_type
```

Definition at line 57 of file tfm_boot_data.c.

#### 6.6.2.2 partition_id

```
int32_t partition_id
```

Definition at line 56 of file tfm_boot_data.c.

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

- secure_fw/spm/core/tfm_boot_data.c

## 6.7 client_id_region_t Struct Reference

### Data Fields

- int32_t base
- int32_t limit
- void * owner

### 6.7.1 Detailed Description

Definition at line 15 of file tfm_multi_core_client_id.c.

### 6.7.2 Field Documentation

#### 6.7.2.1 base

```
int32_t base
```

Definition at line 16 of file tfm_multi_core_client_id.c.

#### 6.7.2.2 limit

```
int32_t limit
```

Definition at line 17 of file tfm_multi_core_client_id.c.

#### 6.7.2.3 owner

```
void* owner
```

Definition at line 18 of file tfm_multi_core_client_id.c.

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

- secure_fw/partitions/ns_agent_mailbox/tfm_multi_core_client_id.c
- secure_fw/spm/ns_client_ext/tfm_spm_ns_ctx.c

## 6.8 client_params_t Struct Reference

```
#include <secure_fw/spm/include/ffm/mailbox_agent_api.h>
```

Collaboration diagram for client_params_t:

### Data Fields

- int32_t ns_client_id_stateless
- psa_invec * p_invecs
- psa_outvec * p_outvecs

### 6.8.1 Detailed Description

Definition at line 18 of file mailbox_agent_api.h.

### 6.8.2 Field Documentation

#### 6.8.2.1 ns_client_id_stateless

```
int32_t ns_client_id_stateless
```

Definition at line 19 of file mailbox_agent_api.h.

#### 6.8.2.2 p_invecs

```
psa_invec* p_invecs
```

Definition at line 20 of file mailbox_agent_api.h.

#### 6.8.2.3 p_outvecs

```
psa_outvec* p_outvecs
```

Definition at line 21 of file mailbox_agent_api.h.

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

- secure_fw/spm/include/ffm/mailbox_agent_api.h

## 6.9 composite_addr_t Union Reference

```
#include <secure_fw/include/crt_impl_private.h>
```

### Data Fields

- uintptr_t uint_addr
- uint8_t * p_byte
- uint32_t * p_word

### 6.9.1 Detailed Description

Definition at line 16 of file crt_impl_private.h.

### 6.9.2 Field Documentation

#### 6.9.2.1 p_byte

```
uint8_t* p_byte
```

Definition at line 18 of file crt_impl_private.h.

#### 6.9.2.2 p_word

```
uint32_t* p_word
```

Definition at line 19 of file crt_impl_private.h.

#### 6.9.2.3 uint_addr

```
uintptr_t uint_addr
```

Definition at line 17 of file crt_impl_private.h.

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

- secure_fw/include/crt_impl_private.h

## 6.10 connection_t Struct Reference

```
#include <secure_fw/spm/core/spm.h>
```

Collaboration diagram for connection_t:

## Data Fields

- uint32_t status
- struct partition_t * p_client
- const struct service_t * service
- psa_msg_t msg
- uint32_t ctrl_param
- const void * invec_base [PSA_MAX_IOVEC]
- size_t invec_accessed [PSA_MAX_IOVEC]
- void * outvec_base [PSA_MAX_IOVEC]
- size_t outvec_written [PSA_MAX_IOVEC]
- psa_outvec * caller_outvec

### 6.10.1 Detailed Description

Definition at line 73 of file spm.h.

### 6.10.2 Field Documentation

#### 6.10.2.1 caller_outvec

```
psa_outvec* caller_outvec
```

Definition at line 89 of file spm.h.

#### 6.10.2.2 ctrl_param

```
uint32_t ctrl_param
```

Definition at line 84 of file spm.h.

#### 6.10.2.3 invec_accessed

```
size_t invec_accessed[PSA_MAX_IOVEC]
```

Definition at line 86 of file spm.h.

#### 6.10.2.4 invec_base

```
const void* invec_base[PSA_MAX_IOVEC]
```

Definition at line 85 of file spm.h.

#### 6.10.2.5 msg

```
psa_msg_t msg
```

Definition at line 83 of file spm.h.

#### 6.10.2.6 outvec_base

```
void* outvec_base[PSA_MAX_IOVEC]
```

Definition at line 87 of file spm.h.

#### 6.10.2.7 outvec_written

```
size_t outvec_written[PSA_MAX_IOVEC]
```

Definition at line 88 of file spm.h.

#### 6.10.2.8 p_client

```
struct partition_t* p_client
```

Definition at line 81 of file spm.h.

#### 6.10.2.9 service

```
const struct service_t* service
```

Definition at line 82 of file spm.h.

#### 6.10.2.10 status

```
uint32_t status
```

Definition at line 74 of file spm.h.

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

- secure_fw/spm/core/spm.h

## 6.11 context_ctrl_t Struct Reference

```
#include <secure_fw/spm/include/tfm_arch.h>
```

### Data Fields

- uint32_t sp
- uint32_t exc_ret
- uint32_t sp_limit
- uint32_t sp_base

### 6.11.1 Detailed Description

Definition at line 138 of file tfm_arch.h.

### 6.11.2 Field Documentation

#### 6.11.2.1 exc_ret

```
uint32_t exc_ret
```

Definition at line 143 of file tfm_arch.h.

#### 6.11.2.2 sp

```
uint32_t sp
```

Definition at line 139 of file tfm_arch.h.

#### 6.11.2.3 sp_base

```
uint32_t sp_base
```

Definition at line 148 of file tfm_arch.h.

#### 6.11.2.4 sp_limit

```
uint32_t sp_limit
```

Definition at line 147 of file tfm_arch.h.

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

- secure_fw/spm/include/tfm_arch.h

## 6.12 context_flih_ret_t Struct Reference

```
#include <secure_fw/spm/include/tfm_arch.h>
```

Collaboration diagram for context_flih_ret_t:

### Data Fields

- uint64_t stack_seal
- struct tfm_additional_context_t addi_ctx
- uint32_t exc_return
- uint32_t dummy
- uint32_t psp
- uint32_t psplim
- struct tfm_state_context_t state_ctx

### 6.12.1 Detailed Description

Definition at line 155 of file tfm_arch.h.

### 6.12.2 Field Documentation

#### 6.12.2.1 addi_ctx

```
struct tfm_additional_context_t addi_ctx
```

Definition at line 157 of file tfm_arch.h.

#### 6.12.2.2 dummy

```
uint32_t dummy
```

Definition at line 162 of file tfm_arch.h.

#### 6.12.2.3 exc_return

```
uint32_t exc_return
```

Definition at line 161 of file tfm_arch.h.

#### 6.12.2.4 psp

```
uint32_t psp
```

Definition at line 163 of file tfm_arch.h.

#### 6.12.2.5 psplim

```
uint32_t psplim
```

Definition at line 164 of file tfm_arch.h.

#### 6.12.2.6 stack_seal

```
uint64_t stack_seal
```

Definition at line 156 of file tfm_arch.h.

#### 6.12.2.7 state_ctx

```
struct tfm_state_context_t state_ctx
```

Definition at line 165 of file tfm_arch.h.

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

- secure_fw/spm/include/tfm_arch.h

## 6.13 critical_section_t Struct Reference

```
#include <secure_fw/spm/include/critical_section.h>
```

### Data Fields

- uint32_t state

### 6.13.1 Detailed Description

Definition at line 14 of file critical_section.h.

### 6.13.2 Field Documentation

#### 6.13.2.1 state

```
uint32_t state
```

Definition at line 15 of file critical_section.h.

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

- secure_fw/spm/include/critical_section.h

## 6.14 formatted_buffer_t Struct Reference

### Data Fields

- size_t pos
- uint8_t buf [32]

### 6.14.1 Detailed Description

Definition at line 17 of file tfm_sp_log_raw.c.

### 6.14.2 Field Documentation

#### 6.14.2.1 buf

```
uint8_t buf[32]
```

Definition at line 19 of file tfm_sp_log_raw.c.

#### 6.14.2.2 pos

```
size_t pos
```

Definition at line 18 of file tfm_sp_log_raw.c.

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

- secure_fw/partitions/lib/runtime/tfm_sp_log_raw.c

## 6.15 full_context_t Struct Reference

```
#include <secure_fw/spm/include/tfm_arch.h>
```

Collaboration diagram for full_context_t:

### Data Fields

- struct tfm_additional_context_t addi_ctx
- struct tfm_state_context_t stat_ctx

### 6.15.1 Detailed Description

Definition at line 125 of file tfm_arch.h.

### 6.15.2 Field Documentation

#### 6.15.2.1 addi_ctx

```
struct tfm_additional_context_t addi_ctx
```

Definition at line 126 of file tfm_arch.h.

#### 6.15.2.2 stat_ctx

```
struct tfm_state_context_t stat_ctx
```

Definition at line 130 of file tfm_arch.h.

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

- secure_fw/spm/include/tfm_arch.h

## 6.16 fwu_image_info_data_s Struct Reference

Collaboration diagram for fwu_image_info_data_s:

### Data Fields

- struct shared_data_tlv_header header
- uint8_t data [(MCUBOOT_IMAGE_NUMBER ∗(sizeof(struct image_version)+SHARED_DATA_ENTRY_HEADER_SIZE))]

### 6.16.1 Detailed Description

Definition at line 36 of file tfm_mcuboot_fwu.c.

### 6.16.2 Field Documentation

#### 6.16.2.1 data

```
uint8_t data[(MCUBOOT_IMAGE_NUMBER *(sizeof(struct image_version)+ SHARED_DATA_ENTRY_HEADER_SIZE))]
```

Definition at line 38 of file tfm_mcuboot_fwu.c.

#### 6.16.2.2 header

```
struct shared_data_tlv_header header
```

Definition at line 37 of file tfm_mcuboot_fwu.c.

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

- secure_fw/partitions/firmware_update/bootloader/mcuboot/tfm_mcuboot_fwu.c

## 6.17 ifx_debug_policy_t Struct Reference

### Data Fields

- uint32_t access_cfg
- uint32_t syscpuss_ap_ctl
- uint32_t reserved

### 6.17.1 Detailed Description

Definition at line 39 of file ifx_platform_provisioning.c.

### 6.17.2 Field Documentation

#### 6.17.2.1 access_cfg

```
uint32_t access_cfg
```

Definition at line 42 of file ifx_platform_provisioning.c.

#### 6.17.2.2 reserved

```
uint32_t reserved
```

Definition at line 45 of file ifx_platform_provisioning.c.

#### 6.17.2.3 syscpuss_ap_ctl

```
uint32_t syscpuss_ap_ctl
```

Definition at line 44 of file ifx_platform_provisioning.c.

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

- platform/ext/target/infineon/pse84/spe/provisioning/ifx_platform_provisioning.c

## 6.18 ifx_driver_flash_obj_t Struct Reference

Structure containing the FLASH driver instance parameters.

```
#include <platform/ext/target/infineon/common/drivers/flash/flash/ifx_↩
driver_flash.h>
```

### Data Fields

- uint32_t start_address
- ARM_FLASH_INFO flash_info

### 6.18.1 Detailed Description

Structure containing the FLASH driver instance parameters.

FLASH driver uses start_address, flash_info.sector_size and flash_info.sector_count to define a working region that is handled by active instance. Any access outside of this region is invalid and FLASH driver returns ARM_DRIVE↩R_ERROR_PARAMETER in such cases.

ifx_driver_flash interface expects address to be relative to start_address.

**Note**

This structure can be a constant.

Definition at line 38 of file ifx_driver_flash.h.

### 6.18.2 Field Documentation

#### 6.18.2.1 flash_info

```
ARM_FLASH_INFO flash_info
```

flash_info.sector_size - sector size for region. It should be multiple of FLASH row size CY_FLASH_SIZEOF_ROW.

Definition at line 45 of file ifx_driver_flash.h.

#### 6.18.2.2 start_address

```
uint32_t start_address
```

Region start address

Definition at line 39 of file ifx_driver_flash.h.

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

- platform/ext/target/infineon/common/drivers/flash/flash/ifx_driver_flash.h

## 6.19 ifx_driver_rram_obj_t Struct Reference

Structure containing the RRAM driver instance parameters.

```
#include <platform/ext/target/infineon/common/drivers/flash/rram/ifx_driver↩
_rram.h>
```

## Data Fields

- RRAMC_Type * rram_base
- uint32_t start_address
- ARM_FLASH_INFO flash_info

### 6.19.1 Detailed Description

Structure containing the RRAM driver instance parameters.

RRAM flash driver uses start_address, flash_info.sector_size and flash_info.sector_count to define a working region that is handled by active instance. Any access outside of this region is invalid and RRAM flash driver returns ARM_DRIVER_ERROR_PARAMETER in such cases.

ifx_driver_rram interface expects address to be relative to start_address.

**Note**

This structure can be a constant.

Definition at line 44 of file ifx_driver_rram.h.

### 6.19.2 Field Documentation

#### 6.19.2.1 flash_info

```
ARM_FLASH_INFO flash_info
```

flash_info.sector_size - sector size for region. It should be multiple of RRAM block size CY_RRAM_BLOCK_SIZE_BYTES.

Definition at line 52 of file ifx_driver_rram.h.

#### 6.19.2.2 rram_base

```
RRAMC_Type* rram_base
```

The pointer to the RRAMC instance.

Definition at line 45 of file ifx_driver_rram.h.

#### 6.19.2.3 start_address

```
uint32_t start_address
```

Region start address

Definition at line 46 of file ifx_driver_rram.h.

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

- platform/ext/target/infineon/common/drivers/flash/rram/ifx_driver_rram.h

## 6.20 ifx_driver_smif_mem_t Struct Reference

Structure containing the SMIF driver memory configuration.

```
#include <platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver↩
_smif.h>
```

### Data Fields

- SMIF_Type * smif_base
- const cy_stc_smif_block_config_t * block_config
- const cy_stc_smif_mem_config_t * mem_config
- const cy_stc_smif_mem_config_t * mem_config_dual_quad_pair
- cy_stc_smif_context_t * smif_context

### 6.20.1 Detailed Description

Structure containing the SMIF driver memory configuration.

SMIF driver instance represent memory region with own boundaries. Such instance can share common SMIF memory configuration and context.

Definition at line 27 of file ifx_driver_smif.h.

### 6.20.2 Field Documentation

#### 6.20.2.1 block_config

```
const cy_stc_smif_block_config_t* block_config
```

SMIF block configuration used for Cy_SMIF_MemInit

Definition at line 29 of file ifx_driver_smif.h.

#### 6.20.2.2 mem_config

```
const cy_stc_smif_mem_config_t* mem_config
```

SMIF memory configuration

Definition at line 30 of file ifx_driver_smif.h.

#### 6.20.2.3 mem_config_dual_quad_pair

```
const cy_stc_smif_mem_config_t* mem_config_dual_quad_pair
```

SMIF dual-quad pair memory configuration

Definition at line 31 of file ifx_driver_smif.h.

#### 6.20.2.4 smif_base

```
SMIF_Type* smif_base
```

Holds the base address of the SMIF block registers

Definition at line 28 of file ifx_driver_smif.h.

#### 6.20.2.5 smif_context

```
cy_stc_smif_context_t* smif_context
```

SMIF driver context

Definition at line 32 of file ifx_driver_smif.h.

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

- platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif.h

## 6.21 ifx_driver_smif_obj_t Struct Reference

Structure containing the SMIF driver instance data.

```
#include <platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver↩
_smif.h>
```

Collaboration diagram for ifx_driver_smif_obj_t:

### Data Fields

- const ifx_driver_smif_mem_t ∗ memory
- uint32_t offset
- uint32_t size
- struct _ARM_FLASH_INFO ∗ flash_info

### 6.21.1 Detailed Description

Structure containing the SMIF driver instance data.

SMIF flash driver uses offset, size to define a working region that is handled by active instance. Any access outside of this region is invalid and SMIF flash driver returns ARM_DRIVER_ERROR_PARAMETER in such cases.

ifx_driver_smif interface expects address to be relative to offset.

**Note**

This structure is modified by ifx_driver_smif.Initialize, so it should not be a constant.

Definition at line 47 of file ifx_driver_smif.h.

### 6.21.2 Field Documentation

#### 6.21.2.1 flash_info

```
struct _ARM_FLASH_INFO* flash_info
```

Definition at line 55 of file ifx_driver_smif.h.

#### 6.21.2.2 memory

```
const ifx_driver_smif_mem_t* memory
```

Memory configuration must be provided by BSP

Definition at line 48 of file ifx_driver_smif.h.

#### 6.21.2.3 offset

```
uint32_t offset
```

Region offset relative to SMIF memory

Definition at line 50 of file ifx_driver_smif.h.

#### 6.21.2.4 size

```
uint32_t size
```

Region size

Definition at line 51 of file ifx_driver_smif.h.

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

- platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif.h

## 6.22 ifx_flash_driver_instance_t Struct Reference

Flash driver instance.

```
#include <platform/ext/target/infineon/common/drivers/flash/ifx_flash_↩
driver_api.h>
```

Collaboration diagram for ifx_flash_driver_instance_t:

### Data Fields

- ifx_flash_driver_handler_t handler
- ifx_flash_driver_t * driver

### 6.22.1 Detailed Description

Flash driver instance.

You can create a separate driver instance for same flash chip. But each instance can handle different areas of flash memory. This allows to provide isolation of data on driver level too.

Definition at line 78 of file ifx_flash_driver_api.h.

### 6.22.2 Field Documentation

#### 6.22.2.1 driver

```
ifx_flash_driver_t* driver
```

Definition at line 80 of file ifx_flash_driver_api.h.

#### 6.22.2.2 handler

```
ifx_flash_driver_handler_t handler
```

Definition at line 79 of file ifx_flash_driver_api.h.

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

- platform/ext/target/infineon/common/drivers/flash/ifx_flash_driver_api.h

## 6.23 ifx_flash_driver_t Struct Reference

Access structure of the Flash Driver.

```
#include <platform/ext/target/infineon/common/drivers/flash/ifx_flash_↩
driver_api.h>
```

### Data Fields

- ARM_DRIVER_VERSION(∗ GetVersion )(ifx_flash_driver_handler_t handler)
- ARM_FLASH_CAPABILITIES(∗ GetCapabilities )(ifx_flash_driver_handler_t handler)
- int32_t(∗ Initialize )(ifx_flash_driver_handler_t handler, ifx_flash_driver_event_t cb_event)
- int32_t(∗ Uninitialize )(ifx_flash_driver_handler_t handler)
- int32_t(∗ PowerControl )(ifx_flash_driver_handler_t handler, ARM_POWER_STATE state)
- int32_t(∗ ReadData )(ifx_flash_driver_handler_t handler, uint32_t addr, void ∗data, uint32_t cnt)
- int32_t(∗ ProgramData )(ifx_flash_driver_handler_t handler, uint32_t addr, const void ∗data, uint32_t cnt)
- int32_t(∗ EraseSector )(ifx_flash_driver_handler_t handler, uint32_t addr)
- int32_t(∗ EraseChip )(ifx_flash_driver_handler_t handler)
- struct _ARM_FLASH_STATUS(∗ GetStatus )(ifx_flash_driver_handler_t handler)
- ARM_FLASH_INFO ∗(∗ GetInfo )(ifx_flash_driver_handler_t handler)

### 6.23.1 Detailed Description

Access structure of the Flash Driver.

Interface is similar to CMSIS flash driver interface (ARM_DRIVER_FLASH) except that there is an additional driver instance handler (see ifx_flash_driver_instance_t::handler) which is specific to driver implementation.

Definition at line 41 of file ifx_flash_driver_api.h.

### 6.23.2 Field Documentation

#### 6.23.2.1 EraseChip

```
int32_t(* EraseChip(ifx_flash_driver_handler_t handler)
```

Definition at line 64 of file ifx_flash_driver_api.h.

#### 6.23.2.2 EraseSector

```
int32_t(* EraseSector(ifx_flash_driver_handler_t handler, uint32_t addr)
```

Definition at line 61 of file ifx_flash_driver_api.h.

#### 6.23.2.3 GetCapabilities

```
ARM_FLASH_CAPABILITIES(* GetCapabilities(ifx_flash_driver_handler_t handler)
```

Definition at line 45 of file ifx_flash_driver_api.h.

#### 6.23.2.4 GetInfo

```
ARM_FLASH_INFO*(* GetInfo(ifx_flash_driver_handler_t handler)
```

Definition at line 68 of file ifx_flash_driver_api.h.

#### 6.23.2.5 GetStatus

```
struct _ARM_FLASH_STATUS(* GetStatus(ifx_flash_driver_handler_t handler)
```

Definition at line 66 of file ifx_flash_driver_api.h.

#### 6.23.2.6 GetVersion

```
ARM_DRIVER_VERSION(* GetVersion(ifx_flash_driver_handler_t handler)
```

Definition at line 43 of file ifx_flash_driver_api.h.

#### 6.23.2.7 Initialize

```
int32_t(* Initialize(ifx_flash_driver_handler_t handler, ifx_flash_driver_event_t cb_event)
```

Definition at line 47 of file ifx_flash_driver_api.h.

#### 6.23.2.8 PowerControl

```
int32_t(* PowerControl(ifx_flash_driver_handler_t handler, ARM_POWER_STATE state)
```

Definition at line 52 of file ifx_flash_driver_api.h.

#### 6.23.2.9 ProgramData

```
int32_t(* ProgramData(ifx_flash_driver_handler_t handler, uint32_t addr, const void *data,
uint32_t cnt)
```

Definition at line 58 of file ifx_flash_driver_api.h.

#### 6.23.2.10 ReadData

```
int32_t(* ReadData(ifx_flash_driver_handler_t handler, uint32_t addr, void *data, uint32_t cnt)
```

Definition at line 55 of file ifx_flash_driver_api.h.

#### 6.23.2.11 Uninitialize

```
int32_t(* Uninitialize(ifx_flash_driver_handler_t handler)
```

Definition at line 50 of file ifx_flash_driver_api.h.

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

- platform/ext/target/infineon/common/drivers/flash/ifx_flash_driver_api.h

## 6.24 ifx_memory_config_t Struct Reference

```
#include <platform/ext/target/infineon/pse84/config/ifx_platform_spe_types.↩
h>
```

## Data Fields

- MPC_Type * mpc
- cy_en_mpc_size_t mpc_block_size
- uint32_t s_address
- uint32_t size

## 6.24.1 Detailed Description

Definition at line 48 of file ifx_platform_spe_types.h.

## 6.24.2 Field Documentation

### 6.24.2.1 mpc

```
MPC_Type* mpc
```

MPC type

Definition at line 49 of file ifx_platform_spe_types.h.

### 6.24.2.2 mpc_block_size

```
cy_en_mpc_size_t mpc_block_size
```

Size of MPC memory block. MPC configures the attributes of each fixed-sized block for all PCs

Definition at line 50 of file ifx_platform_spe_types.h.

### 6.24.2.3 s_address

```
uint32_t s_address
```

MPC secure region base address

Definition at line 52 of file ifx_platform_spe_types.h.

#### 6.24.2.4 size

```
uint32_t size
```

MPC region size

Definition at line 53 of file ifx_platform_spe_types.h.

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

- platform/ext/target/infineon/pse84/config/ifx_platform_spe_types.h

## 6.25 ifx_mpc_cfg_t Struct Reference

### Data Fields

- uint16_t mem_offset
- uint16_t mem_size
- uint32_t attr

### 6.25.1 Detailed Description

Definition at line 40 of file protection_mpc_sw_policy.c.

### 6.25.2 Field Documentation

#### 6.25.2.1 attr

```
uint32_t attr
```

Definition at line 43 of file protection_mpc_sw_policy.c.

#### 6.25.2.2 mem_offset

```
uint16_t mem_offset
```

Definition at line 41 of file protection_mpc_sw_policy.c.

#### 6.25.2.3 mem_size

```
uint16_t mem_size
```

Definition at line 42 of file protection_mpc_sw_policy.c.

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

- platform/ext/target/infineon/common/spe/protection/protection_mpc_sw_policy.c

## 6.26 ifx_mpc_ext_cache_cfg_info_t Struct Reference

### Data Fields

- uint32_t attr
- uint32_t first_block_idx
- uint32_t last_block_idx

### 6.26.1 Detailed Description

Definition at line 46 of file protection_mpc_sert.c.

### 6.26.2 Field Documentation

#### 6.26.2.1 attr

```
uint32_t attr
```

Bit field with MPC cached attributes

Definition at line 48 of file protection_mpc_sert.c.

#### 6.26.2.2 first_block_idx

```
uint32_t first_block_idx
```

First block index in cached attributes

Definition at line 50 of file protection_mpc_sert.c.

#### 6.26.2.3 last_block_idx

```
uint32_t last_block_idx
```

Last block index in cached attributes

Definition at line 52 of file protection_mpc_sert.c.

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

- platform/ext/target/infineon/common/spe/protection/protection_mpc_sert.c

## 6.27 ifx_mpc_numbered_mmio_config_t Struct Reference

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
mpc_hw_mpc.h>
```

### Data Fields

- ifx_pc_mask_t pc_mask
- ifx_pc_mask_t ns_mask

### 6.27.1 Detailed Description

Definition at line 30 of file protection_mpc_hw_mpc.h.

### 6.27.2 Field Documentation

#### 6.27.2.1 ns_mask

```
ifx_pc_mask_t ns_mask
```

NS bit mask for each PC

Definition at line 34 of file protection_mpc_hw_mpc.h.

#### 6.27.2.2 pc_mask

```
ifx_pc_mask_t pc_mask
```

Mask that define what PCs must be modified. 1 - must be applied new value, 0 - must be left old value no matter what it was

Definition at line 31 of file protection_mpc_hw_mpc.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc.h

## 6.28 ifx_mpc_policy_t Struct Reference

Collaboration diagram for ifx_mpc_policy_t:

### Data Fields

- uint8_t n_ram_mpc
- uint8_t n_flash_mpc
- uint8_t unused [2]
- ifx_mpc_cfg_t mpc_struct [32]

### 6.28.1 Detailed Description

Definition at line 48 of file protection_mpc_sw_policy.c.

### 6.28.2 Field Documentation

#### 6.28.2.1 mpc_struct

```
ifx_mpc_cfg_t mpc_struct[32]
```

Definition at line 52 of file protection_mpc_sw_policy.c.

#### 6.28.2.2 n_flash_mpc

```
uint8_t n_flash_mpc
```

Definition at line 50 of file protection_mpc_sw_policy.c.

#### 6.28.2.3 n_ram_mpc

```
uint8_t n_ram_mpc
```

Definition at line 49 of file protection_mpc_sw_policy.c.

#### 6.28.2.4 unused

```
uint8_t unused[2]
```

Definition at line 51 of file protection_mpc_sw_policy.c.

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

- platform/ext/target/infineon/common/spe/protection/protection_mpc_sw_policy.c

## 6.29 ifx_mpc_raw_region_config_t Struct Reference

Configuration structure for MPC raw region.

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
mpc_hw_mpc.h>
```

### Data Fields

- MPC_Type * mpc_base
- cy_en_mpc_size_t mpc_block_size
- uint32_t offset
- uint32_t size
- ifx_pc_mask_t ns_mask
- ifx_pc_mask_t r_mask
- ifx_pc_mask_t w_mask

### 6.29.1 Detailed Description

Configuration structure for MPC raw region.

Definition at line 51 of file protection_mpc_hw_mpc.h.

### 6.29.2 Field Documentation

#### 6.29.2.1 mpc_base

```
MPC_Type* mpc_base
```

Base address of the MPC controller

Definition at line 52 of file protection_mpc_hw_mpc.h.

#### 6.29.2.2 mpc_block_size

```
cy_en_mpc_size_t mpc_block_size
```

Size of the MPC block

Definition at line 53 of file protection_mpc_hw_mpc.h.

#### 6.29.2.3 ns_mask

```
ifx_pc_mask_t ns_mask
```

NS bit mask for each PC

Definition at line 56 of file protection_mpc_hw_mpc.h.

#### 6.29.2.4 offset

```
uint32_t offset
```

Definition at line 54 of file protection_mpc_hw_mpc.h.

#### 6.29.2.5 r_mask

```
ifx_pc_mask_t r_mask
```

R bit mask for each PC

Definition at line 57 of file protection_mpc_hw_mpc.h.

#### 6.29.2.6 size

```
uint32_t size
```

Definition at line 55 of file protection_mpc_hw_mpc.h.

#### 6.29.2.7 w_mask

```
ifx_pc_mask_t w_mask
```

W bit mask for each PC

Definition at line 58 of file protection_mpc_hw_mpc.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc.h

## 6.30 ifx_mpc_region_config_t Struct Reference

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
mpc_hw_mpc.h>
```

### Data Fields

- uint32_t address
- uint32_t size
- ifx_pc_mask_t ns_mask
- ifx_pc_mask_t r_mask
- ifx_pc_mask_t w_mask

### 6.30.1 Detailed Description

Definition at line 37 of file protection_mpc_hw_mpc.h.

## 6.30.2 Field Documentation

### 6.30.2.1 address

```
uint32_t address
```

Definition at line 38 of file protection_mpc_hw_mpc.h.

### 6.30.2.2 ns_mask

```
ifx_pc_mask_t ns_mask
```

NS bit mask for each PC

Definition at line 40 of file protection_mpc_hw_mpc.h.

### 6.30.2.3 r_mask

```
ifx_pc_mask_t r_mask
```

R bit mask for each PC

Definition at line 41 of file protection_mpc_hw_mpc.h.

### 6.30.2.4 size

```
uint32_t size
```

Definition at line 39 of file protection_mpc_hw_mpc.h.

### 6.30.2.5 w_mask

```
ifx_pc_mask_t w_mask
```

W bit mask for each PC

Definition at line 42 of file protection_mpc_hw_mpc.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc.h

## 6.31 ifx_msc_agc_resp_config_t Struct Reference

```
#include <platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection↩
_regions_cfg.h>
```

### Data Fields

- en_ms_ctl_master_sc_acg_t bus_master
- cy_en_ms_ctl_cfg_gate_resp_t gate_resp
- cy_en_ms_ctl_sec_resp_t sec_resp

### 6.31.1 Detailed Description

Definition at line 33 of file protection_regions_cfg.h.

### 6.31.2 Field Documentation

#### 6.31.2.1 bus_master

```
en_ms_ctl_master_sc_acg_t bus_master
```

Definition at line 34 of file protection_regions_cfg.h.

#### 6.31.2.2 gate_resp

```
cy_en_ms_ctl_cfg_gate_resp_t gate_resp
```

Definition at line 35 of file protection_regions_cfg.h.

#### 6.31.2.3 sec_resp

```
cy_en_ms_ctl_sec_resp_t sec_resp
```

Definition at line 36 of file protection_regions_cfg.h.

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

- platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection_regions_cfg.h

## 6.32 ifx_msc_agc_resp_config_v1_t Struct Reference

```
#include <platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection↩
_regions_cfg.h>
```

### Data Fields

- en_ms_ctl_master_sc_acg_v1_t bus_master
- cy_en_ms_ctl_cfg_gate_resp_t gate_resp
- cy_en_ms_ctl_sec_resp_t sec_resp

### 6.32.1 Detailed Description

Definition at line 39 of file protection_regions_cfg.h.

### 6.32.2 Field Documentation

#### 6.32.2.1 bus_master

```
en_ms_ctl_master_sc_acg_v1_t bus_master
```

Definition at line 40 of file protection_regions_cfg.h.

#### 6.32.2.2 gate_resp

```
cy_en_ms_ctl_cfg_gate_resp_t gate_resp
```

Definition at line 41 of file protection_regions_cfg.h.

#### 6.32.2.3 sec_resp

```
cy_en_ms_ctl_sec_resp_t sec_resp
```

Definition at line 42 of file protection_regions_cfg.h.

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

- platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection_regions_cfg.h

## 6.33 ifx_nv_counters Struct Reference

Struct representing the NV counter data in flash.

### Data Fields

- uint32_t checksum
- uint32_t init_value
- union {
  uint32_t counters [((IFX_TFM_NV_COUNTERS_SECTOR_SIZE
  - sizeof(uint32_t)
  - IFX_CHECKSUM_SIZE)/IFX_NV_COUNTER_SIZE)]
  uint8_t bytes [((IFX_TFM_NV_COUNTERS_SECTOR_SIZE
  - sizeof(uint32_t)
  - IFX_CHECKSUM_SIZE)/IFX_NV_COUNTER_SIZE)
  *sizeof(uint32_t)]
  } nv_cnt

### 6.33.1 Detailed Description

Struct representing the NV counter data in flash.

Definition at line 43 of file nv_counters_flash.c.

### 6.33.2 Field Documentation

#### 6.33.2.1 bytes

```
uint8_t bytes[((IFX_TFM_NV_COUNTERS_SECTOR_SIZE - sizeof(uint32_t) - IFX_CHECKSUM_SIZE)/IFX_NV_COUNTER_SIZE)
*sizeof(uint32_t)]
```

Definition at line 48 of file nv_counters_flash.c.

#### 6.33.2.2 checksum

```
uint32_t checksum
```

Definition at line 44 of file nv_counters_flash.c.

#### 6.33.2.3 counters

```
uint32_t counters[((IFX_TFM_NV_COUNTERS_SECTOR_SIZE - sizeof(uint32_t) - IFX_CHECKSUM_SIZE)/IFX_NV_COUNTER_SIZ
```

Array of NV counters

Definition at line 47 of file nv_counters_flash.c.

#### 6.33.2.4 init_value

```
uint32_t init_value
```

Definition at line 45 of file nv_counters_flash.c.

#### 6.33.2.5 nv_cnt

```
union { ... } nv_cnt
```

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

- platform/ext/target/infineon/common/spe/services/platform/nv_counters_flash.c

## 6.34 ifx_oem_policy_t Struct Reference

Collaboration diagram for ifx_oem_policy_t:

## Data Fields

- ifx_debug_policy_t dbg_policy
- uint32_t rma
- uint8_t debug_key [(65U)]
- uint8_t padding [3]
- uint32_t boundary_scan
- uint32_t warm_boot
- uint32_t reserved

### 6.34.1 Detailed Description

Definition at line 52 of file ifx_platform_provisioning.c.

### 6.34.2 Field Documentation

#### 6.34.2.1 boundary_scan

```
uint32_t boundary_scan
```

Definition at line 58 of file ifx_platform_provisioning.c.

#### 6.34.2.2 dbg_policy

```
ifx_debug_policy_t dbg_policy
```

Definition at line 54 of file ifx_platform_provisioning.c.

#### 6.34.2.3 debug_key

```
uint8_t debug_key[(65U)]
```

Definition at line 56 of file ifx_platform_provisioning.c.

#### 6.34.2.4 padding

```
uint8_t padding[3]
```

Definition at line 57 of file ifx_platform_provisioning.c.

#### 6.34.2.5 reserved

```
uint32_t reserved
```

Definition at line 60 of file ifx_platform_provisioning.c.

#### 6.34.2.6 rma

```
uint32_t rma
```

Definition at line 55 of file ifx_platform_provisioning.c.

#### 6.34.2.7 warm_boot

```
uint32_t warm_boot
```

Definition at line 59 of file ifx_platform_provisioning.c.

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

- platform/ext/target/infineon/pse84/spe/provisioning/ifx_platform_provisioning.c

## 6.35 ifx_partition_info_t Struct Reference

Structure describing partition info.

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
types.h>
```

Collaboration diagram for ifx_partition_info_t:

## Data Fields

- const struct partition_load_info_t ∗ p_ldinfo
- const ifx_partition_load_info_t ∗ ifx_ldinfo
- uintptr_t ifx_domain
- const struct asset_desc_t ∗ p_assets

  *Platform specific assets which can't be added to manifest.*
- uint32_t asset_cnt

  *Number of items in p_assets.*
- const ifx_ppc_regions_config_t ∗ p_peri_regions

  *Partition peripheral assets added by platform or Edge Protect Configurator.*
- uint32_t peri_region_count

  *Number of items in p_peri_regions.*

### 6.35.1 Detailed Description

Structure describing partition info.

Pointer to this structure is used to provide boundary for isolation HAL.

Definition at line 203 of file protection_types.h.

### 6.35.2 Field Documentation

#### 6.35.2.1 asset_cnt

```
uint32_t asset_cnt
```

Number of items in p_assets.

Definition at line 213 of file protection_types.h.

#### 6.35.2.2 ifx_domain

```
uintptr_t ifx_domain
```

Definition at line 209 of file protection_types.h.

#### 6.35.2.3 ifx_ldinfo

```
const ifx_partition_load_info_t* ifx_ldinfo
```

Definition at line 207 of file protection_types.h.

#### 6.35.2.4 p_assets

```
const struct asset_desc_t* p_assets
```

Platform specific assets which can't be added to manifest.

Definition at line 211 of file protection_types.h.

#### 6.35.2.5 p_ldinfo

```
const struct partition_load_info_t* p_ldinfo
```

Definition at line 205 of file protection_types.h.

#### 6.35.2.6 p_peri_regions

```
const ifx_ppc_regions_config_t* p_peri_regions
```

Partition peripheral assets added by platform or Edge Protect Configurator.

Definition at line 219 of file protection_types.h.

#### 6.35.2.7 peri_region_count

```
uint32_t peri_region_count
```

Number of items in p_peri_regions.

Definition at line 221 of file protection_types.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_types.h

## 6.36 ifx_partition_load_info_t Struct Reference

Structure used to store platform specific partition properties.

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
types.h>
```

Collaboration diagram for ifx_partition_load_info_t:

### Data Fields

- IFX_FIH_BOOL privileged

  *Is partition privileged.*
- const ifx_protection_region_config_t * protect_cfg_enabled

  *Partition configuration applied by tfm_hal_bind_boundary or by tfm_hal_activate_boundary when partition is enabled.*

### 6.36.1 Detailed Description

Structure used to store platform specific partition properties.

Definition at line 177 of file protection_types.h.

### 6.36.2 Field Documentation

#### 6.36.2.1 privileged

```
IFX_FIH_BOOL privileged
```

Is partition privileged.

Definition at line 179 of file protection_types.h.

#### 6.36.2.2 protect_cfg_enabled

```
const ifx_protection_region_config_t* protect_cfg_enabled
```

Partition configuration applied by tfm_hal_bind_boundary or by tfm_hal_activate_boundary when partition is enabled.

Definition at line 190 of file protection_types.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_types.h

## 6.37 ifx_plat_epc2_keys_t Struct Reference

```
#include <platform/ext/target/infineon/pse84/epc2/spe/services/crypto/crypto↩
_keys_rram.h>
```

### Data Fields

- uint8_t iak_private [32]
- uint8_t iak_private_checksum [4]
- uint8_t iak_public [65]
- uint8_t iak_public_checksum [4]
- uint8_t huk [32]
- uint8_t huk_checksum [4]

### 6.37.1 Detailed Description

Definition at line 15 of file crypto_keys_rram.h.

### 6.37.2 Field Documentation

#### 6.37.2.1 huk

```
uint8_t huk[32]
```

Definition at line 20 of file crypto_keys_rram.h.

#### 6.37.2.2 huk_checksum

```
uint8_t huk_checksum[4]
```

Definition at line 21 of file crypto_keys_rram.h.

#### 6.37.2.3 iak_private

```
uint8_t iak_private[32]
```

Definition at line 16 of file crypto_keys_rram.h.

#### 6.37.2.4 iak_private_checksum

```
uint8_t iak_private_checksum[4]
```

Definition at line 17 of file crypto_keys_rram.h.

#### 6.37.2.5 iak_public

```
uint8_t iak_public[65]
```

Definition at line 18 of file crypto_keys_rram.h.

#### 6.37.2.6 iak_public_checksum

```
uint8_t iak_public_checksum[4]
```

Definition at line 19 of file crypto_keys_rram.h.

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

- platform/ext/target/infineon/pse84/epc2/spe/services/crypto/crypto_keys_rram.h

## 6.38 ifx_ppc_named_mmio_config_t Struct Reference

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
ppc_v1.h>
```

## Data Fields

- cy_en_ppc_sec_attribute_t sec_attr
- bool allow_unpriv
- ifx_pc_mask_t pc_mask

## 6.38.1 Detailed Description

Definition at line 29 of file protection_ppc_v1.h.

## 6.38.2 Field Documentation

### 6.38.2.1 allow_unpriv

```
bool allow_unpriv
```

Whether unprivileged access is permitted

Definition at line 31 of file protection_ppc_v1.h.

### 6.38.2.2 pc_mask

```
ifx_pc_mask_t pc_mask
```

PC mask

Definition at line 32 of file protection_ppc_v1.h.

### 6.38.2.3 sec_attr

```
cy_en_ppc_sec_attribute_t sec_attr
```

Security attribute

Definition at line 30 of file protection_ppc_v1.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_ppc_v1.h
- platform/ext/target/infineon/common/spe/protection/protection_ppc_v2.h

## 6.39 ifx_ppc_regions_config_t Struct Reference

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
types.h>
```

### Data Fields

- const cy_en_prot_region_t * regions

  *Pointer to the array of regions for the domain.*
- uint32_t region_count

  *Number of items in regions.*

### 6.39.1 Detailed Description

Definition at line 144 of file protection_types.h.

### 6.39.2 Field Documentation

#### 6.39.2.1 region_count

```
uint32_t region_count
```

Number of items in regions.

Definition at line 148 of file protection_types.h.

#### 6.39.2.2 regions

```
const cy_en_prot_region_t* regions
```

Pointer to the array of regions for the domain.

Definition at line 146 of file protection_types.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_types.h

## 6.40 ifx_ppcx_config_t Struct Reference

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
ppc_v1.h>
```

## Data Fields

- cy_en_ppc_sec_attribute_t sec_attr
- bool allow_unpriv
- ifx_pc_mask_t pc_mask
- const cy_en_prot_region_t * regions
- size_t region_count

## 6.40.1 Detailed Description

Definition at line 35 of file protection_ppc_v1.h.

## 6.40.2 Field Documentation

### 6.40.2.1 allow_unpriv

```
bool allow_unpriv
```

Whether unprivileged access is permitted

Definition at line 37 of file protection_ppc_v1.h.

### 6.40.2.2 pc_mask

```
ifx_pc_mask_t pc_mask
```

PC mask

Definition at line 38 of file protection_ppc_v1.h.

### 6.40.2.3 region_count

```
size_t region_count
```

Number of items in /ref regions

Definition at line 40 of file protection_ppc_v1.h.

#### 6.40.2.4 regions

```
const cy_en_prot_region_t * regions
```

Array of PPC regions

Definition at line 39 of file protection_ppc_v1.h.

#### 6.40.2.5 sec_attr

```
cy_en_ppc_sec_attribute_t sec_attr
```

Security attribute

Definition at line 36 of file protection_ppc_v1.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_ppc_v1.h
- platform/ext/target/infineon/common/spe/protection/protection_ppc_v2.h

## 6.41 ifx_protection_region_config_t Struct Reference

Structure used to store platform specific protection properties.

```
#include <platform/ext/target/infineon/common/spe/protection/protection_↩
types.h>
```

### Data Fields

- IFX_FIH_BOOL mpu_regions_enable

### 6.41.1 Detailed Description

Structure used to store platform specific protection properties.

Definition at line 154 of file protection_types.h.

### 6.41.2 Field Documentation

#### 6.41.2.1 mpu_regions_enable

```
IFX_FIH_BOOL mpu_regions_enable
```

Definition at line 161 of file protection_types.h.

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

- platform/ext/target/infineon/common/spe/protection/protection_types.h

## 6.42 ifx_rram_counter_t Struct Reference

### Data Fields

- union {
  ifx_rram_counter_value_t value
  uint8_t bytes [sizeof(ifx_rram_counter_value_t)]
  } data

- uint32_t checksum

### 6.42.1 Detailed Description

Definition at line 23 of file nv_counters_rram.c.

### 6.42.2 Field Documentation

#### 6.42.2.1 bytes

```
uint8_t bytes[sizeof(ifx_rram_counter_value_t)]
```

Definition at line 28 of file nv_counters_rram.c.

#### 6.42.2.2 checksum

```
uint32_t checksum
```

Definition at line 30 of file nv_counters_rram.c.

#### 6.42.2.3 data

```
union { ...  } data
```

#### 6.42.2.4 value

```
ifx_rram_counter_value_t value
```

Definition at line 27 of file nv_counters_rram.c.

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

- platform/ext/target/infineon/common/spe/services/platform/nv_counters_rram.c

## 6.43 irq_load_info_t Struct Reference

```
#include <secure_fw/spm/include/load/interrupt_defs.h>
```

### Data Fields

- enum tfm_hal_status_t(∗ init )(void ∗pt, const struct irq_load_info_t ∗pildi)
- psa_flih_result_t(∗ flih_func )(void)
- int32_t pid
- uint32_t source
- psa_signal_t signal
- int32_t client_id_base
- int32_t client_id_limit

### 6.43.1 Detailed Description

Definition at line 19 of file interrupt_defs.h.

### 6.43.2 Field Documentation

#### 6.43.2.1 client_id_base

```
int32_t client_id_base
```

Definition at line 29 of file interrupt_defs.h.

#### 6.43.2.2 client_id_limit

```
int32_t client_id_limit
```

Definition at line 30 of file interrupt_defs.h.

#### 6.43.2.3 flih_func

```
psa_flih_result_t(* flih_func(void)
```

Definition at line 25 of file interrupt_defs.h.

#### 6.43.2.4 init

```
enum tfm_hal_status_t(* init(void *pt, const struct irq_load_info_t *pildi)
```

Definition at line 24 of file interrupt_defs.h.

#### 6.43.2.5 pid

```
int32_t pid
```

Definition at line 26 of file interrupt_defs.h.

#### 6.43.2.6 signal

```
psa_signal_t signal
```

Definition at line 28 of file interrupt_defs.h.

#### 6.43.2.7 source

```
uint32_t source
```

Definition at line 27 of file interrupt_defs.h.

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

- secure_fw/spm/include/load/interrupt_defs.h

## 6.44 irq_t Struct Reference

```
#include <secure_fw/spm/include/load/interrupt_defs.h>
```

Collaboration diagram for irq_t:

### Data Fields

- void * p_pt
- const struct irq_load_info_t * p_ildi

### 6.44.1 Detailed Description

Definition at line 34 of file interrupt_defs.h.

### 6.44.2 Field Documentation

#### 6.44.2.1 p_ildi

```
const struct irq_load_info_t* p_ildi
```

Definition at line 36 of file interrupt_defs.h.

#### 6.44.2.2 p_pt

```
void* p_pt
```

Definition at line 35 of file interrupt_defs.h.

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

- secure_fw/spm/include/load/interrupt_defs.h

## 6.45 its_block_meta_t Struct Reference

Structure to store information about each physical flash memory block.

```
#include <secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash↩
_fs_mblock.h>
```

### Data Fields

- uint32_t phy_id
- size_t data_start
- size_t free_size

### 6.45.1 Detailed Description

Structure to store information about each physical flash memory block.

**Note**

This structure is programmed to flash, so its size must be padded to a multiple of the maximum required flash program unit.

Definition at line 130 of file its_flash_fs_mblock.h.

### 6.45.2 Field Documentation

#### 6.45.2.1 data_start

```
size_t data_start
```

Offset from the beginning of the block to the ∗ location where the data starts

Definition at line 131 of file its_flash_fs_mblock.h.

#### 6.45.2.2 free_size

```
size_t free_size
```

Number of bytes free at end of block (set during ∗ block compaction for gap reuse)

Definition at line 131 of file its_flash_fs_mblock.h.

#### 6.45.2.3 phy_id

```
uint32_t phy_id
```

Physical ID of this logical block

Definition at line 131 of file its_flash_fs_mblock.h.

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

- secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.h

## 6.46 its_file_meta_t Struct Reference

Structure to store file metadata.

```
#include <secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash↩
_fs_mblock.h>
```

### Data Fields

- uint32_t lblock
- size_t data_idx
- size_t cur_size
- size_t max_size
- uint32_t flags
- uint8_t id [ITS_FILE_ID_SIZE]

### 6.46.1 Detailed Description

Structure to store file metadata.

**Note**

This structure is programmed to flash, so its size must be padded to a multiple of the maximum required flash program unit.

Definition at line 167 of file its_flash_fs_mblock.h.

### 6.46.2 Field Documentation

#### 6.46.2.1 cur_size

```
size_t cur_size
```

Definition at line 168 of file its_flash_fs_mblock.h.

#### 6.46.2.2 data_idx

```
size_t data_idx
```

Definition at line 168 of file its_flash_fs_mblock.h.

#### 6.46.2.3 flags

```
uint32_t flags
```

Definition at line 168 of file its_flash_fs_mblock.h.

#### 6.46.2.4 id

```
uint8_t id[ITS_FILE_ID_SIZE]
```

Definition at line 168 of file its_flash_fs_mblock.h.

#### 6.46.2.5 lblock

```
uint32_t lblock
```

Definition at line 168 of file its_flash_fs_mblock.h.

#### 6.46.2.6 max_size

```
size_t max_size
```

Definition at line 168 of file its_flash_fs_mblock.h.

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

- secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.h

## 6.47 its_flash_config_t Struct Reference

Structure containing ITS flash driver instance configuration used by ITS flash FS layer.

```
#include <secure_fw/partitions/internal_trusted_storage/flash/its_flash_↩
hal.h>
```

### Data Fields

- const void * context
- uint32_t flash_area_addr
- uint32_t block_size
- uint16_t num_blocks
- uint16_t program_unit
- uint8_t erase_val

### 6.47.1 Detailed Description

Structure containing ITS flash driver instance configuration used by ITS flash FS layer.

There are values which should be initialized by its_flash_ops_t::init of ops or set to constant values during allocation of this structure:

- erase_val
- program_unit

Following members are setup by ITS:

- block_size
- flash_area_addr
- num_blocks

Definition at line 77 of file its_flash_hal.h.

### 6.47.2 Field Documentation

#### 6.47.2.1 block_size

```
uint32_t block_size
```

Size of a logical filesystem erase unit, a multiple of sector_size.

Definition at line 83 of file its_flash_hal.h.

#### 6.47.2.2 context

```
const void* context
```

ITS flash driver context

Definition at line 78 of file its_flash_hal.h.

#### 6.47.2.3 erase_val

```
uint8_t erase_val
```

Value of a byte after erase (usually 0xFF)

Definition at line 88 of file its_flash_hal.h.

#### 6.47.2.4 flash_area_addr

```
uint32_t flash_area_addr
```

Base address of the flash region in flash memory space. It's not mandatory that this address is mapped to MCU address space.

Definition at line 79 of file its_flash_hal.h.

#### 6.47.2.5 num_blocks

```
uint16_t num_blocks
```

Number of logical erase blocks

Definition at line 86 of file its_flash_hal.h.

#### 6.47.2.6 program_unit

```
uint16_t program_unit
```

Minimum size of a program operation

Definition at line 87 of file its_flash_hal.h.

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

- secure_fw/partitions/internal_trusted_storage/flash/its_flash_hal.h

## 6.48 its_flash_fs_config_t Struct Reference

Structure containing the flash filesystem configuration parameters.

```
#include <secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash↩
_fs.h>
```

Collaboration diagram for its_flash_fs_config_t:

### Data Fields

- struct its_flash_config_t * flash_cfg
- const struct its_flash_ops_t * ops
- uint16_t max_file_size
- uint16_t max_num_files

### 6.48.1 Detailed Description

Structure containing the flash filesystem configuration parameters.

Definition at line 49 of file its_flash_fs.h.

### 6.48.2 Field Documentation

#### 6.48.2.1 flash_cfg

```
struct its_flash_config_t* flash_cfg
```

Pointer to the flash device

Definition at line 50 of file its_flash_fs.h.

#### 6.48.2.2 max_file_size

```
uint16_t max_file_size
```

Maximum file size

Definition at line 52 of file its_flash_fs.h.

#### 6.48.2.3 max_num_files

```
uint16_t max_num_files
```

Maximum number of files

Definition at line 53 of file its_flash_fs.h.

#### 6.48.2.4 ops

```
const struct its_flash_ops_t* ops
```

Filesystem flash operations

Definition at line 51 of file its_flash_fs.h.

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

- secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs.h

## 6.49 its_flash_fs_ctx_t Struct Reference

Structure to store the ITS flash file system context.

```
#include <secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash↩
_fs_mblock.h>
```

Collaboration diagram for its_flash_fs_ctx_t:

## Data Fields

- const struct its_flash_fs_config_t * cfg
- struct its_metadata_block_header_t meta_block_header
- uint32_t active_metablock
- uint32_t scratch_metablock

### 6.49.1 Detailed Description

Structure to store the ITS flash file system context.

Definition at line 181 of file its_flash_fs_mblock.h.

### 6.49.2 Field Documentation

#### 6.49.2.1 active_metablock

```
uint32_t active_metablock
```

Active metadata block

Definition at line 186 of file its_flash_fs_mblock.h.

#### 6.49.2.2 cfg

```
const struct its_flash_fs_config_t* cfg
```

Filesystem configuration

Definition at line 182 of file its_flash_fs_mblock.h.

#### 6.49.2.3 meta_block_header

```
struct its_metadata_block_header_t meta_block_header
```

Metadata block header

Definition at line 183 of file its_flash_fs_mblock.h.

#### 6.49.2.4 scratch_metablock

```
uint32_t scratch_metablock
```

Scratch metadata block

Definition at line 187 of file its_flash_fs_mblock.h.

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

- secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.h

## 6.50 its_flash_fs_file_info_t Struct Reference

Structure containing file information.

```
#include <secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash↩
_fs.h>
```

### Data Fields

- size_t size_current
- size_t size_max
- uint32_t flags

### 6.50.1 Detailed Description

Structure containing file information.

This structure is not written to the filesystem, it is used by the file system functions to simplify accessing the containing information.

Definition at line 76 of file its_flash_fs.h.

### 6.50.2 Field Documentation

#### 6.50.2.1 flags

```
uint32_t flags
```

Flags set when the file was created

Definition at line 79 of file its_flash_fs.h.

#### 6.50.2.2 size_current

```
size_t size_current
```

The current size of the file in bytes

Definition at line 77 of file its_flash_fs.h.

#### 6.50.2.3 size_max

```
size_t size_max
```

The maximum size of the file in bytes.

Definition at line 78 of file its_flash_fs.h.

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

- secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs.h

## 6.51 its_flash_nand_dev_t Struct Reference

```
#include <secure_fw/partitions/internal_trusted_storage/flash/its_flash_↩
nand.h>
```

### Data Fields

- ARM_DRIVER_FLASH * driver
- uint32_t sector_size
- uint32_t buf_block_id_0
- uint32_t buf_block_id_1
- uint8_t * write_buf_0
- uint8_t * write_buf_1
- size_t buf_size

### 6.51.1 Detailed Description

Definition at line 30 of file its_flash_nand.h.

### 6.51.2 Field Documentation

#### 6.51.2.1 buf_block_id_0

```
uint32_t buf_block_id_0
```

Definition at line 37 of file its_flash_nand.h.

#### 6.51.2.2 buf_block_id_1

```
uint32_t buf_block_id_1
```

Definition at line 38 of file its_flash_nand.h.

#### 6.51.2.3 buf_size

```
size_t buf_size
```

Definition at line 41 of file its_flash_nand.h.

#### 6.51.2.4 driver

```
ARM_DRIVER_FLASH* driver
```

Definition at line 31 of file its_flash_nand.h.

#### 6.51.2.5 sector_size

```
uint32_t sector_size
```

Definition at line 33 of file its_flash_nand.h.

#### 6.51.2.6 write_buf_0

```
uint8_t* write_buf_0
```

Definition at line 39 of file its_flash_nand.h.

#### 6.51.2.7 write_buf_1

```
uint8_t* write_buf_1
```

Definition at line 40 of file its_flash_nand.h.

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

- secure_fw/partitions/internal_trusted_storage/flash/its_flash_nand.h

## 6.52 its_flash_ops_t Struct Reference

Structure containing the ITS flash driver operations.

```
#include <secure_fw/partitions/internal_trusted_storage/flash/its_flash_↩
hal.h>
```

### Data Fields

- psa_status_t(∗ init )(struct its_flash_config_t ∗cfg)

  *Initializes the flash device.*
- psa_status_t(∗ read )(const struct its_flash_config_t ∗cfg, uint32_t block_id, uint8_t ∗buf, size_t offset, size↩_t size)

  *Reads block data from the position specified by block ID and offset.*
- psa_status_t(∗ write )(const struct its_flash_config_t ∗cfg, uint32_t block_id, const uint8_t ∗buf, size_t offset, size_t size)

  *Writes block data to the position specified by block ID and offset.*
- psa_status_t(∗ flush )(const struct its_flash_config_t ∗cfg, uint32_t block_id)

  *Flushes modifications to a block to flash. Must be called after a sequence of calls to write() (including via its_flash↩_block_to_block_move()) for one block ID, before any call to the same functions for a different block ID.*
- psa_status_t(∗ erase )(const struct its_flash_config_t ∗cfg, uint32_t block_id)

  *Erases block ID data.*

### 6.52.1 Detailed Description

Structure containing the ITS flash driver operations.

Definition at line 94 of file its_flash_hal.h.

### 6.52.2 Field Documentation

#### 6.52.2.1 erase

```
psa_status_t(* erase(const struct its_flash_config_t *cfg, uint32_t block_id)
```

Erases block ID data.

**Parameters**

| | | |
|---|---|---|
| in | *cfg* | Flash driver configuration |
| in | *block↩ _id* | Block ID |

**Note**

This function assumes the input value is valid.

**Returns**

Returns PSA_SUCCESS if the function is executed correctly. Otherwise, it returns PSA_ERROR_STORA↩ GE_FAILURE.

Definition at line 177 of file its_flash_hal.h.

#### 6.52.2.2 flush

```
psa_status_t(* flush(const struct its_flash_config_t *cfg, uint32_t block_id)
```

Flushes modifications to a block to flash. Must be called after a sequence of calls to write() (including via its_flash↩ _block_to_block_move()) for one block ID, before any call to the same functions for a different block ID.

**Parameters**

| | | |
|---|---|---|
| in | *cfg* | Flash driver configuration |
| in | *block↩ _id* | Block ID |

**Note**

It is permitted for write() to commit block updates immediately, in which case this function is a no-op.

**Returns**

Returns PSA_SUCCESS if the function is executed correctly. Otherwise, it returns PSA_ERROR_STORA↩ GE_FAILURE.

Definition at line 163 of file its_flash_hal.h.

#### 6.52.2.3 init

```
psa_status_t(* init(struct its_flash_config_t *cfg)
```

Initializes the flash device.

**Parameters**

| | |
|---|---|
| *[in/out]* | cfg Flash driver configuration |

**Returns**

Returns PSA_SUCCESS if the function is executed correctly. Otherwise, it returns PSA_ERROR_STORA↩GE_FAILURE.

Definition at line 103 of file its_flash_hal.h.

#### 6.52.2.4 read

```
psa_status_t(* read(const struct its_flash_config_t *cfg, uint32_t block_id, uint8_t *buf,
size_t offset, size_t size)
```

Reads block data from the position specified by block ID and offset.

**Parameters**

| | | |
|---|---|---|
| in | *cfg* | Flash driver configuration |
| in | *block↩_id* | Block ID |
| out | *buf* | Buffer pointer to store the data read |
| in | *offset* | Offset position from the init of the block |
| in | *size* | Number of bytes to read |

**Note**

This function assumes all input values are valid. That is, the address range, based on block_id, offset and size, is a valid range in flash.

**Returns**

Returns PSA_SUCCESS if the function is executed correctly. Otherwise, it returns PSA_ERROR_STORA↩GE_FAILURE.

Definition at line 122 of file its_flash_hal.h.

#### 6.52.2.5 write

```
psa_status_t(* write(const struct its_flash_config_t *cfg, uint32_t block_id, const uint8_↩
t *buf, size_t offset, size_t size)
```

Writes block data to the position specified by block ID and offset.

**Parameters**

| | | |
|---|---|---|
| in | *cfg* | Flash driver configuration |
| in | *block↩ _id* | Block ID |
| in | *buf* | Buffer pointer to the write data |
| in | *offset* | Offset position from the init of the block |
| in | *size* | Number of bytes to write |

**Note**

This function assumes all input values are valid. That is, the address range, based on block_id, offset and size, is a valid range in flash.

**Returns**

Returns PSA_SUCCESS if the function is executed correctly. Otherwise, it returns PSA_ERROR_STORA↩ GE_FAILURE.

Definition at line 143 of file its_flash_hal.h.

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

- secure_fw/partitions/internal_trusted_storage/flash/its_flash_hal.h

## 6.53 its_flash_ram_dev_t Struct Reference

ITS RAM driver context.

```
#include <secure_fw/partitions/internal_trusted_storage/flash/its_flash_↩
ram.h>
```

### Data Fields

- uint8_t ∗ buffer
- uint32_t size

### 6.53.1 Detailed Description

ITS RAM driver context.

Definition at line 31 of file its_flash_ram.h.

### 6.53.2 Field Documentation

#### 6.53.2.1 buffer

```
uint8_t* buffer
```

Definition at line 33 of file its_flash_ram.h.

#### 6.53.2.2 size

```
uint32_t size
```

Definition at line 36 of file its_flash_ram.h.

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

- secure_fw/partitions/internal_trusted_storage/flash/its_flash_ram.h

## 6.54 its_metadata_block_header_comp_t Struct Reference

The struture of metadata block header in ITS_BACKWARD_SUPPORTED_VERSION.

```
#include <secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash↩
_fs_mblock.h>
```

### Data Fields

- uint32_t scratch_dblock
- uint8_t fs_version
- uint8_t active_swap_count

### 6.54.1 Detailed Description

The struture of metadata block header in ITS_BACKWARD_SUPPORTED_VERSION.

**Note**

The active_swap_count must be the last member to allow it to be programmed last.

This structure is programmed to flash, so its size must be padded to a multiple of the maximum required flash program unit.

Definition at line 106 of file its_flash_fs_mblock.h.

### 6.54.2 Field Documentation

#### 6.54.2.1 active_swap_count

```
uint8_t active_swap_count
```

Number of times the metadata blocks have ∗ been swapped

Definition at line 107 of file its_flash_fs_mblock.h.

#### 6.54.2.2 fs_version

```
uint8_t fs_version
```

Filesystem version

Definition at line 107 of file its_flash_fs_mblock.h.

#### 6.54.2.3 scratch_dblock

```
uint32_t scratch_dblock
```

Physical block ID of the data ∗ section's scratch block

Definition at line 107 of file its_flash_fs_mblock.h.

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

- secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.h

## 6.55 its_metadata_block_header_t Struct Reference

Structure to store the metadata block header.

```
#include <secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash↩
_fs_mblock.h>
```

### Data Fields

- uint32_t scratch_dblock
- uint8_t fs_version
- uint8_t metadata_xor
- uint8_t active_swap_count

### 6.55.1 Detailed Description

Structure to store the metadata block header.

**Note**

The active_swap_count must be the last member to allow it to be programmed last. The fs_version must be at the same position as it is in its_metadata_block_header_comp_t.

This structure is programmed to flash, so its size must be padded to a multiple of the maximum required flash program unit.

Definition at line 75 of file its_flash_fs_mblock.h.

### 6.55.2 Field Documentation

#### 6.55.2.1 active_swap_count

```
uint8_t active_swap_count
```

Number of times the metadata blocks have ∗ been swapped

Definition at line 76 of file its_flash_fs_mblock.h.

#### 6.55.2.2 fs_version

```
uint8_t fs_version
```

Filesystem version

Definition at line 76 of file its_flash_fs_mblock.h.

#### 6.55.2.3 metadata_xor

```
uint8_t metadata_xor
```

XOR value based on the whole metadata(not ∗ including the metadata block header)

Definition at line 76 of file its_flash_fs_mblock.h.

#### 6.55.2.4 scratch_dblock

```
uint32_t scratch_dblock
```

Physical block ID of the data ∗ section's scratch block

Definition at line 76 of file its_flash_fs_mblock.h.

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

- secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_fs_mblock.h

## 6.56 mailbox_init_t Struct Reference

```
#include <interface/include/multi_core/tfm_mailbox.h>
```

Collaboration diagram for mailbox_init_t:

### Public Member Functions

- struct mailbox_status_t ∗status __ALIGNED (32)

### Data Fields

- uint32_t slot_count
- struct mailbox_slot_t ∗ slots

### 6.56.1 Detailed Description

Definition at line 157 of file tfm_mailbox.h.

### 6.56.2 Member Function Documentation

#### 6.56.2.1 __ALIGNED()

```
struct mailbox_status_t* status __ALIGNED (
            32  )
```

### 6.56.3 Field Documentation

#### 6.56.3.1 slot_count

```
uint32_t slot_count
```

Definition at line 162 of file tfm_mailbox.h.

#### 6.56.3.2 slots

```
struct mailbox_slot_t* slots
```

Definition at line 165 of file tfm_mailbox.h.

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

- interface/include/multi_core/tfm_mailbox.h

## 6.57 mailbox_msg_t Struct Reference

```
#include <interface/include/multi_core/tfm_mailbox.h>
```

Collaboration diagram for mailbox_msg_t:

### Data Fields

- uint32_t call_type
- struct psa_client_params_t params
- int32_t client_id

### 6.57.1 Detailed Description

Definition at line 98 of file tfm_mailbox.h.

### 6.57.2 Field Documentation

#### 6.57.2.1 call_type

```
uint32_t call_type
```

Definition at line 99 of file tfm_mailbox.h.

#### 6.57.2.2 client_id

```
int32_t client_id
```

Definition at line 104 of file tfm_mailbox.h.

#### 6.57.2.3 params

```
struct psa_client_params_t params
```

Definition at line 100 of file tfm_mailbox.h.

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

- interface/include/multi_core/tfm_mailbox.h

## 6.58 mailbox_reply_t Struct Reference

```
#include <interface/include/multi_core/tfm_mailbox.h>
```

### Data Fields

- int32_t return_val
- size_t out_vec_len [PSA_MAX_IOVEC]

### 6.58.1 Detailed Description

Definition at line 116 of file tfm_mailbox.h.

### 6.58.2 Field Documentation

#### 6.58.2.1 out_vec_len

```
size_t out_vec_len[PSA_MAX_IOVEC]
```

Definition at line 118 of file tfm_mailbox.h.

#### 6.58.2.2 return_val

```
int32_t return_val
```

Definition at line 117 of file tfm_mailbox.h.

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

- interface/include/multi_core/tfm_mailbox.h

## 6.59 mailbox_slot_t Struct Reference

```
#include <interface/include/multi_core/tfm_mailbox.h>
```

### Public Member Functions

- struct mailbox_msg_t msg __ALIGNED (32)
- struct mailbox_reply_t reply __ALIGNED (32)

### 6.59.1 Detailed Description

Definition at line 127 of file tfm_mailbox.h.

### 6.59.2 Member Function Documentation

#### 6.59.2.1 __ALIGNED() [1/2]

```
struct mailbox_msg_t msg __ALIGNED (
            32  )
```

#### 6.59.2.2 __ALIGNED() [2/2]

```
struct mailbox_reply_t reply __ALIGNED (
            32  )
```

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

- interface/include/multi_core/tfm_mailbox.h

## 6.60 mailbox_status_t Struct Reference

```
#include <interface/include/multi_core/tfm_mailbox.h>
```

### Data Fields

- mailbox_queue_status_t pend_slots
- mailbox_queue_status_t replied_slots

### 6.60.1 Detailed Description

Definition at line 143 of file tfm_mailbox.h.

### 6.60.2 Field Documentation

#### 6.60.2.1 pend_slots

```
mailbox_queue_status_t pend_slots
```

Definition at line 144 of file tfm_mailbox.h.

#### 6.60.2.2 replied_slots

```
mailbox_queue_status_t replied_slots
```

Definition at line 147 of file tfm_mailbox.h.

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

- interface/include/multi_core/tfm_mailbox.h

## 6.61 mpu_armv8m_region_cfg_t Struct Reference

```
#include <platform/ext/target/infineon/common/drivers/protection/mpu_armv8m↩
_drv.h>
```

### Data Fields

- uint32_t region_base
- uint32_t region_limit
- uint32_t region_attridx
- enum mpu_armv8m_attr_exec_t attr_exec
- enum mpu_armv8m_attr_access_t attr_access
- enum mpu_armv8m_attr_shared_t attr_sh

### 6.61.1 Detailed Description

Definition at line 49 of file mpu_armv8m_drv.h.

### 6.61.2 Field Documentation

#### 6.61.2.1 attr_access

```
enum mpu_armv8m_attr_access_t attr_access
```

Definition at line 54 of file mpu_armv8m_drv.h.

#### 6.61.2.2 attr_exec

```
enum mpu_armv8m_attr_exec_t attr_exec
```

Definition at line 53 of file mpu_armv8m_drv.h.

#### 6.61.2.3 attr_sh

```
enum mpu_armv8m_attr_shared_t attr_sh
```

Definition at line 55 of file mpu_armv8m_drv.h.

#### 6.61.2.4 region_attridx

```
uint32_t region_attridx
```

Definition at line 52 of file mpu_armv8m_drv.h.

#### 6.61.2.5 region_base

```
uint32_t region_base
```

Definition at line 50 of file mpu_armv8m_drv.h.

#### 6.61.2.6 region_limit

```
uint32_t region_limit
```

Definition at line 51 of file mpu_armv8m_drv.h.

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

- platform/ext/target/infineon/common/drivers/protection/mpu_armv8m_drv.h

## 6.62 ns_mailbox_queue_t Struct Reference

```
#include <platform/ext/target/infineon/common/interface/include/multi_↩
core/tfm_ns_mailbox.h>
```

Collaboration diagram for ns_mailbox_queue_t:

## Public Member Functions

- struct mailbox_status_t status __ALIGNED (32)
- struct ns_mailbox_slot_t slots_ns[NUM_MAILBOX_QUEUE_SLOT] __ALIGNED (32)

## Data Fields

- struct mailbox_slot_t slots [NUM_MAILBOX_QUEUE_SLOT]
- mailbox_queue_status_t empty_slots
- bool is_full

### 6.62.1 Detailed Description

Definition at line 70 of file tfm_ns_mailbox.h.

### 6.62.2 Member Function Documentation

#### 6.62.2.1 __ALIGNED() [1/2]

```
struct mailbox_status_t status __ALIGNED (
            32  )
```

#### 6.62.2.2 __ALIGNED() [2/2]

```
struct ns_mailbox_slot_t slots_ns [NUM_MAILBOX_QUEUE_SLOT] __ALIGNED (
            32  )
```

### 6.62.3 Field Documentation

#### 6.62.3.1 empty_slots

```
mailbox_queue_status_t empty_slots
```

Definition at line 77 of file tfm_ns_mailbox.h.

#### 6.62.3.2 is_full

```
bool is_full
```

Definition at line 92 of file tfm_ns_mailbox.h.

#### 6.62.3.3 slots

```
struct mailbox_slot_t slots[NUM_MAILBOX_QUEUE_SLOT]
```

Definition at line 72 of file tfm_ns_mailbox.h.

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

- platform/ext/target/infineon/common/interface/include/multi_core/tfm_ns_mailbox.h

## 6.63 ns_mailbox_req_t Struct Reference

Collaboration diagram for ns_mailbox_req_t:

### Data Fields

- uint32_t call_type
- const struct psa_client_params_t * params_ptr
- int32_t client_id
- const void * owner
- struct mailbox_reply_t * reply
- uint8_t * woken_flag

### 6.63.1 Detailed Description

Definition at line 22 of file tfm_ns_mailbox_thread.c.

### 6.63.2 Field Documentation

#### 6.63.2.1 call_type

```
uint32_t call_type
```

Definition at line 23 of file tfm_ns_mailbox_thread.c.

#### 6.63.2.2 client_id

```
int32_t client_id
```

Definition at line 27 of file tfm_ns_mailbox_thread.c.

#### 6.63.2.3 owner

```
const void* owner
```

Definition at line 34 of file tfm_ns_mailbox_thread.c.

#### 6.63.2.4 params_ptr

```
const struct psa_client_params_t* params_ptr
```

Definition at line 24 of file tfm_ns_mailbox_thread.c.

#### 6.63.2.5 reply

```
struct mailbox_reply_t* reply
```

Definition at line 35 of file tfm_ns_mailbox_thread.c.

#### 6.63.2.6 woken_flag

```
uint8_t* woken_flag
```

Definition at line 39 of file tfm_ns_mailbox_thread.c.

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

- interface/src/multi_core/tfm_ns_mailbox_thread.c

## 6.64 ns_mailbox_slot_t Struct Reference

```
#include <platform/ext/target/infineon/common/interface/include/multi_↩
core/tfm_ns_mailbox.h>
```

Collaboration diagram for ns_mailbox_slot_t:

### Data Fields

- const void * owner
- struct mailbox_reply_t * reply
- bool is_woken

### 6.64.1 Detailed Description

Definition at line 51 of file tfm_ns_mailbox.h.

### 6.64.2 Field Documentation

#### 6.64.2.1 is_woken

```
bool is_woken
```

Definition at line 62 of file tfm_ns_mailbox.h.

#### 6.64.2.2 owner

```
const void* owner
```

Definition at line 52 of file tfm_ns_mailbox.h.

#### 6.64.2.3 reply

```
struct mailbox_reply_t* reply
```

Definition at line 53 of file tfm_ns_mailbox.h.

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

- platform/ext/target/infineon/common/interface/include/multi_core/tfm_ns_mailbox.h

## 6.65 ns_mailbox_stats_res_t Struct Reference

The structure to hold the statistics result of NSPE mailbox.

```
#include <interface/include/multi_core/tfm_ns_mailbox_test.h>
```

### Data Fields

- uint8_t avg_nr_slots
- uint8_t avg_nr_slots_tenths

### 6.65.1 Detailed Description

The structure to hold the statistics result of NSPE mailbox.

Definition at line 24 of file tfm_ns_mailbox_test.h.

### 6.65.2 Field Documentation

#### 6.65.2.1 avg_nr_slots

```
uint8_t avg_nr_slots
```

Definition at line 25 of file tfm_ns_mailbox_test.h.

#### 6.65.2.2 avg_nr_slots_tenths

```
uint8_t avg_nr_slots_tenths
```

Definition at line 29 of file tfm_ns_mailbox_test.h.

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

- interface/include/multi_core/tfm_ns_mailbox_test.h

## 6.66 partition_head_t Struct Reference

```
#include <secure_fw/spm/include/load/spm_load_api.h>
```

Collaboration diagram for partition_head_t:

### Data Fields

- uint32_t reserved
- struct partition_t * next

### 6.66.1 Detailed Description

Definition at line 50 of file spm_load_api.h.

### 6.66.2 Field Documentation

#### 6.66.2.1 next

```
struct partition_t* next
```

Definition at line 52 of file spm_load_api.h.

#### 6.66.2.2 reserved

```
uint32_t reserved
```

Definition at line 51 of file spm_load_api.h.

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

- secure_fw/spm/include/load/spm_load_api.h

## 6.67 partition_load_info_t Struct Reference

```
#include <secure_fw/spm/include/load/partition_defs.h>
```

### Data Fields

- uint32_t psa_ff_ver
- int32_t pid
- uint32_t flags
- uintptr_t entry
- size_t stack_size
- size_t heap_size
- uint32_t ndeps
- uint32_t nservices
- uint32_t nassets
- uint32_t nirqs
- int32_t client_id_base
- int32_t client_id_limit

### 6.67.1 Detailed Description

Definition at line 92 of file partition_defs.h.

## 6.67.2 Field Documentation

### 6.67.2.1 client_id_base

```
int32_t client_id_base
```

Definition at line 103 of file partition_defs.h.

### 6.67.2.2 client_id_limit

```
int32_t client_id_limit
```

Definition at line 104 of file partition_defs.h.

### 6.67.2.3 entry

```
uintptr_t entry
```

Definition at line 96 of file partition_defs.h.

### 6.67.2.4 flags

```
uint32_t flags
```

Definition at line 95 of file partition_defs.h.

### 6.67.2.5 heap_size

```
size_t heap_size
```

Definition at line 98 of file partition_defs.h.

#### 6.67.2.6 nassets

```
uint32_t nassets
```

Definition at line 101 of file partition_defs.h.

#### 6.67.2.7 ndeps

```
uint32_t ndeps
```

Definition at line 99 of file partition_defs.h.

#### 6.67.2.8 nirqs

```
uint32_t nirqs
```

Definition at line 102 of file partition_defs.h.

#### 6.67.2.9 nservices

```
uint32_t nservices
```

Definition at line 100 of file partition_defs.h.

#### 6.67.2.10 pid

```
int32_t pid
```

Definition at line 94 of file partition_defs.h.

#### 6.67.2.11 psa_ff_ver

```
uint32_t psa_ff_ver
```

Definition at line 93 of file partition_defs.h.

#### 6.67.2.12 stack_size

```
size_t stack_size
```

Definition at line 97 of file partition_defs.h.

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

- secure_fw/spm/include/load/partition_defs.h

## 6.68 partition_t Struct Reference

```
#include <secure_fw/spm/core/spm.h>
```

Collaboration diagram for partition_t:

### Data Fields

- const struct partition_load_info_t * p_ldinf
- uintptr_t boundary
- uint32_t signals_allowed
- uint32_t signals_waiting
- volatile uint32_t signals_asserted
- uint32_t state
- struct connection_t * p_handles
- struct partition_t * next

### 6.68.1 Detailed Description

Definition at line 107 of file spm.h.

## 6.68.2 Field Documentation

### 6.68.2.1 boundary

```
uintptr_t boundary
```

Definition at line 109 of file spm.h.

### 6.68.2.2 next

```
struct partition_t* next
```

Definition at line 122 of file spm.h.

### 6.68.2.3 p_handles

```
struct connection_t* p_handles
```

Definition at line 121 of file spm.h.

### 6.68.2.4 p_ldinf

```
const struct partition_load_info_t* p_ldinf
```

Definition at line 108 of file spm.h.

### 6.68.2.5 signals_allowed

```
uint32_t signals_allowed
```

Definition at line 110 of file spm.h.

#### 6.68.2.6 signals_asserted

```
volatile uint32_t signals_asserted
```

Definition at line 112 of file spm.h.

#### 6.68.2.7 signals_waiting

```
uint32_t signals_waiting
```

Definition at line 111 of file spm.h.

#### 6.68.2.8 state

```
uint32_t state
```

Definition at line 119 of file spm.h.

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

- secure_fw/spm/core/spm.h

## 6.69 partition_tfm_sp_idle_load_info_t Struct Reference

Collaboration diagram for partition_tfm_sp_idle_load_info_t:

### Data Fields

- struct partition_load_info_t load_info
- uintptr_t stack_addr
- uintptr_t heap_addr

### 6.69.1 Detailed Description

Definition at line 28 of file load_info_idle_sp.c.

### 6.69.2 Field Documentation

#### 6.69.2.1 heap_addr

```
uintptr_t heap_addr
```

Definition at line 33 of file load_info_idle_sp.c.

#### 6.69.2.2 load_info

```
struct partition_load_info_t load_info
```

Definition at line 30 of file load_info_idle_sp.c.

#### 6.69.2.3 stack_addr

```
uintptr_t stack_addr
```

Definition at line 32 of file load_info_idle_sp.c.

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

- secure_fw/partitions/idle_partition/load_info_idle_sp.c

## 6.70 partition_tfm_sp_ns_agent_tz_load_info_t Struct Reference

Collaboration diagram for partition_tfm_sp_ns_agent_tz_load_info_t:

### Data Fields

- struct partition_load_info_t load_info
- uintptr_t stack_addr
- uintptr_t heap_addr

### 6.70.1 Detailed Description

Definition at line 39 of file load_info_ns_agent_tz.c.

### 6.70.2 Field Documentation

#### 6.70.2.1 heap_addr

```
uintptr_t heap_addr
```

Definition at line 44 of file load_info_ns_agent_tz.c.

#### 6.70.2.2 load_info

```
struct partition_load_info_t load_info
```

Definition at line 41 of file load_info_ns_agent_tz.c.

#### 6.70.2.3 stack_addr

```
uintptr_t stack_addr
```

Definition at line 43 of file load_info_ns_agent_tz.c.

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

- secure_fw/partitions/ns_agent_tz/load_info_ns_agent_tz.c

## 6.71 ps_crypto_t Union Reference

```
#include <secure_fw/partitions/protected_storage/crypto/ps_crypto_interface.↩
h>
```

## Data Fields

- struct {
  uint8_t tag [PS_TAG_LEN_BYTES]
  psa_storage_uid_t uid
  int32_t client_id
  uint8_t iv [PS_IV_LEN_BYTES]
  } ref

### 6.71.1 Detailed Description

Definition at line 27 of file ps_crypto_interface.h.

### 6.71.2 Field Documentation

#### 6.71.2.1 client_id

```
int32_t client_id
```

Owner client ID for key label

Definition at line 31 of file ps_crypto_interface.h.

#### 6.71.2.2 iv

```
uint8_t iv[PS_IV_LEN_BYTES]
```

IV value of AEAD object

Definition at line 32 of file ps_crypto_interface.h.

#### 6.71.2.3 ref

```
struct { ... } ref
```

#### 6.71.2.4 tag

```
uint8_t tag[PS_TAG_LEN_BYTES]
```

MAC value of AEAD object

Definition at line 29 of file ps_crypto_interface.h.

#### 6.71.2.5 uid

```
psa_storage_uid_t uid
```

UID for key label

Definition at line 30 of file ps_crypto_interface.h.

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

- secure_fw/partitions/protected_storage/crypto/ps_crypto_interface.h

## 6.72 ps_obj_header_t Struct Reference

Metadata attached as a header to object data before storage.

```
#include <secure_fw/partitions/protected_storage/ps_object_defs.h>
```

Collaboration diagram for ps_obj_header_t:

### Data Fields

- uint32_t version
- uint32_t fid
- struct ps_object_info_t info

### 6.72.1 Detailed Description

Metadata attached as a header to object data before storage.

Definition at line 38 of file ps_object_defs.h.

### 6.72.2 Field Documentation

#### 6.72.2.1 fid

```
uint32_t fid
```

File ID

Definition at line 43 of file ps_object_defs.h.

#### 6.72.2.2 info

```
struct ps_object_info_t info
```

Object information

Definition at line 45 of file ps_object_defs.h.

#### 6.72.2.3 version

```
uint32_t version
```

Object version

Definition at line 42 of file ps_object_defs.h.

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

- secure_fw/partitions/protected_storage/ps_object_defs.h

## 6.73 ps_obj_table_ctx_t Struct Reference

Object table context structure.

Collaboration diagram for ps_obj_table_ctx_t:

### Data Fields

- struct ps_obj_table_t obj_table
- uint8_t active_table
- uint8_t scratch_table

### 6.73.1 Detailed Description

Object table context structure.

Object table init context structure.

Definition at line 141 of file ps_object_table.c.

### 6.73.2 Field Documentation

#### 6.73.2.1 active_table

```
uint8_t active_table
```

Active object table

Definition at line 143 of file ps_object_table.c.

#### 6.73.2.2 obj_table

```
struct ps_obj_table_t obj_table
```

Object tables

Definition at line 142 of file ps_object_table.c.

#### 6.73.2.3 scratch_table

```
uint8_t scratch_table
```

Scratch object table

Definition at line 144 of file ps_object_table.c.

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

- secure_fw/partitions/protected_storage/ps_object_table.c

## 6.74 ps_obj_table_entry_t Struct Reference

### Data Fields

- uint32_t version
- psa_storage_uid_t uid
- int32_t client_id

### 6.74.1 Detailed Description

Definition at line 41 of file ps_object_table.c.

### 6.74.2 Field Documentation

#### 6.74.2.1 client_id

```
int32_t client_id
```

Client ID

Definition at line 51 of file ps_object_table.c.

#### 6.74.2.2 uid

```
psa_storage_uid_t uid
```

Object UID

Definition at line 50 of file ps_object_table.c.

#### 6.74.2.3 version

```
uint32_t version
```

File version

Definition at line 48 of file ps_object_table.c.

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

- secure_fw/partitions/protected_storage/ps_object_table.c

## 6.75 ps_obj_table_info_t Struct Reference

Object table information structure.

```
#include <secure_fw/partitions/protected_storage/ps_object_table.h>
```

### Data Fields

- uint32_t fid
- uint32_t version

### 6.75.1 Detailed Description

Object table information structure.

Definition at line 26 of file ps_object_table.h.

### 6.75.2 Field Documentation

#### 6.75.2.1 fid

```
uint32_t fid
```

File ID in the file system

Definition at line 27 of file ps_object_table.h.

#### 6.75.2.2 version

```
uint32_t version
```

Object version

Definition at line 34 of file ps_object_table.h.

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

- secure_fw/partitions/protected_storage/ps_object_table.h

## 6.76 ps_obj_table_init_ctx_t Struct Reference

Collaboration diagram for ps_obj_table_init_ctx_t:

### Data Fields

- struct ps_obj_table_t * p_table [2]
- enum ps_obj_table_state table_state [2]

### 6.76.1 Detailed Description

Definition at line 215 of file ps_object_table.c.

### 6.76.2 Field Documentation

#### 6.76.2.1 p_table

```
struct ps_obj_table_t* p_table[2]
```

Pointers to object tables

Definition at line 216 of file ps_object_table.c.

#### 6.76.2.2 table_state

```
enum ps_obj_table_state table_state[2]
```

Array to indicate if the object table X is valid

Definition at line 219 of file ps_object_table.c.

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

- secure_fw/partitions/protected_storage/ps_object_table.c

## 6.77 ps_obj_table_t Struct Reference

Object table structure.

Collaboration diagram for ps_obj_table_t:

## Data Fields

- uint8_t version
- uint8_t swap_count
- struct ps_obj_table_entry_t obj_db [(PS_NUM_ASSETS+1)]

## 6.77.1 Detailed Description

Object table structure.

Definition at line 65 of file ps_object_table.c.

## 6.77.2 Field Documentation

### 6.77.2.1 obj_db

```
struct ps_obj_table_entry_t obj_db[(PS_NUM_ASSETS+1)]
```

Table's entries

Definition at line 78 of file ps_object_table.c.

### 6.77.2.2 swap_count

```
uint8_t swap_count
```

Swap counter to distinguish 2 different object tables.

Definition at line 73 of file ps_object_table.c.

### 6.77.2.3 version

```
uint8_t version
```

PS object system version.

Definition at line 70 of file ps_object_table.c.

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

- secure_fw/partitions/protected_storage/ps_object_table.c

## 6.78 ps_object_info_t Struct Reference

Object information.

```
#include <secure_fw/partitions/protected_storage/ps_object_defs.h>
```

### Data Fields

- uint32_t current_size
- uint32_t max_size
- psa_storage_create_flags_t create_flags

### 6.78.1 Detailed Description

Object information.

Definition at line 27 of file ps_object_defs.h.

### 6.78.2 Field Documentation

#### 6.78.2.1 create_flags

```
psa_storage_create_flags_t create_flags
```

Object creation flags

Definition at line 30 of file ps_object_defs.h.

#### 6.78.2.2 current_size

```
uint32_t current_size
```

Current size of the object content in bytes

Definition at line 28 of file ps_object_defs.h.

#### 6.78.2.3 max_size

```
uint32_t max_size
```

Maximum size of the object content in bytes

Definition at line 29 of file ps_object_defs.h.

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

- secure_fw/partitions/protected_storage/ps_object_defs.h

## 6.79 ps_object_t Struct Reference

The object to be written to the file system below. Made up of the object header and the object data.

```
#include <secure_fw/partitions/protected_storage/ps_object_defs.h>
```

Collaboration diagram for ps_object_t:

### Data Fields

- struct ps_obj_header_t header
- uint8_t data [PS_MAX_ASSET_SIZE]

### 6.79.1 Detailed Description

The object to be written to the file system below. Made up of the object header and the object data.

Definition at line 64 of file ps_object_defs.h.

### 6.79.2 Field Documentation

#### 6.79.2.1 data

```
uint8_t data[ PS_MAX_ASSET_SIZE]
```

Object data (and tag if encrypted)

Definition at line 66 of file ps_object_defs.h.

#### 6.79.2.2 header

```
struct ps_obj_header_t header
```

Object header

Definition at line 65 of file ps_object_defs.h.

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

- secure_fw/partitions/protected_storage/ps_object_defs.h

## 6.80 psa_client_params_t Struct Reference

```
#include <interface/include/multi_core/tfm_mailbox.h>
```

Collaboration diagram for psa_client_params_t:

## Data Fields

- union {
  - struct {
    - uint32_t sid
  - } psa_version_params
  - struct {
    - uint32_t sid
    - uint32_t version
  - } psa_connect_params
  - struct {
    - psa_handle_t handle
    - int32_t type
    - psa_invec in_vec [PSA_MAX_IOVEC]
    - size_t in_len
    - psa_outvec out_vec [PSA_MAX_IOVEC]
    - size_t out_len
  - } psa_call_params
  - struct {
    - psa_handle_t handle
  - } psa_close_params
- } psa_params

### 6.80.1 Detailed Description

Definition at line 71 of file tfm_mailbox.h.

### 6.80.2 Field Documentation

#### 6.80.2.1 handle

```
psa_handle_t handle
```

Definition at line 83 of file tfm_mailbox.h.

#### 6.80.2.2 in_len

```
size_t in_len
```

Definition at line 86 of file tfm_mailbox.h.

#### 6.80.2.3 in_vec

```
psa_invec in_vec[PSA_MAX_IOVEC]
```

Definition at line 85 of file tfm_mailbox.h.

#### 6.80.2.4 out_len

```
size_t out_len
```

Definition at line 88 of file tfm_mailbox.h.

#### 6.80.2.5 out_vec

```
psa_outvec out_vec[PSA_MAX_IOVEC]
```

Definition at line 87 of file tfm_mailbox.h.

#### 6.80.2.6 psa_call_params

```
struct { ... } psa_call_params
```

#### 6.80.2.7 psa_close_params

```
struct { ... } psa_close_params
```

#### 6.80.2.8 psa_connect_params

```
struct { ... } psa_connect_params
```

#### 6.80.2.9 psa_params

```
union { ... } psa_params
```

#### 6.80.2.10 psa_version_params

```
struct { ... } psa_version_params
```

#### 6.80.2.11 sid

```
uint32_t sid
```

Definition at line 74 of file tfm_mailbox.h.

#### 6.80.2.12 type

```
int32_t type
```

Definition at line 84 of file tfm_mailbox.h.

#### 6.80.2.13 version

```
uint32_t version
```

Definition at line 79 of file tfm_mailbox.h.

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

- interface/include/multi_core/tfm_mailbox.h

## 6.81 psa_fwu_component_info_t Struct Reference

Information about the firmware store for a firmware component.

```
#include <interface/include/psa/update.h>
```

Collaboration diagram for psa_fwu_component_info_t:

## Data Fields

- uint8_t state
- psa_status_t error
- psa_fwu_image_version_t version
- uint32_t max_size
- uint32_t flags
- uint32_t location
- psa_fwu_impl_info_t impl

### 6.81.1 Detailed Description

Information about the firmware store for a firmware component.

Definition at line 155 of file update.h.

### 6.81.2 Field Documentation

#### 6.81.2.1 error

```
psa_status_t error
```

Definition at line 159 of file update.h.

#### 6.81.2.2 flags

```
uint32_t flags
```

Definition at line 165 of file update.h.

#### 6.81.2.3 impl

```
psa_fwu_impl_info_t impl
```

Definition at line 169 of file update.h.

#### 6.81.2.4 location

```
uint32_t location
```

Definition at line 167 of file update.h.

#### 6.81.2.5 max_size

```
uint32_t max_size
```

Definition at line 163 of file update.h.

#### 6.81.2.6 state

```
uint8_t state
```

Definition at line 157 of file update.h.

#### 6.81.2.7 version

```
psa_fwu_image_version_t version
```

Definition at line 161 of file update.h.

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

- interface/include/psa/update.h

## 6.82 psa_fwu_image_version_t Struct Reference

Version information about a firmware image.

```
#include <interface/include/psa/update.h>
```

### Data Fields

- uint8_t major
- uint8_t minor
- uint16_t patch
- uint32_t build

### 6.82.1 Detailed Description

Version information about a firmware image.

Definition at line 76 of file update.h.

### 6.82.2 Field Documentation

#### 6.82.2.1 build

```
uint32_t build
```

Definition at line 84 of file update.h.

#### 6.82.2.2 major

```
uint8_t major
```

Definition at line 78 of file update.h.

#### 6.82.2.3 minor

```
uint8_t minor
```

Definition at line 80 of file update.h.

#### 6.82.2.4 patch

```
uint16_t patch
```

Definition at line 82 of file update.h.

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

- interface/include/psa/update.h

## 6.83 psa_fwu_impl_info_t Struct Reference

The implementation-specific data in the component information structure.

```
#include <interface/include/psa/update.h>
```

### Data Fields

- uint8_t candidate_digest [TFM_FWU_MAX_DIGEST_SIZE]

### 6.83.1 Detailed Description

The implementation-specific data in the component information structure.

Definition at line 147 of file update.h.

### 6.83.2 Field Documentation

#### 6.83.2.1 candidate_digest

```
uint8_t candidate_digest[TFM_FWU_MAX_DIGEST_SIZE]
```

Definition at line 149 of file update.h.

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

- interface/include/psa/update.h

## 6.84 psa_invec Struct Reference

```
#include <interface/include/psa/client.h>
```

### Data Fields

- const void * base
- size_t len

### 6.84.1 Detailed Description

A read-only input memory region provided to an RoT Service.

Definition at line 80 of file client.h.

### 6.84.2 Field Documentation

#### 6.84.2.1 base

```
const void* base
```

the start address of the memory buffer

Definition at line 81 of file client.h.

#### 6.84.2.2 len

```
size_t len
```

the size in bytes

Definition at line 82 of file client.h.

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

- interface/include/psa/client.h

## 6.85 psa_msg_t Struct Reference

```
#include <interface/include/psa/service.h>
```

### Data Fields

- int32_t type
- psa_handle_t handle
- int32_t client_id
- void * rhandle
- size_t in_size [PSA_MAX_IOVEC]
- size_t out_size [PSA_MAX_IOVEC]

### 6.85.1 Detailed Description

Describe a message received by an RoT Service after calling psa_get().

Definition at line 70 of file service.h.

## 6.85.2 Field Documentation

### 6.85.2.1 client_id

```
int32_t client_id
```

Definition at line 79 of file service.h.

### 6.85.2.2 handle

```
psa_handle_t handle
```

Definition at line 76 of file service.h.

### 6.85.2.3 in_size

```
size_t in_size[PSA_MAX_IOVEC]
```

Definition at line 90 of file service.h.

### 6.85.2.4 out_size

```
size_t out_size[PSA_MAX_IOVEC]
```

Definition at line 93 of file service.h.

### 6.85.2.5 rhandle

```
void* rhandle
```

Definition at line 85 of file service.h.

#### 6.85.2.6 type

```
int32_t type
```

Definition at line 71 of file service.h.

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

- interface/include/psa/service.h

## 6.86 psa_outvec Struct Reference

```
#include <interface/include/psa/client.h>
```

### Data Fields

- void * base
- size_t len

### 6.86.1 Detailed Description

A writable output memory region provided to an RoT Service.

Definition at line 88 of file client.h.

### 6.86.2 Field Documentation

#### 6.86.2.1 base

```
void* base
```

the start address of the memory buffer

Definition at line 89 of file client.h.

#### 6.86.2.2 len

```
size_t len
```

the size in bytes

Definition at line 90 of file client.h.

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

- interface/include/psa/client.h

## 6.87 psa_storage_info_t Struct Reference

```
#include <interface/include/psa/storage_common.h>
```

### Data Fields

- size_t capacity
- size_t size
- psa_storage_create_flags_t flags

### 6.87.1 Detailed Description

Definition at line 34 of file storage_common.h.

### 6.87.2 Field Documentation

#### 6.87.2.1 capacity

```
size_t capacity
```

Definition at line 35 of file storage_common.h.

#### 6.87.2.2 flags

```
psa_storage_create_flags_t flags
```

Definition at line 37 of file storage_common.h.

#### 6.87.2.3 size

```
size_t size
```

Definition at line 36 of file storage_common.h.

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

- interface/include/psa/storage_common.h

## 6.88 service_head_t Struct Reference

```
#include <secure_fw/spm/include/load/spm_load_api.h>
```

Collaboration diagram for service_head_t:

### Data Fields

- uint32_t reserved
- struct service_t * next

### 6.88.1 Detailed Description

Definition at line 56 of file spm_load_api.h.

### 6.88.2 Field Documentation

#### 6.88.2.1 next

```
struct service_t* next
```

Definition at line 58 of file spm_load_api.h.

#### 6.88.2.2 reserved

```
uint32_t reserved
```

Definition at line 57 of file spm_load_api.h.

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

- secure_fw/spm/include/load/spm_load_api.h

## 6.89 service_load_info_t Struct Reference

```
#include <secure_fw/spm/include/load/service_defs.h>
```

### Data Fields

- uintptr_t name_strid
- uint32_t sid
- uint32_t flags
- uint32_t version
- uintptr_t sfn

### 6.89.1 Detailed Description

Definition at line 47 of file service_defs.h.

### 6.89.2 Field Documentation

#### 6.89.2.1 flags

```
uint32_t flags
```

Definition at line 50 of file service_defs.h.

#### 6.89.2.2 name_strid

```
uintptr_t name_strid
```

Definition at line 48 of file service_defs.h.

#### 6.89.2.3 sfn

```
uintptr_t sfn
```

Definition at line 52 of file service_defs.h.

#### 6.89.2.4 sid

```
uint32_t sid
```

Definition at line 49 of file service_defs.h.

#### 6.89.2.5 version

```
uint32_t version
```

Definition at line 51 of file service_defs.h.

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

- secure_fw/spm/include/load/service_defs.h

## 6.90 service_t Struct Reference

```
#include <secure_fw/spm/core/spm.h>
```

Collaboration diagram for service_t:

### Data Fields

- const struct service_load_info_t ∗ p_ldinf
- struct partition_t ∗ partition
- struct service_t ∗ next

### 6.90.1 Detailed Description

Definition at line 126 of file spm.h.

### 6.90.2 Field Documentation

#### 6.90.2.1 next

```
struct service_t* next
```

Definition at line 129 of file spm.h.

#### 6.90.2.2 p_ldinf

```
const struct service_load_info_t* p_ldinf
```

Definition at line 127 of file spm.h.

#### 6.90.2.3 partition

```
struct partition_t* partition
```

Definition at line 128 of file spm.h.

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

- secure_fw/spm/core/spm.h

## 6.91 shared_data_tlv_entry Struct Reference

```
#include <secure_fw/spm/include/boot/tfm_boot_status.h>
```

### Data Fields

- uint16_t tlv_type
- uint16_t tlv_len

### 6.91.1 Detailed Description

Shared data TLV entry header format. All fields in little endian.

| **tlv_type(16)** | **tlv_len(16)** |

| **Raw data** |

Definition at line 179 of file tfm_boot_status.h.

### 6.91.2 Field Documentation

#### 6.91.2.1 tlv_len

`uint16_t tlv_len`
Definition at line 181 of file tfm_boot_status.h.

#### 6.91.2.2 tlv_type

`uint16_t tlv_type`
Definition at line 180 of file tfm_boot_status.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/include/boot/tfm_boot_status.h

## 6.92 shared_data_tlv_header Struct Reference

```
#include <secure_fw/spm/include/boot/tfm_boot_status.h>
```

### Data Fields

- uint16_t tlv_magic
- uint16_t tlv_tot_len

### 6.92.1 Detailed Description

Shared data TLV header. All fields in little endian.

| **tlv_magic(16)** | **tlv_tot_len(16)** |

Definition at line 163 of file tfm_boot_status.h.

### 6.92.2 Field Documentation

#### 6.92.2.1 tlv_magic

`uint16_t tlv_magic`
Definition at line 164 of file tfm_boot_status.h.

#### 6.92.2.2 tlv_tot_len

`uint16_t tlv_tot_len`
Definition at line 165 of file tfm_boot_status.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/include/boot/tfm_boot_status.h

## 6.93 tfm_additional_context_t Struct Reference

`#include <secure_fw/spm/include/tfm_arch.h>`

### Data Fields

- uint32_t integ_sign
- uint32_t reserved
- uint32_t callee [8]

### 6.93.1 Detailed Description

Definition at line 103 of file tfm_arch.h.

### 6.93.2 Field Documentation

#### 6.93.2.1 callee

`uint32_t callee[8]`
Definition at line 106 of file tfm_arch.h.

#### 6.93.2.2 integ_sign

`uint32_t integ_sign`
Definition at line 104 of file tfm_arch.h.

#### 6.93.2.3 reserved

`uint32_t reserved`
Definition at line 105 of file tfm_arch.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/include/tfm_arch.h

## 6.94 tfm_boot_data Struct Reference

Store the data for the runtime SW.
`#include <secure_fw/spm/include/boot/tfm_boot_status.h>`

Collaboration diagram for tfm_boot_data:

## Data Fields

- struct shared_data_tlv_header header
- uint8_t data [ ]

### 6.94.1 Detailed Description

Store the data for the runtime SW.
Definition at line 189 of file tfm_boot_status.h.

### 6.94.2 Field Documentation

#### 6.94.2.1 data

`uint8_t data[]`
Definition at line 191 of file tfm_boot_status.h.

#### 6.94.2.2 header

`struct shared_data_tlv_header header`
Definition at line 190 of file tfm_boot_status.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/include/boot/tfm_boot_status.h

## 6.95 tfm_builtin_key_t Struct Reference

A structure which describes a builtin key slot.

## Data Fields

- uint8_t key [(48)]
- size_t key_len
- psa_key_attributes_t attr
- uint32_t is_loaded

### 6.95.1 Detailed Description

A structure which describes a builtin key slot.
Definition at line 27 of file tfm_builtin_key_loader.c.

### 6.95.2 Field Documentation

#### 6.95.2.1 attr

`psa_key_attributes_t attr`
Key attributes associated to the key
Definition at line 30 of file tfm_builtin_key_loader.c.

#### 6.95.2.2 is_loaded

`uint32_t is_loaded`
Boolean indicating whether the slot is being used
Definition at line 31 of file tfm_builtin_key_loader.c.

#### 6.95.2.3 key

`uint8_t key[(48)]`
Raw key material, 4-byte aligned
Definition at line 28 of file tfm_builtin_key_loader.c.

#### 6.95.2.4 key_len

`size_t key_len`
Size of the key material held in the key buffer
Definition at line 29 of file tfm_builtin_key_loader.c.
The documentation for this struct was generated from the following file:

- secure_fw/partitions/crypto/psa_driver_api/tfm_builtin_key_loader.c

## 6.96 tfm_crypto_aead_pack_input Struct Reference

This type is used to overcome a limitation in the number of maximum IOVECs that can be used especially in psa↩_aead_encrypt and psa_aead_decrypt. By using this type we pack the nonce and the actual nonce_length at part of the same structure.
`#include <interface/include/tfm_crypto_defs.h>`

### Data Fields

- uint8_t nonce [(16u)]
- uint32_t nonce_length

### 6.96.1 Detailed Description

This type is used to overcome a limitation in the number of maximum IOVECs that can be used especially in psa↩_aead_encrypt and psa_aead_decrypt. By using this type we pack the nonce and the actual nonce_length at part of the same structure.
Definition at line 34 of file tfm_crypto_defs.h.

### 6.96.2 Field Documentation

#### 6.96.2.1 nonce

`uint8_t nonce[(16u)]`
Definition at line 35 of file tfm_crypto_defs.h.

#### 6.96.2.2 nonce_length

`uint32_t nonce_length`
Definition at line 36 of file tfm_crypto_defs.h.
The documentation for this struct was generated from the following file:

- interface/include/tfm_crypto_defs.h

## 6.97 tfm_crypto_key_id_s Struct Reference

The type which describes a key identifier to the Crypto service. The key identifiers must clearly provide a dedicated indication of the entity owner which owns the key.
`#include <secure_fw/partitions/crypto/tfm_crypto_key.h>`

### Data Fields

- psa_key_id_t key_id
- int32_t owner

### 6.97.1 Detailed Description

The type which describes a key identifier to the Crypto service. The key identifiers must clearly provide a dedicated indication of the entity owner which owns the key.
A macro to perform static initialisation of a.
structure
Definition at line 23 of file tfm_crypto_key.h.

### 6.97.2 Field Documentation

#### 6.97.2.1 key_id

`psa_key_id_t key_id`
Key ID for the key itself
Definition at line 24 of file tfm_crypto_key.h.

#### 6.97.2.2 owner

`int32_t owner`
ID of the entity owning the key
Definition at line 25 of file tfm_crypto_key.h.
The documentation for this struct was generated from the following file:

- secure_fw/partitions/crypto/tfm_crypto_key.h

## 6.98 tfm_crypto_operation_s Struct Reference

A type describing the context stored in Secure memory by the TF-M Crypto service to support multipart calls on secure side.

### Data Fields

- uint32_t in_use
- int32_t owner
- enum tfm_crypto_operation_type type
- union {
  psa_cipher_operation_t cipher
  psa_mac_operation_t mac
  psa_hash_operation_t hash
  psa_key_derivation_operation_t key_deriv
  psa_aead_operation_t aead
  } operation

### 6.98.1 Detailed Description

A type describing the context stored in Secure memory by the TF-M Crypto service to support multipart calls on secure side.
Definition at line 35 of file crypto_alloc.c.

### 6.98.2 Field Documentation

#### 6.98.2.1 aead

`psa_aead_operation_t aead`
AEAD operation context
Definition at line 46 of file crypto_alloc.c.

#### 6.98.2.2 cipher

`psa_cipher_operation_t cipher`
Cipher operation context
Definition at line 42 of file crypto_alloc.c.

#### 6.98.2.3 hash

`psa_hash_operation_t hash`
Hash operation context
Definition at line 44 of file crypto_alloc.c.

#### 6.98.2.4 in_use

`uint32_t in_use`
Indicates if the operation is in use
Definition at line 36 of file crypto_alloc.c.

#### 6.98.2.5 key_deriv

`psa_key_derivation_operation_t key_deriv`
Key derivation operation context
Definition at line 45 of file crypto_alloc.c.

#### 6.98.2.6 mac

`psa_mac_operation_t mac`
MAC operation context
Definition at line 43 of file crypto_alloc.c.

#### 6.98.2.7 operation

`union { ... } operation`

#### 6.98.2.8 owner

`int32_t owner`
Indicates an ID of the owner of the context
Definition at line 37 of file crypto_alloc.c.

#### 6.98.2.9 type

`enum tfm_crypto_operation_type type`
Type of the operation
Definition at line 40 of file crypto_alloc.c.
The documentation for this struct was generated from the following file:

- secure_fw/partitions/crypto/crypto_alloc.c

## 6.99 tfm_crypto_pack_iovec Struct Reference

Structure used to pack non-pointer types in a call to PSA Crypto APIs.
`#include <interface/include/tfm_crypto_defs.h>`
Collaboration diagram for tfm_crypto_pack_iovec:

## Data Fields

- psa_key_id_t key_id
- psa_algorithm_t alg
- uint32_t op_handle
- size_t ad_length
- size_t plaintext_length
- struct tfm_crypto_aead_pack_input aead_in
- uint16_t function_id
- uint16_t step
- union {
  size_t capacity
  uint64_t value
  };

### 6.99.1 Detailed Description

Structure used to pack non-pointer types in a call to PSA Crypto APIs.
Definition at line 43 of file tfm_crypto_defs.h.

### 6.99.2 Field Documentation

#### 6.99.2.1 "@6

```
union { ... }
```

#### 6.99.2.2 ad_length

```
size_t ad_length
```

Additional Data length for multipart AEAD
Definition at line 49 of file tfm_crypto_defs.h.

#### 6.99.2.3 aead_in

```
struct tfm_crypto_aead_pack_input aead_in
```

Packs AEAD-related inputs
Definition at line 52 of file tfm_crypto_defs.h.

#### 6.99.2.4 alg

```
psa_algorithm_t alg
```

Algorithm
Definition at line 45 of file tfm_crypto_defs.h.

#### 6.99.2.5 capacity

```
size_t capacity
```

Key derivation capacity
Definition at line 60 of file tfm_crypto_defs.h.

#### 6.99.2.6 function_id

`uint16_t function_id`
Used to identify the function in the API dispatcher to the service backend See tfm_crypto_func_sid for detail
Definition at line 54 of file tfm_crypto_defs.h.

#### 6.99.2.7 key_id

`psa_key_id_t key_id`
Key id
Definition at line 44 of file tfm_crypto_defs.h.

#### 6.99.2.8 op_handle

`uint32_t op_handle`
Client context handle associated to a multipart operation
Definition at line 46 of file tfm_crypto_defs.h.

#### 6.99.2.9 plaintext_length

`size_t plaintext_length`
Plaintext length for multipart AEAD
Definition at line 50 of file tfm_crypto_defs.h.

#### 6.99.2.10 step

`uint16_t step`
Key derivation step
Definition at line 58 of file tfm_crypto_defs.h.

#### 6.99.2.11 value

`uint64_t value`
Key derivation integer for update
Definition at line 61 of file tfm_crypto_defs.h.
The documentation for this struct was generated from the following file:

- interface/include/tfm_crypto_defs.h

## 6.100 tfm_fwu_ctx_s Struct Reference

### Data Fields

- psa_status_t error
- uint8_t component_state
- bool in_use

### 6.100.1 Detailed Description

Definition at line 23 of file tfm_fwu_req_mngr.c.

### 6.100.2 Field Documentation

#### 6.100.2.1 component_state

`uint8_t component_state`
Definition at line 25 of file tfm_fwu_req_mngr.c.

#### 6.100.2.2 error

`psa_status_t error`
Definition at line 24 of file tfm_fwu_req_mngr.c.

#### 6.100.2.3 in_use

`bool in_use`
Definition at line 26 of file tfm_fwu_req_mngr.c.
The documentation for this struct was generated from the following file:

- secure_fw/partitions/firmware_update/tfm_fwu_req_mngr.c

## 6.101 tfm_fwu_mcuboot_ctx_s Struct Reference

### Data Fields

- const struct flash_area * fap
- size_t loaded_size

### 6.101.1 Detailed Description

Definition at line 41 of file tfm_mcuboot_fwu.c.

### 6.101.2 Field Documentation

#### 6.101.2.1 fap

`const struct flash_area* fap`
Definition at line 43 of file tfm_mcuboot_fwu.c.

#### 6.101.2.2 loaded_size

`size_t loaded_size`
Definition at line 46 of file tfm_mcuboot_fwu.c.
The documentation for this struct was generated from the following file:

- secure_fw/partitions/firmware_update/bootloader/mcuboot/tfm_mcuboot_fwu.c

## 6.102 tfm_ns_ctx_mgr_t Struct Reference

```
#include <secure_fw/spm/ns_client_ext/tfm_ns_ctx_shm.h>
```

Collaboration diagram for tfm_ns_ctx_mgr_t:

## Data Fields

- uint8_t ns_ctx_limit
- uint8_t active_ns_ctx_index
- struct tfm_ns_ctx_t * ns_ctx_data

### 6.102.1 Detailed Description

Definition at line 19 of file tfm_ns_ctx_shm.h.

### 6.102.2 Field Documentation

#### 6.102.2.1 active_ns_ctx_index

`uint8_t active_ns_ctx_index`
Definition at line 24 of file tfm_ns_ctx_shm.h.

#### 6.102.2.2 ns_ctx_data

`struct tfm_ns_ctx_t* ns_ctx_data`
Definition at line 30 of file tfm_ns_ctx_shm.h.

#### 6.102.2.3 ns_ctx_limit

`uint8_t ns_ctx_limit`
Definition at line 21 of file tfm_ns_ctx_shm.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/ns_client_ext/tfm_ns_ctx_shm.h

## 6.103 tfm_ns_ctx_t Struct Reference

```
#include <secure_fw/spm/ns_client_ext/tfm_ns_ctx_shm.h>
```

### Data Fields

- int32_t nsid
- uint8_t gid
- uint8_t tid
- uint8_t ref_cnt

### 6.103.1 Detailed Description

Definition at line 12 of file tfm_ns_ctx_shm.h.

### 6.103.2 Field Documentation

#### 6.103.2.1 gid

`uint8_t gid`
Definition at line 14 of file tfm_ns_ctx_shm.h.

#### 6.103.2.2 nsid

`int32_t nsid`
Definition at line 13 of file tfm_ns_ctx_shm.h.

#### 6.103.2.3 ref_cnt

`uint8_t ref_cnt`
Definition at line 16 of file tfm_ns_ctx_shm.h.

#### 6.103.2.4 tid

`uint8_t tid`
Definition at line 15 of file tfm_ns_ctx_shm.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/ns_client_ext/tfm_ns_ctx_shm.h

## 6.104 tfm_original_iovec_t Struct Reference

`#include <secure_fw/spm/include/ffm/original_vector_api.h>`

Collaboration diagram for tfm_original_iovec_t:

## Data Fields

- psa_invec invec [PSA_MAX_IOVEC]
- psa_outvec outvec [PSA_MAX_IOVEC]
- size_t invec_count
- size_t outvec_count

### 6.104.1 Detailed Description

Structure to hold original input and output vectors and their counts
Definition at line 20 of file original_vector_api.h.

### 6.104.2 Field Documentation

#### 6.104.2.1 invec

`psa_invec invec[PSA_MAX_IOVEC]`
Definition at line 21 of file original_vector_api.h.

#### 6.104.2.2 invec_count

`size_t invec_count`
Definition at line 23 of file original_vector_api.h.

#### 6.104.2.3 outvec

`psa_outvec outvec[PSA_MAX_IOVEC]`
Definition at line 22 of file original_vector_api.h.

#### 6.104.2.4 outvec_count

`size_t outvec_count`
Definition at line 24 of file original_vector_api.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/include/ffm/original_vector_api.h

## 6.105 tfm_pool_chunk_t Struct Reference

`#include <secure_fw/spm/core/tfm_pools.h>`
Collaboration diagram for tfm_pool_chunk_t:

### Data Fields

- struct tfm_pool_chunk_t * next
- uint32_t magic
- uint8_t data [ ]

### 6.105.1 Detailed Description

Definition at line 19 of file tfm_pools.h.

### 6.105.2 Field Documentation

#### 6.105.2.1 data

`uint8_t data[]`
Definition at line 22 of file tfm_pools.h.

#### 6.105.2.2 magic

`uint32_t magic`
Definition at line 21 of file tfm_pools.h.

#### 6.105.2.3 next

`struct tfm_pool_chunk_t* next`
Definition at line 20 of file tfm_pools.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/core/tfm_pools.h

## 6.106 tfm_pool_instance_t Struct Reference

`#include <secure_fw/spm/core/tfm_pools.h>`

Collaboration diagram for tfm_pool_instance_t:

## Data Fields

- struct tfm_pool_chunk_t * next
- size_t chunksz
- size_t pool_sz
- uint8_t chunks [ ]

## 6.106.1 Detailed Description

Definition at line 25 of file tfm_pools.h.

## 6.106.2 Field Documentation

### 6.106.2.1 chunks

`uint8_t chunks[]`
Definition at line 29 of file tfm_pools.h.

### 6.106.2.2 chunksz

`size_t chunksz`
Definition at line 27 of file tfm_pools.h.

### 6.106.2.3 next

`struct tfm_pool_chunk_t* next`
Definition at line 26 of file tfm_pools.h.

### 6.106.2.4 pool_sz

`size_t pool_sz`
Definition at line 28 of file tfm_pools.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/core/tfm_pools.h

## 6.107 tfm_state_context_t Struct Reference

`#include <secure_fw/spm/include/tfm_arch.h>`

### Data Fields

- uint32_t r0
- uint32_t r1
- uint32_t r2
- uint32_t r3
- uint32_t r12
- uint32_t lr
- uint32_t ra
- uint32_t xpsr

### 6.107.1 Detailed Description

Definition at line 91 of file tfm_arch.h.

### 6.107.2 Field Documentation

#### 6.107.2.1 lr

`uint32_t lr`

Definition at line 97 of file tfm_arch.h.

#### 6.107.2.2 r0

`uint32_t r0`

Definition at line 92 of file tfm_arch.h.

#### 6.107.2.3 r1

`uint32_t r1`

Definition at line 93 of file tfm_arch.h.

#### 6.107.2.4 r12

`uint32_t r12`

Definition at line 96 of file tfm_arch.h.

#### 6.107.2.5 r2

`uint32_t r2`

Definition at line 94 of file tfm_arch.h.

#### 6.107.2.6 r3

`uint32_t r3`

Definition at line 95 of file tfm_arch.h.

#### 6.107.2.7 ra

`uint32_t ra`
Definition at line 98 of file tfm_arch.h.

#### 6.107.2.8 xpsr

`uint32_t xpsr`
Definition at line 99 of file tfm_arch.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/include/tfm_arch.h

## 6.108 thread_t Struct Reference

`#include <secure_fw/spm/core/thread.h>`
Collaboration diagram for thread_t:

### Data Fields

- uint8_t priority
- uint8_t state
- uint16_t flags
- struct context_ctrl_t * p_context_ctrl
- struct thread_t * next

### 6.108.1 Detailed Description

Definition at line 45 of file thread.h.

### 6.108.2 Field Documentation

#### 6.108.2.1 flags

`uint16_t flags`
Definition at line 48 of file thread.h.

#### 6.108.2.2 next

`struct thread_t* next`
Definition at line 50 of file thread.h.

#### 6.108.2.3 p_context_ctrl

`struct context_ctrl_t* p_context_ctrl`
Definition at line 49 of file thread.h.

#### 6.108.2.4 priority

`uint8_t priority`
Definition at line 46 of file thread.h.

#### 6.108.2.5 state

`uint8_t state`
Definition at line 47 of file thread.h.
The documentation for this struct was generated from the following file:

- secure_fw/spm/core/thread.h

## 6.109 u64_in_u32_regs_t Union Reference

`#include <secure_fw/spm/include/aapcs_local.h>`

### Data Fields

- struct {
  uint32_t r0
  uint32_t r1
  } u32_regs

- uint64_t u64_val

### 6.109.1 Detailed Description

Definition at line 37 of file aapcs_local.h.

### 6.109.2 Field Documentation

#### 6.109.2.1 r0

`uint32_t r0`
Definition at line 39 of file aapcs_local.h.

#### 6.109.2.2 r1

`uint32_t r1`
Definition at line 40 of file aapcs_local.h.

#### 6.109.2.3 u32_regs

```
struct { ... } u32_regs
```

#### 6.109.2.4 u64_val

```
uint64_t u64_val
```

Definition at line 43 of file aapcs_local.h.
The documentation for this union was generated from the following file:

- secure_fw/spm/include/aapcs_local.h

## 6.110 vectors Struct Reference

Collaboration diagram for vectors:

### Data Fields

- psa_invec in_vec [PSA_MAX_IOVEC]
- psa_outvec out_vec [PSA_MAX_IOVEC]
- size_t out_len
- bool in_use

### 6.110.1 Detailed Description

Definition at line 48 of file tfm_spe_mailbox.c.

### 6.110.2 Field Documentation

#### 6.110.2.1 in_use

```
bool in_use
```

Definition at line 52 of file tfm_spe_mailbox.c.

#### 6.110.2.2 in_vec

```
psa_invec in_vec[PSA_MAX_IOVEC]
```

Definition at line 49 of file tfm_spe_mailbox.c.

#### 6.110.2.3 out_len

`size_t out_len`
Definition at line 51 of file tfm_spe_mailbox.c.

#### 6.110.2.4 out_vec

`psa_outvec out_vec[PSA_MAX_IOVEC]`
Definition at line 50 of file tfm_spe_mailbox.c.
The documentation for this struct was generated from the following file:

- secure_fw/partitions/ns_agent_mailbox/tfm_spe_mailbox.c

# Chapter 7

# File Documentation

## 7.1 docs/doxygen/mainpage.dox File Reference

## 7.2 interface/dir_interface.dox File Reference

## 7.3 interface/include/crypto_keys/tfm_builtin_key_ids.h File Reference

### Enumerations

- enum tfm_builtin_key_id_t {
  TFM_BUILTIN_KEY_ID_MIN = 0x7FFF815Bu, TFM_BUILTIN_KEY_ID_HUK, TFM_BUILTIN_KEY_ID_IAK, TFM_BUILTIN_KEY_ID_PLAT_SPECIFIC_MIN = 0x7FFF816Bu, TFM_BUILTIN_KEY_ID_MAX = 0x7FFF817Bu, TFM_BUILTIN_KEY_ID_MIN = ((psa_key_id_t)0x7fff0000), TFM_BUILTIN_KEY_ID_ATTEST_PRIV, TFM_BUILTIN_KEY_ID_ATTEST_PUB, TFM_BUILTIN_KEY_ID_MAX }

  *The persistent key identifiers for TF-M builtin keys.*

### 7.3.1 Enumeration Type Documentation

#### 7.3.1.1 tfm_builtin_key_id_t

`enum tfm_builtin_key_id_t`

The persistent key identifiers for TF-M builtin keys.

**Note**

The value of TFM_BUILTIN_KEY_ID_MIN (and therefore of the whole range) is completely arbitrary except for being inside the PSA builtin keys range. The range is specified by the limits defined through MBEDTLS↩_PSA_KEY_ID_BUILTIN_MIN and MBEDTLS_PSA_KEY_ID_BUILTIN_MAX

**Enumerator**

| | |
|---:|---|
| TFM_BUILTIN_KEY_ID_MIN | |
| TFM_BUILTIN_KEY_ID_HUK | |
| TFM_BUILTIN_KEY_ID_IAK | |
| TFM_BUILTIN_KEY_ID_PLAT_SPECIFIC_MIN | |
| TFM_BUILTIN_KEY_ID_MAX | |
| TFM_BUILTIN_KEY_ID_MIN | |
| TFM_BUILTIN_KEY_ID_ATTEST_PRIV | |
| TFM_BUILTIN_KEY_ID_ATTEST_PUB | |
| TFM_BUILTIN_KEY_ID_MAX | |

Definition at line 19 of file tfm_builtin_key_ids.h.

## 7.4 platform/ext/target/infineon/pse84/epc2/spe/services/crypto/tfm_↩ builtin_key_ids.h File Reference

```
#include "psa/crypto.h"
#include "static_checks.h"
```

Include dependency graph for tfm_builtin_key_ids.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define MBEDTLS_PSA_KEY_ID_BUILTIN_MIN ((psa_key_id_t)0x7fff0000)

  *The persistent key identifiers for TF-M builtin keys.*
- #define TFM_BUILTIN_KEY_ID_HUK TFM_BUILTIN_KEY_ID_MIN
- #define TFM_BUILTIN_KEY_ID_IAK TFM_BUILTIN_KEY_ID_ATTEST_PRIV
- #define TFM_BUILTIN_KEY_ID_IAK_PUB TFM_BUILTIN_KEY_ID_ATTEST_PUB

### Enumerations

- enum tfm_builtin_key_id_t {
  TFM_BUILTIN_KEY_ID_MIN = 0x7FFF815Bu, TFM_BUILTIN_KEY_ID_HUK, TFM_BUILTIN_KEY_ID_IAK,
  TFM_BUILTIN_KEY_ID_PLAT_SPECIFIC_MIN = 0x7FFF816Bu,
  TFM_BUILTIN_KEY_ID_MAX = 0x7FFF817Bu, TFM_BUILTIN_KEY_ID_MIN = ((psa_key_id_t)0x7fff0000),
  TFM_BUILTIN_KEY_ID_ATTEST_PRIV, TFM_BUILTIN_KEY_ID_ATTEST_PUB,
  TFM_BUILTIN_KEY_ID_MAX }

### 7.4.1 Macro Definition Documentation

#### 7.4.1.1 MBEDTLS_PSA_KEY_ID_BUILTIN_MIN

```
#define MBEDTLS_PSA_KEY_ID_BUILTIN_MIN ((psa_key_id_t)0x7fff0000)
```

The persistent key identifiers for TF-M builtin keys.
The value of TFM_BUILTIN_KEY_ID_MIN (and therefore of the whole range) is completely arbitrary except for being inside the PSA builtin keys range.
Definition at line 30 of file tfm_builtin_key_ids.h.

#### 7.4.1.2 TFM_BUILTIN_KEY_ID_HUK

```
#define TFM_BUILTIN_KEY_ID_HUK TFM_BUILTIN_KEY_ID_MIN
```

Definition at line 40 of file tfm_builtin_key_ids.h.

#### 7.4.1.3 TFM_BUILTIN_KEY_ID_IAK

```
#define TFM_BUILTIN_KEY_ID_IAK TFM_BUILTIN_KEY_ID_ATTEST_PRIV
```

Definition at line 41 of file tfm_builtin_key_ids.h.

#### 7.4.1.4 TFM_BUILTIN_KEY_ID_IAK_PUB

```
#define TFM_BUILTIN_KEY_ID_IAK_PUB TFM_BUILTIN_KEY_ID_ATTEST_PUB
```

Definition at line 42 of file tfm_builtin_key_ids.h.

### 7.4.2 Enumeration Type Documentation

#### 7.4.2.1 tfm_builtin_key_id_t

```
enum tfm_builtin_key_id_t
```

**Enumerator**

| | |
|---|---|
| TFM_BUILTIN_KEY_ID_MIN | |
| TFM_BUILTIN_KEY_ID_HUK | |
| TFM_BUILTIN_KEY_ID_IAK | |
| TFM_BUILTIN_KEY_ID_PLAT_SPECIFIC_MIN | |
| TFM_BUILTIN_KEY_ID_MAX | |
| TFM_BUILTIN_KEY_ID_MIN | |
| TFM_BUILTIN_KEY_ID_ATTEST_PRIV | |
| TFM_BUILTIN_KEY_ID_ATTEST_PUB | |
| TFM_BUILTIN_KEY_ID_MAX | |

Definition at line 33 of file tfm_builtin_key_ids.h.

## 7.5 interface/include/multi_core/tfm_mailbox.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <stddef.h>
#include "cmsis_compiler.h"
#include "psa/client.h"
#include "tfm_mailbox_config.h"
```

Include dependency graph for tfm_mailbox.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct psa_client_params_t
- struct mailbox_msg_t
- struct mailbox_reply_t
- struct mailbox_slot_t
- struct mailbox_status_t
- struct mailbox_init_t

## Macros

- #define MAILBOX_CACHE_LINE_SIZE 32
- #define MAILBOX_ALIGN __ALIGNED(MAILBOX_CACHE_LINE_SIZE)
- #define MAILBOX_PSA_FRAMEWORK_VERSION (0x1)
- #define MAILBOX_PSA_VERSION (0x2)
- #define MAILBOX_PSA_CONNECT (0x3)
- #define MAILBOX_PSA_CALL (0x4)
- #define MAILBOX_PSA_CLOSE (0x5)
- #define MAILBOX_SUCCESS (0)
- #define MAILBOX_QUEUE_FULL (INT32_MIN + 1)
- #define MAILBOX_INVAL_PARAMS (INT32_MIN + 2)
- #define MAILBOX_NO_PERMS (INT32_MIN + 3)
- #define MAILBOX_NO_PEND_EVENT (INT32_MIN + 4)
- #define MAILBOX_CHAN_BUSY (INT32_MIN + 5)
- #define MAILBOX_CALLBACK_REG_ERROR (INT32_MIN + 6)
- #define MAILBOX_INIT_ERROR (INT32_MIN + 7)
- #define MAILBOX_GENERIC_ERROR (INT32_MIN + 8)

## Typedefs

- typedef uint32_t mailbox_queue_status_t

## Functions

- struct mailbox_slot_t __ALIGNED (32)

## Variables

- mailbox_queue_status_t pend_slots
- mailbox_queue_status_t replied_slots
- struct mailbox_init_t __ALIGNED

## 7.5.1 Macro Definition Documentation

### 7.5.1.1 MAILBOX_ALIGN

```
#define MAILBOX_ALIGN __ALIGNED(MAILBOX_CACHE_LINE_SIZE)
```

Definition at line 46 of file tfm_mailbox.h.

### 7.5.1.2 MAILBOX_CACHE_LINE_SIZE

```
#define MAILBOX_CACHE_LINE_SIZE 32
```

Definition at line 44 of file tfm_mailbox.h.

### 7.5.1.3 MAILBOX_CALLBACK_REG_ERROR

```
#define MAILBOX_CALLBACK_REG_ERROR (INT32_MIN + 6)
```

Definition at line 64 of file tfm_mailbox.h.

### 7.5.1.4 MAILBOX_CHAN_BUSY

```
#define MAILBOX_CHAN_BUSY (INT32_MIN + 5)
```

Definition at line 63 of file tfm_mailbox.h.

### 7.5.1.5 MAILBOX_GENERIC_ERROR

```
#define MAILBOX_GENERIC_ERROR (INT32_MIN + 8)
```

Definition at line 66 of file tfm_mailbox.h.

### 7.5.1.6 MAILBOX_INIT_ERROR

```
#define MAILBOX_INIT_ERROR (INT32_MIN + 7)
```

Definition at line 65 of file tfm_mailbox.h.

### 7.5.1.7 MAILBOX_INVAL_PARAMS

```
#define MAILBOX_INVAL_PARAMS (INT32_MIN + 2)
```

Definition at line 60 of file tfm_mailbox.h.

#### 7.5.1.8 MAILBOX_NO_PEND_EVENT

```
#define MAILBOX_NO_PEND_EVENT (INT32_MIN + 4)
```

Definition at line 62 of file tfm_mailbox.h.

#### 7.5.1.9 MAILBOX_NO_PERMS

```
#define MAILBOX_NO_PERMS (INT32_MIN + 3)
```

Definition at line 61 of file tfm_mailbox.h.

#### 7.5.1.10 MAILBOX_PSA_CALL

```
#define MAILBOX_PSA_CALL (0x4)
```

Definition at line 54 of file tfm_mailbox.h.

#### 7.5.1.11 MAILBOX_PSA_CLOSE

```
#define MAILBOX_PSA_CLOSE (0x5)
```

Definition at line 55 of file tfm_mailbox.h.

#### 7.5.1.12 MAILBOX_PSA_CONNECT

```
#define MAILBOX_PSA_CONNECT (0x3)
```

Definition at line 53 of file tfm_mailbox.h.

#### 7.5.1.13 MAILBOX_PSA_FRAMEWORK_VERSION

```
#define MAILBOX_PSA_FRAMEWORK_VERSION (0x1)
```

Definition at line 51 of file tfm_mailbox.h.

#### 7.5.1.14 MAILBOX_PSA_VERSION

```
#define MAILBOX_PSA_VERSION (0x2)
```

Definition at line 52 of file tfm_mailbox.h.

#### 7.5.1.15 MAILBOX_QUEUE_FULL

```
#define MAILBOX_QUEUE_FULL (INT32_MIN + 1)
```

Definition at line 59 of file tfm_mailbox.h.

#### 7.5.1.16 MAILBOX_SUCCESS

```
#define MAILBOX_SUCCESS (0)
```

Definition at line 58 of file tfm_mailbox.h.

### 7.5.2 Typedef Documentation

#### 7.5.2.1 mailbox_queue_status_t

```
typedef uint32_t mailbox_queue_status_t
```

Definition at line 134 of file tfm_mailbox.h.

### 7.5.3 Function Documentation

#### 7.5.3.1 __ALIGNED()

```
struct mailbox_slot_t __ALIGNED (
            32  )
```

### 7.5.4 Variable Documentation

#### 7.5.4.1 __ALIGNED

```
struct mailbox_status_t __ALIGNED
```

#### 7.5.4.2 pend_slots

```
mailbox_queue_status_t pend_slots
```

Definition at line 135 of file tfm_mailbox.h.

#### 7.5.4.3 replied_slots

```
mailbox_queue_status_t replied_slots
```

Definition at line 138 of file tfm_mailbox.h.

## 7.6 interface/include/multi_core/tfm_multi_core_api.h File Reference

```
#include <stdint.h>
```

Include dependency graph for tfm_multi_core_api.h:

This graph shows which files directly or indirectly include this file:

## Functions

- int32_t tfm_ns_wait_for_s_cpu_ready (void)

  *Called on the non-secure CPU. Flags that the non-secure side has completed its initialization. Waits, if necessary, for the secure CPU to flag that it has completed its initialization.*

- int32_t tfm_platform_ns_wait_for_s_cpu_ready (void)

  *Synchronisation with secure CPU, platform-specific implementation. Flags that the non-secure side has completed its initialization. Waits, if necessary, for the secure CPU to flag that it has completed its initialization.*

### 7.6.1 Function Documentation

#### 7.6.1.1 tfm_ns_wait_for_s_cpu_ready()

```
int32_t tfm_ns_wait_for_s_cpu_ready (
            void  )
```

Called on the non-secure CPU. Flags that the non-secure side has completed its initialization. Waits, if necessary, for the secure CPU to flag that it has completed its initialization.

**Returns**

Return 0 if succeeds.

Otherwise, return specific error code.

Definition at line 10 of file tfm_multi_core_ns_api.c.
Here is the call graph for this function:

#### 7.6.1.2 tfm_platform_ns_wait_for_s_cpu_ready()

```
int32_t tfm_platform_ns_wait_for_s_cpu_ready (
            void  )
```

Synchronisation with secure CPU, platform-specific implementation. Flags that the non-secure side has completed its initialization. Waits, if necessary, for the secure CPU to flag that it has completed its initialization.

**Return values**

| | |
|---|---|
| *Return* | 0 if succeeds. |
| *Otherwise,return* | specific error code. |

Definition at line 82 of file platform_ns_mailbox.c.
Here is the caller graph for this function:

## 7.7 interface/include/multi_core/tfm_ns_mailbox.h File Reference

## 7.8 platform/ext/target/infineon/common/interface/include/multi_core/tfm_ns_mailbox.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "cmsis.h"
#include "tfm_mailbox.h"
```

Include dependency graph for tfm_ns_mailbox.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct ns_mailbox_slot_t
- struct ns_mailbox_queue_t

## Macros

- #define MAILBOX_CLEAN_CACHE(addr, size) { __DSB(); }
- #define MAILBOX_INVALIDATE_CACHE(addr, size) do {} while (0)
- #define MAILBOX_DISABLE_CACHE() do {} while (0)
- #define MAILBOX_ENABLE_CACHE() do {} while (0)
- #define DCACHE_IS_ENABLED false
- #define tfm_ns_mailbox_thread_runner(args) do {} while (0)
- #define tfm_ns_mailbox_os_wait_reply() do {} while (0)
- #define tfm_ns_mailbox_os_wake_task_isr(task) do {} while (0)
- #define tfm_ns_mailbox_os_spin_lock() do {} while (0)
- #define tfm_ns_mailbox_os_spin_unlock() do {} while (0)
- #define tfm_ns_mailbox_client_call ifx_mtb_mailbox_client_call

## Functions

- int32_t tfm_ns_multi_core_boot (struct ns_mailbox_queue_t *queue)

  *Performs multicore boot sequence in NSPE.*
- int32_t tfm_ns_mailbox_init (struct ns_mailbox_queue_t *queue)

  *NSPE mailbox initialization.*
- int32_t tfm_ns_mailbox_client_call (uint32_t call_type, const struct psa_client_params_t *params, int32_t client_id, struct mailbox_reply_t *reply)

  *Send PSA client call to SPE via mailbox. Wait and fetch PSA client call result.*
- int32_t tfm_ns_mailbox_hal_init (struct ns_mailbox_queue_t *queue)

  *Platform specific NSPE mailbox initialization. Invoked by tfm_ns_mailbox_init().*
- int32_t tfm_ns_mailbox_hal_notify_peer (void)

  *Notify SPE to deal with the PSA client call sent via mailbox.*
- uint32_t tfm_ns_mailbox_hal_enter_critical (void)

  *Enter critical section of NSPE mailbox.*
- void tfm_ns_mailbox_hal_exit_critical (uint32_t state)

  *Exit critical section of NSPE mailbox.*
- uint32_t tfm_ns_mailbox_hal_enter_critical_isr (void)

  *Enter critical section of NSPE mailbox in IRQ handler.*
- void tfm_ns_mailbox_hal_exit_critical_isr (uint32_t state)

  *Enter critical section of NSPE mailbox in IRQ handler.*
- int32_t ifx_mtb_mailbox_client_call (uint32_t call_type, const struct psa_client_params_t *params, int32_t client_id, struct mailbox_reply_t *reply)

  *Send PSA client call to on core NSPE via MTB mailbox. Wait and fetch PSA client call result.*

## 7.8.1 Macro Definition Documentation

### 7.8.1.1 DCACHE_IS_ENABLED

```
#define DCACHE_IS_ENABLED false
```

Definition at line 37 of file tfm_ns_mailbox.h.

### 7.8.1.2 MAILBOX_CLEAN_CACHE

```
#define MAILBOX_CLEAN_CACHE(
            addr,
            size ) { __DSB(); }
```

### 7.8.1.3 MAILBOX_DISABLE_CACHE

```
#define MAILBOX_DISABLE_CACHE( ) do {} while (0)
```

Definition at line 35 of file tfm_ns_mailbox.h.

### 7.8.1.4 MAILBOX_ENABLE_CACHE

```
#define MAILBOX_ENABLE_CACHE( ) do {} while (0)
```

Definition at line 36 of file tfm_ns_mailbox.h.

### 7.8.1.5 MAILBOX_INVALIDATE_CACHE

```
#define MAILBOX_INVALIDATE_CACHE(
            addr,
            size ) do {} while (0)
```

### 7.8.1.6 tfm_ns_mailbox_client_call

```
#define tfm_ns_mailbox_client_call ifx_mtb_mailbox_client_call
```

Definition at line 22 of file tfm_ns_mailbox.h.

### 7.8.1.7 tfm_ns_mailbox_os_spin_lock

```
#define tfm_ns_mailbox_os_spin_lock( ) do {} while (0)
```

Definition at line 376 of file tfm_ns_mailbox.h.

### 7.8.1.8 tfm_ns_mailbox_os_spin_unlock

```
#define tfm_ns_mailbox_os_spin_unlock( ) do {} while (0)
```

Definition at line 378 of file tfm_ns_mailbox.h.

### 7.8.1.9 tfm_ns_mailbox_os_wait_reply

```
#define tfm_ns_mailbox_os_wait_reply( ) do {} while (0)
```

Definition at line 343 of file tfm_ns_mailbox.h.

#### 7.8.1.10 tfm_ns_mailbox_os_wake_task_isr

```
#define tfm_ns_mailbox_os_wake_task_isr(
              task ) do {} while (0)
```

Definition at line 345 of file tfm_ns_mailbox.h.

#### 7.8.1.11 tfm_ns_mailbox_thread_runner

```
#define tfm_ns_mailbox_thread_runner(
              args ) do {} while (0)
```

Definition at line 147 of file tfm_ns_mailbox.h.

### 7.8.2 Function Documentation

#### 7.8.2.1 ifx_mtb_mailbox_client_call()

```
int32_t ifx_mtb_mailbox_client_call (
            uint32_t call_type,
            const struct psa_client_params_t * params,
            int32_t client_id,
            struct mailbox_reply_t * reply )
```

Send PSA client call to on core NSPE via MTB mailbox. Wait and fetch PSA client call result.

**Parameters**

| | | |
|---|---|---|
| in | *call_type* | PSA client call type |
| in | *params* | Parameters used for PSA client call |
| in | *client_id* | Optional client ID of non-secure caller. It is required to identify the non-secure caller when NSPE OS enforces non-secure task isolation. |
| out | *reply* | The buffer written with PSA client call result. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | The PSA client call is completed successfully. |
| *Other* | return code Operation failed with an error code. |

Definition at line 23 of file ifx_mtb_mailbox.c.

#### 7.8.2.2 tfm_ns_mailbox_client_call()

```
int32_t tfm_ns_mailbox_client_call (
            uint32_t call_type,
            const struct psa_client_params_t * params,
            int32_t client_id,
            struct mailbox_reply_t * reply )
```

Send PSA client call to SPE via mailbox. Wait and fetch PSA client call result.

**Parameters**

| | | |
|---|---|---|
| in | *call_type* | PSA client call type |
| in | *params* | Parameters used for PSA client call |
| in | *client_id* | Optional client ID of non-secure caller. It is required to identify the non-secure caller when NSPE OS enforces non-secure task isolation. |

**Parameters**

| | | |
|---|---|---|
| out | *reply* | The buffer written with PSA client call result. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | The PSA client call is completed successfully. |
| *Other* | return code Operation failed with an error code. |

Definition at line 188 of file tfm_ns_mailbox.c.

#### 7.8.2.3 tfm_ns_mailbox_hal_enter_critical()

```
uint32_t tfm_ns_mailbox_hal_enter_critical (
            void  )
```

Enter critical section of NSPE mailbox.

**Note**

The implementation depends on platform specific hardware and use case.

**Returns**

Platform specific critical section state. Use it to exit from critical section by passing result to tfm_ns_mailbox_hal_exit_critical.

Definition at line 187 of file platform_ns_mailbox.c.
Here is the caller graph for this function:

#### 7.8.2.4 tfm_ns_mailbox_hal_enter_critical_isr()

```
uint32_t tfm_ns_mailbox_hal_enter_critical_isr (
            void  )
```

Enter critical section of NSPE mailbox in IRQ handler.

**Note**

The implementation depends on platform specific hardware and use case.

**Returns**

Platform specific critical section state. Use it to exit from critical section by passing result to tfm_ns_mailbox_hal_exit_critical_isr.

Definition at line 216 of file platform_ns_mailbox.c.
Here is the call graph for this function:

#### 7.8.2.5 tfm_ns_mailbox_hal_exit_critical()

```
void tfm_ns_mailbox_hal_exit_critical (
            uint32_t state )
```

Exit critical section of NSPE mailbox.

**Note**

The implementation depends on platform specific hardware and use case.

**Parameters**

| | | |
|---|---|---|
| in | *state* | Critical section state returned by tfm_ns_mailbox_hal_enter_critical. |

Definition at line 206 of file platform_ns_mailbox.c.
Here is the caller graph for this function:

#### 7.8.2.6 tfm_ns_mailbox_hal_exit_critical_isr()

```
void tfm_ns_mailbox_hal_exit_critical_isr (
            uint32_t state )
```

Enter critical section of NSPE mailbox in IRQ handler.

**Note**

The implementation depends on platform specific hardware and use case.

**Parameters**

| | | |
|---|---|---|
| in | *state* | Critical section state returned by tfm_ns_mailbox_hal_enter_critical_isr. |

Definition at line 225 of file platform_ns_mailbox.c.
Here is the call graph for this function:

#### 7.8.2.7 tfm_ns_mailbox_hal_init()

```
int32_t tfm_ns_mailbox_hal_init (
            struct ns_mailbox_queue_t * queue )
```

Platform specific NSPE mailbox initialization. Invoked by tfm_ns_mailbox_init().

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The base address of NSPE mailbox queue to be initialized. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 133 of file platform_ns_mailbox.c.

#### 7.8.2.8 tfm_ns_mailbox_hal_notify_peer()

```
int32_t tfm_ns_mailbox_hal_notify_peer (
            void )
```

Notify SPE to deal with the PSA client call sent via mailbox.

**Note**

The implementation depends on platform specific hardware and use case.

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 118 of file platform_ns_mailbox.c.

#### 7.8.2.9 tfm_ns_mailbox_init()

```
int32_t tfm_ns_mailbox_init (
            struct ns_mailbox_queue_t * queue )
```

NSPE mailbox initialization.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The base address of NSPE mailbox queue to be initialized. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 333 of file tfm_ns_mailbox.c.
Here is the caller graph for this function:

#### 7.8.2.10 tfm_ns_multi_core_boot()

```
int32_t tfm_ns_multi_core_boot (
            struct ns_mailbox_queue_t * queue )
```

Performs multicore boot sequence in NSPE.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The base address of NSPE mailbox queue to be initialized. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 26 of file platform_ns_mailbox.c.

Here is the call graph for this function:

## 7.9 interface/include/multi_core/tfm_ns_mailbox_test.h File Reference

```
#include <stdint.h>
#include "tfm_ns_mailbox.h"
```

Include dependency graph for tfm_ns_mailbox_test.h:

### Data Structures

- struct ns_mailbox_stats_res_t

  *The structure to hold the statistics result of NSPE mailbox.*

### Functions

- void tfm_ns_mailbox_tx_stats_init (struct ns_mailbox_queue_t *ns_queue)

  *Initialize the statistics module in TF-M NSPE mailbox.*

- int32_t tfm_ns_mailbox_tx_stats_reinit (void)

  *Re-initialize the statistics module in TF-M NSPE mailbox. Clean up statistics data.*

- void tfm_ns_mailbox_tx_stats_update (void)

  *Update the statistics result of NSPE mailbox message transmission.*

- void tfm_ns_mailbox_stats_avg_slot (struct ns_mailbox_stats_res_t *stats_res)

  *Calculate the average number of used NS mailbox queue slots each time NS task requires a queue slot to submit mailbox message, which is recorded in NS mailbox statisitics module.*

## 7.9.1 Function Documentation

### 7.9.1.1 tfm_ns_mailbox_stats_avg_slot()

```
void tfm_ns_mailbox_stats_avg_slot (
            struct ns_mailbox_stats_res_t * stats_res )
```

Calculate the average number of used NS mailbox queue slots each time NS task requires a queue slot to submit mailbox message, which is recorded in NS mailbox statisitics module.

**Note**

This function is only available when multi-core tests are enabled.

**Parameters**

| | | |
|---|---|---|
| in | *stats_res* | The buffer to be written with ns_mailbox_stats_res_t. |

**Returns**

Return the calculation result.

### 7.9.1.2 tfm_ns_mailbox_tx_stats_init()

```
void tfm_ns_mailbox_tx_stats_init (
            struct ns_mailbox_queue_t * ns_queue )
```

Initialize the statistics module in TF-M NSPE mailbox.

**Note**

This function is only available when multi-core tests are enabled.

**Parameters**

| | | |
|---|---|---|
| in | *ns_queue* | The NSPE mailbox queue to be tracked. |

### 7.9.1.3 tfm_ns_mailbox_tx_stats_reinit()

```
int32_t tfm_ns_mailbox_tx_stats_reinit (
            void )
```

Re-initialize the statistics module in TF-M NSPE mailbox. Clean up statistics data.

**Note**

This function is only available when multi-core tests are enabled.

**Returns**

MAILBOX_SUCCESS if the operation succeeded, or other return code in case of error

### 7.9.1.4 tfm_ns_mailbox_tx_stats_update()

```
void tfm_ns_mailbox_tx_stats_update (
            void )
```

Update the statistics result of NSPE mailbox message transmission.

**Note**

This function is only available when multi-core tests are enabled.

## 7.10 interface/include/os_wrapper/common.h File Reference

`#include <stdint.h>`
Include dependency graph for common.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define OS_WRAPPER_SUCCESS (0x0U)
- #define OS_WRAPPER_ERROR (0xFFFFFFFFU)
- #define OS_WRAPPER_WAIT_FOREVER (0xFFFFFFFFU)
- #define OS_WRAPPER_DEFAULT_STACK_SIZE (-1)

### 7.10.1 Macro Definition Documentation

#### 7.10.1.1 OS_WRAPPER_DEFAULT_STACK_SIZE

`#define OS_WRAPPER_DEFAULT_STACK_SIZE (-1)`
Definition at line 20 of file common.h.

#### 7.10.1.2 OS_WRAPPER_ERROR

`#define OS_WRAPPER_ERROR (0xFFFFFFFFU)`
Definition at line 18 of file common.h.

#### 7.10.1.3 OS_WRAPPER_SUCCESS

```
#define OS_WRAPPER_SUCCESS (0x0U)
```

Definition at line 17 of file common.h.

#### 7.10.1.4 OS_WRAPPER_WAIT_FOREVER

```
#define OS_WRAPPER_WAIT_FOREVER (0xFFFFFFFFU)
```

Definition at line 19 of file common.h.

## 7.11 interface/include/os_wrapper/kernel.h File Reference

```
#include <stdbool.h>
#include "os_wrapper/common.h"
```

Include dependency graph for kernel.h:

This graph shows which files directly or indirectly include this file:

## Functions

- bool os_wrapper_is_kernel_running (void)

  *Returns value that indicates whether RTOS kernel is running.*

### 7.11.1 Function Documentation

#### 7.11.1.1 os_wrapper_is_kernel_running()

```
bool os_wrapper_is_kernel_running (
            void  )
```

Returns value that indicates whether RTOS kernel is running.

**Returns**

true if RTOS kernel is running. false if RTOS kernel is not running.

Here is the caller graph for this function:

## 7.12 interface/include/os_wrapper/mutex.h File Reference

```
#include "os_wrapper/common.h"
```

Include dependency graph for mutex.h:

This graph shows which files directly or indirectly include this file:

## Functions

- void ∗ os_wrapper_mutex_create (void)

  *Creates a mutex for mutual exclusion of resources.*
- uint32_t os_wrapper_mutex_acquire (void ∗handle, uint32_t timeout)

  *Acquires a mutex that is created by os_wrapper_mutex_create()*
- uint32_t os_wrapper_mutex_release (void ∗handle)

  *Releases the mutex acquired previously.*
- uint32_t os_wrapper_mutex_delete (void ∗handle)

  *Deletes a mutex that is created by os_wrapper_mutex_create()*

### 7.12.1 Function Documentation

#### 7.12.1.1 os_wrapper_mutex_acquire()

```
uint32_t os_wrapper_mutex_acquire (
            void * handle,
            uint32_t timeout )
```

Acquires a mutex that is created by os_wrapper_mutex_create()

**Parameters**

| | | |
|---|---|---|
| in | *handle* | The handle of the mutex to acquire. Should be one of the handles returned by os_wrapper_mutex_create() |
| in | *timeout* | The maximum amount of time(in tick periods) for the thread to wait for the mutex to be available. If timeout is zero, the function will return immediately. Setting timeout to OS_WRAPPER_WAIT_FOREVER will cause the thread to wait indefinitely |

**Returns**

OS_WRAPPER_SUCCESS on success or OS_WRAPPER_ERROR on error or timeout

Here is the caller graph for this function:

#### 7.12.1.2 os_wrapper_mutex_create()

```
void* os_wrapper_mutex_create (
            void  )
```

Creates a mutex for mutual exclusion of resources.

**Returns**

The handle of the created mutex on success or NULL on error

#### 7.12.1.3 os_wrapper_mutex_delete()

```
uint32_t os_wrapper_mutex_delete (
            void * handle )
```

Deletes a mutex that is created by os_wrapper_mutex_create()

**Parameters**

| | | |
|---|---|---|
| in | *handle* | The handle of the mutex to be deleted |

**Returns**

OS_WRAPPER_SUCCESS on success or OS_WRAPPER_ERROR on error

#### 7.12.1.4 os_wrapper_mutex_release()

```
uint32_t os_wrapper_mutex_release (
            void * handle )
```

Releases the mutex acquired previously.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | The handle of the mutex that has been acquired |

**Returns**

OS_WRAPPER_SUCCESS on success or OS_WRAPPER_ERROR on error

## 7.13 interface/include/psa/client.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "psa/error.h"
```

Include dependency graph for client.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct psa_invec
- struct psa_outvec

### Macros

- #define IOVEC_LEN(arr) ((uint32_t)(sizeof(arr)/sizeof(arr[0])))
- #define PSA_FRAMEWORK_VERSION (0x0101u)

- #define PSA_VERSION_NONE (0u)
- #define PSA_NULL_HANDLE ((psa_handle_t)0)
- #define PSA_HANDLE_IS_VALID(handle) ((psa_handle_t)(handle) > 0)
- #define PSA_HANDLE_TO_ERROR(handle) ((psa_status_t)(handle))
- #define PSA_MAX_IOVEC (4u)
- #define PSA_CALL_TYPE_MIN (0)
- #define PSA_CALL_TYPE_MAX (INT16_MAX)
- #define PSA_IPC_CALL (0)

## Typedefs

- typedef int32_t psa_handle_t
- typedef struct psa_invec psa_invec
- typedef struct psa_outvec psa_outvec

## Functions

- uint32_t psa_framework_version (void)

  *Retrieve the version of the PSA Framework API that is implemented.*
- uint32_t psa_version (uint32_t sid)

  *Retrieve the version of an RoT Service or indicate that it is not present on this system.*
- psa_handle_t psa_connect (uint32_t sid, uint32_t version)

  *Connect to an RoT Service by its SID.*
- psa_status_t psa_call (psa_handle_t handle, int32_t type, const psa_invec *in_vec, size_t in_len, psa_outvec *out_vec, size_t out_len)

  *Call an RoT Service on an established connection.*
- void psa_close (psa_handle_t handle)

  *Close a connection to an RoT Service.*

## 7.13.1 Macro Definition Documentation

### 7.13.1.1 IOVEC_LEN

```
#define IOVEC_LEN(
            arr ) ((uint32_t)(sizeof(arr)/sizeof(arr[0])))
```

Definition at line 21 of file client.h.

### 7.13.1.2 PSA_CALL_TYPE_MAX

```
#define PSA_CALL_TYPE_MAX (INT16_MAX)
```

Definition at line 68 of file client.h.

### 7.13.1.3 PSA_CALL_TYPE_MIN

```
#define PSA_CALL_TYPE_MIN (0)
```

The minimum and maximum value in THIS implementation that can be passed as the type parameter in a call to psa_call().

Definition at line 67 of file client.h.

#### 7.13.1.4 PSA_FRAMEWORK_VERSION

```
#define PSA_FRAMEWORK_VERSION (0x0101u)
```

The version of the PSA Framework API that is being used to build the calling firmware. Only part of features of FF-M v1.1 have been implemented. FF-M v1.1 is compatible with v1.0.
Definition at line 31 of file client.h.

#### 7.13.1.5 PSA_HANDLE_IS_VALID

```
#define PSA_HANDLE_IS_VALID(
            handle ) ((psa_handle_t)(handle) > 0)
```

Tests whether a handle value returned by psa_connect() is valid.
Definition at line 48 of file client.h.

#### 7.13.1.6 PSA_HANDLE_TO_ERROR

```
#define PSA_HANDLE_TO_ERROR(
            handle ) ((psa_status_t)(handle))
```

Converts the handle value returned from a failed call psa_connect() into an error code.
Definition at line 54 of file client.h.

#### 7.13.1.7 PSA_IPC_CALL

```
#define PSA_IPC_CALL (0)
```

An IPC message type that indicates a generic client request.
Definition at line 73 of file client.h.

#### 7.13.1.8 PSA_MAX_IOVEC

```
#define PSA_MAX_IOVEC (4u)
```

Maximum number of input and output vectors for a request to psa_call().
Definition at line 59 of file client.h.

#### 7.13.1.9 PSA_NULL_HANDLE

```
#define PSA_NULL_HANDLE ((psa_handle_t)0)
```

The zero-value null handle can be assigned to variables used in clients and RoT Services, indicating that there is no current connection or message.
Definition at line 43 of file client.h.

#### 7.13.1.10 PSA_VERSION_NONE

```
#define PSA_VERSION_NONE (0u)
```

Return value from psa_version() if the requested RoT Service is not present in the system.
Definition at line 37 of file client.h.

### 7.13.2 Typedef Documentation

#### 7.13.2.1 psa_handle_t

```
typedef int32_t psa_handle_t
```

Definition at line 75 of file client.h.

#### 7.13.2.2 psa_invec

```
typedef struct psa_invec psa_invec
```

A read-only input memory region provided to an RoT Service.

#### 7.13.2.3 psa_outvec

```
typedef struct psa_outvec psa_outvec
```

A writable output memory region provided to an RoT Service.

### 7.13.3 Function Documentation

#### 7.13.3.1 psa_call()

```
psa_status_t psa_call (
            psa_handle_t handle,
            int32_t type,
            const psa_invec * in_vec,
            size_t in_len,
            psa_outvec * out_vec,
            size_t out_len )
```

Call an RoT Service on an established connection.

**Note**

FF-M 1.0 proposes 6 parameters for psa_call but the secure gateway ABI support at most 4 parameters. T↩ F-M chooses to encode 'in_len', 'out_len', and 'type' into a 32-bit integer to improve efficiency. Compared with struct-based encoding, this method saves extra memory check and memory copy operation. The disadvantage is that the 'type' range has to be reduced into a 16-bit integer. So with this encoding, the valid range for 'type' is 0-32767.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | A handle to an established connection. |
| in | *type* | The request type. Must be zero( PSA_IPC_CALL) or positive. |
| in | *in_vec* | Array of input psa_invec structures. |
| in | *in_len* | Number of input psa_invec structures. |
| in,out | *out_vec* | Array of output psa_outvec structures. |
| in | *out_len* | Number of output psa_outvec structures. |

**Return values**

| | |
|---|---|
| *>=0* | RoT Service-specific status value. |
| *<0* | RoT Service-specific error code. |

**Return values**

| | |
|---|---|
| *PSA_ERROR_PROGRAMMER_ERROR* | The connection has been terminated by the RoT Service. The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• An invalid handle was passed.<br>• The connection is already handling a request.<br>• type $<$ 0.<br>• An invalid memory reference was provided.<br>• in_len + out_len $>$ PSA_MAX_IOVEC.<br>• The message is unrecognized by the RoT Service or incorrectly formatted. |

Definition at line 100 of file tfm_multi_core_psa_ns_api.c.

Here is the caller graph for this function:

#### 7.13.3.2 psa_close()

```
void psa_close (
            psa_handle_t handle )
```

Close a connection to an RoT Service.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | A handle to an established connection, or the null handle. |

**Return values**

| | |
|---|---|
| *void* | Success. |
| *PROGRAMMER ERROR* | The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• An invalid handle was provided that is not the null handle.<br>• The connection is currently handling a request. |

Definition at line 143 of file tfm_multi_core_psa_ns_api.c.

#### 7.13.3.3 psa_connect()

```
psa_handle_t psa_connect (
            uint32_t sid,
            uint32_t version )
```

Connect to an RoT Service by its SID.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of the RoT Service to connect to. |
| in | *version* | Requested version of the RoT Service. |

**Return values**

| | |
|---|---|
| > | 0 A handle for the connection. |
| *PSA_ERROR_CONNECTION_REFUSED* | The SPM or RoT Service has refused the connection. |
| *PSA_ERROR_CONNECTION_BUSY* | The SPM or RoT Service cannot make the connection at the moment. |
| *PROGRAMMER ERROR* | The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• The RoT Service ID is not present.<br>• The RoT Service version is not supported.<br>• The caller is not allowed to access the RoT service. |

Definition at line 79 of file tfm_multi_core_psa_ns_api.c.

Here is the call graph for this function:

#### 7.13.3.4 psa_framework_version()

```
uint32_t psa_framework_version (
            void  )
```

Retrieve the version of the PSA Framework API that is implemented.

**Returns**

version The version of the PSA Framework implementation that is providing the runtime services to the caller. The major and minor version are encoded as follows:

- version[15:8] – major version number.
- version[7:0] – minor version number.

Definition at line 41 of file tfm_multi_core_psa_ns_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.13.3.5 psa_version()

```
uint32_t psa_version (
            uint32_t sid )
```

Retrieve the version of an RoT Service or indicate that it is not present on this system.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of the RoT Service to query. |

**Return values**

| | |
|---|---|
| *PSA_VERSION_NONE* | The RoT Service is not implemented, or the caller is not permitted to access the service. |
| > | 0 The version of the implemented RoT Service. |

Definition at line 59 of file tfm_multi_core_psa_ns_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.14 interface/include/psa/error.h File Reference

Standard error codes for the SPM and RoT Services.

```
#include <stdint.h>
```

Include dependency graph for error.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define PSA_SUCCESS ((psa_status_t)0)
- #define PSA_ERROR_PROGRAMMER_ERROR ((psa_status_t)-129)
- #define PSA_ERROR_CONNECTION_REFUSED ((psa_status_t)-130)
- #define PSA_ERROR_CONNECTION_BUSY ((psa_status_t)-131)
- #define PSA_ERROR_GENERIC_ERROR ((psa_status_t)-132)
- #define PSA_ERROR_NOT_PERMITTED ((psa_status_t)-133)
- #define PSA_ERROR_NOT_SUPPORTED ((psa_status_t)-134)
- #define PSA_ERROR_INVALID_ARGUMENT ((psa_status_t)-135)
- #define PSA_ERROR_INVALID_HANDLE ((psa_status_t)-136)
- #define PSA_ERROR_BAD_STATE ((psa_status_t)-137)
- #define PSA_ERROR_BUFFER_TOO_SMALL ((psa_status_t)-138)
- #define PSA_ERROR_ALREADY_EXISTS ((psa_status_t)-139)
- #define PSA_ERROR_DOES_NOT_EXIST ((psa_status_t)-140)
- #define PSA_ERROR_INSUFFICIENT_MEMORY ((psa_status_t)-141)
- #define PSA_ERROR_INSUFFICIENT_STORAGE ((psa_status_t)-142)
- #define PSA_ERROR_INSUFFICIENT_DATA ((psa_status_t)-143)
- #define PSA_ERROR_SERVICE_FAILURE ((psa_status_t)-144)
- #define PSA_ERROR_COMMUNICATION_FAILURE ((psa_status_t)-145)
- #define PSA_ERROR_STORAGE_FAILURE ((psa_status_t)-146)
- #define PSA_ERROR_HARDWARE_FAILURE ((psa_status_t)-147)
- #define PSA_ERROR_INVALID_SIGNATURE ((psa_status_t)-149)
- #define PSA_ERROR_CORRUPTION_DETECTED ((psa_status_t)-151)
- #define PSA_ERROR_DEPENDENCY_NEEDED ((psa_status_t)-156)
- #define PSA_ERROR_FLASH_ABUSE ((psa_status_t)-160)
- #define PSA_ERROR_INSUFFICIENT_POWER ((psa_status_t)-161)

## Typedefs

- typedef int32_t psa_status_t

### 7.14.1 Detailed Description

Standard error codes for the SPM and RoT Services.

### 7.14.2 Macro Definition Documentation

#### 7.14.2.1 PSA_ERROR_ALREADY_EXISTS

```
#define PSA_ERROR_ALREADY_EXISTS ((psa_status_t)-139)
```

Definition at line 42 of file error.h.

#### 7.14.2.2 PSA_ERROR_BAD_STATE

```
#define PSA_ERROR_BAD_STATE ((psa_status_t)-137)
```

Definition at line 40 of file error.h.

#### 7.14.2.3 PSA_ERROR_BUFFER_TOO_SMALL

```
#define PSA_ERROR_BUFFER_TOO_SMALL ((psa_status_t)-138)
```

Definition at line 41 of file error.h.

#### 7.14.2.4 PSA_ERROR_COMMUNICATION_FAILURE

```
#define PSA_ERROR_COMMUNICATION_FAILURE ((psa_status_t)-145)
```

Definition at line 48 of file error.h.

#### 7.14.2.5 PSA_ERROR_CONNECTION_BUSY

```
#define PSA_ERROR_CONNECTION_BUSY ((psa_status_t)-131)
```

Definition at line 34 of file error.h.

#### 7.14.2.6 PSA_ERROR_CONNECTION_REFUSED

```
#define PSA_ERROR_CONNECTION_REFUSED ((psa_status_t)-130)
```

Definition at line 33 of file error.h.

#### 7.14.2.7 PSA_ERROR_CORRUPTION_DETECTED

```
#define PSA_ERROR_CORRUPTION_DETECTED ((psa_status_t)-151)
```

Definition at line 52 of file error.h.

#### 7.14.2.8 PSA_ERROR_DEPENDENCY_NEEDED

```
#define PSA_ERROR_DEPENDENCY_NEEDED ((psa_status_t)-156)
```

Definition at line 53 of file error.h.

#### 7.14.2.9 PSA_ERROR_DOES_NOT_EXIST

```
#define PSA_ERROR_DOES_NOT_EXIST ((psa_status_t)-140)
```

Definition at line 43 of file error.h.

#### 7.14.2.10 PSA_ERROR_FLASH_ABUSE

```
#define PSA_ERROR_FLASH_ABUSE ((psa_status_t)-160)
```

Definition at line 54 of file error.h.

#### 7.14.2.11 PSA_ERROR_GENERIC_ERROR

```
#define PSA_ERROR_GENERIC_ERROR ((psa_status_t)-132)
```

Definition at line 35 of file error.h.

#### 7.14.2.12 PSA_ERROR_HARDWARE_FAILURE

```
#define PSA_ERROR_HARDWARE_FAILURE ((psa_status_t)-147)
```

Definition at line 50 of file error.h.

#### 7.14.2.13 PSA_ERROR_INSUFFICIENT_DATA

```
#define PSA_ERROR_INSUFFICIENT_DATA ((psa_status_t)-143)
```

Definition at line 46 of file error.h.

#### 7.14.2.14 PSA_ERROR_INSUFFICIENT_MEMORY

```
#define PSA_ERROR_INSUFFICIENT_MEMORY ((psa_status_t)-141)
```

Definition at line 44 of file error.h.

#### 7.14.2.15 PSA_ERROR_INSUFFICIENT_POWER

```
#define PSA_ERROR_INSUFFICIENT_POWER ((psa_status_t)-161)
```

Definition at line 55 of file error.h.

#### 7.14.2.16 PSA_ERROR_INSUFFICIENT_STORAGE

```
#define PSA_ERROR_INSUFFICIENT_STORAGE ((psa_status_t)-142)
```

Definition at line 45 of file error.h.

#### 7.14.2.17 PSA_ERROR_INVALID_ARGUMENT

```
#define PSA_ERROR_INVALID_ARGUMENT ((psa_status_t)-135)
```

Definition at line 38 of file error.h.

#### 7.14.2.18 PSA_ERROR_INVALID_HANDLE

```
#define PSA_ERROR_INVALID_HANDLE ((psa_status_t)-136)
```

Definition at line 39 of file error.h.

#### 7.14.2.19 PSA_ERROR_INVALID_SIGNATURE

```
#define PSA_ERROR_INVALID_SIGNATURE ((psa_status_t)-149)
```

Definition at line 51 of file error.h.

#### 7.14.2.20 PSA_ERROR_NOT_PERMITTED

```
#define PSA_ERROR_NOT_PERMITTED ((psa_status_t)-133)
```

Definition at line 36 of file error.h.

#### 7.14.2.21 PSA_ERROR_NOT_SUPPORTED

```
#define PSA_ERROR_NOT_SUPPORTED ((psa_status_t)-134)
```

Definition at line 37 of file error.h.

#### 7.14.2.22 PSA_ERROR_PROGRAMMER_ERROR

```
#define PSA_ERROR_PROGRAMMER_ERROR ((psa_status_t)-129)
```

Definition at line 32 of file error.h.

#### 7.14.2.23 PSA_ERROR_SERVICE_FAILURE

```
#define PSA_ERROR_SERVICE_FAILURE ((psa_status_t)-144)
```

Definition at line 47 of file error.h.

#### 7.14.2.24 PSA_ERROR_STORAGE_FAILURE

```
#define PSA_ERROR_STORAGE_FAILURE ((psa_status_t)-146)
```

Definition at line 49 of file error.h.

#### 7.14.2.25 PSA_SUCCESS

```
#define PSA_SUCCESS ((psa_status_t)0)
```

Definition at line 30 of file error.h.

### 7.14.3 Typedef Documentation

#### 7.14.3.1 psa_status_t

```
typedef int32_t psa_status_t
```

Definition at line 27 of file error.h.

## 7.15 interface/include/psa/internal_trusted_storage.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "psa/error.h"
#include "psa/storage_common.h"
```

Include dependency graph for internal_trusted_storage.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define PSA_ITS_API_VERSION_MAJOR 1
- #define PSA_ITS_API_VERSION_MINOR 0

## Functions

- psa_status_t psa_its_set (psa_storage_uid_t uid, size_t data_length, const void ∗p_data, psa_storage_create_flags_t create_flags)

  *Create a new, or modify an existing, uid/value pair.*
- psa_status_t psa_its_get (psa_storage_uid_t uid, size_t data_offset, size_t data_size, void ∗p_data, size_t ∗p_data_length)

  *Retrieve data associated with a provided UID.*
- psa_status_t psa_its_get_info (psa_storage_uid_t uid, struct psa_storage_info_t ∗p_info)

  *Retrieve the metadata about the provided uid.*
- psa_status_t psa_its_remove (psa_storage_uid_t uid)

  *Remove the provided uid and its associated data from the storage.*

### 7.15.1 Macro Definition Documentation

#### 7.15.1.1 PSA_ITS_API_VERSION_MAJOR

```
#define PSA_ITS_API_VERSION_MAJOR 1
```

This file describes the PSA Internal Trusted Storage API The major version number of the PSA ITS API
Definition at line 23 of file internal_trusted_storage.h.

#### 7.15.1.2 PSA_ITS_API_VERSION_MINOR

```
#define PSA_ITS_API_VERSION_MINOR 0
```

The minor version number of the PSA ITS API
Definition at line 26 of file internal_trusted_storage.h.

### 7.15.2 Function Documentation

#### 7.15.2.1 psa_its_get()

```
psa_status_t psa_its_get (
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_size,
            void * p_data,
            size_t * p_data_length )
```

Retrieve data associated with a provided UID.

Retrieves up to `data_size` bytes of the data associated with `uid`, starting at `data_offset` bytes from the beginning of the data. Upon successful completion, the data will be placed in the `p_data` buffer, which must be at least `data_size` bytes in size. The length of the data returned will be in `p_data_length`. If `data_size` is 0, the contents of `p_data_length` will be set to zero.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The uid value |
| in | *data_offset* | The starting offset of the data requested |
| in | *data_size* | The amount of data requested |
| out | *p_data* | On success, the buffer where the data will be placed |
| out | *p_data_length* | On success, this will contain size of the data placed in `p_data` |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided `uid` value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided arguments (`p_data`, `p_data_length`) is invalid, for example is `NULL` or references memory the caller cannot access. In addition, this can also happen if `data_offset` is larger than the size of the data associated with `uid` |

Definition at line 32 of file tfm_its_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.15.2.2 psa_its_get_info()

```
psa_status_t psa_its_get_info (
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Retrieve the metadata about the provided uid.
Retrieves the metadata stored for a given `uid` as a `psa_storage_info_t` structure.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |
| out | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |

**Return values**

| | |
|---|---|
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_info`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 61 of file tfm_its_api.c.
Here is the call graph for this function:

#### 7.15.2.3 psa_its_remove()

```
psa_status_t psa_its_remove (
            psa_storage_uid_t uid )
```

Remove the provided uid and its associated data from the storage.
Deletes the data from internal storage.

**Parameters**

| | | |
|---|---|---|
| `in` | *uid* | The `uid` value |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, wrong flags and so on) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |

Definition at line 81 of file tfm_its_api.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.15.2.4 psa_its_set()

```
psa_status_t psa_its_set (
            psa_storage_uid_t uid,
            size_t data_length,
            const void * p_data,
            psa_storage_create_flags_t create_flags )
```

Create a new, or modify an existing, uid/value pair.
Stores data in the internal storage.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The identifier for the data |
| in | *data_length* | The size in bytes of the data in `p_data` |
| in | *p_data* | A buffer containing the data |
| in | *create_flags* | The flags that the data will be stored with |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided `uid` value was already created with PSA_STORAGE_FLAG_WRITE_ONCE |

**Return values**

| | |
|---|---|
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because one or more of the flags provided in `create_flags` is not supported or is not valid |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because there was insufficient space on the storage medium |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_data`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 13 of file tfm_its_api.c.
Here is the call graph for this function:

## 7.16 interface/include/psa/lifecycle.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define PSA_LIFECYCLE_PSA_STATE_MASK (0xff00u)
- #define PSA_LIFECYCLE_IMP_STATE_MASK (0x00ffu)
- #define PSA_LIFECYCLE_UNKNOWN (0x0000u)
- #define PSA_LIFECYCLE_ASSEMBLY_AND_TEST (0x1000u)
- #define PSA_LIFECYCLE_PSA_ROT_PROVISIONING (0x2000u)
- #define PSA_LIFECYCLE_SECURED (0x3000u)
- #define PSA_LIFECYCLE_NON_PSA_ROT_DEBUG (0x4000u)
- #define PSA_LIFECYCLE_RECOVERABLE_PSA_ROT_DEBUG (0x5000u)
- #define PSA_LIFECYCLE_DECOMMISSIONED (0x6000u)

### Functions

- uint32_t psa_rot_lifecycle_state (void)

## 7.16.1 Macro Definition Documentation

### 7.16.1.1 PSA_LIFECYCLE_ASSEMBLY_AND_TEST

```
#define PSA_LIFECYCLE_ASSEMBLY_AND_TEST (0x1000u)
```

Definition at line 18 of file lifecycle.h.

### 7.16.1.2 PSA_LIFECYCLE_DECOMMISSIONED

```
#define PSA_LIFECYCLE_DECOMMISSIONED (0x6000u)
```

Definition at line 23 of file lifecycle.h.

### 7.16.1.3 PSA_LIFECYCLE_IMP_STATE_MASK

```
#define PSA_LIFECYCLE_IMP_STATE_MASK (0x00ffu)
```

Definition at line 16 of file lifecycle.h.

### 7.16.1.4 PSA_LIFECYCLE_NON_PSA_ROT_DEBUG

```
#define PSA_LIFECYCLE_NON_PSA_ROT_DEBUG (0x4000u)
```

Definition at line 21 of file lifecycle.h.

### 7.16.1.5 PSA_LIFECYCLE_PSA_ROT_PROVISIONING

```
#define PSA_LIFECYCLE_PSA_ROT_PROVISIONING (0x2000u)
```

Definition at line 19 of file lifecycle.h.

### 7.16.1.6 PSA_LIFECYCLE_PSA_STATE_MASK

```
#define PSA_LIFECYCLE_PSA_STATE_MASK (0xff00u)
```

Definition at line 15 of file lifecycle.h.

### 7.16.1.7 PSA_LIFECYCLE_RECOVERABLE_PSA_ROT_DEBUG

```
#define PSA_LIFECYCLE_RECOVERABLE_PSA_ROT_DEBUG (0x5000u)
```

Definition at line 22 of file lifecycle.h.

### 7.16.1.8 PSA_LIFECYCLE_SECURED

```
#define PSA_LIFECYCLE_SECURED (0x3000u)
```

Definition at line 20 of file lifecycle.h.

### 7.16.1.9 PSA_LIFECYCLE_UNKNOWN

```
#define PSA_LIFECYCLE_UNKNOWN (0x0000u)
```

Definition at line 17 of file lifecycle.h.

## 7.16.2 Function Documentation

#### 7.16.2.1 psa_rot_lifecycle_state()

```
uint32_t psa_rot_lifecycle_state (
            void  )
```

Definition at line 74 of file psa_api_ipc.c.

## 7.17 interface/include/psa/protected_storage.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "psa/error.h"
#include "psa/storage_common.h"
```

Include dependency graph for protected_storage.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define PSA_PS_API_VERSION_MAJOR 1

  *PSA_PS_API_VERSION version.*
- #define PSA_PS_API_VERSION_MINOR 0

### Functions

- psa_status_t psa_ps_set (psa_storage_uid_t uid, size_t data_length, const void ∗p_data, psa_storage_create_flags_t create_flags)

  *Create a new, or modify an existing, uid/value pair.*

- psa_status_t psa_ps_get (psa_storage_uid_t uid, size_t data_offset, size_t data_size, void *p_data, size_t *p_data_length)

  *Retrieve data associated with a provided uid.*
- psa_status_t psa_ps_get_info (psa_storage_uid_t uid, struct psa_storage_info_t *p_info)

  *Retrieve the metadata about the provided uid.*
- psa_status_t psa_ps_remove (psa_storage_uid_t uid)

  *Remove the provided uid and its associated data from the storage.*
- psa_status_t psa_ps_create (psa_storage_uid_t uid, size_t capacity, psa_storage_create_flags_t create_↩flags)

  *Reserves storage for the specified uid.*
- psa_status_t psa_ps_set_extended (psa_storage_uid_t uid, size_t data_offset, size_t data_length, const void *p_data)

  *Sets partial data into an asset.*
- uint32_t psa_ps_get_support (void)

  *Lists optional features.*

## 7.17.1 Macro Definition Documentation

### 7.17.1.1 PSA_PS_API_VERSION_MAJOR

```
#define PSA_PS_API_VERSION_MAJOR 1
```

PSA_PS_API_VERSION version.
Major and minor PSA_PS_API_VERSION numbers
Definition at line 28 of file protected_storage.h.

### 7.17.1.2 PSA_PS_API_VERSION_MINOR

```
#define PSA_PS_API_VERSION_MINOR 0
```

Definition at line 29 of file protected_storage.h.

## 7.17.2 Function Documentation

### 7.17.2.1 psa_ps_create()

```
psa_status_t psa_ps_create (
            psa_storage_uid_t uid,
            size_t capacity,
            psa_storage_create_flags_t create_flags )
```

Reserves storage for the specified uid.
Upon success, the capacity of the storage will be capacity, and the size will be 0. It is only necessary to call this function for assets that will be written with the psa_ps_set_extended function. If only the psa_ps_set function is needed, calls to this function are redundant.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |
| in | *capacity* | The capacity to be allocated in bytes |
| in | *create_flags* | Flags indicating properties of storage |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because the capacity is bigger than the current available space |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because the function is not implemented or one or more create_flags are not supported. |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because uid was 0 or create_flags specified flags that are not defined in the API. |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed due to an unspecified error |
| *PSA_ERROR_ALREADY_EXISTS* | Storage for the specified uid already exists |

Definition at line 94 of file tfm_ps_api.c.

#### 7.17.2.2 psa_ps_get()

```
psa_status_t psa_ps_get (
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_size,
            void * p_data,
            size_t * p_data_length )
```

Retrieve data associated with a provided uid.

Retrieves up to `data_size` bytes of the data associated with `uid`, starting at `data_offset` bytes from the beginning of the data. Upon successful completion, the data will be placed in the `p_data` buffer, which must be at least `data_size` bytes in size. The length of the data returned will be in `p_data_length`. If `data_size` is 0, the contents of `p_data_length` will be set to zero.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The uid value |
| in | *data_offset* | The starting offset of the data requested |
| in | *data_size* | The amount of data requested |
| out | *p_data* | On success, the buffer where the data will be placed |
| out | *p_data_length* | On success, this will contain size of the data placed in `p_data` |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided arguments (`p_data`, `p_data_length`) is invalid, for example is `NULL` or references memory the caller cannot access. In addition, this can also happen if `data_offset` is larger than the size of the data associated with `uid` |

**Return values**

| | |
|---|---|
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided `uid` value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the data associated with the UID was corrupt |
| *PSA_ERROR_INVALID_SIGNATURE* | The operation failed because the data associated with the UID failed authentication |

Definition at line 32 of file tfm_ps_api.c.
Here is the call graph for this function:

#### 7.17.2.3 psa_ps_get_info()

```
psa_status_t psa_ps_get_info (
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Retrieve the metadata about the provided uid.
Retrieves the metadata stored for a given `uid`

**Parameters**

| | | |
|---|---|---|
| `in` | *uid* | The `uid` value |
| `out` | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_info`) is invalid, for example is `NULL` or references memory the caller cannot access |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

**Return values**

| | |
|---|---|
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the data associated with the UID was corrupt |

Definition at line 61 of file tfm_ps_api.c.
Here is the call graph for this function:

#### 7.17.2.4 psa_ps_get_support()

```
uint32_t psa_ps_get_support (
            void  )
```

Lists optional features.

**Returns**

A bitmask with flags set for all of the optional features supported by the implementation.Currently defined flags are limited to PSA_STORAGE_SUPPORT_SET_EXTENDED

Definition at line 115 of file tfm_ps_api.c.
Here is the call graph for this function:

#### 7.17.2.5 psa_ps_remove()

```
psa_status_t psa_ps_remove (
            psa_storage_uid_t uid )
```

Remove the provided uid and its associated data from the storage.
Removes previously stored data and any associated metadata, including rollback protection data.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, wrong flags and so on) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 80 of file tfm_ps_api.c.
Here is the call graph for this function:

### 7.17.2.6 psa_ps_set()

```
psa_status_t psa_ps_set (
            psa_storage_uid_t uid,
            size_t data_length,
            const void * p_data,
            psa_storage_create_flags_t create_flags )
```

Create a new, or modify an existing, uid/value pair.
Stores data in the protected storage.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The identifier for the data |
| in | *data_length* | The size in bytes of the data in `p_data` |
| in | *p_data* | A buffer containing the data |
| in | *create_flags* | The flags that the data will be stored with |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |

**Return values**

| | |
|---|---|
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided `uid` value was already created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_data`) is invalid, for example is `NULL` or references memory the caller cannot access |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because one or more of the flags provided in `create_flags` is not supported or is not valid |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because there was insufficient space on the storage medium |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 13 of file tfm_ps_api.c.
Here is the call graph for this function:

#### 7.17.2.7 psa_ps_set_extended()

```
psa_status_t psa_ps_set_extended (
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_length,
            const void * p_data )
```

Sets partial data into an asset.
Before calling this function, the storage must have been reserved with a call to psa_ps_create. It can also be used to overwrite data in an asset that was created with a call to psa_ps_set. Calling this function with data_length = 0 is permitted, which will make no change to the stored data.This function can overwrite existing data and/or extend it up to the capacity for the uid specified in psa_ps_create, but cannot create gaps.
That is, it has preconditions:

- data_offset <= size
- data_offset + data_length <= capacity and postconditions:
- size = max(size, data_offset + data_length)
- capacity unchanged.

**Parameters**

| | | |
|---|---|---|
| `in` | *uid* | The `uid` value |
| `in` | *data_offset* | Offset within the asset to start the write |
| `in` | *data_length* | The size in bytes of the data in p_data to write |
| `in` | *p_data* | Pointer to a buffer which contains the data to write |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The asset exists, the input parameters are correct and the data is correctly written in the physical storage. |
| *PSA_ERROR_STORAGE_FAILURE* | The data was not written correctly in the physical storage |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the preconditions listed above regarding data_offset, size, or data_length was violated. |
| *PSA_ERROR_DOES_NOT_EXIST* | The specified uid was not found |
| *PSA_ERROR_NOT_SUPPORTED* | The implementation of the API does not support this function |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed due to an unspecified error |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the existing data has been corrupted. |
| *PSA_ERROR_INVALID_SIGNATURE* | The operation failed because the existing data failed authentication (MAC check failed). |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because it was attempted on an asset which was written with the flag PSA_STORAGE_FLAG_WRITE_ONCE |

Definition at line 104 of file tfm_ps_api.c.

## 7.18 interface/include/psa/service.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "config_impl.h"
#include "psa/client.h"
#include "psa/error.h"
#include "psa/framework_feature.h"
```

Include dependency graph for service.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct psa_msg_t

## Macros

- #define PSA_POLL (0x00000000u)
- #define PSA_BLOCK (0x80000000u)
- #define PSA_WAIT_ANY (0xFFFFFFFFu)
- #define PSA_DOORBELL (0x00000008u)
- #define PSA_IPC_CONNECT (-1)
- #define PSA_IPC_DISCONNECT (-2)
- #define PSA_FLIH_NO_SIGNAL ((psa_flih_result_t) 0)
- #define PSA_FLIH_SIGNAL ((psa_flih_result_t) 1)

## Typedefs

- typedef uint32_t psa_signal_t
- typedef uint32_t psa_irq_status_t
- typedef uint32_t psa_flih_result_t
- typedef struct psa_msg_t psa_msg_t

## Functions

- psa_signal_t psa_wait (psa_signal_t signal_mask, uint32_t timeout)

  *Return the Secure Partition interrupt signals that have been asserted from a subset of signals provided by the caller.*
- psa_status_t psa_get (psa_signal_t signal, psa_msg_t *msg)

  *Retrieve the message which corresponds to a given RoT Service signal and remove the message from the RoT Service queue.*
- void psa_set_rhandle (psa_handle_t msg_handle, void *rhandle)

  *Associate some RoT Service private data with a client connection.*
- size_t psa_read (psa_handle_t msg_handle, uint32_t invec_idx, void *buffer, size_t num_bytes)

  *Read a message parameter or part of a message parameter from a client input vector.*
- size_t psa_skip (psa_handle_t msg_handle, uint32_t invec_idx, size_t num_bytes)

  *Skip over part of a client input vector.*
- void psa_write (psa_handle_t msg_handle, uint32_t outvec_idx, const void *buffer, size_t num_bytes)

  *Write a message response to a client output vector.*
- void psa_reply (psa_handle_t msg_handle, psa_status_t status)

  *Complete handling of a specific message and unblock the client.*
- void psa_notify (int32_t partition_id)

  *Send a PSA_DOORBELL signal to a specific Secure Partition.*
- void psa_clear (void)

  *Clear the PSA_DOORBELL signal.*
- void psa_eoi (psa_signal_t irq_signal)

  *Inform the SPM that an interrupt has been handled (end of interrupt).*
- void psa_panic (void)

  *Terminate execution within the calling Secure Partition and will not return.*

- void psa_irq_enable (psa_signal_t irq_signal)

  *Enable an interrupt.*
- psa_irq_status_t psa_irq_disable (psa_signal_t irq_signal)

  *Disable an interrupt and return the status of the interrupt prior to being disabled by this call.*
- void psa_reset_signal (psa_signal_t irq_signal)

  *Reset the signal for an interrupt that is using FLIH handling.*

## 7.18.1 Macro Definition Documentation

### 7.18.1.1 PSA_BLOCK

```
#define PSA_BLOCK (0x80000000u)
```

A timeout value that requests a blocking wait operation.
Definition at line 36 of file service.h.

### 7.18.1.2 PSA_DOORBELL

```
#define PSA_DOORBELL (0x00000008u)
```

The signal number for the Secure Partition doorbell.
Definition at line 46 of file service.h.

### 7.18.1.3 PSA_FLIH_NO_SIGNAL

```
#define PSA_FLIH_NO_SIGNAL ((psa_flih_result_t) 0)
```

Definition at line 55 of file service.h.

### 7.18.1.4 PSA_FLIH_SIGNAL

```
#define PSA_FLIH_SIGNAL ((psa_flih_result_t) 1)
```

Definition at line 56 of file service.h.

### 7.18.1.5 PSA_IPC_CONNECT

```
#define PSA_IPC_CONNECT (-1)
```

Definition at line 50 of file service.h.

### 7.18.1.6 PSA_IPC_DISCONNECT

```
#define PSA_IPC_DISCONNECT (-2)
```

Definition at line 52 of file service.h.

### 7.18.1.7 PSA_POLL

```
#define PSA_POLL (0x00000000u)
```

A timeout value that requests a polling wait operation.
Definition at line 31 of file service.h.

#### 7.18.1.8 PSA_WAIT_ANY

```
#define PSA_WAIT_ANY (0xFFFFFFFFu)
```

A mask value that includes all Secure Partition signals.
Definition at line 41 of file service.h.

### 7.18.2 Typedef Documentation

#### 7.18.2.1 psa_flih_result_t

```
typedef uint32_t psa_flih_result_t
```

Definition at line 65 of file service.h.

#### 7.18.2.2 psa_irq_status_t

```
typedef uint32_t psa_irq_status_t
```

Definition at line 62 of file service.h.

#### 7.18.2.3 psa_msg_t

```
typedef struct psa_msg_t psa_msg_t
```

Describe a message received by an RoT Service after calling psa_get().

#### 7.18.2.4 psa_signal_t

```
typedef uint32_t psa_signal_t
```

Definition at line 59 of file service.h.

### 7.18.3 Function Documentation

#### 7.18.3.1 psa_clear()

```
void psa_clear (
            void  )
```

Clear the PSA_DOORBELL signal.

**Return values**

| | |
|---|---|
| *void* | Success. |
| *PROGRAMMER ERROR* | The Secure Partition's doorbell signal is not currently asserted. |

#### 7.18.3.2 psa_eoi()

```
void psa_eoi (
            psa_signal_t irq_signal )
```

Inform the SPM that an interrupt has been handled (end of interrupt).

**Parameters**

| | | |
|---|---|---|
| in | *irq_signal* | The interrupt signal that has been processed. |

**Return values**

| | |
|---|---|
| *void* | Success. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• irq_signal is not an interrupt signal.<br>• irq_signal indicates more than one signal.<br>• irq_signal is not currently asserted.<br>• The interrupt is not using SLIH. |

#### 7.18.3.3 psa_get()

```
psa_status_t psa_get (
            psa_signal_t signal,
            psa_msg_t * msg )
```

Retrieve the message which corresponds to a given RoT Service signal and remove the message from the RoT Service queue.

**Parameters**

| | | |
|---|---|---|
| in | *signal* | The signal value for an asserted RoT Service. |
| out | *msg* | Pointer to psa_msg_t object for receiving the message. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success, ∗msg will contain the delivered message. |
| *PSA_ERROR_DOES_NOT_EXIST* | Message could not be delivered. |
| *PROGRAMMER ERROR* | The call is invalid because one or more of the following are true:<br>• signal has more than a single bit set.<br>• signal does not correspond to an RoT Service.<br>• The RoT Service signal is not currently asserted.<br>• The msg pointer provided is not a valid memory reference. |

Definition at line 42 of file psa_api_ipc.c.
Here is the caller graph for this function:

#### 7.18.3.4 psa_irq_disable()

```
psa_irq_status_t psa_irq_disable (
            psa_signal_t irq_signal )
```

Disable an interrupt and return the status of the interrupt prior to being disabled by this call.

**Parameters**

| | | |
|---|---|---|
| in | *irq_signal* | The signal for the interrupt to be disabled. This must have a single bit set, which must be the signal value for an interrupt in the calling Secure Partition. |

**Return values**

| | |
|---|---|
| *0* | The interrupt was disabled prior to this call. 1 The interrupt was enabled prior to this call. |
| *PROGRAMMER ERROR* | If one or more of the following are true:<br>• *irq_signal* is not an interrupt signal.<br>• *irq_signal* indicates more than one signal. |

**Note**

The current implementation always return 1. Do not use the return.

#### 7.18.3.5 psa_irq_enable()

```
void psa_irq_enable (
            psa_signal_t irq_signal )
```

Enable an interrupt.

**Parameters**

| | | |
|---|---|---|
| in | *irq_signal* | The signal for the interrupt to be enabled. This must have a single bit set, which must be the signal value for an interrupt in the calling Secure Partition. |

**Return values**

| | |
|---|---|
| *void* | |
| *PROGRAMMER ERROR* | If one or more of the following are true:<br>• *irq_signal* is not an interrupt signal.<br>• *irq_signal* indicates more than one signal. |

#### 7.18.3.6 psa_notify()

```
void psa_notify (
            int32_t partition_id )
```

Send a PSA_DOORBELL signal to a specific Secure Partition.

**Parameters**

| | | |
|---|---|---|
| in | *partition↩ _id* | Secure Partition ID of the target partition. |

**Return values**

| | |
|---|---|
| *void* | Success. |
| *PROGRAMMER ERROR* | partition_id does not correspond to a Secure Partition. |

#### 7.18.3.7 psa_panic()

```
void psa_panic (
            void  )
```

Terminate execution within the calling Secure Partition and will not return.

**Return values**

| | |
|---|---|
| *Does not return* | |

Definition at line 69 of file psa_api_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.18.3.8 psa_read()

```
size_t psa_read (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            void * buffer,
            size_t num_bytes )
```

Read a message parameter or part of a message parameter from a client input vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of the input vector to read from. Must be less than PSA_MAX_IOVEC. |
| out | *buffer* | Buffer in the Secure Partition to copy the requested data to. |
| in | *num_bytes* | Maximum number of bytes to be read from the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes copied. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a PSA_IPC_CALL message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• the memory reference for buffer is invalid or not writable. |

Definition at line 47 of file psa_api_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.18.3.9 psa_reply()

```
void psa_reply (
            psa_handle_t msg_handle,
            psa_status_t status )
```

Complete handling of a specific message and unblock the client.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *status* | Message result value to be reported to the client. |

**Return values**

| | |
|---|---|
| *void* | Success. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• An invalid status code is specified for the type of message. |

Definition at line 64 of file psa_api_ipc.c.
Here is the caller graph for this function:

#### 7.18.3.10 psa_reset_signal()

```
void psa_reset_signal (
            psa_signal_t irq_signal )
```

Reset the signal for an interrupt that is using FLIH handling.

**Parameters**

| | | |
|---|---|---|
| in | *irq_signal* | The interrupt signal to be reset. This must have a single bit set, corresponding to a currently asserted signal for an interrupt that is defined to use FLIH handling. |

**Return values**

| | |
|---|---|
| *void* | |
| *Programmer Error* | if one or more of the following are true:<br>• *irq_signal* is not a signal for an interrupt that is specified with FLIH handling in the Secure Partition manifest.<br>• *irq_signal* indicates more than one signal.<br>• *irq_signal* is not currently asserted. |

#### 7.18.3.11 psa_set_rhandle()

```
void psa_set_rhandle (
            psa_handle_t msg_handle,
            void * rhandle )
```

Associate some RoT Service private data with a client connection.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *rhandle* | Reverse handle allocated by the RoT Service. |

**Return values**

| | |
|---|---|
| *void* | Success, rhandle will be provided with all subsequent messages delivered on this connection. |
| *PROGRAMMER ERROR* | msg_handle is invalid. |

#### 7.18.3.12 psa_skip()

```
size_t psa_skip (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            size_t num_bytes )
```

Skip over part of a client input vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of input vector to skip from. Must be less than PSA_MAX_IOVEC. |
| in | *num_bytes* | Maximum number of bytes to skip in the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes skipped. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC. |

Definition at line 53 of file psa_api_ipc.c.
Here is the call graph for this function:

#### 7.18.3.13 psa_wait()

```
psa_signal_t psa_wait (
            psa_signal_t signal_mask,
            uint32_t timeout )
```

Return the Secure Partition interrupt signals that have been asserted from a subset of signals provided by the caller.

**Parameters**

| | | |
|---|---|---|
| in | *signal_mask* | A set of signals to query. Signals that are not in this set will be ignored. |
| in | *timeout* | Specify either blocking PSA_BLOCK or polling PSA_POLL operation. |

**Return values**

| | |
|---|---|
| *>0* | At least one signal is asserted. |
| *0* | No signals are asserted. This is only seen when a polling timeout is used. |

Definition at line 37 of file psa_api_ipc.c.

Here is the caller graph for this function:

#### 7.18.3.14 psa_write()

```
void psa_write (
            psa_handle_t msg_handle,
            uint32_t outvec_idx,
            const void * buffer,
            size_t num_bytes )
```

Write a message response to a client output vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| out | *outvec_idx* | Index of output vector in message to write to. Must be less than PSA_MAX_IOVEC. |
| in | *buffer* | Buffer with the data to write. |
| in | *num_bytes* | Number of bytes to write to the client output vector. |

**Return values**

| | |
|---|---|
| *void* | Success |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• outvec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• The memory reference for buffer is invalid.<br>• The call attempts to write data past the end of the client output vector. |

Definition at line 58 of file psa_api_ipc.c.

Here is the call graph for this function:

Here is the caller graph for this function:

## 7.19 interface/include/psa/storage_common.h File Reference

```
#include <stddef.h>
#include <stdint.h>
```

Include dependency graph for storage_common.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct psa_storage_info_t

## Macros

- #define PSA_STORAGE_FLAG_NONE 0u
- #define PSA_STORAGE_FLAG_WRITE_ONCE (1u << 0)
- #define PSA_STORAGE_FLAG_NO_CONFIDENTIALITY (1u << 1)
- #define PSA_STORAGE_FLAG_NO_REPLAY_PROTECTION (1u << 2)
- #define PSA_STORAGE_SUPPORT_SET_EXTENDED (1u << 0)
- #define PSA_ERROR_INVALID_SIGNATURE ((psa_status_t)-149)
- #define PSA_ERROR_DATA_CORRUPT ((psa_status_t)-152)

## Typedefs

- typedef uint32_t psa_storage_create_flags_t
- typedef uint64_t psa_storage_uid_t

### 7.19.1 Macro Definition Documentation

#### 7.19.1.1 PSA_ERROR_DATA_CORRUPT

```
#define PSA_ERROR_DATA_CORRUPT ((psa_status_t)-152)
```

Definition at line 43 of file storage_common.h.

#### 7.19.1.2 PSA_ERROR_INVALID_SIGNATURE

```
#define PSA_ERROR_INVALID_SIGNATURE ((psa_status_t)-149)
```

Definition at line 42 of file storage_common.h.

#### 7.19.1.3 PSA_STORAGE_FLAG_NO_CONFIDENTIALITY

```
#define PSA_STORAGE_FLAG_NO_CONFIDENTIALITY (1u << 1)
```

Definition at line 29 of file storage_common.h.

#### 7.19.1.4 PSA_STORAGE_FLAG_NO_REPLAY_PROTECTION

```
#define PSA_STORAGE_FLAG_NO_REPLAY_PROTECTION (1u << 2)
```

Definition at line 30 of file storage_common.h.

#### 7.19.1.5 PSA_STORAGE_FLAG_NONE

```
#define PSA_STORAGE_FLAG_NONE 0u
```

Definition at line 27 of file storage_common.h.

#### 7.19.1.6 PSA_STORAGE_FLAG_WRITE_ONCE

```
#define PSA_STORAGE_FLAG_WRITE_ONCE (1u << 0)
```

Definition at line 28 of file storage_common.h.

#### 7.19.1.7 PSA_STORAGE_SUPPORT_SET_EXTENDED

```
#define PSA_STORAGE_SUPPORT_SET_EXTENDED (1u << 0)
```

Definition at line 40 of file storage_common.h.

### 7.19.2 Typedef Documentation

#### 7.19.2.1 psa_storage_create_flags_t

```
typedef uint32_t psa_storage_create_flags_t
```

Definition at line 21 of file storage_common.h.

#### 7.19.2.2 psa_storage_uid_t

```
typedef uint64_t psa_storage_uid_t
```

Definition at line 23 of file storage_common.h.

## 7.20 interface/include/psa/update.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "psa/error.h"
#include "psa/fwu_config.h"
#include "tfm_fwu_defs.h"
```

Include dependency graph for update.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct psa_fwu_image_version_t

  *Version information about a firmware image.*
- struct psa_fwu_impl_info_t

  *The implementation-specific data in the component information structure.*
- struct psa_fwu_component_info_t

  *Information about the firmware store for a firmware component.*

## Macros

- #define PSA_FWU_API_VERSION_MAJOR 1

  *The major version of this implementation of the Firmware Update API.*
- #define PSA_FWU_API_VERSION_MINOR 0

  *The minor version of this implementation of the Firmware Update API.*
- #define PSA_ERROR_DEPENDENCY_NEEDED ((psa_status_t)-156)

  *A status code that indicates that the firmware of another component requires updating.*
- #define PSA_ERROR_FLASH_ABUSE ((psa_status_t)-160)

  *A status code that indicates that the system is limiting i/o operations to avoid rapid flash exhaustion.*
- #define PSA_ERROR_INSUFFICIENT_POWER ((psa_status_t)-161)

  *A status code that indicates that the system does not have enough power to carry out the request.*
- #define PSA_SUCCESS_REBOOT ((psa_status_t)+1)

  *The action was completed successfully and requires a system reboot to complete installation.*
- #define PSA_SUCCESS_RESTART ((psa_status_t)+2)

  *The action was completed successfully and requires a restart of the component to complete installation.*
- #define PSA_FWU_READY 0u

  *The READY state: the component is ready to start another update.*
- #define PSA_FWU_WRITING 1u

  *The WRITING state: a new firmware image is being written to the firmware store.*
- #define PSA_FWU_CANDIDATE 2u

  *The CANDIDATE state: a new firmware image is ready for installation.*
- #define PSA_FWU_STAGED 3u

  *The STAGED state: a new firmware image is queued for installation.*
- #define PSA_FWU_FAILED 4u

*The FAILED state: a firmware update has been cancelled or has failed.*

- #define PSA_FWU_TRIAL 5u

  *The TRIAL state: a new firmware image requires testing prior to acceptance of the update.*
- #define PSA_FWU_REJECTED 6u

  *The REJECTED state: a new firmware image has been rejected after testing.*
- #define PSA_FWU_UPDATED 7u

  *The UPDATED state: a firmware update has been successful, and the new image is now active.*
- #define PSA_FWU_FLAG_VOLATILE_STAGING 0x00000001u

  *Flag to indicate whether the image data in the component staging area is discarded at system reset.*
- #define PSA_FWU_FLAG_ENCRYPTION 0x00000002u

  *Flag to indicate whether a firmware component expects encrypted images during an update.*
- #define PSA_FWU_MAX_WRITE_SIZE TFM_CONFIG_FWU_MAX_WRITE_SIZE

  *The maximum permitted size for block in psa_fwu_write(), in bytes.*

## Typedefs

- typedef uint8_t psa_fwu_component_t

  *Firmware component type identifier.*
- typedef struct psa_fwu_image_version_t psa_fwu_image_version_t

  *Version information about a firmware image.*
- typedef struct psa_fwu_component_info_t psa_fwu_component_info_t

  *Information about the firmware store for a firmware component.*

## Functions

- psa_status_t psa_fwu_query (psa_fwu_component_t component, psa_fwu_component_info_t *info)

  *Retrieve the firmware store information for a specific firmware component.*
- psa_status_t psa_fwu_start (psa_fwu_component_t component, const void *manifest, size_t manifest_size)

  *Begin a firmware update operation for a specific firmware component.*
- psa_status_t psa_fwu_write (psa_fwu_component_t component, size_t image_offset, const void *block, size_t block_size)

  *Write a firmware image, or part of a firmware image, to its staging area.*
- psa_status_t psa_fwu_finish (psa_fwu_component_t component)

  *Mark a firmware image in the staging area as ready for installation.*
- psa_status_t psa_fwu_cancel (psa_fwu_component_t component)

  *Abandon an update that is in WRITING or CANDIDATE state.*
- psa_status_t psa_fwu_clean (psa_fwu_component_t component)

  *Prepare the component for another update.*
- psa_status_t psa_fwu_install (void)

  *Start the installation of all firmware images that have been prepared for update.*
- psa_status_t psa_fwu_request_reboot (void)

  *Requests the platform to reboot.*
- psa_status_t psa_fwu_reject (psa_status_t error)

  *Abandon an installation that is in STAGED or TRIAL state.*
- psa_status_t psa_fwu_accept (void)

  *Accept a firmware update that is currently in TRIAL state.*

### 7.20.1 Macro Definition Documentation

#### 7.20.1.1 PSA_ERROR_DEPENDENCY_NEEDED

```
#define PSA_ERROR_DEPENDENCY_NEEDED ((psa_status_t)-156)
```

A status code that indicates that the firmware of another component requires updating.
Definition at line 42 of file update.h.

#### 7.20.1.2 PSA_ERROR_FLASH_ABUSE

```
#define PSA_ERROR_FLASH_ABUSE ((psa_status_t)-160)
```

A status code that indicates that the system is limiting i/o operations to avoid rapid flash exhaustion.
Definition at line 48 of file update.h.

#### 7.20.1.3 PSA_ERROR_INSUFFICIENT_POWER

```
#define PSA_ERROR_INSUFFICIENT_POWER ((psa_status_t)-161)
```

A status code that indicates that the system does not have enough power to carry out the request.
Definition at line 54 of file update.h.

#### 7.20.1.4 PSA_FWU_API_VERSION_MAJOR

```
#define PSA_FWU_API_VERSION_MAJOR 1
```

The major version of this implementation of the Firmware Update API.
Definition at line 31 of file update.h.

#### 7.20.1.5 PSA_FWU_API_VERSION_MINOR

```
#define PSA_FWU_API_VERSION_MINOR 0
```

The minor version of this implementation of the Firmware Update API.
Definition at line 36 of file update.h.

#### 7.20.1.6 PSA_FWU_CANDIDATE

```
#define PSA_FWU_CANDIDATE 2u
```

The CANDIDATE state: a new firmware image is ready for installation.
Definition at line 101 of file update.h.

#### 7.20.1.7 PSA_FWU_FAILED

```
#define PSA_FWU_FAILED 4u
```

The FAILED state: a firmware update has been cancelled or has failed.
Definition at line 111 of file update.h.

#### 7.20.1.8 PSA_FWU_FLAG_ENCRYPTION

```
#define PSA_FWU_FLAG_ENCRYPTION 0x00000002u
```

Flag to indicate whether a firmware component expects encrypted images during an update.
Definition at line 141 of file update.h.

#### 7.20.1.9 PSA_FWU_FLAG_VOLATILE_STAGING

```
#define PSA_FWU_FLAG_VOLATILE_STAGING 0x00000001u
```

Flag to indicate whether the image data in the component staging area is discarded at system reset.

Definition at line 135 of file update.h.

#### 7.20.1.10 PSA_FWU_MAX_WRITE_SIZE

```
#define PSA_FWU_MAX_WRITE_SIZE TFM_CONFIG_FWU_MAX_WRITE_SIZE
```

The maximum permitted size for block in psa_fwu_write(), in bytes.
Definition at line 201 of file update.h.

#### 7.20.1.11 PSA_FWU_READY

```
#define PSA_FWU_READY 0u
```

The READY state: the component is ready to start another update.
Definition at line 90 of file update.h.

#### 7.20.1.12 PSA_FWU_REJECTED

```
#define PSA_FWU_REJECTED 6u
```

The REJECTED state: a new firmware image has been rejected after testing.
Definition at line 123 of file update.h.

#### 7.20.1.13 PSA_FWU_STAGED

```
#define PSA_FWU_STAGED 3u
```

The STAGED state: a new firmware image is queued for installation.
Definition at line 106 of file update.h.

#### 7.20.1.14 PSA_FWU_TRIAL

```
#define PSA_FWU_TRIAL 5u
```

The TRIAL state: a new firmware image requires testing prior to acceptance of the update.
Definition at line 117 of file update.h.

#### 7.20.1.15 PSA_FWU_UPDATED

```
#define PSA_FWU_UPDATED 7u
```

The UPDATED state: a firmware update has been successful, and the new image is now active.
Definition at line 129 of file update.h.

#### 7.20.1.16 PSA_FWU_WRITING

```
#define PSA_FWU_WRITING 1u
```

The WRITING state: a new firmware image is being written to the firmware store.
Definition at line 96 of file update.h.

#### 7.20.1.17 PSA_SUCCESS_REBOOT

```
#define PSA_SUCCESS_REBOOT ((psa_status_t)+1)
```

The action was completed successfully and requires a system reboot to complete installation.
Definition at line 60 of file update.h.

#### 7.20.1.18 PSA_SUCCESS_RESTART

```
#define PSA_SUCCESS_RESTART ((psa_status_t)+2)
```

The action was completed successfully and requires a restart of the component to complete installation.
Definition at line 66 of file update.h.

### 7.20.2 Typedef Documentation

#### 7.20.2.1 psa_fwu_component_info_t

```
typedef struct psa_fwu_component_info_t psa_fwu_component_info_t
```

Information about the firmware store for a firmware component.

#### 7.20.2.2 psa_fwu_component_t

```
typedef uint8_t psa_fwu_component_t
```

Firmware component type identifier.
Definition at line 71 of file update.h.

#### 7.20.2.3 psa_fwu_image_version_t

```
typedef struct psa_fwu_image_version_t psa_fwu_image_version_t
```

Version information about a firmware image.

### 7.20.3 Function Documentation

#### 7.20.3.1 psa_fwu_accept()

```
psa_status_t psa_fwu_accept (
            void  )
```

Accept a firmware update that is currently in TRIAL state.

**Returns**

Result status.

Definition at line 96 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.20.3.2 psa_fwu_cancel()

```
psa_status_t psa_fwu_cancel (
            psa_fwu_component_t component )
```

Abandon an update that is in WRITING or CANDIDATE state.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component to be cancelled. |

**Returns**

Result status.

Definition at line 56 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.20.3.3 psa_fwu_clean()

```
psa_status_t psa_fwu_clean (
            psa_fwu_component_t component )
```

Prepare the component for another update.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component to tidy up. |

**Returns**

Result status.

Definition at line 66 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.20.3.4 psa_fwu_finish()

```
psa_status_t psa_fwu_finish (
            psa_fwu_component_t component )
```

Mark a firmware image in the staging area as ready for installation.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component to install. |

**Returns**

Result status.

Definition at line 40 of file tfm_fwu_api.c.
Here is the call graph for this function:

### 7.20.3.5 psa_fwu_install()

```
psa_status_t psa_fwu_install (
            void  )
```

Start the installation of all firmware images that have been prepared for update.

**Returns**

Result status.

Definition at line 50 of file tfm_fwu_api.c.
Here is the call graph for this function:

### 7.20.3.6 psa_fwu_query()

```
psa_status_t psa_fwu_query (
            psa_fwu_component_t component,
            psa_fwu_component_info_t * info )
```

Retrieve the firmware store information for a specific firmware component.

**Parameters**

| | |
|---|---|
| *component* | Firmware component for which information is requested. |
| *info* | Output parameter for component information. |

**Returns**

Result status.

Definition at line 76 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.20.3.7 psa_fwu_reject()

```
psa_status_t psa_fwu_reject (
            psa_status_t error )
```

Abandon an installation that is in STAGED or TRIAL state.

**Parameters**

| | |
|---|---|
| *error* | An application-specific error code chosen by the application. |

**Returns**

Result status.

Definition at line 102 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.20.3.8 psa_fwu_request_reboot()

```
psa_status_t psa_fwu_request_reboot (
            void )
```

Requests the platform to reboot.

**Returns**

Result status.

Definition at line 90 of file tfm_fwu_api.c.

Here is the call graph for this function:

#### 7.20.3.9 psa_fwu_start()

```
psa_status_t psa_fwu_start (
            psa_fwu_component_t component,
            const void * manifest,
            size_t manifest_size )
```

Begin a firmware update operation for a specific firmware component.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component to be updated. |
| *manifest* | A pointer to a buffer containing a detached manifest for the update. |
| *manifest_size* | The size of the detached manifest. |

**Returns**

Result status.

Definition at line 12 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.20.3.10 psa_fwu_write()

```
psa_status_t psa_fwu_write (
            psa_fwu_component_t component,
            size_t image_offset,
            const void * block,
            size_t block_size )
```

Write a firmware image, or part of a firmware image, to its staging area.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component being updated. |
| *image_offset* | The offset of the data block in the whole image. |
| *block* | A buffer containing a block of image data. |
| *block_size* | Size of block, in bytes. |

**Returns**

Result status.

Definition at line 25 of file tfm_fwu_api.c.
Here is the call graph for this function:

## 7.21 interface/include/tfm_attest_defs.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define TFM_ATTEST_GET_TOKEN 1001
- #define TFM_ATTEST_GET_TOKEN_SIZE 1002

### 7.21.1 Macro Definition Documentation

#### 7.21.1.1 TFM_ATTEST_GET_TOKEN

```
#define TFM_ATTEST_GET_TOKEN 1001
```

Definition at line 16 of file tfm_attest_defs.h.

#### 7.21.1.2 TFM_ATTEST_GET_TOKEN_SIZE

```
#define TFM_ATTEST_GET_TOKEN_SIZE 1002
```

Definition at line 17 of file tfm_attest_defs.h.

## 7.22 interface/include/tfm_attest_iat_defs.h File Reference

```
#include "config_tfm.h"
```

Include dependency graph for tfm_attest_iat_defs.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define IAT_SW_COMPONENT_MEASUREMENT_TYPE (1)
- #define IAT_SW_COMPONENT_MEASUREMENT_VALUE (2)
- #define IAT_SW_COMPONENT_VERSION (4)
- #define IAT_SW_COMPONENT_SIGNER_ID (5)
- #define IAT_SW_COMPONENT_MEASUREMENT_DESC (6)

## 7.22.1 Macro Definition Documentation

#### 7.22.1.1 IAT_SW_COMPONENT_MEASUREMENT_DESC

```
#define IAT_SW_COMPONENT_MEASUREMENT_DESC (6)
```

Definition at line 58 of file tfm_attest_iat_defs.h.

#### 7.22.1.2 IAT_SW_COMPONENT_MEASUREMENT_TYPE

```
#define IAT_SW_COMPONENT_MEASUREMENT_TYPE (1)
```

Definition at line 53 of file tfm_attest_iat_defs.h.

#### 7.22.1.3 IAT_SW_COMPONENT_MEASUREMENT_VALUE

```
#define IAT_SW_COMPONENT_MEASUREMENT_VALUE (2)
```

Definition at line 54 of file tfm_attest_iat_defs.h.

#### 7.22.1.4 IAT_SW_COMPONENT_SIGNER_ID

```
#define IAT_SW_COMPONENT_SIGNER_ID (5)
```

Definition at line 57 of file tfm_attest_iat_defs.h.

#### 7.22.1.5 IAT_SW_COMPONENT_VERSION

```
#define IAT_SW_COMPONENT_VERSION (4)
```

Definition at line 56 of file tfm_attest_iat_defs.h.

# 7.23 interface/include/tfm_crypto_defs.h File Reference

```
#include "psa/crypto.h"
#include "tfm_builtin_key_ids.h"
```

Include dependency graph for tfm_crypto_defs.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct tfm_crypto_aead_pack_input

  *This type is used to overcome a limitation in the number of maximum IOVECs that can be used especially in psa_aead_encrypt and psa_aead_decrypt. By using this type we pack the nonce and the actual nonce_length at part of the same structure.*

- struct tfm_crypto_pack_iovec

  *Structure used to pack non-pointer types in a call to PSA Crypto APIs.*

## Macros

- #define TFM_CRYPTO_MAX_NONCE_LENGTH (16u)

  *The maximum supported length of a nonce through the TF-M interfaces.*

- #define RANDOM_FUNCS X(TFM_CRYPTO_GENERATE_RANDOM)
- #define KEY_MANAGEMENT_FUNCS
- #define HASH_FUNCS
- #define MAC_FUNCS
- #define CIPHER_FUNCS
- #define AEAD_FUNCS
- #define ASYM_SIGN_FUNCS
- #define ASYM_ENCRYPT_FUNCS
- #define KEY_DERIVATION_FUNCS
- #define BASE__VALUE(x) ((uint16_t)((((uint16_t)(x)) << 8) & 0xFF00))
- #define X(FUNCTION_NAME) FUNCTION_NAME ## _SID,
- #define TFM_CRYPTO_GET_GROUP_ID(_function_id) ((enum tfm_crypto_group_id_t)(((uint16_t)(_function_id) >> 8) & 0xFF))

  *This macro is used to extract the group_id from an encoded function id by accessing the upper 8 bits. A _function_id is uint16_t type.*

## Enumerations

- enum tfm_crypto_group_id_t {
  TFM_CRYPTO_GROUP_ID_RANDOM = UINT8_C(1), TFM_CRYPTO_GROUP_ID_KEY_MANAGEMENT = UINT8_C(2), TFM_CRYPTO_GROUP_ID_HASH = UINT8_C(3), TFM_CRYPTO_GROUP_ID_MAC = U↩INT8_C(4),
  TFM_CRYPTO_GROUP_ID_CIPHER = UINT8_C(5), TFM_CRYPTO_GROUP_ID_AEAD = UINT8_C(6), TFM_CRYPTO_GROUP_ID_ASYM_SIGN = UINT8_C(7), TFM_CRYPTO_GROUP_ID_ASYM_ENCRYPT = UINT8_C(8),
  TFM_CRYPTO_GROUP_ID_KEY_DERIVATION = UINT8_C(9) }

  *Type associated to the group of a function encoding. There can be nine groups (Random, Key management, Hash, MAC, Cipher, AEAD, Asym sign, Asym encrypt, Key derivation).*

- enum tfm_crypto_func_sid_t {
  BASE__RANDOM = ((uint16_t)((((uint16_t)( TFM_CRYPTO_GROUP_ID_RANDOM )) << 8) & 0xFF00)) - 1, TFM_CRYPTO_GENERATE_RANDOM_SID, BASE__KEY_MANAGEMENT = ((uint16_t)((((uint16_t)( TF↩M_CRYPTO_GROUP_ID_KEY_MANAGEMENT )) << 8) & 0xFF00)) - 1, TFM_CRYPTO_GET_KEY_ATTRIBUTES_SID, TFM_CRYPTO_OPEN_KEY_SID, TFM_CRYPTO_CLOSE_KEY_SID, TFM_CRYPTO_IMPORT_KEY_SID, TFM_CRYPTO_DESTROY_KEY_SID,
  TFM_CRYPTO_EXPORT_KEY_SID, TFM_CRYPTO_EXPORT_PUBLIC_KEY_SID, TFM_CRYPTO_PURGE_KEY_SID, TFM_CRYPTO_COPY_KEY_SID,
  TFM_CRYPTO_GENERATE_KEY_SID, BASE__HASH = ((uint16_t)((((uint16_t)( TFM_CRYPTO_GROU↩P_ID_HASH )) << 8) & 0xFF00)) - 1, TFM_CRYPTO_HASH_COMPUTE_SID, TFM_CRYPTO_HASH_COMPARE_SID, TFM_CRYPTO_HASH_SETUP_SID, TFM_CRYPTO_HASH_UPDATE_SID, TFM_CRYPTO_HASH_CLONE_SID, TFM_CRYPTO_HASH_FINISH_SID,
  TFM_CRYPTO_HASH_VERIFY_SID, TFM_CRYPTO_HASH_ABORT_SID, BASE__MAC = ((uint16_↩t)((((uint16_t)( TFM_CRYPTO_GROUP_ID_MAC )) << 8) & 0xFF00)) - 1, TFM_CRYPTO_MAC_COMPUTE_SID, TFM_CRYPTO_MAC_VERIFY_SID, TFM_CRYPTO_MAC_SIGN_SETUP_SID, TFM_CRYPTO_MAC_VERIFY_SETUP_SID, TFM_CRYPTO_MAC_UPDATE_SID,
  TFM_CRYPTO_MAC_SIGN_FINISH_SID, TFM_CRYPTO_MAC_VERIFY_FINISH_SID, TFM_CRYPTO_MAC_ABORT_SID, BASE__CIPHER = ((uint16_t)((((uint16_t)( TFM_CRYPTO_GROUP_ID_CIPHER )) << 8) & 0xFF00)) - 1, TFM_CRYPTO_CIPHER_ENCRYPT_SID, TFM_CRYPTO_CIPHER_DECRYPT_SID, TFM_CRYPTO_CIPHER_ENCRYPT_S TFM_CRYPTO_CIPHER_DECRYPT_SETUP_SID,
  TFM_CRYPTO_CIPHER_GENERATE_IV_SID, TFM_CRYPTO_CIPHER_SET_IV_SID, TFM_CRYPTO_CIPHER_UPDATE_S TFM_CRYPTO_CIPHER_FINISH_SID,
  TFM_CRYPTO_CIPHER_ABORT_SID, BASE__AEAD = ((uint16_t)((((uint16_t)( TFM_CRYPTO_GROUP↩_ID_AEAD )) << 8) & 0xFF00)) - 1, TFM_CRYPTO_AEAD_ENCRYPT_SID, TFM_CRYPTO_AEAD_DECRYPT_SID, TFM_CRYPTO_AEAD_ENCRYPT_SETUP_SID, TFM_CRYPTO_AEAD_DECRYPT_SETUP_SID, TFM_CRYPTO_AEAD_GE TFM_CRYPTO_AEAD_SET_NONCE_SID,
  TFM_CRYPTO_AEAD_SET_LENGTHS_SID, TFM_CRYPTO_AEAD_UPDATE_AD_SID, TFM_CRYPTO_AEAD_UPDATE_S TFM_CRYPTO_AEAD_FINISH_SID,
  TFM_CRYPTO_AEAD_VERIFY_SID, TFM_CRYPTO_AEAD_ABORT_SID, BASE__ASYM_SIGN = ((uint16_t)((((uint16_t)( TFM_CRYPTO_GROUP_ID_ASYM_SIGN )) << 8) & 0xFF00)) - 1, TFM_CRYPTO_ASYMMETRIC_SI TFM_CRYPTO_ASYMMETRIC_VERIFY_MESSAGE_SID, TFM_CRYPTO_ASYMMETRIC_SIGN_HASH_SID,
  TFM_CRYPTO_ASYMMETRIC_VERIFY_HASH_SID, BASE__ASYM_ENCRYPT = ((uint16_t)((((uint16_t)( TFM_CRYPTO_GROUP_ID_ASYM_ENCRYPT )) << 8) & 0xFF00)) - 1,
  TFM_CRYPTO_ASYMMETRIC_ENCRYPT_SID, TFM_CRYPTO_ASYMMETRIC_DECRYPT_SID, BASE__KEY_DERIVATIO = ((uint16_t)((((uint16_t)( TFM_CRYPTO_GROUP_ID_KEY_DERIVATION )) << 8) & 0xFF00)) - 1,
  TFM_CRYPTO_RAW_KEY_AGREEMENT_SID,
  TFM_CRYPTO_KEY_DERIVATION_SETUP_SID, TFM_CRYPTO_KEY_DERIVATION_GET_CAPACITY_SID, TFM_CRYPTO_KEY_DERIVATION_SET_CAPACITY_SID, TFM_CRYPTO_KEY_DERIVATION_INPUT_BYTES_SID, TFM_CRYPTO_KEY_DERIVATION_INPUT_KEY_SID, TFM_CRYPTO_KEY_DERIVATION_INPUT_INTEGER_SID, TFM_CRYPTO_KEY_DERIVATION_KEY_AGREEMENT_SID, TFM_CRYPTO_KEY_DERIVATION_OUTPUT_BYTES_SID, TFM_CRYPTO_KEY_DERIVATION_OUTPUT_KEY_SID, TFM_CRYPTO_KEY_DERIVATION_ABORT_SID
  }

  *This type defines numerical progressive values identifying a function API exposed through the interfaces (S or NS). It's used to dispatch the requests from S/NS to the corresponding API implementation in the Crypto service backend.*

## 7.23.1 Macro Definition Documentation

### 7.23.1.1 AEAD_FUNCS

```
#define AEAD_FUNCS
```

**Value:**

```
    X(TFM_CRYPTO_AEAD_ENCRYPT)                  \
    X(TFM_CRYPTO_AEAD_DECRYPT)                  \
    X(TFM_CRYPTO_AEAD_ENCRYPT_SETUP)            \
    X(TFM_CRYPTO_AEAD_DECRYPT_SETUP)            \
    X(TFM_CRYPTO_AEAD_GENERATE_NONCE)           \
    X(TFM_CRYPTO_AEAD_SET_NONCE)                \
    X(TFM_CRYPTO_AEAD_SET_LENGTHS)              \
    X(TFM_CRYPTO_AEAD_UPDATE_AD)                \
    X(TFM_CRYPTO_AEAD_UPDATE)                   \
    X(TFM_CRYPTO_AEAD_FINISH)                   \
    X(TFM_CRYPTO_AEAD_VERIFY)                   \
    X(TFM_CRYPTO_AEAD_ABORT)
```

Definition at line 129 of file tfm_crypto_defs.h.

### 7.23.1.2 ASYM_ENCRYPT_FUNCS

```
#define ASYM_ENCRYPT_FUNCS
```

**Value:**

```
    X(TFM_CRYPTO_ASYMMETRIC_ENCRYPT)            \
    X(TFM_CRYPTO_ASYMMETRIC_DECRYPT)
```

Definition at line 149 of file tfm_crypto_defs.h.

### 7.23.1.3 ASYM_SIGN_FUNCS

```
#define ASYM_SIGN_FUNCS
```

**Value:**

```
    X(TFM_CRYPTO_ASYMMETRIC_SIGN_MESSAGE)       \
    X(TFM_CRYPTO_ASYMMETRIC_VERIFY_MESSAGE)     \
    X(TFM_CRYPTO_ASYMMETRIC_SIGN_HASH)          \
    X(TFM_CRYPTO_ASYMMETRIC_VERIFY_HASH)
```

Definition at line 143 of file tfm_crypto_defs.h.

### 7.23.1.4 BASE__VALUE

```
#define BASE__VALUE(
              x ) ((uint16_t)((((uint16_t)(x)) << 8) & 0xFF00))
```

Definition at line 166 of file tfm_crypto_defs.h.

### 7.23.1.5 CIPHER_FUNCS

```
#define CIPHER_FUNCS
```

**Value:**

```
    X(TFM_CRYPTO_CIPHER_ENCRYPT)                \
    X(TFM_CRYPTO_CIPHER_DECRYPT)                \
    X(TFM_CRYPTO_CIPHER_ENCRYPT_SETUP)          \
    X(TFM_CRYPTO_CIPHER_DECRYPT_SETUP)          \
    X(TFM_CRYPTO_CIPHER_GENERATE_IV)            \
    X(TFM_CRYPTO_CIPHER_SET_IV)                 \
    X(TFM_CRYPTO_CIPHER_UPDATE)                 \
    X(TFM_CRYPTO_CIPHER_FINISH)                 \
    X(TFM_CRYPTO_CIPHER_ABORT)
```

Definition at line 118 of file tfm_crypto_defs.h.

### 7.23.1.6 HASH_FUNCS

```
#define HASH_FUNCS
```

**Value:**

```
X(TFM_CRYPTO_HASH_COMPUTE)                    \
X(TFM_CRYPTO_HASH_COMPARE)                    \
X(TFM_CRYPTO_HASH_SETUP)                      \
X(TFM_CRYPTO_HASH_UPDATE)                     \
X(TFM_CRYPTO_HASH_CLONE)                      \
X(TFM_CRYPTO_HASH_FINISH)                     \
X(TFM_CRYPTO_HASH_VERIFY)                     \
X(TFM_CRYPTO_HASH_ABORT)
```

Definition at line 98 of file tfm_crypto_defs.h.

#### 7.23.1.7 KEY_DERIVATION_FUNCS

```
#define KEY_DERIVATION_FUNCS
```

**Value:**

```
X(TFM_CRYPTO_RAW_KEY_AGREEMENT)               \
X(TFM_CRYPTO_KEY_DERIVATION_SETUP)            \
X(TFM_CRYPTO_KEY_DERIVATION_GET_CAPACITY)     \
X(TFM_CRYPTO_KEY_DERIVATION_SET_CAPACITY)     \
X(TFM_CRYPTO_KEY_DERIVATION_INPUT_BYTES)      \
X(TFM_CRYPTO_KEY_DERIVATION_INPUT_KEY)        \
X(TFM_CRYPTO_KEY_DERIVATION_INPUT_INTEGER)    \
X(TFM_CRYPTO_KEY_DERIVATION_KEY_AGREEMENT)    \
X(TFM_CRYPTO_KEY_DERIVATION_OUTPUT_BYTES)     \
X(TFM_CRYPTO_KEY_DERIVATION_OUTPUT_KEY)       \
X(TFM_CRYPTO_KEY_DERIVATION_ABORT)
```

Definition at line 153 of file tfm_crypto_defs.h.

#### 7.23.1.8 KEY_MANAGEMENT_FUNCS

```
#define KEY_MANAGEMENT_FUNCS
```

**Value:**

```
X(TFM_CRYPTO_GET_KEY_ATTRIBUTES)              \
X(TFM_CRYPTO_OPEN_KEY)                        \
X(TFM_CRYPTO_CLOSE_KEY)                       \
X(TFM_CRYPTO_IMPORT_KEY)                      \
X(TFM_CRYPTO_DESTROY_KEY)                     \
X(TFM_CRYPTO_EXPORT_KEY)                      \
X(TFM_CRYPTO_EXPORT_PUBLIC_KEY)               \
X(TFM_CRYPTO_PURGE_KEY)                       \
X(TFM_CRYPTO_COPY_KEY)                        \
X(TFM_CRYPTO_GENERATE_KEY)
```

Definition at line 86 of file tfm_crypto_defs.h.

#### 7.23.1.9 MAC_FUNCS

```
#define MAC_FUNCS
```

**Value:**

```
X(TFM_CRYPTO_MAC_COMPUTE)                     \
X(TFM_CRYPTO_MAC_VERIFY)                      \
X(TFM_CRYPTO_MAC_SIGN_SETUP)                  \
X(TFM_CRYPTO_MAC_VERIFY_SETUP)                \
X(TFM_CRYPTO_MAC_UPDATE)                      \
X(TFM_CRYPTO_MAC_SIGN_FINISH)                 \
X(TFM_CRYPTO_MAC_VERIFY_FINISH)               \
X(TFM_CRYPTO_MAC_ABORT)
```

Definition at line 108 of file tfm_crypto_defs.h.

#### 7.23.1.10 RANDOM_FUNCS

```
#define RANDOM_FUNCS X(TFM_CRYPTO_GENERATE_RANDOM)
```

Definition at line 83 of file tfm_crypto_defs.h.

#### 7.23.1.11 TFM_CRYPTO_GET_GROUP_ID

```
#define TFM_CRYPTO_GET_GROUP_ID(
```

```
_function_id ) ((enum tfm_crypto_group_id_t)(((uint16_t)(_function_id) >> 8) &
0xFF))
```

This macro is used to extract the group_id from an encoded function id by accessing the upper 8 bits. A *_function_id* is uint16_t type.
Definition at line 207 of file tfm_crypto_defs.h.

#### 7.23.1.12 TFM_CRYPTO_MAX_NONCE_LENGTH

```
#define TFM_CRYPTO_MAX_NONCE_LENGTH (16u)
```

The maximum supported length of a nonce through the TF-M interfaces.
Definition at line 26 of file tfm_crypto_defs.h.

#### 7.23.1.13 X

```
#define X(
            FUNCTION_NAME ) FUNCTION_NAME ## _SID,
```

Definition at line 181 of file tfm_crypto_defs.h.

### 7.23.2 Enumeration Type Documentation

#### 7.23.2.1 tfm_crypto_func_sid_t

```
enum tfm_crypto_func_sid_t
```

This type defines numerical progressive values identifying a function API exposed through the interfaces (S or NS). It's used to dispatch the requests from S/NS to the corresponding API implementation in the Crypto service backend.

**Note**

Each function SID is encoded as uint16_t. +----------+----------+ | Group ID | Func ID | +----------+----------+ (MSB)15 8 7 0(LSB)

**Enumerator**

| | |
|---|---|
| BASE_RANDOM | |
| TFM_CRYPTO_GENERATE_RANDOM_SID | |
| BASE_KEY_MANAGEMENT | |
| TFM_CRYPTO_GET_KEY_ATTRIBUTES_SID | |
| TFM_CRYPTO_OPEN_KEY_SID | |
| TFM_CRYPTO_CLOSE_KEY_SID | |
| TFM_CRYPTO_IMPORT_KEY_SID | |
| TFM_CRYPTO_DESTROY_KEY_SID | |
| TFM_CRYPTO_EXPORT_KEY_SID | |
| TFM_CRYPTO_EXPORT_PUBLIC_KEY_SID | |
| TFM_CRYPTO_PURGE_KEY_SID | |
| TFM_CRYPTO_COPY_KEY_SID | |
| TFM_CRYPTO_GENERATE_KEY_SID | |
| BASE_HASH | |
| TFM_CRYPTO_HASH_COMPUTE_SID | |
| TFM_CRYPTO_HASH_COMPARE_SID | |
| TFM_CRYPTO_HASH_SETUP_SID | |
| TFM_CRYPTO_HASH_UPDATE_SID | |
| TFM_CRYPTO_HASH_CLONE_SID | |

**Enumerator**

| | |
|---|---|
| TFM_CRYPTO_HASH_FINISH_SID | |
| TFM_CRYPTO_HASH_VERIFY_SID | |
| TFM_CRYPTO_HASH_ABORT_SID | |
| BASE__MAC | |
| TFM_CRYPTO_MAC_COMPUTE_SID | |
| TFM_CRYPTO_MAC_VERIFY_SID | |
| TFM_CRYPTO_MAC_SIGN_SETUP_SID | |
| TFM_CRYPTO_MAC_VERIFY_SETUP_SID | |
| TFM_CRYPTO_MAC_UPDATE_SID | |
| TFM_CRYPTO_MAC_SIGN_FINISH_SID | |
| TFM_CRYPTO_MAC_VERIFY_FINISH_SID | |
| TFM_CRYPTO_MAC_ABORT_SID | |
| BASE__CIPHER | |
| TFM_CRYPTO_CIPHER_ENCRYPT_SID | |
| TFM_CRYPTO_CIPHER_DECRYPT_SID | |
| TFM_CRYPTO_CIPHER_ENCRYPT_SETUP_SID | |
| TFM_CRYPTO_CIPHER_DECRYPT_SETUP_SID | |
| TFM_CRYPTO_CIPHER_GENERATE_IV_SID | |
| TFM_CRYPTO_CIPHER_SET_IV_SID | |
| TFM_CRYPTO_CIPHER_UPDATE_SID | |
| TFM_CRYPTO_CIPHER_FINISH_SID | |
| TFM_CRYPTO_CIPHER_ABORT_SID | |
| BASE__AEAD | |
| TFM_CRYPTO_AEAD_ENCRYPT_SID | |
| TFM_CRYPTO_AEAD_DECRYPT_SID | |
| TFM_CRYPTO_AEAD_ENCRYPT_SETUP_SID | |
| TFM_CRYPTO_AEAD_DECRYPT_SETUP_SID | |
| TFM_CRYPTO_AEAD_GENERATE_NONCE_SID | |
| TFM_CRYPTO_AEAD_SET_NONCE_SID | |
| TFM_CRYPTO_AEAD_SET_LENGTHS_SID | |
| TFM_CRYPTO_AEAD_UPDATE_AD_SID | |
| TFM_CRYPTO_AEAD_UPDATE_SID | |
| TFM_CRYPTO_AEAD_FINISH_SID | |
| TFM_CRYPTO_AEAD_VERIFY_SID | |
| TFM_CRYPTO_AEAD_ABORT_SID | |
| BASE__ASYM_SIGN | |
| TFM_CRYPTO_ASYMMETRIC_SIGN_MESSAGE_SID | |
| TFM_CRYPTO_ASYMMETRIC_VERIFY_MESSAGE_SID | |
| TFM_CRYPTO_ASYMMETRIC_SIGN_HASH_SID | |
| TFM_CRYPTO_ASYMMETRIC_VERIFY_HASH_SID | |
| BASE__ASYM_ENCRYPT | |
| TFM_CRYPTO_ASYMMETRIC_ENCRYPT_SID | |
| TFM_CRYPTO_ASYMMETRIC_DECRYPT_SID | |
| BASE__KEY_DERIVATION | |
| TFM_CRYPTO_RAW_KEY_AGREEMENT_SID | |
| TFM_CRYPTO_KEY_DERIVATION_SETUP_SID | |
| TFM_CRYPTO_KEY_DERIVATION_GET_CAPACITY_SID | |
| TFM_CRYPTO_KEY_DERIVATION_SET_CAPACITY_SID | |
| TFM_CRYPTO_KEY_DERIVATION_INPUT_BYTES_SID | |

**Enumerator**

| | |
|---|---|
| TFM_CRYPTO_KEY_DERIVATION_INPUT_KEY_SID | |
| TFM_CRYPTO_KEY_DERIVATION_INPUT_INTEGER_SID | |
| TFM_CRYPTO_KEY_DERIVATION_KEY_AGREEMENT_SID | |
| TFM_CRYPTO_KEY_DERIVATION_OUTPUT_BYTES_SID | |
| TFM_CRYPTO_KEY_DERIVATION_OUTPUT_KEY_SID | |
| TFM_CRYPTO_KEY_DERIVATION_ABORT_SID | |

Definition at line 180 of file tfm_crypto_defs.h.

#### 7.23.2.2 tfm_crypto_group_id_t

`enum tfm_crypto_group_id_t`

Type associated to the group of a function encoding. There can be nine groups (Random, Key management, Hash, MAC, Cipher, AEAD, Asym sign, Asym encrypt, Key derivation).

**Enumerator**

| | |
|---|---|
| TFM_CRYPTO_GROUP_ID_RANDOM | |
| TFM_CRYPTO_GROUP_ID_KEY_MANAGEMENT | |
| TFM_CRYPTO_GROUP_ID_HASH | |
| TFM_CRYPTO_GROUP_ID_MAC | |
| TFM_CRYPTO_GROUP_ID_CIPHER | |
| TFM_CRYPTO_GROUP_ID_AEAD | |
| TFM_CRYPTO_GROUP_ID_ASYM_SIGN | |
| TFM_CRYPTO_GROUP_ID_ASYM_ENCRYPT | |
| TFM_CRYPTO_GROUP_ID_KEY_DERIVATION | |

Definition at line 70 of file tfm_crypto_defs.h.

## 7.24 interface/include/tfm_fwu_defs.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define TFM_FWU_START 1001
- #define TFM_FWU_WRITE 1002
- #define TFM_FWU_FINISH 1003
- #define TFM_FWU_CANCEL 1004
- #define TFM_FWU_INSTALL 1005
- #define TFM_FWU_CLEAN 1006
- #define TFM_FWU_REJECT 1007
- #define TFM_FWU_REQUEST_REBOOT 1008
- #define TFM_FWU_ACCEPT 1009
- #define TFM_FWU_QUERY 1010

### 7.24.1 Macro Definition Documentation

#### 7.24.1.1 TFM_FWU_ACCEPT

```
#define TFM_FWU_ACCEPT 1009
```

Definition at line 24 of file tfm_fwu_defs.h.

#### 7.24.1.2 TFM_FWU_CANCEL

```
#define TFM_FWU_CANCEL 1004
```

Definition at line 19 of file tfm_fwu_defs.h.

#### 7.24.1.3 TFM_FWU_CLEAN

```
#define TFM_FWU_CLEAN 1006
```

Definition at line 21 of file tfm_fwu_defs.h.

#### 7.24.1.4 TFM_FWU_FINISH

```
#define TFM_FWU_FINISH 1003
```

Definition at line 18 of file tfm_fwu_defs.h.

#### 7.24.1.5 TFM_FWU_INSTALL

```
#define TFM_FWU_INSTALL 1005
```

Definition at line 20 of file tfm_fwu_defs.h.

#### 7.24.1.6 TFM_FWU_QUERY

```
#define TFM_FWU_QUERY 1010
```

Definition at line 25 of file tfm_fwu_defs.h.

#### 7.24.1.7 TFM_FWU_REJECT

```
#define TFM_FWU_REJECT 1007
```

Definition at line 22 of file tfm_fwu_defs.h.

#### 7.24.1.8 TFM_FWU_REQUEST_REBOOT

```
#define TFM_FWU_REQUEST_REBOOT 1008
```

Definition at line 23 of file tfm_fwu_defs.h.

#### 7.24.1.9 TFM_FWU_START

```
#define TFM_FWU_START 1001
```

Definition at line 16 of file tfm_fwu_defs.h.

#### 7.24.1.10 TFM_FWU_WRITE

```
#define TFM_FWU_WRITE 1002
```

Definition at line 17 of file tfm_fwu_defs.h.

## 7.25 interface/include/tfm_its_defs.h File Reference

This graph shows which files directly or indirectly include this file:

## Macros

- #define TFM_ITS_INVALID_UID 0
- #define TFM_ITS_SET 1001
- #define TFM_ITS_GET 1002
- #define TFM_ITS_GET_INFO 1003
- #define TFM_ITS_REMOVE 1004

### 7.25.1 Macro Definition Documentation

#### 7.25.1.1 TFM_ITS_GET

```
#define TFM_ITS_GET 1002
```

Definition at line 20 of file tfm_its_defs.h.

#### 7.25.1.2 TFM_ITS_GET_INFO

```
#define TFM_ITS_GET_INFO 1003
```

Definition at line 21 of file tfm_its_defs.h.

#### 7.25.1.3 TFM_ITS_INVALID_UID

```
#define TFM_ITS_INVALID_UID 0
```

Definition at line 16 of file tfm_its_defs.h.

#### 7.25.1.4 TFM_ITS_REMOVE

```
#define TFM_ITS_REMOVE 1004
```

Definition at line 22 of file tfm_its_defs.h.

#### 7.25.1.5 TFM_ITS_SET

```
#define TFM_ITS_SET 1001
```

Definition at line 19 of file tfm_its_defs.h.

## 7.26 interface/include/tfm_ns_client_ext.h File Reference

```
#include <stdint.h>
```

Include dependency graph for tfm_ns_client_ext.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define TFM_NS_CLIENT_INVALID_TOKEN 0xFFFFFFFF
- #define TFM_NS_CLIENT_ERR_SUCCESS 0x0
- #define TFM_NS_CLIENT_ERR_INVALID_TOKEN 0x1
- #define TFM_NS_CLIENT_ERR_INVALID_NSID 0x2
- #define TFM_NS_CLIENT_ERR_INVALID_ACCESS 0x3

## Functions

- uint32_t tfm_nsce_init (uint32_t ctx_requested)

  *Initialize the non-secure client extension.*
- uint32_t tfm_nsce_acquire_ctx (uint8_t group_id, uint8_t thread_id)

  *Acquire the context for a non-secure client.*
- uint32_t tfm_nsce_release_ctx (uint32_t token)

  *Release the context for the non-secure client.*
- uint32_t tfm_nsce_load_ctx (uint32_t token, int32_t nsid)

  *Load the context for the non-secure client.*
- uint32_t tfm_nsce_save_ctx (uint32_t token)

  *Save the context for the non-secure client.*

## 7.26.1 Macro Definition Documentation

### 7.26.1.1 TFM_NS_CLIENT_ERR_INVALID_ACCESS

```
#define TFM_NS_CLIENT_ERR_INVALID_ACCESS 0x3
```

Definition at line 23 of file tfm_ns_client_ext.h.

### 7.26.1.2 TFM_NS_CLIENT_ERR_INVALID_NSID

```
#define TFM_NS_CLIENT_ERR_INVALID_NSID 0x2
```

Definition at line 22 of file tfm_ns_client_ext.h.

### 7.26.1.3 TFM_NS_CLIENT_ERR_INVALID_TOKEN

```
#define TFM_NS_CLIENT_ERR_INVALID_TOKEN 0x1
```

Definition at line 21 of file tfm_ns_client_ext.h.

### 7.26.1.4 TFM_NS_CLIENT_ERR_SUCCESS

```
#define TFM_NS_CLIENT_ERR_SUCCESS 0x0
```

Definition at line 20 of file tfm_ns_client_ext.h.

### 7.26.1.5 TFM_NS_CLIENT_INVALID_TOKEN

```
#define TFM_NS_CLIENT_INVALID_TOKEN 0xFFFFFFFF
```

Definition at line 17 of file tfm_ns_client_ext.h.

## 7.26.2 Function Documentation

### 7.26.2.1 tfm_nsce_acquire_ctx()

```
uint32_t tfm_nsce_acquire_ctx (
            uint8_t group_id,
            uint8_t thread_id )
```

Acquire the context for a non-secure client.

This function should be called before a non-secure client calling the PSA API into TF-M. It is to request the allocation of the context for the upcoming service call from that non-secure client. The non-secure clients in one group share the same context. The thread ID is used to identify the different non-secure clients.

**Parameters**

| | | |
|---|---|---|
| in | *group_id* | The group ID of the non-secure client |
| in | *thread_id* | The thread ID of the non-secure client |

**Returns**

Returns the token of the allocated context. 0xFFFFFFFF means the allocation failed and the token is invalid.

Definition at line 54 of file tfm_ns_client_ext.c.
Here is the call graph for this function:

#### 7.26.2.2 tfm_nsce_init()

```
uint32_t tfm_nsce_init (
            uint32_t ctx_requested )
```

Initialize the non-secure client extension.
This function should be called before any other non-secure client APIs. It gives NSPE the opportunity to initialize the non-secure client extension in TF-M. Also, NSPE can get the number of allocated non-secure client context slots in the return value. That is useful if NSPE wants to decide the group (context) assignment at runtime.

**Parameters**

| | | |
|---|---|---|
| in | *ctx_requested* | The number of non-secure context requested from the NS entity. If request maximum available context, then set it to 0. |

**Returns**

Returns the number of non-secure context allocated to the NS entity. The allocated context number <= maximum supported context number. If the initialization is failed, then 0 is returned.

Definition at line 36 of file tfm_ns_client_ext.c.
Here is the call graph for this function:

#### 7.26.2.3 tfm_nsce_load_ctx()

```
uint32_t tfm_nsce_load_ctx (
```

```
            uint32_t token,
            int32_t nsid )
```

Load the context for the non-secure client.

This function should be called when a non-secure client is going to be scheduled in at the non-secure side. The caller is usually the scheduler of the RTOS. The non-secure client ID is managed by the non-secure world and passed to TF-M as the input parameter of TF-M.

**Parameters**

| | | |
|---|---|---|
| in | *token* | The token returned by tfm_nsce_acquire_ctx |
| in | *nsid* | The non-secure client ID for this client |

**Returns**

Returns the error code.

Definition at line 102 of file tfm_ns_client_ext.c.

Here is the call graph for this function:

#### 7.26.2.4 tfm_nsce_release_ctx()

```
uint32_t tfm_nsce_release_ctx (
            uint32_t token )
```

Release the context for the non-secure client.

This function should be called when a non-secure client is going to be terminated or will not call TF-M secure services in the future. It is to release the context allocated for the calling non-secure client. If the calling non-secure client is the only thread in the group, then the context will be deallocated. Otherwise, the context will still be taken for the other threads in the group.

**Parameters**

| | | |
|---|---|---|
| in | *token* | The token returned by tfm_nsce_acquire_ctx |

**Returns**

Returns the error code.

Definition at line 73 of file tfm_ns_client_ext.c.
Here is the call graph for this function:

#### 7.26.2.5 tfm_nsce_save_ctx()

```
uint32_t tfm_nsce_save_ctx (
            uint32_t token )
```

Save the context for the non-secure client.
This function should be called when a non-secure client is going to be scheduled out at the non-secure side. The caller is usually the scheduler of the RTOS.

**Parameters**

| | | |
|---|---|---|
| in | *token* | The token returned by tfm_nsce_acquire_ctx |

**Returns**

Returns the error code.

Definition at line 135 of file tfm_ns_client_ext.c.
Here is the call graph for this function:

## 7.27 interface/include/tfm_ns_interface.h File Reference

```
#include <stdint.h>
#include "tfm_veneers.h"
```

Include dependency graph for tfm_ns_interface.h:

This graph shows which files directly or indirectly include this file:

## Typedefs

- typedef int32_t(∗ veneer_fn) (uint32_t arg0, uint32_t arg1, uint32_t arg2, uint32_t arg3)

## Functions

- int32_t tfm_ns_interface_dispatch (veneer_fn fn, uint32_t arg0, uint32_t arg1, uint32_t arg2, uint32_t arg3)

  *NS interface, veneer function dispatcher.*
- uint32_t tfm_ns_interface_init (void)

  *NS interface initialization function.*

## 7.27.1 Typedef Documentation

### 7.27.1.1 veneer_fn

```
typedef int32_t(* veneer_fn) (uint32_t arg0, uint32_t arg1, uint32_t arg2, uint32_t arg3)
```

Definition at line 19 of file tfm_ns_interface.h.

## 7.27.2 Function Documentation

### 7.27.2.1 tfm_ns_interface_dispatch()

```
int32_t tfm_ns_interface_dispatch (
            veneer_fn fn,
            uint32_t arg0,
            uint32_t arg1,
            uint32_t arg2,
            uint32_t arg3 )
```

NS interface, veneer function dispatcher.
This function implements the dispatching mechanism for the desired veneer function, to be called with the parameters described from arg0 to arg3.

**Note**

NSPE can use default implementation of this function or implement this function according to NS specific implementation and actual usage scenario.

**Parameters**

| | | |
|---|---|---|
| in | *fn* | Function pointer to the veneer function desired |
| in | *arg0* | Argument 0 of fn |
| in | *arg1* | Argument 1 of fn |
| in | *arg2* | Argument 2 of fn |
| in | *arg3* | Argument 3 of fn |

**Returns**

Returns the same return value of the requested veneer function

**Note**

This API must ensure the return value is from the veneer function. Other unrecoverable errors must be considered as fatal error and should not return.

Definition at line 26 of file tfm_ns_interface_bare_metal.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.27.2.2 tfm_ns_interface_init()

```
uint32_t tfm_ns_interface_init (
            void  )
```

NS interface initialization function.
This function initializes TF-M NS interface.

**Note**

NSPE can use default implementation of this function or implement this function according to NS specific implementation and actual usage scenario.

**Returns**

OS_WRAPPER_SUCCESS on success or OS_WRAPPER_ERROR on error

Definition at line 33 of file tfm_ns_interface_bare_metal.c.

## 7.28 interface/include/tfm_platform_api.h File Reference

```
#include <limits.h>
#include <stdbool.h>
```

```
#include <stdint.h>
#include "psa/client.h"
```

Include dependency graph for tfm_platform_api.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define TFM_PLATFORM_API_VERSION_MAJOR (0)

  *TFM secure partition platform API version.*
- #define TFM_PLATFORM_API_VERSION_MINOR (3)
- #define TFM_PLATFORM_API_ID_NV_READ (1010)
- #define TFM_PLATFORM_API_ID_NV_INCREMENT (1011)
- #define TFM_PLATFORM_API_ID_SYSTEM_RESET (1012)
- #define TFM_PLATFORM_API_ID_IOCTL (1013)

## Typedefs

- typedef int32_t tfm_platform_ioctl_req_t

## Enumerations

- enum tfm_platform_err_t {
  TFM_PLATFORM_ERR_SUCCESS = 0, TFM_PLATFORM_ERR_SYSTEM_ERROR, TFM_PLATFORM_ERR_INVALID_PAR
  TFM_PLATFORM_ERR_NOT_SUPPORTED,
  TFM_PLATFORM_ERR_FORCE_INT_SIZE = INT_MAX }

  *Platform service error types.*

## Functions

- enum tfm_platform_err_t tfm_platform_system_reset (void)

  *Resets the system.*
- enum tfm_platform_err_t tfm_platform_ioctl (tfm_platform_ioctl_req_t request, psa_invec *input, psa_outvec *output)

  *Performs a platform-specific service.*
- enum tfm_platform_err_t tfm_platform_nv_counter_increment (uint32_t counter_id)

  *Increments the given non-volatile (NV) counter by one.*
- enum tfm_platform_err_t tfm_platform_nv_counter_read (uint32_t counter_id, uint32_t size, uint8_t *val)

  *Reads the given non-volatile (NV) counter.*

## 7.28.1 Macro Definition Documentation

### 7.28.1.1 TFM_PLATFORM_API_ID_IOCTL

```
#define TFM_PLATFORM_API_ID_IOCTL (1013)
```

Definition at line 29 of file tfm_platform_api.h.

### 7.28.1.2 TFM_PLATFORM_API_ID_NV_INCREMENT

```
#define TFM_PLATFORM_API_ID_NV_INCREMENT (1011)
```

Definition at line 27 of file tfm_platform_api.h.

### 7.28.1.3 TFM_PLATFORM_API_ID_NV_READ

```
#define TFM_PLATFORM_API_ID_NV_READ (1010)
```

Definition at line 26 of file tfm_platform_api.h.

### 7.28.1.4 TFM_PLATFORM_API_ID_SYSTEM_RESET

```
#define TFM_PLATFORM_API_ID_SYSTEM_RESET (1012)
```

Definition at line 28 of file tfm_platform_api.h.

### 7.28.1.5 TFM_PLATFORM_API_VERSION_MAJOR

```
#define TFM_PLATFORM_API_VERSION_MAJOR (0)
```

TFM secure partition platform API version.
Definition at line 23 of file tfm_platform_api.h.

### 7.28.1.6 TFM_PLATFORM_API_VERSION_MINOR

```
#define TFM_PLATFORM_API_VERSION_MINOR (3)
```

Definition at line 24 of file tfm_platform_api.h.

## 7.28.2 Typedef Documentation

#### 7.28.2.1 tfm_platform_ioctl_req_t

```
typedef int32_t tfm_platform_ioctl_req_t
```

Definition at line 47 of file tfm_platform_api.h.

### 7.28.3 Enumeration Type Documentation

#### 7.28.3.1 tfm_platform_err_t

```
enum tfm_platform_err_t
```

Platform service error types.

**Enumerator**

| | |
|---|---|
| TFM_PLATFORM_ERR_SUCCESS | |
| TFM_PLATFORM_ERR_SYSTEM_ERROR | |
| TFM_PLATFORM_ERR_INVALID_PARAM | |
| TFM_PLATFORM_ERR_NOT_SUPPORTED | |
| TFM_PLATFORM_ERR_FORCE_INT_SIZE | |

Definition at line 37 of file tfm_platform_api.h.

### 7.28.4 Function Documentation

#### 7.28.4.1 tfm_platform_ioctl()

```
enum tfm_platform_err_t tfm_platform_ioctl (
            tfm_platform_ioctl_req_t request,
            psa_invec * input,
            psa_outvec * output )
```

Performs a platform-specific service.

**Parameters**

| | | |
|---|---|---|
| in | *request* | Request identifier (valid values vary based on the platform) |
| in | *input* | Input buffer to the requested service (or NULL) |
| in,out | *output* | Output buffer to the requested service (or NULL) |

**Returns**

Returns values as specified by the tfm_platform_err_t

Definition at line 30 of file tfm_platform_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.28.4.2 tfm_platform_nv_counter_increment()

```
enum tfm_platform_err_t tfm_platform_nv_counter_increment (
            uint32_t counter_id )
```

Increments the given non-volatile (NV) counter by one.

**Parameters**

| | | |
|---|---|---|
| in | *counter↩ _id* | NV counter ID. |

**Returns**

TFM_PLATFORM_ERR_SUCCESS if the value is read correctly. Otherwise, it returns TFM_PLATFORM_↩ ERR_SYSTEM_ERROR.

Definition at line 67 of file tfm_platform_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.28.4.3 tfm_platform_nv_counter_read()

```
enum tfm_platform_err_t tfm_platform_nv_counter_read (
            uint32_t counter_id,
```

```
uint32_t size,
uint8_t * val )
```

Reads the given non-volatile (NV) counter.

**Parameters**

| | | |
|---|---|---|
| in | *counter↩ _id* | NV counter ID. |
| in | *size* | Size of the buffer to store NV counter value in bytes. |
| out | *val* | Pointer to store the current NV counter value. |

**Returns**

TFM_PLATFORM_ERR_SUCCESS if the value is read correctly. Otherwise, it returns TFM_PLATFORM_↩ ERR_SYSTEM_ERROR.

Definition at line 87 of file tfm_platform_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.28.4.4 tfm_platform_system_reset()

```
enum tfm_platform_err_t tfm_platform_system_reset (
            void  )
```

Resets the system.

**Returns**

Returns values as specified by the tfm_platform_err_t

Definition at line 13 of file tfm_platform_api.c.
Here is the call graph for this function:

## 7.29 interface/include/tfm_ps_defs.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define TFM_PS_INVALID_UID 0
- #define TFM_PS_SET 1001
- #define TFM_PS_GET 1002
- #define TFM_PS_GET_INFO 1003
- #define TFM_PS_REMOVE 1004
- #define TFM_PS_GET_SUPPORT 1005

### 7.29.1 Macro Definition Documentation

#### 7.29.1.1 TFM_PS_GET

```
#define TFM_PS_GET 1002
```

Definition at line 20 of file tfm_ps_defs.h.

#### 7.29.1.2 TFM_PS_GET_INFO

```
#define TFM_PS_GET_INFO 1003
```

Definition at line 21 of file tfm_ps_defs.h.

#### 7.29.1.3 TFM_PS_GET_SUPPORT

```
#define TFM_PS_GET_SUPPORT 1005
```

Definition at line 23 of file tfm_ps_defs.h.

#### 7.29.1.4 TFM_PS_INVALID_UID

```
#define TFM_PS_INVALID_UID 0
```

Definition at line 16 of file tfm_ps_defs.h.

#### 7.29.1.5 TFM_PS_REMOVE

```
#define TFM_PS_REMOVE 1004
```

Definition at line 22 of file tfm_ps_defs.h.

#### 7.29.1.6 TFM_PS_SET

```
#define TFM_PS_SET 1001
```

Definition at line 19 of file tfm_ps_defs.h.

## 7.30 interface/include/tfm_psa_call_pack.h File Reference

```
#include "psa/client.h"
```

Include dependency graph for tfm_psa_call_pack.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define TYPE_MASK 0xFFFFUL
- #define IN_LEN_OFFSET 24
- #define IN_LEN_MASK (0x7UL << IN_LEN_OFFSET)
- #define OUT_LEN_OFFSET 16
- #define OUT_LEN_MASK (0x7UL << OUT_LEN_OFFSET)
- #define NS_VEC_DESC_BIT 0x80000000UL
- #define NS_INVEC_OFFSET 27
- #define NS_INVEC_BIT (1UL << NS_INVEC_OFFSET)
- #define NS_OUTVEC_OFFSET 19
- #define NS_OUTVEC_BIT (1UL << NS_OUTVEC_OFFSET)
- #define PARAM_PACK(type, in_len, out_len)
- #define PARAM_UNPACK_TYPE(ctrl_param) ((int32_t)(int16_t)((ctrl_param) & TYPE_MASK))
- #define PARAM_UNPACK_IN_LEN(ctrl_param) ((size_t)(((ctrl_param) & IN_LEN_MASK) >> IN_LEN_OFFSET))
- #define PARAM_UNPACK_OUT_LEN(ctrl_param) ((size_t)(((ctrl_param) & OUT_LEN_MASK) >> OUT_LEN_OFFSET))
- #define PARAM_SET_NS_VEC(ctrl_param) ((ctrl_param) | NS_VEC_DESC_BIT)
- #define PARAM_IS_NS_VEC(ctrl_param) ((ctrl_param) & NS_VEC_DESC_BIT)
- #define PARAM_SET_NS_INVEC(ctrl_param) ((ctrl_param) | NS_INVEC_BIT)
- #define PARAM_IS_NS_INVEC(ctrl_param) ((ctrl_param) & NS_INVEC_BIT)
- #define PARAM_SET_NS_OUTVEC(ctrl_param) ((ctrl_param) | NS_OUTVEC_BIT)
- #define PARAM_IS_NS_OUTVEC(ctrl_param) ((ctrl_param) & NS_OUTVEC_BIT)
- #define PARAM_HAS_IOVEC(ctrl_param) ((ctrl_param) != (uint32_t)PARAM_UNPACK_TYPE(ctrl_param))

## Functions

- psa_status_t tfm_psa_call_pack (psa_handle_t handle, uint32_t ctrl_param, const psa_invec *in_vec, psa_outvec *out_vec)

## 7.30.1 Macro Definition Documentation

### 7.30.1.1 IN_LEN_MASK

```
#define IN_LEN_MASK (0x7UL << IN_LEN_OFFSET)
```

Definition at line 29 of file tfm_psa_call_pack.h.

### 7.30.1.2 IN_LEN_OFFSET

```
#define IN_LEN_OFFSET 24
```

Definition at line 28 of file tfm_psa_call_pack.h.

### 7.30.1.3 NS_INVEC_BIT

```
#define NS_INVEC_BIT (1UL << NS_INVEC_OFFSET)
```

Definition at line 42 of file tfm_psa_call_pack.h.

### 7.30.1.4 NS_INVEC_OFFSET

```
#define NS_INVEC_OFFSET 27
```

Definition at line 41 of file tfm_psa_call_pack.h.

#### 7.30.1.5 NS_OUTVEC_BIT

```
#define NS_OUTVEC_BIT (1UL << NS_OUTVEC_OFFSET)
```

Definition at line 44 of file tfm_psa_call_pack.h.

#### 7.30.1.6 NS_OUTVEC_OFFSET

```
#define NS_OUTVEC_OFFSET 19
```

Definition at line 43 of file tfm_psa_call_pack.h.

#### 7.30.1.7 NS_VEC_DESC_BIT

```
#define NS_VEC_DESC_BIT 0x80000000UL
```

Definition at line 39 of file tfm_psa_call_pack.h.

#### 7.30.1.8 OUT_LEN_MASK

```
#define OUT_LEN_MASK (0x7UL << OUT_LEN_OFFSET)
```

Definition at line 32 of file tfm_psa_call_pack.h.

#### 7.30.1.9 OUT_LEN_OFFSET

```
#define OUT_LEN_OFFSET 16
```

Definition at line 31 of file tfm_psa_call_pack.h.

#### 7.30.1.10 PARAM_HAS_IOVEC

```
#define PARAM_HAS_IOVEC(
            ctrl_param ) ((ctrl_param) != (uint32_t)PARAM_UNPACK_TYPE(ctrl_param))
```

Definition at line 68 of file tfm_psa_call_pack.h.

#### 7.30.1.11 PARAM_IS_NS_INVEC

```
#define PARAM_IS_NS_INVEC(
            ctrl_param ) ((ctrl_param) & NS_INVEC_BIT)
```

Definition at line 64 of file tfm_psa_call_pack.h.

#### 7.30.1.12 PARAM_IS_NS_OUTVEC

```
#define PARAM_IS_NS_OUTVEC(
            ctrl_param ) ((ctrl_param) & NS_OUTVEC_BIT)
```

Definition at line 66 of file tfm_psa_call_pack.h.

#### 7.30.1.13 PARAM_IS_NS_VEC

```
#define PARAM_IS_NS_VEC(
            ctrl_param ) ((ctrl_param) & NS_VEC_DESC_BIT)
```

Definition at line 61 of file tfm_psa_call_pack.h.

#### 7.30.1.14 PARAM_PACK

```
#define PARAM_PACK(
            type,
            in_len,
            out_len )
```

**Value:**

```
(((uint32_t)(type)) & TYPE_MASK)                          | \
 (((uint32_t)(in_len)) « IN_LEN_OFFSET) & IN_LEN_MASK) | \
 (((uint32_t)(out_len)) « OUT_LEN_OFFSET) & OUT_LEN_MASK))
```

Definition at line 46 of file tfm_psa_call_pack.h.

#### 7.30.1.15 PARAM_SET_NS_INVEC

```
#define PARAM_SET_NS_INVEC(
            ctrl_param ) ((ctrl_param) | NS_INVEC_BIT)
```

Definition at line 63 of file tfm_psa_call_pack.h.

#### 7.30.1.16 PARAM_SET_NS_OUTVEC

```
#define PARAM_SET_NS_OUTVEC(
            ctrl_param ) ((ctrl_param) | NS_OUTVEC_BIT)
```

Definition at line 65 of file tfm_psa_call_pack.h.

#### 7.30.1.17 PARAM_SET_NS_VEC

```
#define PARAM_SET_NS_VEC(
            ctrl_param ) ((ctrl_param) | NS_VEC_DESC_BIT)
```

Definition at line 60 of file tfm_psa_call_pack.h.

#### 7.30.1.18 PARAM_UNPACK_IN_LEN

```
#define PARAM_UNPACK_IN_LEN(
            ctrl_param ) ((size_t)(((ctrl_param) & IN_LEN_MASK) >> IN_LEN_OFFSET))
```

Definition at line 54 of file tfm_psa_call_pack.h.

#### 7.30.1.19 PARAM_UNPACK_OUT_LEN

```
#define PARAM_UNPACK_OUT_LEN(
            ctrl_param ) ((size_t)(((ctrl_param) & OUT_LEN_MASK) >> OUT_LEN_OFFSET))
```

Definition at line 57 of file tfm_psa_call_pack.h.

#### 7.30.1.20 PARAM_UNPACK_TYPE

```
#define PARAM_UNPACK_TYPE(
            ctrl_param ) ((int32_t)(int16_t)((ctrl_param) & TYPE_MASK))
```

Definition at line 51 of file tfm_psa_call_pack.h.

#### 7.30.1.21 TYPE_MASK

```
#define TYPE_MASK 0xFFFFUL
```

Definition at line 26 of file tfm_psa_call_pack.h.

### 7.30.2 Function Documentation

#### 7.30.2.1 tfm_psa_call_pack()

```
psa_status_t tfm_psa_call_pack (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * in_vec,
            psa_outvec * out_vec )
```

Definition at line 28 of file psa_api_ipc.c.

## 7.31 interface/include/tfm_veneers.h File Reference

```
#include <stdint.h>
#include "psa/client.h"
```

Include dependency graph for tfm_veneers.h:

This graph shows which files directly or indirectly include this file:

## Functions

- uint32_t tfm_psa_framework_version_veneer (void)

  *Retrieve the version of the PSA Framework API that is implemented.*
- uint32_t tfm_psa_version_veneer (uint32_t sid)

  *Return version of secure function provided by secure binary.*
- psa_handle_t tfm_psa_connect_veneer (uint32_t sid, uint32_t version)

  *Connect to secure function.*
- psa_status_t tfm_psa_call_veneer (psa_handle_t handle, uint32_t ctrl_param, const psa_invec *in_vec, psa_outvec *out_vec)

  *Call a secure function referenced by a connection handle.*
- void tfm_psa_close_veneer (psa_handle_t handle)

  *Close connection to secure function referenced by a connection handle.*

## 7.31.1 Function Documentation

### 7.31.1.1 tfm_psa_call_veneer()

```
psa_status_t tfm_psa_call_veneer (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * in_vec,
            psa_outvec * out_vec )
```

Call a secure function referenced by a connection handle.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Handle to connection. |
| in | *ctrl_param* | Parameters combined in uint32_t, includes request type, in_num and out_num. |
| in | *in_vec* | Array of input psa_invec structures. |
| in,out | *out_vec* | Array of output psa_outvec structures. |

**Returns**

Returns psa_status_t status code.

Definition at line 57 of file psa_api_veneers.c.

#### 7.31.1.2 tfm_psa_close_veneer()

```
void tfm_psa_close_veneer (
            psa_handle_t handle )
```

Close connection to secure function referenced by a connection handle.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Handle to connection |

Definition at line 113 of file psa_api_veneers.c.

#### 7.31.1.3 tfm_psa_connect_veneer()

```
psa_handle_t tfm_psa_connect_veneer (
            uint32_t sid,
            uint32_t version )
```

Connect to secure function.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of secure service. |
| in | *version* | Version of SF requested by client. |

**Returns**

Returns handle to connection.

Definition at line 104 of file psa_api_veneers.c.
Here is the caller graph for this function:

#### 7.31.1.4 tfm_psa_framework_version_veneer()

```
uint32_t tfm_psa_framework_version_veneer (
            void )
```

Retrieve the version of the PSA Framework API that is implemented.

**Returns**

The version of the PSA Framework.

Definition at line 29 of file psa_api_veneers.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.31.1.5 tfm_psa_version_veneer()

```
uint32_t tfm_psa_version_veneer (
            uint32_t sid )
```

Return version of secure function provided by secure binary.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of secure service. |

**Returns**

Version number of secure function.

Definition at line 43 of file psa_api_veneers.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.32 interface/src/multi_core/tfm_multi_core_ns_api.c File Reference

`#include "tfm_multi_core_api.h"`
Include dependency graph for tfm_multi_core_ns_api.c:

## Functions

- int32_t tfm_ns_wait_for_s_cpu_ready (void)

  *Called on the non-secure CPU. Flags that the non-secure side has completed its initialization. Waits, if necessary, for the secure CPU to flag that it has completed its initialization.*

### 7.32.1 Function Documentation

#### 7.32.1.1 tfm_ns_wait_for_s_cpu_ready()

```
int32_t tfm_ns_wait_for_s_cpu_ready (
            void  )
```

Called on the non-secure CPU. Flags that the non-secure side has completed its initialization. Waits, if necessary, for the secure CPU to flag that it has completed its initialization.

**Returns**

Return 0 if succeeds.

Otherwise, return specific error code.

Definition at line 10 of file tfm_multi_core_ns_api.c.
Here is the call graph for this function:

## 7.33 interface/src/multi_core/tfm_multi_core_psa_ns_api.c File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include "psa/client.h"
#include "psa/error.h"
#include "tfm_ns_mailbox.h"
```

Include dependency graph for tfm_multi_core_psa_ns_api.c:

## Macros

- #define NON_SECURE_CLIENT_ID (-1)
- #define PSA_INTER_CORE_COMM_ERR (INT32_MIN + 0xFF)

## Functions

- uint32_t psa_framework_version (void)

  *Retrieve the version of the PSA Framework API that is implemented.*
- uint32_t psa_version (uint32_t sid)

  *Retrieve the version of an RoT Service or indicate that it is not present on this system.*
- psa_handle_t psa_connect (uint32_t sid, uint32_t version)

  *Connect to an RoT Service by its SID.*
- psa_status_t psa_call (psa_handle_t handle, int32_t type, const psa_invec *in_vec, size_t in_len, psa_outvec *out_vec, size_t out_len)

  *Call an RoT Service on an established connection.*
- void psa_close (psa_handle_t handle)

  *Close a connection to an RoT Service.*

## 7.33.1 Macro Definition Documentation

### 7.33.1.1 NON_SECURE_CLIENT_ID

```
#define NON_SECURE_CLIENT_ID (-1)
```

Definition at line 30 of file tfm_multi_core_psa_ns_api.c.

### 7.33.1.2 PSA_INTER_CORE_COMM_ERR

```
#define PSA_INTER_CORE_COMM_ERR (INT32_MIN + 0xFF)
```

Definition at line 37 of file tfm_multi_core_psa_ns_api.c.

## 7.33.2 Function Documentation

### 7.33.2.1 psa_call()

```
psa_status_t psa_call (
            psa_handle_t handle,
            int32_t type,
            const psa_invec * in_vec,
            size_t in_len,
            psa_outvec * out_vec,
            size_t out_len )
```

Call an RoT Service on an established connection.

**Note**

FF-M 1.0 proposes 6 parameters for psa_call but the secure gateway ABI support at most 4 parameters. T↩ F-M chooses to encode 'in_len', 'out_len', and 'type' into a 32-bit integer to improve efficiency. Compared with struct-based encoding, this method saves extra memory check and memory copy operation. The disadvantage is that the 'type' range has to be reduced into a 16-bit integer. So with this encoding, the valid range for 'type' is 0-32767.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | A handle to an established connection. |
| in | *type* | The request type. Must be zero( PSA_IPC_CALL) or positive. |
| in | *in_vec* | Array of input psa_invec structures. |
| in | *in_len* | Number of input psa_invec structures. |
| in,out | *out_vec* | Array of output psa_outvec structures. |
| in | *out_len* | Number of output psa_outvec structures. |

**Return values**

| | |
|---|---|
| *>=0* | RoT Service-specific status value. |
| *<0* | RoT Service-specific error code. |
| *PSA_ERROR_PROGRAMMER_ERROR* | The connection has been terminated by the RoT Service. The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• An invalid handle was passed.<br>• The connection is already handling a request.<br>• type < 0.<br>• An invalid memory reference was provided.<br>• in_len + out_len > PSA_MAX_IOVEC.<br>• The message is unrecognized by the RoT Service or incorrectly formatted. |

Definition at line 100 of file tfm_multi_core_psa_ns_api.c.

#### 7.33.2.2 psa_close()

```
void psa_close (
            psa_handle_t handle )
```

Close a connection to an RoT Service.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | A handle to an established connection, or the null handle. |

**Return values**

| | |
|---|---|
| *void* | Success. |
| *PROGRAMMER ERROR* | The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• An invalid handle was provided that is not the null handle.<br>• The connection is currently handling a request. |

Definition at line 143 of file tfm_multi_core_psa_ns_api.c.

#### 7.33.2.3 psa_connect()

```
psa_handle_t psa_connect (
            uint32_t sid,
            uint32_t version )
```

Connect to an RoT Service by its SID.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of the RoT Service to connect to. |
| in | *version* | Requested version of the RoT Service. |

**Return values**

| | |
|---|---|
| > | 0 A handle for the connection. |
| *PSA_ERROR_CONNECTION_REFUSED* | The SPM or RoT Service has refused the connection. |
| *PSA_ERROR_CONNECTION_BUSY* | The SPM or RoT Service cannot make the connection at the moment. |
| *PROGRAMMER ERROR* | The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• The RoT Service ID is not present.<br>• The RoT Service version is not supported.<br>• The caller is not allowed to access the RoT service. |

Definition at line 79 of file tfm_multi_core_psa_ns_api.c.

Here is the call graph for this function:

#### 7.33.2.4 psa_framework_version()

```
uint32_t psa_framework_version (
            void  )
```

Retrieve the version of the PSA Framework API that is implemented.

**Returns**

version The version of the PSA Framework implementation that is providing the runtime services to the caller. The major and minor version are encoded as follows:

- version[15:8] – major version number.
- version[7:0] – minor version number.

Definition at line 41 of file tfm_multi_core_psa_ns_api.c.
Here is the call graph for this function:

#### 7.33.2.5 psa_version()

```
uint32_t psa_version (
            uint32_t sid )
```

Retrieve the version of an RoT Service or indicate that it is not present on this system.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of the RoT Service to query. |

**Return values**

| | |
|---|---|
| *PSA_VERSION_NONE* | The RoT Service is not implemented, or the caller is not permitted to access the service. |
| > | 0 The version of the implemented RoT Service. |

Definition at line 59 of file tfm_multi_core_psa_ns_api.c.
Here is the call graph for this function:

## 7.34 interface/src/multi_core/tfm_ns_mailbox.c File Reference

```
#include <string.h>
#include "tfm_ns_mailbox.h"
```

Include dependency graph for tfm_ns_mailbox.c:

### Functions

- int32_t tfm_ns_mailbox_client_call (uint32_t call_type, const struct psa_client_params_t *params, int32_t client_id, struct mailbox_reply_t *reply)

  *Send PSA client call to SPE via mailbox. Wait and fetch PSA client call result.*
- int32_t tfm_ns_mailbox_init (struct ns_mailbox_queue_t *queue)

  *NSPE mailbox initialization.*

### 7.34.1 Function Documentation

#### 7.34.1.1 tfm_ns_mailbox_client_call()

```
int32_t tfm_ns_mailbox_client_call (
            uint32_t call_type,
            const struct psa_client_params_t * params,
            int32_t client_id,
            struct mailbox_reply_t * reply )
```

Send PSA client call to SPE via mailbox. Wait and fetch PSA client call result.

**Parameters**

| | | |
|---|---|---|
| in | *call_type* | PSA client call type |
| in | *params* | Parameters used for PSA client call |
| in | *client_id* | Optional client ID of non-secure caller. It is required to identify the non-secure caller when NSPE OS enforces non-secure task isolation. |
| out | *reply* | The buffer written with PSA client call result. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | The PSA client call is completed successfully. |
| *Other* | return code Operation failed with an error code. |

Definition at line 188 of file tfm_ns_mailbox.c.

#### 7.34.1.2 tfm_ns_mailbox_init()

```
int32_t tfm_ns_mailbox_init (
            struct ns_mailbox_queue_t * queue )
```

NSPE mailbox initialization.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The base address of NSPE mailbox queue to be initialized. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 333 of file tfm_ns_mailbox.c.

## 7.35 interface/src/multi_core/tfm_ns_mailbox_thread.c File Reference

```
#include <string.h>
#include "tfm_ns_mailbox.h"
```

Include dependency graph for tfm_ns_mailbox_thread.c:

## Data Structures

- struct ns_mailbox_req_t

## Macros

- #define NOT_WOKEN 0x0
- #define WOKEN_UP 0x5C

## Functions

- int32_t tfm_ns_mailbox_client_call (uint32_t call_type, const struct psa_client_params_t *params, int32_t client_id, struct mailbox_reply_t *reply)

  *Send PSA client call to SPE via mailbox. Wait and fetch PSA client call result.*
- void tfm_ns_mailbox_thread_runner (void *args)
- int32_t tfm_ns_mailbox_wake_reply_owner_isr (void)
- int32_t tfm_ns_mailbox_init (struct ns_mailbox_queue_t *queue)

  *NSPE mailbox initialization.*

## 7.35.1 Macro Definition Documentation

### 7.35.1.1 NOT_WOKEN

```
#define NOT_WOKEN 0x0
```

Definition at line 15 of file tfm_ns_mailbox_thread.c.

### 7.35.1.2 WOKEN_UP

```
#define WOKEN_UP 0x5C
```

Definition at line 16 of file tfm_ns_mailbox_thread.c.

## 7.35.2 Function Documentation

#### 7.35.2.1 tfm_ns_mailbox_client_call()

```
int32_t tfm_ns_mailbox_client_call (
            uint32_t call_type,
            const struct psa_client_params_t * params,
            int32_t client_id,
            struct mailbox_reply_t * reply )
```

Send PSA client call to SPE via mailbox. Wait and fetch PSA client call result.

**Parameters**

| | | |
|---|---|---|
| in | *call_type* | PSA client call type |
| in | *params* | Parameters used for PSA client call |
| in | *client_id* | Optional client ID of non-secure caller. It is required to identify the non-secure caller when NSPE OS enforces non-secure task isolation. |
| out | *reply* | The buffer written with PSA client call result. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | The PSA client call is completed successfully. |
| *Other* | return code Operation failed with an error code. |

Definition at line 192 of file tfm_ns_mailbox_thread.c.

#### 7.35.2.2 tfm_ns_mailbox_init()

```
int32_t tfm_ns_mailbox_init (
            struct ns_mailbox_queue_t * queue )
```

NSPE mailbox initialization.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The base address of NSPE mailbox queue to be initialized. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 334 of file tfm_ns_mailbox_thread.c.
Here is the caller graph for this function:

#### 7.35.2.3 tfm_ns_mailbox_thread_runner()

```
void tfm_ns_mailbox_thread_runner (
            void * args )
```

Definition at line 226 of file tfm_ns_mailbox_thread.c.

#### 7.35.2.4 tfm_ns_mailbox_wake_reply_owner_isr()

```
int32_t tfm_ns_mailbox_wake_reply_owner_isr (
            void  )
```

Definition at line 254 of file tfm_ns_mailbox_thread.c.

## 7.36 interface/src/os_wrapper/tfm_ns_interface_bare_metal.c File Reference

```
#include "os_wrapper/common.h"
#include "tfm_ns_interface.h"
```

Include dependency graph for tfm_ns_interface_bare_metal.c:

### Functions

- int32_t tfm_ns_interface_dispatch (veneer_fn fn, uint32_t arg0, uint32_t arg1, uint32_t arg2, uint32_t arg3)

  *NS interface, veneer function dispatcher.*
- uint32_t tfm_ns_interface_init (void)

  *NS interface initialization function.*

### 7.36.1 Function Documentation

#### 7.36.1.1 tfm_ns_interface_dispatch()

```
int32_t tfm_ns_interface_dispatch (
            veneer_fn fn,
            uint32_t arg0,
            uint32_t arg1,
            uint32_t arg2,
            uint32_t arg3 )
```

NS interface, veneer function dispatcher.
This function implements the dispatching mechanism for the desired veneer function, to be called with the parameters described from arg0 to arg3.

**Note**

NSPE can use default implementation of this function or implement this function according to NS specific implementation and actual usage scenario.

**Parameters**

| | | |
|---|---|---|
| in | *fn* | Function pointer to the veneer function desired |
| in | *arg0* | Argument 0 of fn |
| in | *arg1* | Argument 1 of fn |
| in | *arg2* | Argument 2 of fn |
| in | *arg3* | Argument 3 of fn |

**Returns**

Returns the same return value of the requested veneer function

**Note**

This API must ensure the return value is from the veneer function. Other unrecoverable errors must be considered as fatal error and should not return.

Definition at line 26 of file tfm_ns_interface_bare_metal.c.

#### 7.36.1.2 tfm_ns_interface_init()

```
uint32_t tfm_ns_interface_init (
            void )
```

NS interface initialization function.
This function initializes TF-M NS interface.

**Note**

NSPE can use default implementation of this function or implement this function according to NS specific implementation and actual usage scenario.

**Returns**

OS_WRAPPER_SUCCESS on success or OS_WRAPPER_ERROR on error

Definition at line 33 of file tfm_ns_interface_bare_metal.c.

## 7.37 interface/src/os_wrapper/tfm_ns_interface_rtos.c File Reference

```
#include <stdint.h>
#include "os_wrapper/mutex.h"
```

```
#include "os_wrapper/kernel.h"
#include "tfm_ns_interface.h"
```

Include dependency graph for tfm_ns_interface_rtos.c:

## Functions

- int32_t tfm_ns_interface_dispatch (veneer_fn fn, uint32_t arg0, uint32_t arg1, uint32_t arg2, uint32_t arg3)

  *NS interface, veneer function dispatcher.*
- uint32_t tfm_ns_interface_init (void)

  *NS interface initialization function.*

## 7.37.1 Function Documentation

### 7.37.1.1 tfm_ns_interface_dispatch()

```
int32_t tfm_ns_interface_dispatch (
            veneer_fn fn,
            uint32_t arg0,
            uint32_t arg1,
            uint32_t arg2,
            uint32_t arg3 )
```

NS interface, veneer function dispatcher.

This function implements the dispatching mechanism for the desired veneer function, to be called with the parameters described from arg0 to arg3.

**Note**

NSPE can use default implementation of this function or implement this function according to NS specific implementation and actual usage scenario.

**Parameters**

| | | |
|---|---|---|
| in | *fn* | Function pointer to the veneer function desired |
| in | *arg0* | Argument 0 of fn |
| in | *arg1* | Argument 1 of fn |

**Parameters**

| | | |
|---|---|---|
| in | *arg2* | Argument 2 of fn |
| in | *arg3* | Argument 3 of fn |

**Returns**

Returns the same return value of the requested veneer function

**Note**

This API must ensure the return value is from the veneer function. Other unrecoverable errors must be considered as fatal error and should not return.

Definition at line 34 of file tfm_ns_interface_rtos.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.37.1.2 tfm_ns_interface_init()

```
uint32_t tfm_ns_interface_init (
            void  )
```

NS interface initialization function.
This function initializes TF-M NS interface.

**Note**

NSPE can use default implementation of this function or implement this function according to NS specific implementation and actual usage scenario.

**Returns**

OS_WRAPPER_SUCCESS on success or OS_WRAPPER_ERROR on error

Definition at line 62 of file tfm_ns_interface_rtos.c.

## 7.38 interface/src/tfm_attest_api.c File Reference

```
#include "psa/initial_attestation.h"
#include "psa/client.h"
#include "psa_manifest/sid.h"
#include "tfm_attest_defs.h"
```

Include dependency graph for tfm_attest_api.c:

### Functions

- psa_status_t psa_initial_attest_get_token (const uint8_t *auth_challenge, size_t challenge_size, uint8_t *token_buf, size_t token_buf_size, size_t *token_size)
- psa_status_t psa_initial_attest_get_token_size (size_t challenge_size, size_t *token_size)

### 7.38.1 Function Documentation

#### 7.38.1.1 psa_initial_attest_get_token()

```
psa_status_t psa_initial_attest_get_token (
            const uint8_t * auth_challenge,
            size_t challenge_size,
            uint8_t * token_buf,
            size_t token_buf_size,
            size_t * token_size )
```

Definition at line 14 of file tfm_attest_api.c.

Here is the call graph for this function:

#### 7.38.1.2 psa_initial_attest_get_token_size()

```
psa_status_t psa_initial_attest_get_token_size (
            size_t challenge_size,
            size_t * token_size )
```

Definition at line 41 of file tfm_attest_api.c.
Here is the call graph for this function:

## 7.39 interface/src/tfm_crypto_api.c File Reference

```
#include <stdlib.h>
#include <string.h>
#include "tfm_crypto_defs.h"
#include "psa/client.h"
#include "psa_manifest/sid.h"
```

Include dependency graph for tfm_crypto_api.c:

## Macros

- #define API_DISPATCH(in_vec, out_vec)
- #define API_DISPATCH_NO_OUTVEC(in_vec)
- #define TFM_CRYPTO_API(ret, fun) ret fun

  *Define the function signature of a TF-M Crypto API with return type ret and PSA Crypto API function name fun.*

## Functions

- psa_status_t psa_crypto_init (void)
- psa_status_t psa_open_key (psa_key_id_t id, psa_key_id_t *key)
- psa_status_t psa_close_key (psa_key_id_t key)
- psa_status_t psa_import_key (const psa_key_attributes_t *attributes, const uint8_t *data, size_t data_length, psa_key_id_t *key)
- psa_status_t psa_destroy_key (psa_key_id_t key)
- psa_status_t psa_get_key_attributes (psa_key_id_t key, psa_key_attributes_t *attributes)
- psa_status_t psa_export_key (psa_key_id_t key, uint8_t *data, size_t data_size, size_t *data_length)
- psa_status_t psa_export_public_key (psa_key_id_t key, uint8_t *data, size_t data_size, size_t *data_length)
- psa_status_t psa_purge_key (psa_key_id_t key)
- psa_status_t psa_copy_key (psa_key_id_t source_key, const psa_key_attributes_t *attributes, psa_key_id↩_t *target_key)
- psa_status_t psa_cipher_generate_iv (psa_cipher_operation_t *operation, uint8_t *iv, size_t iv_size, size_t *iv_length)
- psa_status_t psa_cipher_set_iv (psa_cipher_operation_t *operation, const uint8_t *iv, size_t iv_length)
- psa_status_t psa_cipher_encrypt_setup (psa_cipher_operation_t *operation, psa_key_id_t key, psa_↩algorithm_t alg)
- psa_status_t psa_cipher_decrypt_setup (psa_cipher_operation_t *operation, psa_key_id_t key, psa_↩algorithm_t alg)
- psa_status_t psa_cipher_update (psa_cipher_operation_t *operation, const uint8_t *input, size_t input_↩length, uint8_t *output, size_t output_size, size_t *output_length)
- psa_status_t psa_cipher_abort (psa_cipher_operation_t *operation)
- psa_status_t psa_cipher_finish (psa_cipher_operation_t *operation, uint8_t *output, size_t output_size, size_t *output_length)
- psa_status_t psa_hash_setup (psa_hash_operation_t *operation, psa_algorithm_t alg)
- psa_status_t psa_hash_update (psa_hash_operation_t *operation, const uint8_t *input, size_t input_length)
- psa_status_t psa_hash_finish (psa_hash_operation_t *operation, uint8_t *hash, size_t hash_size, size_↩t *hash_length)

- psa_status_t psa_hash_verify (psa_hash_operation_t *operation, const uint8_t *hash, size_t hash_length)
- psa_status_t psa_hash_abort (psa_hash_operation_t *operation)
- psa_status_t psa_hash_clone (const psa_hash_operation_t *source_operation, psa_hash_operation_t *target_operation)
- psa_status_t psa_hash_compute (psa_algorithm_t alg, const uint8_t *input, size_t input_length, uint8_t *hash, size_t hash_size, size_t *hash_length)
- psa_status_t psa_hash_compare (psa_algorithm_t alg, const uint8_t *input, size_t input_length, const uint8_t *hash, size_t hash_length)
- psa_status_t psa_mac_sign_setup (psa_mac_operation_t *operation, psa_key_id_t key, psa_algorithm_t alg)
- psa_status_t psa_mac_verify_setup (psa_mac_operation_t *operation, psa_key_id_t key, psa_algorithm_t alg)
- psa_status_t psa_mac_update (psa_mac_operation_t *operation, const uint8_t *input, size_t input_length)
- psa_status_t psa_mac_sign_finish (psa_mac_operation_t *operation, uint8_t *mac, size_t mac_size, size_t *mac_length)
- psa_status_t psa_mac_verify_finish (psa_mac_operation_t *operation, const uint8_t *mac, size_t mac_length)
- psa_status_t psa_mac_abort (psa_mac_operation_t *operation)
- psa_status_t psa_aead_encrypt (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *nonce, size_t nonce_length, const uint8_t *additional_data, size_t additional_data_length, const uint8_t *plaintext, size_t plaintext_length, uint8_t *ciphertext, size_t ciphertext_size, size_t *ciphertext_length)
- psa_status_t psa_aead_decrypt (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *nonce, size_t nonce_length, const uint8_t *additional_data, size_t additional_data_length, const uint8_t *ciphertext, size_t ciphertext_length, uint8_t *plaintext, size_t plaintext_size, size_t *plaintext_length)
- psa_status_t psa_aead_encrypt_setup (psa_aead_operation_t *operation, psa_key_id_t key, psa_algorithm_t alg)
- psa_status_t psa_aead_decrypt_setup (psa_aead_operation_t *operation, psa_key_id_t key, psa_algorithm_t alg)
- psa_status_t psa_aead_generate_nonce (psa_aead_operation_t *operation, uint8_t *nonce, size_t nonce_size, size_t *nonce_length)
- psa_status_t psa_aead_set_nonce (psa_aead_operation_t *operation, const uint8_t *nonce, size_t nonce_length)
- psa_status_t psa_aead_set_lengths (psa_aead_operation_t *operation, size_t ad_length, size_t plaintext_length)
- psa_status_t psa_aead_update_ad (psa_aead_operation_t *operation, const uint8_t *input, size_t input_length)
- psa_status_t psa_aead_update (psa_aead_operation_t *operation, const uint8_t *input, size_t input_length, uint8_t *output, size_t output_size, size_t *output_length)
- psa_status_t psa_aead_finish (psa_aead_operation_t *operation, uint8_t *ciphertext, size_t ciphertext_size, size_t *ciphertext_length, uint8_t *tag, size_t tag_size, size_t *tag_length)
- psa_status_t psa_aead_verify (psa_aead_operation_t *operation, uint8_t *plaintext, size_t plaintext_size, size_t *plaintext_length, const uint8_t *tag, size_t tag_length)
- psa_status_t psa_aead_abort (psa_aead_operation_t *operation)
- psa_status_t psa_sign_message (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *input, size_t input_length, uint8_t *signature, size_t signature_size, size_t *signature_length)
- psa_status_t psa_verify_message (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *input, size_t input_length, const uint8_t *signature, size_t signature_length)
- psa_status_t psa_sign_hash (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *hash, size_t hash_length, uint8_t *signature, size_t signature_size, size_t *signature_length)
- psa_status_t psa_verify_hash (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *hash, size_t hash_length, const uint8_t *signature, size_t signature_length)
- psa_status_t psa_asymmetric_encrypt (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *input, size_t input_length, const uint8_t *salt, size_t salt_length, uint8_t *output, size_t output_size, size_t *output_length)
- psa_status_t psa_asymmetric_decrypt (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *input, size_t input_length, const uint8_t *salt, size_t salt_length, uint8_t *output, size_t output_size, size_t *output_length)

- psa_status_t psa_key_derivation_get_capacity (const psa_key_derivation_operation_t *operation, size_t *capacity)
- psa_status_t psa_key_derivation_output_bytes (psa_key_derivation_operation_t *operation, uint8_t *output, size_t output_length)
- psa_status_t psa_key_derivation_input_key (psa_key_derivation_operation_t *operation, psa_key_derivation_step_t step, psa_key_id_t key)
- psa_status_t psa_key_derivation_abort (psa_key_derivation_operation_t *operation)
- psa_status_t psa_key_derivation_key_agreement (psa_key_derivation_operation_t *operation, psa_key_derivation_step_t step, psa_key_id_t private_key, const uint8_t *peer_key, size_t peer_key_length)
- psa_status_t psa_generate_random (uint8_t *output, size_t output_size)
- psa_status_t psa_generate_key (const psa_key_attributes_t *attributes, psa_key_id_t *key)
- psa_status_t psa_mac_compute (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *input, size_t input_length, uint8_t *mac, size_t mac_size, size_t *mac_length)
- psa_status_t psa_mac_verify (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *input, size_t input_length, const uint8_t *mac, const size_t mac_length)
- psa_status_t psa_cipher_encrypt (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *input, size_t input_length, uint8_t *output, size_t output_size, size_t *output_length)
- psa_status_t psa_cipher_decrypt (psa_key_id_t key, psa_algorithm_t alg, const uint8_t *input, size_t input_length, uint8_t *output, size_t output_size, size_t *output_length)
- psa_status_t psa_raw_key_agreement (psa_algorithm_t alg, psa_key_id_t private_key, const uint8_t *peer_key, size_t peer_key_length, uint8_t *output, size_t output_size, size_t *output_length)
- psa_status_t psa_key_derivation_setup (psa_key_derivation_operation_t *operation, psa_algorithm_t alg)
- psa_status_t psa_key_derivation_set_capacity (psa_key_derivation_operation_t *operation, size_t capacity)
- psa_status_t psa_key_derivation_input_bytes (psa_key_derivation_operation_t *operation, psa_key_derivation_step_t step, const uint8_t *data, size_t data_length)
- psa_status_t psa_key_derivation_output_key (const psa_key_attributes_t *attributes, psa_key_derivation_operation_t *operation, psa_key_id_t *key)
- psa_status_t psa_key_derivation_input_integer (psa_key_derivation_operation_t *operation, psa_key_derivation_step_t step, uint64_t value)
- psa_status_t psa_key_derivation_verify_bytes (psa_key_derivation_operation_t *operation, const uint8_t *expected_output, size_t output_length)
- psa_status_t psa_key_derivation_verify_key (psa_key_derivation_operation_t *operation, psa_key_id_t expected)
- void psa_reset_key_attributes (psa_key_attributes_t *attributes)

## 7.39.1 Macro Definition Documentation

### 7.39.1.1 API_DISPATCH

```
#define API_DISPATCH(
            in_vec,
            out_vec )
```

**Value:**

```
psa_call(TFM_CRYPTO_HANDLE, PSA_IPC_CALL,  \
         in_vec, IOVEC_LEN(in_vec),        \
         out_vec, IOVEC_LEN(out_vec))
```

Definition at line 18 of file tfm_crypto_api.c.

### 7.39.1.2 API_DISPATCH_NO_OUTVEC

```
#define API_DISPATCH_NO_OUTVEC(
            in_vec )
```

**Value:**

```
psa_call(TFM_CRYPTO_HANDLE, PSA_IPC_CALL,  \
         in_vec, IOVEC_LEN(in_vec),        \
         (psa_outvec *)NULL, 0)
```

Definition at line 22 of file tfm_crypto_api.c.

#### 7.39.1.3 TFM_CRYPTO_API

```
#define TFM_CRYPTO_API(
            ret,
            fun ) ret fun
```

Define the function signature of a TF-M Crypto API with return type *ret* and PSA Crypto API function name *fun*.

**Parameters**

| | |
|---|---|
| *ret* | return type associated to the API |
| *fun* | API name (e.g. a PSA Crypto API function name) |

**Returns**

Function signature

Definition at line 57 of file tfm_crypto_api.c.

### 7.39.2 Function Documentation

#### 7.39.2.1 psa_aead_abort()

```
psa_status_t psa_aead_abort (
            psa_aead_operation_t * operation )
```

Definition at line 1080 of file tfm_crypto_api.c.

#### 7.39.2.2 psa_aead_decrypt()

```
psa_status_t psa_aead_decrypt (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * nonce,
            size_t nonce_length,
            const uint8_t * additional_data,
            size_t additional_data_length,
            const uint8_t * ciphertext,
            size_t ciphertext_length,
            uint8_t * plaintext,
            size_t plaintext_size,
            size_t * plaintext_length )
```

Definition at line 742 of file tfm_crypto_api.c.
Here is the call graph for this function:

#### 7.39.2.3 psa_aead_decrypt_setup()

```
psa_status_t psa_aead_decrypt_setup (
            psa_aead_operation_t * operation,
            psa_key_id_t key,
            psa_algorithm_t alg )
```

Definition at line 826 of file tfm_crypto_api.c.

#### 7.39.2.4 psa_aead_encrypt()

```
psa_status_t psa_aead_encrypt (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * nonce,
            size_t nonce_length,
            const uint8_t * additional_data,
            size_t additional_data_length,
            const uint8_t * plaintext,
            size_t plaintext_length,
            uint8_t * ciphertext,
            size_t ciphertext_size,
            size_t * ciphertext_length )
```

Definition at line 681 of file tfm_crypto_api.c.
Here is the call graph for this function:

#### 7.39.2.5 psa_aead_encrypt_setup()

```
psa_status_t psa_aead_encrypt_setup (
            psa_aead_operation_t * operation,
            psa_key_id_t key,
            psa_algorithm_t alg )
```

Definition at line 803 of file tfm_crypto_api.c.

#### 7.39.2.6 psa_aead_finish()

```
psa_status_t psa_aead_finish (
            psa_aead_operation_t * operation,
            uint8_t * ciphertext,
            size_t ciphertext_size,
            size_t * ciphertext_length,
            uint8_t * tag,
            size_t tag_size,
            size_t * tag_length )
```

Definition at line 980 of file tfm_crypto_api.c.

Here is the call graph for this function:

#### 7.39.2.7 psa_aead_generate_nonce()

```
psa_status_t psa_aead_generate_nonce (
            psa_aead_operation_t * operation,
            uint8_t * nonce,
            size_t nonce_size,
            size_t * nonce_length )
```

Definition at line 849 of file tfm_crypto_api.c.

#### 7.39.2.8 psa_aead_set_lengths()

```
psa_status_t psa_aead_set_lengths (
            psa_aead_operation_t * operation,
            size_t ad_length,
            size_t plaintext_length )
```

Definition at line 892 of file tfm_crypto_api.c.

#### 7.39.2.9 psa_aead_set_nonce()

```
psa_status_t psa_aead_set_nonce (
            psa_aead_operation_t * operation,
            const uint8_t * nonce,
            size_t nonce_length )
```

Definition at line 873 of file tfm_crypto_api.c.

#### 7.39.2.10 psa_aead_update()

```
psa_status_t psa_aead_update (
            psa_aead_operation_t * operation,
            const uint8_t * input,
            size_t input_length,
            uint8_t * output,
            size_t output_size,
            size_t * output_length )
```

Definition at line 942 of file tfm_crypto_api.c.

Here is the call graph for this function:

#### 7.39.2.11 psa_aead_update_ad()

```
psa_status_t psa_aead_update_ad (
            psa_aead_operation_t * operation,
            const uint8_t * input,
            size_t input_length )
```

Definition at line 912 of file tfm_crypto_api.c.
Here is the call graph for this function:

#### 7.39.2.12 psa_aead_verify()

```
psa_status_t psa_aead_verify (
            psa_aead_operation_t * operation,
            uint8_t * plaintext,
            size_t plaintext_size,
            size_t * plaintext_length,
            const uint8_t * tag,
            size_t tag_length )
```

Definition at line 1032 of file tfm_crypto_api.c.
Here is the call graph for this function:

#### 7.39.2.13 psa_asymmetric_decrypt()

```
psa_status_t psa_asymmetric_decrypt (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            const uint8_t * salt,
            size_t salt_length,
            uint8_t * output,
            size_t output_size,
            size_t * output_length )
```

Definition at line 1245 of file tfm_crypto_api.c.
Here is the call graph for this function:

#### 7.39.2.14 psa_asymmetric_encrypt()

```
psa_status_t psa_asymmetric_encrypt (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            const uint8_t * salt,
            size_t salt_length,
            uint8_t * output,
            size_t output_size,
            size_t * output_length )
```

Definition at line 1200 of file tfm_crypto_api.c.
Here is the call graph for this function:

#### 7.39.2.15 psa_cipher_abort()

```
psa_status_t psa_cipher_abort (
            psa_cipher_operation_t * operation )
```

Definition at line 343 of file tfm_crypto_api.c.

#### 7.39.2.16 psa_cipher_decrypt()

```
psa_status_t psa_cipher_decrypt (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            uint8_t * output,
            size_t output_size,
            size_t * output_length )
```

Definition at line 1511 of file tfm_crypto_api.c.

#### 7.39.2.17 psa_cipher_decrypt_setup()

```
psa_status_t psa_cipher_decrypt_setup (
            psa_cipher_operation_t * operation,
            psa_key_id_t key,
            psa_algorithm_t alg )
```

Definition at line 294 of file tfm_crypto_api.c.

#### 7.39.2.18 psa_cipher_encrypt()

```
psa_status_t psa_cipher_encrypt (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            uint8_t * output,
            size_t output_size,
            size_t * output_length )
```

Definition at line 1482 of file tfm_crypto_api.c.

#### 7.39.2.19 psa_cipher_encrypt_setup()

```
psa_status_t psa_cipher_encrypt_setup (
            psa_cipher_operation_t * operation,
            psa_key_id_t key,
            psa_algorithm_t alg )
```

Definition at line 273 of file tfm_crypto_api.c.

#### 7.39.2.20 psa_cipher_finish()

```
psa_status_t psa_cipher_finish (
            psa_cipher_operation_t * operation,
            uint8_t * output,
            size_t output_size,
            size_t * output_length )
```

Definition at line 360 of file tfm_crypto_api.c.

#### 7.39.2.21 psa_cipher_generate_iv()

```
psa_status_t psa_cipher_generate_iv (
            psa_cipher_operation_t * operation,
            uint8_t * iv,
            size_t iv_size,
            size_t * iv_length )
```

Definition at line 231 of file tfm_crypto_api.c.

#### 7.39.2.22 psa_cipher_set_iv()

```
psa_status_t psa_cipher_set_iv (
            psa_cipher_operation_t * operation,
            const uint8_t * iv,
            size_t iv_length )
```

Definition at line 256 of file tfm_crypto_api.c.

#### 7.39.2.23 psa_cipher_update()

```
psa_status_t psa_cipher_update (
            psa_cipher_operation_t * operation,
            const uint8_t * input,
            size_t input_length,
            uint8_t * output,
            size_t output_size,
            size_t * output_length )
```

Definition at line 315 of file tfm_crypto_api.c.

#### 7.39.2.24 psa_close_key()

```
psa_status_t psa_close_key (
            psa_key_id_t key )
```

Definition at line 85 of file tfm_crypto_api.c.

#### 7.39.2.25 psa_copy_key()

```
psa_status_t psa_copy_key (
            psa_key_id_t source_key,
            const psa_key_attributes_t * attributes,
            psa_key_id_t * target_key )
```

Definition at line 210 of file tfm_crypto_api.c.

#### 7.39.2.26 psa_crypto_init()

```
psa_status_t psa_crypto_init (
            void  )
```

Definition at line 60 of file tfm_crypto_api.c.

#### 7.39.2.27 psa_destroy_key()

```
psa_status_t psa_destroy_key (
            psa_key_id_t key )
```

Definition at line 118 of file tfm_crypto_api.c.

#### 7.39.2.28 psa_export_key()

```
psa_status_t psa_export_key (
            psa_key_id_t key,
            uint8_t * data,
            size_t data_size,
            size_t * data_length )
```

Definition at line 148 of file tfm_crypto_api.c.

#### 7.39.2.29 psa_export_public_key()

```
psa_status_t psa_export_public_key (
            psa_key_id_t key,
            uint8_t * data,
            size_t data_size,
            size_t * data_length )
```

Definition at line 172 of file tfm_crypto_api.c.

#### 7.39.2.30 psa_generate_key()

```
psa_status_t psa_generate_key (
            const psa_key_attributes_t * attributes,
            psa_key_id_t * key )
```

Definition at line 1412 of file tfm_crypto_api.c.

#### 7.39.2.31 psa_generate_random()

```
psa_status_t psa_generate_random (
            uint8_t * output,
            size_t output_size )
```

Definition at line 1390 of file tfm_crypto_api.c.
Here is the caller graph for this function:

#### 7.39.2.32 psa_get_key_attributes()

```
psa_status_t psa_get_key_attributes (
            psa_key_id_t key,
            psa_key_attributes_t * attributes )
```

Definition at line 131 of file tfm_crypto_api.c.

#### 7.39.2.33 psa_hash_abort()

```
psa_status_t psa_hash_abort (
            psa_hash_operation_t * operation )
```

Definition at line 468 of file tfm_crypto_api.c.

#### 7.39.2.34 psa_hash_clone()

```
psa_status_t psa_hash_clone (
            const psa_hash_operation_t * source_operation,
            psa_hash_operation_t * target_operation )
```

Definition at line 485 of file tfm_crypto_api.c.

#### 7.39.2.35 psa_hash_compare()

```
psa_status_t psa_hash_compare (
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            const uint8_t * hash,
            size_t hash_length )
```

Definition at line 539 of file tfm_crypto_api.c.

#### 7.39.2.36 psa_hash_compute()

```
psa_status_t psa_hash_compute (
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            uint8_t * hash,
            size_t hash_size,
            size_t * hash_length )
```

Definition at line 510 of file tfm_crypto_api.c.

#### 7.39.2.37 psa_hash_finish()

```
psa_status_t psa_hash_finish (
            psa_hash_operation_t * operation,
            uint8_t * hash,
            size_t hash_size,
            size_t * hash_length )
```

Definition at line 422 of file tfm_crypto_api.c.

#### 7.39.2.38 psa_hash_setup()

```
psa_status_t psa_hash_setup (
            psa_hash_operation_t * operation,
            psa_algorithm_t alg )
```

Definition at line 386 of file tfm_crypto_api.c.

#### 7.39.2.39 psa_hash_update()

```
psa_status_t psa_hash_update (
            psa_hash_operation_t * operation,
            const uint8_t * input,
            size_t input_length )
```

Definition at line 405 of file tfm_crypto_api.c.

#### 7.39.2.40 psa_hash_verify()

```
psa_status_t psa_hash_verify (
            psa_hash_operation_t * operation,
            const uint8_t * hash,
            size_t hash_length )
```

Definition at line 448 of file tfm_crypto_api.c.

#### 7.39.2.41 psa_import_key()

```
psa_status_t psa_import_key (
            const psa_key_attributes_t * attributes,
            const uint8_t * data,
            size_t data_length,
            psa_key_id_t * key )
```

Definition at line 98 of file tfm_crypto_api.c.

#### 7.39.2.42 psa_key_derivation_abort()

```
psa_status_t psa_key_derivation_abort (
            psa_key_derivation_operation_t * operation )
```

Definition at line 1350 of file tfm_crypto_api.c.

#### 7.39.2.43 psa_key_derivation_get_capacity()

```
psa_status_t psa_key_derivation_get_capacity (
            const psa_key_derivation_operation_t * operation,
            size_t * capacity )
```

Definition at line 1290 of file tfm_crypto_api.c.

#### 7.39.2.44 psa_key_derivation_input_bytes()

```
psa_status_t psa_key_derivation_input_bytes (
            psa_key_derivation_operation_t * operation,
            psa_key_derivation_step_t step,
            const uint8_t * data,
            size_t data_length )
```

Definition at line 1607 of file tfm_crypto_api.c.
Here is the caller graph for this function:

#### 7.39.2.45 psa_key_derivation_input_integer()

```
psa_status_t psa_key_derivation_input_integer (
            psa_key_derivation_operation_t * operation,
            psa_key_derivation_step_t step,
            uint64_t value )
```

Definition at line 1649 of file tfm_crypto_api.c.

#### 7.39.2.46 psa_key_derivation_input_key()

```
psa_status_t psa_key_derivation_input_key (
            psa_key_derivation_operation_t * operation,
            psa_key_derivation_step_t step,
            psa_key_id_t key )
```

Definition at line 1331 of file tfm_crypto_api.c.
Here is the caller graph for this function:

#### 7.39.2.47 psa_key_derivation_key_agreement()

```
psa_status_t psa_key_derivation_key_agreement (
            psa_key_derivation_operation_t * operation,
            psa_key_derivation_step_t step,
            psa_key_id_t private_key,
            const uint8_t * peer_key,
            size_t peer_key_length )
```

Definition at line 1368 of file tfm_crypto_api.c.

#### 7.39.2.48 psa_key_derivation_output_bytes()

```
psa_status_t psa_key_derivation_output_bytes (
            psa_key_derivation_operation_t * operation,
            uint8_t * output,
            size_t output_length )
```

Definition at line 1310 of file tfm_crypto_api.c.

#### 7.39.2.49 psa_key_derivation_output_key()

```
psa_status_t psa_key_derivation_output_key (
            const psa_key_attributes_t * attributes,
            psa_key_derivation_operation_t * operation,
            psa_key_id_t * key )
```

Definition at line 1627 of file tfm_crypto_api.c.

#### 7.39.2.50 psa_key_derivation_set_capacity()

```
psa_status_t psa_key_derivation_set_capacity (
            psa_key_derivation_operation_t * operation,
            size_t capacity )
```

Definition at line 1590 of file tfm_crypto_api.c.

#### 7.39.2.51 psa_key_derivation_setup()

```
psa_status_t psa_key_derivation_setup (
            psa_key_derivation_operation_t * operation,
            psa_algorithm_t alg )
```

Definition at line 1571 of file tfm_crypto_api.c.
Here is the caller graph for this function:

#### 7.39.2.52 psa_key_derivation_verify_bytes()

```
psa_status_t psa_key_derivation_verify_bytes (
            psa_key_derivation_operation_t * operation,
            const uint8_t * expected_output,
            size_t output_length )
```

Definition at line 1668 of file tfm_crypto_api.c.

#### 7.39.2.53 psa_key_derivation_verify_key()

```
psa_status_t psa_key_derivation_verify_key (
            psa_key_derivation_operation_t * operation,
            psa_key_id_t expected )
```

Definition at line 1677 of file tfm_crypto_api.c.

#### 7.39.2.54 psa_mac_abort()

```
psa_status_t psa_mac_abort (
            psa_mac_operation_t * operation )
```

Definition at line 664 of file tfm_crypto_api.c.

#### 7.39.2.55 psa_mac_compute()

```
psa_status_t psa_mac_compute (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
```

```
            uint8_t * mac,
            size_t mac_size,
            size_t * mac_length )
```

Definition at line 1431 of file tfm_crypto_api.c.

#### 7.39.2.56 psa_mac_sign_finish()

```
psa_status_t psa_mac_sign_finish (
            psa_mac_operation_t * operation,
            uint8_t * mac,
            size_t mac_size,
            size_t * mac_length )
```

Definition at line 618 of file tfm_crypto_api.c.

#### 7.39.2.57 psa_mac_sign_setup()

```
psa_status_t psa_mac_sign_setup (
            psa_mac_operation_t * operation,
            psa_key_id_t key,
            psa_algorithm_t alg )
```

Definition at line 559 of file tfm_crypto_api.c.

#### 7.39.2.58 psa_mac_update()

```
psa_status_t psa_mac_update (
            psa_mac_operation_t * operation,
            const uint8_t * input,
            size_t input_length )
```

Definition at line 601 of file tfm_crypto_api.c.

#### 7.39.2.59 psa_mac_verify()

```
psa_status_t psa_mac_verify (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            const uint8_t * mac,
            const size_t mac_length )
```

Definition at line 1460 of file tfm_crypto_api.c.

#### 7.39.2.60 psa_mac_verify_finish()

```
psa_status_t psa_mac_verify_finish (
            psa_mac_operation_t * operation,
            const uint8_t * mac,
            size_t mac_length )
```

Definition at line 644 of file tfm_crypto_api.c.

#### 7.39.2.61 psa_mac_verify_setup()

```
psa_status_t psa_mac_verify_setup (
            psa_mac_operation_t * operation,
```

```
            psa_key_id_t key,
            psa_algorithm_t alg )
```

Definition at line 580 of file tfm_crypto_api.c.

#### 7.39.2.62 psa_open_key()

```
psa_status_t psa_open_key (
            psa_key_id_t id,
            psa_key_id_t * key )
```

Definition at line 68 of file tfm_crypto_api.c.

#### 7.39.2.63 psa_purge_key()

```
psa_status_t psa_purge_key (
            psa_key_id_t key )
```

Definition at line 197 of file tfm_crypto_api.c.

#### 7.39.2.64 psa_raw_key_agreement()

```
psa_status_t psa_raw_key_agreement (
            psa_algorithm_t alg,
            psa_key_id_t private_key,
            const uint8_t * peer_key,
            size_t peer_key_length,
            uint8_t * output,
            size_t output_size,
            size_t * output_length )
```

Definition at line 1540 of file tfm_crypto_api.c.

#### 7.39.2.65 psa_reset_key_attributes()

```
void psa_reset_key_attributes (
            psa_key_attributes_t * attributes )
```

Definition at line 1690 of file tfm_crypto_api.c.
Here is the call graph for this function:

#### 7.39.2.66 psa_sign_hash()

```
psa_status_t psa_sign_hash (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * hash,
            size_t hash_length,
```

```
            uint8_t * signature,
            size_t signature_size,
            size_t * signature_length )
```

Definition at line 1148 of file tfm_crypto_api.c.

#### 7.39.2.67 psa_sign_message()

```
psa_status_t psa_sign_message (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            uint8_t * signature,
            size_t signature_size,
            size_t * signature_length )
```

Definition at line 1097 of file tfm_crypto_api.c.

#### 7.39.2.68 psa_verify_hash()

```
psa_status_t psa_verify_hash (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * hash,
            size_t hash_length,
            const uint8_t * signature,
            size_t signature_length )
```

Definition at line 1178 of file tfm_crypto_api.c.

#### 7.39.2.69 psa_verify_message()

```
psa_status_t psa_verify_message (
            psa_key_id_t key,
            psa_algorithm_t alg,
            const uint8_t * input,
            size_t input_length,
            const uint8_t * signature,
            size_t signature_length )
```

Definition at line 1126 of file tfm_crypto_api.c.

## 7.40 interface/src/tfm_fwu_api.c File Reference

```
#include "psa/client.h"
#include "psa/update.h"
#include "psa_manifest/sid.h"
```

Include dependency graph for tfm_fwu_api.c:

## Functions

- psa_status_t psa_fwu_start (psa_fwu_component_t component, const void *manifest, size_t manifest_size)

  *Begin a firmware update operation for a specific firmware component.*
- psa_status_t psa_fwu_write (psa_fwu_component_t component, size_t image_offset, const void *block, size_t block_size)

  *Write a firmware image, or part of a firmware image, to its staging area.*
- psa_status_t psa_fwu_finish (psa_fwu_component_t component)

  *Mark a firmware image in the staging area as ready for installation.*
- psa_status_t psa_fwu_install (void)

  *Start the installation of all firmware images that have been prepared for update.*
- psa_status_t psa_fwu_cancel (psa_fwu_component_t component)

  *Abandon an update that is in WRITING or CANDIDATE state.*
- psa_status_t psa_fwu_clean (psa_fwu_component_t component)

  *Prepare the component for another update.*
- psa_status_t psa_fwu_query (psa_fwu_component_t component, psa_fwu_component_info_t *info)

  *Retrieve the firmware store information for a specific firmware component.*
- psa_status_t psa_fwu_request_reboot (void)

  *Requests the platform to reboot.*
- psa_status_t psa_fwu_accept (void)

  *Accept a firmware update that is currently in TRIAL state.*
- psa_status_t psa_fwu_reject (psa_status_t error)

  *Abandon an installation that is in STAGED or TRIAL state.*

### 7.40.1 Function Documentation

#### 7.40.1.1 psa_fwu_accept()

```
psa_status_t psa_fwu_accept (
            void  )
```

Accept a firmware update that is currently in TRIAL state.

**Returns**

Result status.

Definition at line 96 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.2 psa_fwu_cancel()

```
psa_status_t psa_fwu_cancel (
            psa_fwu_component_t component )
```

Abandon an update that is in WRITING or CANDIDATE state.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component to be cancelled. |

**Returns**

Result status.

Definition at line 56 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.3 psa_fwu_clean()

```
psa_status_t psa_fwu_clean (
            psa_fwu_component_t component )
```

Prepare the component for another update.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component to tidy up. |

**Returns**

Result status.

Definition at line 66 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.4 psa_fwu_finish()

```
psa_status_t psa_fwu_finish (
            psa_fwu_component_t component )
```

Mark a firmware image in the staging area as ready for installation.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component to install. |

**Returns**

Result status.

Definition at line 40 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.5 psa_fwu_install()

```
psa_status_t psa_fwu_install (
            void )
```

Start the installation of all firmware images that have been prepared for update.

**Returns**

Result status.

Definition at line 50 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.6 psa_fwu_query()

```
psa_status_t psa_fwu_query (
            psa_fwu_component_t component,
            psa_fwu_component_info_t * info )
```

Retrieve the firmware store information for a specific firmware component.

**Parameters**

| | |
|---|---|
| *component* | Firmware component for which information is requested. |
| *info* | Output parameter for component information. |

**Returns**

Result status.

Definition at line 76 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.7 psa_fwu_reject()

```
psa_status_t psa_fwu_reject (
            psa_status_t error )
```

Abandon an installation that is in STAGED or TRIAL state.

**Parameters**

| | |
|---|---|
| *error* | An application-specific error code chosen by the application. |

**Returns**

Result status.

Definition at line 102 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.8 psa_fwu_request_reboot()

```
psa_status_t psa_fwu_request_reboot (
            void )
```

Requests the platform to reboot.

**Returns**

Result status.

Definition at line 90 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.9 psa_fwu_start()

```
psa_status_t psa_fwu_start (
            psa_fwu_component_t component,
            const void * manifest,
            size_t manifest_size )
```

Begin a firmware update operation for a specific firmware component.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component to be updated. |
| *manifest* | A pointer to a buffer containing a detached manifest for the update. |
| *manifest_size* | The size of the detached manifest. |

**Returns**

Result status.

Definition at line 12 of file tfm_fwu_api.c.
Here is the call graph for this function:

#### 7.40.1.10 psa_fwu_write()

```
psa_status_t psa_fwu_write (
            psa_fwu_component_t component,
            size_t image_offset,
            const void * block,
            size_t block_size )
```

Write a firmware image, or part of a firmware image, to its staging area.

**Parameters**

| | |
|---|---|
| *component* | Identifier of the firmware component being updated. |
| *image_offset* | The offset of the data block in the whole image. |
| *block* | A buffer containing a block of image data. |
| *block_size* | Size of block, in bytes. |

**Returns**

Result status.

Definition at line 25 of file tfm_fwu_api.c.
Here is the call graph for this function:

## 7.41 interface/src/tfm_its_api.c File Reference

```
#include "psa/client.h"
#include "psa/internal_trusted_storage.h"
#include "psa_manifest/sid.h"
```

`#include "tfm_its_defs.h"`
Include dependency graph for tfm_its_api.c:

## Functions

- psa_status_t psa_its_set (psa_storage_uid_t uid, size_t data_length, const void ∗p_data, psa_storage_create_flags_t create_flags)

  *Create a new, or modify an existing, uid/value pair.*
- psa_status_t psa_its_get (psa_storage_uid_t uid, size_t data_offset, size_t data_size, void ∗p_data, size_t ∗p_data_length)

  *Retrieve data associated with a provided UID.*
- psa_status_t psa_its_get_info (psa_storage_uid_t uid, struct psa_storage_info_t ∗p_info)

  *Retrieve the metadata about the provided uid.*
- psa_status_t psa_its_remove (psa_storage_uid_t uid)

  *Remove the provided uid and its associated data from the storage.*

## 7.41.1 Function Documentation

### 7.41.1.1 psa_its_get()

```
psa_status_t psa_its_get (
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_size,
            void * p_data,
            size_t * p_data_length )
```

Retrieve data associated with a provided UID.
Retrieves up to `data_size` bytes of the data associated with `uid`, starting at `data_offset` bytes from the beginning of the data. Upon successful completion, the data will be placed in the `p_data` buffer, which must be at least `data_size` bytes in size. The length of the data returned will be in `p_data_length`. If `data_size` is 0, the contents of `p_data_length` will be set to zero.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The uid value |
| in | *data_offset* | The starting offset of the data requested |
| in | *data_size* | The amount of data requested |
| out | *p_data* | On success, the buffer where the data will be placed |

**Parameters**

| | | |
|---|---|---|
| out | *p_data_length* | On success, this will contain size of the data placed in `p_data` |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided `uid` value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided arguments (`p_data`, `p_data_length`) is invalid, for example is `NULL` or references memory the caller cannot access. In addition, this can also happen if `data_offset` is larger than the size of the data associated with `uid` |

Definition at line 32 of file tfm_its_api.c.
Here is the call graph for this function:

### 7.41.1.2 psa_its_get_info()

```
psa_status_t psa_its_get_info (
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Retrieve the metadata about the provided uid.
Retrieves the metadata stored for a given `uid` as a `psa_storage_info_t` structure.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |
| out | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_info`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 61 of file tfm_its_api.c.
Here is the call graph for this function:

#### 7.41.1.3 psa_its_remove()

```
psa_status_t psa_its_remove (
            psa_storage_uid_t uid )
```

Remove the provided uid and its associated data from the storage.
Deletes the data from internal storage.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, wrong flags and so on) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |

Definition at line 81 of file tfm_its_api.c.

Here is the call graph for this function:

#### 7.41.1.4 psa_its_set()

```
psa_status_t psa_its_set (
            psa_storage_uid_t uid,
            size_t data_length,
            const void * p_data,
            psa_storage_create_flags_t create_flags )
```

Create a new, or modify an existing, uid/value pair.
Stores data in the internal storage.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The identifier for the data |
| in | *data_length* | The size in bytes of the data in `p_data` |
| in | *p_data* | A buffer containing the data |
| in | *create_flags* | The flags that the data will be stored with |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided `uid` value was already created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because one or more of the flags provided in `create_flags` is not supported or is not valid |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because there was insufficient space on the storage medium |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_data`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 13 of file tfm_its_api.c.

Here is the call graph for this function:

## 7.42 interface/src/tfm_platform_api.c File Reference

```
#include <stdbool.h>
#include "tfm_platform_api.h"
#include "psa/client.h"
#include "psa_manifest/sid.h"
```

Include dependency graph for tfm_platform_api.c:

### Functions

- enum tfm_platform_err_t tfm_platform_system_reset (void)

  *Resets the system.*
- enum tfm_platform_err_t tfm_platform_ioctl (tfm_platform_ioctl_req_t request, psa_invec *input, psa_outvec *output)

  *Performs a platform-specific service.*
- enum tfm_platform_err_t tfm_platform_nv_counter_increment (uint32_t counter_id)

  *Increments the given non-volatile (NV) counter by one.*
- enum tfm_platform_err_t tfm_platform_nv_counter_read (uint32_t counter_id, uint32_t size, uint8_t *val)

  *Reads the given non-volatile (NV) counter.*

## 7.42.1 Function Documentation

### 7.42.1.1 tfm_platform_ioctl()

```
enum tfm_platform_err_t tfm_platform_ioctl (
            tfm_platform_ioctl_req_t request,
            psa_invec * input,
            psa_outvec * output )
```

Performs a platform-specific service.

**Parameters**

| | | |
|---|---|---|
| in | *request* | Request identifier (valid values vary based on the platform) |
| in | *input* | Input buffer to the requested service (or NULL) |
| in,out | *output* | Output buffer to the requested service (or NULL) |

**Returns**

Returns values as specified by the tfm_platform_err_t

Definition at line 30 of file tfm_platform_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.42.1.2 tfm_platform_nv_counter_increment()

```
enum tfm_platform_err_t tfm_platform_nv_counter_increment (
            uint32_t counter_id )
```

Increments the given non-volatile (NV) counter by one.

**Parameters**

| | | |
|---|---|---|
| in | *counter↩ _id* | NV counter ID. |

**Returns**

TFM_PLATFORM_ERR_SUCCESS if the value is read correctly. Otherwise, it returns TFM_PLATFORM_↩ ERR_SYSTEM_ERROR.

Definition at line 67 of file tfm_platform_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.42.1.3 tfm_platform_nv_counter_read()

```
enum tfm_platform_err_t tfm_platform_nv_counter_read (
            uint32_t counter_id,
            uint32_t size,
            uint8_t * val )
```

Reads the given non-volatile (NV) counter.

**Parameters**

| | | |
|---|---|---|
| in | *counter↩ _id* | NV counter ID. |
| in | *size* | Size of the buffer to store NV counter value in bytes. |
| out | *val* | Pointer to store the current NV counter value. |

**Returns**

TFM_PLATFORM_ERR_SUCCESS if the value is read correctly. Otherwise, it returns TFM_PLATFORM_↩ ERR_SYSTEM_ERROR.

Definition at line 87 of file tfm_platform_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.42.1.4 tfm_platform_system_reset()

```
enum tfm_platform_err_t tfm_platform_system_reset (
            void  )
```

Resets the system.

**Returns**

Returns values as specified by the tfm_platform_err_t

Definition at line 13 of file tfm_platform_api.c.
Here is the call graph for this function:

## 7.43 interface/src/tfm_ps_api.c File Reference

```
#include "psa/client.h"
#include "psa/protected_storage.h"
```

```
#include "psa_manifest/sid.h"
#include "tfm_ps_defs.h"
```

Include dependency graph for tfm_ps_api.c:

## Functions

- psa_status_t psa_ps_set (psa_storage_uid_t uid, size_t data_length, const void *p_data, psa_storage_create_flags_t create_flags)

  *Create a new, or modify an existing, uid/value pair.*
- psa_status_t psa_ps_get (psa_storage_uid_t uid, size_t data_offset, size_t data_size, void *p_data, size_t *p_data_length)

  *Retrieve data associated with a provided uid.*
- psa_status_t psa_ps_get_info (psa_storage_uid_t uid, struct psa_storage_info_t *p_info)

  *Retrieve the metadata about the provided uid.*
- psa_status_t psa_ps_remove (psa_storage_uid_t uid)

  *Remove the provided uid and its associated data from the storage.*
- psa_status_t psa_ps_create (psa_storage_uid_t uid, size_t size, psa_storage_create_flags_t create_flags)

  *Reserves storage for the specified uid.*
- psa_status_t psa_ps_set_extended (psa_storage_uid_t uid, size_t data_offset, size_t data_length, const void *p_data)

  *Sets partial data into an asset.*
- uint32_t psa_ps_get_support (void)

  *Lists optional features.*

## 7.43.1 Function Documentation

### 7.43.1.1 psa_ps_create()

```
psa_status_t psa_ps_create (
            psa_storage_uid_t uid,
            size_t capacity,
            psa_storage_create_flags_t create_flags )
```

Reserves storage for the specified uid.

Upon success, the capacity of the storage will be capacity, and the size will be 0. It is only necessary to call this function for assets that will be written with the psa_ps_set_extended function. If only the psa_ps_set function is needed, calls to this function are redundant.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |
| in | *capacity* | The capacity to be allocated in bytes |
| in | *create_flags* | Flags indicating properties of storage |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because the capacity is bigger than the current available space |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because the function is not implemented or one or more create_flags are not supported. |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because uid was 0 or create_flags specified flags that are not defined in the API. |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed due to an unspecified error |
| *PSA_ERROR_ALREADY_EXISTS* | Storage for the specified uid already exists |

Definition at line 94 of file tfm_ps_api.c.

#### 7.43.1.2 psa_ps_get()

```
psa_status_t psa_ps_get (
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_size,
            void * p_data,
            size_t * p_data_length )
```

Retrieve data associated with a provided uid.

Retrieves up to `data_size` bytes of the data associated with `uid`, starting at `data_offset` bytes from the beginning of the data. Upon successful completion, the data will be placed in the `p_data` buffer, which must be at least `data_size` bytes in size. The length of the data returned will be in `p_data_length`. If `data_size` is 0, the contents of `p_data_length` will be set to zero.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The uid value |
| in | *data_offset* | The starting offset of the data requested |
| in | *data_size* | The amount of data requested |
| out | *p_data* | On success, the buffer where the data will be placed |
| out | *p_data_length* | On success, this will contain size of the data placed in `p_data` |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided arguments (`p_data`, `p_data_length`) is invalid, for example is `NULL` or references memory the caller cannot access. In addition, this can also happen if `data_offset` is larger than the size of the data associated with `uid` |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided `uid` value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the data associated with the UID was corrupt |
| *PSA_ERROR_INVALID_SIGNATURE* | The operation failed because the data associated with the UID failed authentication |

Definition at line 32 of file tfm_ps_api.c.
Here is the call graph for this function:

### 7.43.1.3 psa_ps_get_info()

```
psa_status_t psa_ps_get_info (
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Retrieve the metadata about the provided uid.
Retrieves the metadata stored for a given `uid`

**Parameters**

| | | |
|---|---|---|
| `in` | *uid* | The `uid` value |
| `out` | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_info`) is invalid, for example is `NULL` or references memory the caller cannot access |

**Return values**

| | |
|---|---|
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the data associated with the UID was corrupt |

Definition at line 61 of file tfm_ps_api.c.
Here is the call graph for this function:

#### 7.43.1.4 psa_ps_get_support()

```
uint32_t psa_ps_get_support (
            void  )
```

Lists optional features.

**Returns**

A bitmask with flags set for all of the optional features supported by the implementation.Currently defined flags are limited to PSA_STORAGE_SUPPORT_SET_EXTENDED

Definition at line 115 of file tfm_ps_api.c.
Here is the call graph for this function:

#### 7.43.1.5 psa_ps_remove()

```
psa_status_t psa_ps_remove (
            psa_storage_uid_t uid )
```

Remove the provided uid and its associated data from the storage.
Removes previously stored data and any associated metadata, including rollback protection data.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, wrong flags and so on) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 80 of file tfm_ps_api.c.
Here is the call graph for this function:

### 7.43.1.6 psa_ps_set()

```
psa_status_t psa_ps_set (
            psa_storage_uid_t uid,
            size_t data_length,
            const void * p_data,
            psa_storage_create_flags_t create_flags )
```

Create a new, or modify an existing, uid/value pair.
Stores data in the protected storage.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The identifier for the data |
| in | *data_length* | The size in bytes of the data in `p_data` |
| in | *p_data* | A buffer containing the data |
| in | *create_flags* | The flags that the data will be stored with |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided `uid` value was already created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_data`) is invalid, for example is `NULL` or references memory the caller cannot access |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because one or more of the flags provided in `create_flags` is not supported or is not valid |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because there was insufficient space on the storage medium |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 13 of file tfm_ps_api.c.
Here is the call graph for this function:

#### 7.43.1.7 psa_ps_set_extended()

```
psa_status_t psa_ps_set_extended (
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_length,
            const void * p_data )
```

Sets partial data into an asset.
Before calling this function, the storage must have been reserved with a call to psa_ps_create. It can also be used to overwrite data in an asset that was created with a call to psa_ps_set. Calling this function with data_length = 0 is permitted, which will make no change to the stored data.This function can overwrite existing data and/or extend it up to the capacity for the uid specified in psa_ps_create, but cannot create gaps.
That is, it has preconditions:

- data_offset <= size
- data_offset + data_length <= capacity and postconditions:
- size = max(size, data_offset + data_length)
- capacity unchanged.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |
| in | *data_offset* | Offset within the asset to start the write |
| in | *data_length* | The size in bytes of the data in p_data to write |
| in | *p_data* | Pointer to a buffer which contains the data to write |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The asset exists, the input parameters are correct and the data is correctly written in the physical storage. |
| *PSA_ERROR_STORAGE_FAILURE* | The data was not written correctly in the physical storage |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the preconditions listed above regarding data_offset, size, or data_length was violated. |
| *PSA_ERROR_DOES_NOT_EXIST* | The specified uid was not found |
| *PSA_ERROR_NOT_SUPPORTED* | The implementation of the API does not support this function |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed due to an unspecified error |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the existing data has been corrupted. |
| *PSA_ERROR_INVALID_SIGNATURE* | The operation failed because the existing data failed authentication (MAC check failed). |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because it was attempted on an asset which was written with the flag PSA_STORAGE_FLAG_WRITE_ONCE |

Definition at line 104 of file tfm_ps_api.c.

## 7.44 interface/src/tfm_psa_call.c File Reference

```
#include <stdint.h>
#include "psa/client.h"
#include "psa/service.h"
#include "tfm_psa_call_pack.h"
```

Include dependency graph for tfm_psa_call.c:

## Functions

- psa_status_t psa_call (psa_handle_t handle, int32_t type, const psa_invec *in_vec, size_t in_len, psa_outvec *out_vec, size_t out_len)

  *Call an RoT Service on an established connection.*

## 7.44.1 Function Documentation

### 7.44.1.1 psa_call()

```
psa_status_t psa_call (
            psa_handle_t handle,
            int32_t type,
            const psa_invec * in_vec,
            size_t in_len,
            psa_outvec * out_vec,
            size_t out_len )
```

Call an RoT Service on an established connection.

**Note**

FF-M 1.0 proposes 6 parameters for psa_call but the secure gateway ABI support at most 4 parameters. T↩ F-M chooses to encode 'in_len', 'out_len', and 'type' into a 32-bit integer to improve efficiency. Compared with struct-based encoding, this method saves extra memory check and memory copy operation. The disadvantage is that the 'type' range has to be reduced into a 16-bit integer. So with this encoding, the valid range for 'type' is 0-32767.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | A handle to an established connection. |
| in | *type* | The request type. Must be zero( PSA_IPC_CALL) or positive. |
| in | *in_vec* | Array of input psa_invec structures. |
| in | *in_len* | Number of input psa_invec structures. |
| in,out | *out_vec* | Array of output psa_outvec structures. |
| in | *out_len* | Number of output psa_outvec structures. |

**Return values**

| | |
|---|---|
| *>=0* | RoT Service-specific status value. |
| *<0* | RoT Service-specific error code. |
| *PSA_ERROR_PROGRAMMER_ERROR* | The connection has been terminated by the RoT Service. The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• An invalid handle was passed.<br>• The connection is already handling a request.<br>• type < 0.<br>• An invalid memory reference was provided.<br>• in_len + out_len > PSA_MAX_IOVEC.<br>• The message is unrecognized by the RoT Service or incorrectly formatted. |

Definition at line 13 of file tfm_psa_call.c.

## 7.45 interface/src/tfm_tz_psa_ns_api.c File Reference

```
#include "psa/client.h"
#include "tfm_ns_interface.h"
#include "tfm_psa_call_pack.h"
```

Include dependency graph for tfm_tz_psa_ns_api.c:

## Functions

- uint32_t psa_framework_version (void)

  *Retrieve the version of the PSA Framework API that is implemented.*

- uint32_t psa_version (uint32_t sid)

  *Retrieve the version of an RoT Service or indicate that it is not present on this system.*

- psa_status_t psa_call (psa_handle_t handle, int32_t type, const psa_invec ∗in_vec, size_t in_len, psa_outvec ∗out_vec, size_t out_len)

  *Call an RoT Service on an established connection.*

- psa_handle_t psa_connect (uint32_t sid, uint32_t version)

  *Connect to an RoT Service by its SID.*

- void psa_close (psa_handle_t handle)

  *Close a connection to an RoT Service.*

### 7.45.1 Function Documentation

#### 7.45.1.1 psa_call()

```
psa_status_t psa_call (
            psa_handle_t handle,
            int32_t type,
            const psa_invec * in_vec,
            size_t in_len,
            psa_outvec * out_vec,
            size_t out_len )
```

Call an RoT Service on an established connection.

**Note**

FF-M 1.0 proposes 6 parameters for psa_call but the secure gateway ABI support at most 4 parameters. TF-M chooses to encode 'in_len', 'out_len', and 'type' into a 32-bit integer to improve efficiency. Compared with struct-based encoding, this method saves extra memory check and memory copy operation. The disadvantage is that the 'type' range has to be reduced into a 16-bit integer. So with this encoding, the valid range for 'type' is 0-32767.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | A handle to an established connection. |
| in | *type* | The request type. Must be zero( PSA_IPC_CALL) or positive. |
| in | *in_vec* | Array of input psa_invec structures. |
| in | *in_len* | Number of input psa_invec structures. |
| in,out | *out_vec* | Array of output psa_outvec structures. |
| in | *out_len* | Number of output psa_outvec structures. |

**Return values**

| | |
|---|---|
| *>=0* | RoT Service-specific status value. |
| *<0* | RoT Service-specific error code. |
| *PSA_ERROR_PROGRAMMER_ERROR* | The connection has been terminated by the RoT Service. The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• An invalid handle was passed.<br>• The connection is already handling a request.<br>• type < 0.<br>• An invalid memory reference was provided.<br>• in_len + out_len > PSA_MAX_IOVEC.<br>• The message is unrecognized by the RoT Service or incorrectly formatted. |

Definition at line 34 of file tfm_tz_psa_ns_api.c.

Here is the caller graph for this function:

### 7.45.1.2 psa_close()

```
void psa_close (
            psa_handle_t handle )
```

Close a connection to an RoT Service.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | A handle to an established connection, or the null handle. |

**Return values**

| | |
|---|---|
| *void* | Success. |
| *PROGRAMMER ERROR* | The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• An invalid handle was provided that is not the null handle.<br>• The connection is currently handling a request. |

Definition at line 60 of file tfm_tz_psa_ns_api.c.

#### 7.45.1.3 psa_connect()

```
psa_handle_t psa_connect (
            uint32_t sid,
            uint32_t version )
```

Connect to an RoT Service by its SID.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of the RoT Service to connect to. |
| in | *version* | Requested version of the RoT Service. |

**Return values**

| | |
|---|---|
| > | 0 A handle for the connection. |
| *PSA_ERROR_CONNECTION_REFUSED* | The SPM or RoT Service has refused the connection. |
| *PSA_ERROR_CONNECTION_BUSY* | The SPM or RoT Service cannot make the connection at the moment. |
| *PROGRAMMER ERROR* | The call is a PROGRAMMER ERROR if one or more of the following are true:<br>• The RoT Service ID is not present.<br>• The RoT Service version is not supported.<br>• The caller is not allowed to access the RoT service. |

Definition at line 55 of file tfm_tz_psa_ns_api.c.

Here is the call graph for this function:

#### 7.45.1.4 psa_framework_version()

```
uint32_t psa_framework_version (
            void  )
```

Retrieve the version of the PSA Framework API that is implemented.

**Returns**

version The version of the PSA Framework implementation that is providing the runtime services to the caller. The major and minor version are encoded as follows:

- version[15:8] – major version number.
- version[7:0] – minor version number.

Definition at line 14 of file tfm_tz_psa_ns_api.c.
Here is the call graph for this function:

#### 7.45.1.5 psa_version()

```
uint32_t psa_version (
            uint32_t sid )
```

Retrieve the version of an RoT Service or indicate that it is not present on this system.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of the RoT Service to query. |

**Return values**

| | |
|---|---|
| *PSA_VERSION_NONE* | The RoT Service is not implemented, or the caller is not permitted to access the service. |

**Return values**

| | |
|---|---|
| > | 0 The version of the implemented RoT Service. |

Definition at line 24 of file tfm_tz_psa_ns_api.c.
Here is the call graph for this function:

## 7.46 platform/ext/target/infineon/common/device/include/device_definition.h File Reference

```
#include "device_cfg.h"
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for device_definition.h:

This graph shows which files directly or indirectly include this file:

## 7.47 platform/ext/target/infineon/common/device/include/ifx_startup.h File Reference

```
#include "static_checks.h"
```

Include dependency graph for ifx_startup.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IS_POWER_OF_TWO(x) ((x) && !((x) & ((x) - 1)))
- #define IFX_VECTOR_TABLE_ATTRIBUTE __attribute__((aligned(VECTORTABLE_ALIGN))) __VECTO↩R_TABLE_ATTRIBUTE
- #define DEFAULT_IRQ_HANDLER(handler_name) __NO_RETURN void handler_name(void) __attribute↩__ ((weak, alias("Default_Handler")));

## Typedefs

- typedef void(∗ VECTOR_TABLE_Type) (void)

## Variables

- const VECTOR_TABLE_Type __vector_table [VECTORTABLE_SIZE]

### 7.47.1 Macro Definition Documentation

#### 7.47.1.1 DEFAULT_IRQ_HANDLER

```
#define DEFAULT_IRQ_HANDLER(
            handler_name ) __NO_RETURN void handler_name(void) __attribute__ ((weak, alias("Default↩
_Handler")));
```

Definition at line 61 of file ifx_startup.h.

#### 7.47.1.2 IFX_VECTOR_TABLE_ATTRIBUTE

```
#define IFX_VECTOR_TABLE_ATTRIBUTE __attribute__((aligned(VECTORTABLE_ALIGN))) __VECTOR_TABL↩
E_ATTRIBUTE
```

Definition at line 39 of file ifx_startup.h.

#### 7.47.1.3 IS_POWER_OF_TWO

```
#define IS_POWER_OF_TWO(
              x ) ((x) && !((x) & ((x) - 1)))
```

Definition at line 15 of file ifx_startup.h.

### 7.47.2 Typedef Documentation

#### 7.47.2.1 VECTOR_TABLE_Type

```
typedef void(* VECTOR_TABLE_Type) (void)
```

Definition at line 35 of file ifx_startup.h.

### 7.47.3 Variable Documentation

#### 7.47.3.1 __vector_table

```
const VECTOR_TABLE_Type __vector_table[VECTORTABLE_SIZE]
```

## 7.48 platform/ext/target/infineon/common/device/src/device_definition.c File Reference

This file defines exports the structures based on the peripheral definitions from device_cfg.h. This retarget file is meant to be used as a helper for baremetal applications and/or as an example of how to configure the generic driver structures.

```
#include "device_definition.h"
#include "static_checks.h"
#include "tfm_peripherals_def.h"
#include "tfm_plat_defs.h"
```

Include dependency graph for device_definition.c:

### 7.48.1 Detailed Description

This file defines exports the structures based on the peripheral definitions from device_cfg.h. This retarget file is meant to be used as a helper for baremetal applications and/or as an example of how to configure the generic driver structures.

## 7.49 platform/ext/target/infineon/common/drivers/assets/ifx_assets_↩ rram.c File Reference

```
#include "cy_pdl.h"
#include "ifx_assets_rram.h"
#include "ifx_driver_rram.h"
#include "ifx_se_crc32.h"
```

Include dependency graph for ifx_assets_rram.c:

### Functions

- int32_t ifx_assets_rram_read_block (uint32_t address, void ∗data, uint32_t length)
- int32_t ifx_assets_rram_read_asset (uint32_t address, void ∗asset, size_t size)

### 7.49.1 Function Documentation

#### 7.49.1.1 ifx_assets_rram_read_asset()

```
int32_t ifx_assets_rram_read_asset (
            uint32_t address,
            void * asset,
            size_t size )
```

Definition at line 40 of file ifx_assets_rram.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.49.1.2 ifx_assets_rram_read_block()

```
int32_t ifx_assets_rram_read_block (
            uint32_t address,
            void * data,
            uint32_t length )
```

Definition at line 15 of file ifx_assets_rram.c.
Here is the caller graph for this function:

## 7.50 platform/ext/target/infineon/common/drivers/assets/ifx_assets_rram.h File Reference

```
#include "Driver_Common.h"
#include <stdint.h>
#include <stddef.h>
```

Include dependency graph for ifx_assets_rram.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_ASSETS_RRAM_CHECKSUM_SIZE (4)

## Functions

- int32_t ifx_assets_rram_read_block (uint32_t address, void *data, uint32_t length)
- int32_t ifx_assets_rram_read_asset (uint32_t address, void *asset, size_t size)

### 7.50.1 Macro Definition Documentation

#### 7.50.1.1 IFX_ASSETS_RRAM_CHECKSUM_SIZE

```
#define IFX_ASSETS_RRAM_CHECKSUM_SIZE (4)
```

Definition at line 18 of file ifx_assets_rram.h.

## 7.50.2 Function Documentation

### 7.50.2.1 ifx_assets_rram_read_asset()

```
int32_t ifx_assets_rram_read_asset (
            uint32_t address,
            void * asset,
            size_t size )
```

Definition at line 40 of file ifx_assets_rram.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.50.2.2 ifx_assets_rram_read_block()

```
int32_t ifx_assets_rram_read_block (
            uint32_t address,
            void * data,
            uint32_t length )
```

Definition at line 15 of file ifx_assets_rram.c.
Here is the caller graph for this function:

## 7.51 platform/ext/target/infineon/common/drivers/flash/flash/ifx_driver_flash.c File Reference

```
#include "ifx_driver_flash.h"
#include "ifx_driver_private.h"
#include "ifx_utils.h"
#include "static_checks.h"
#include <string.h>
```

Include dependency graph for ifx_driver_flash.c:

### Macros

- #define IFX_DRIVER_FLASH_VERSION ARM_DRIVER_VERSION_MAJOR_MINOR(0x1U, 0x0U)
- #define IFX_DRIVER_FLASH_EVENT ifx_flash_driver_event_t
- #define IFX_DRIVER_FLASH_FUNCTION_ARGUMENTS(...) ifx_flash_driver_handler_t handler __VA_OPT__(,) __VA_ARGS__
- #define IFX_DRIVER_FLASH_FUNCTION_PARAMETERS(...) handler __VA_OPT__(,) __VA_ARGS__
- #define IFX_DRIVER_FLASH_INSTANCE (handler)
- #define IFX_UNUSED_FLASH_DRIVER_INSTANCE IFX_UNUSED(handler)
- #define IFX_FDF_ARGS(...) IFX_DRIVER_FLASH_FUNCTION_ARGUMENTS(__VA_ARGS__)
- #define IFX_FDF_PARAMS(...) IFX_DRIVER_FLASH_FUNCTION_PARAMETERS(__VA_ARGS__)

### Variables

- const struct ifx_flash_driver_t ifx_driver_flash

### 7.51.1 Macro Definition Documentation

#### 7.51.1.1 IFX_DRIVER_FLASH_EVENT

```
#define IFX_DRIVER_FLASH_EVENT ifx_flash_driver_event_t
```

Definition at line 35 of file ifx_driver_flash.c.

#### 7.51.1.2 IFX_DRIVER_FLASH_FUNCTION_ARGUMENTS

```
#define IFX_DRIVER_FLASH_FUNCTION_ARGUMENTS(
            ...  ) ifx_flash_driver_handler_t handler __VA_OPT__(,) __VA_ARGS__
```

Definition at line 42 of file ifx_driver_flash.c.

#### 7.51.1.3 IFX_DRIVER_FLASH_FUNCTION_PARAMETERS

```
#define IFX_DRIVER_FLASH_FUNCTION_PARAMETERS(
                ...  ) handler __VA_OPT__(,) __VA_ARGS__
```

Definition at line 43 of file ifx_driver_flash.c.

#### 7.51.1.4 IFX_DRIVER_FLASH_INSTANCE

```
#define IFX_DRIVER_FLASH_INSTANCE (handler)
```

Definition at line 47 of file ifx_driver_flash.c.

#### 7.51.1.5 IFX_DRIVER_FLASH_VERSION

```
#define IFX_DRIVER_FLASH_VERSION ARM_DRIVER_VERSION_MAJOR_MINOR(0x1U, 0x0U)
```

Definition at line 22 of file ifx_driver_flash.c.

#### 7.51.1.6 IFX_FDF_ARGS

```
#define IFX_FDF_ARGS(
                ...  ) IFX_DRIVER_FLASH_FUNCTION_ARGUMENTS(__VA_ARGS__)
```

Definition at line 52 of file ifx_driver_flash.c.

#### 7.51.1.7 IFX_FDF_PARAMS

```
#define IFX_FDF_PARAMS(
                ...  ) IFX_DRIVER_FLASH_FUNCTION_PARAMETERS(__VA_ARGS__)
```

Definition at line 54 of file ifx_driver_flash.c.

#### 7.51.1.8 IFX_UNUSED_FLASH_DRIVER_INSTANCE

```
#define IFX_UNUSED_FLASH_DRIVER_INSTANCE IFX_UNUSED(handler)
```

Definition at line 48 of file ifx_driver_flash.c.

### 7.51.2 Variable Documentation

#### 7.51.2.1 ifx_driver_flash

```
const struct ifx_flash_driver_t ifx_driver_flash
```

**Initial value:**

```
= {
    .GetVersion = ifx_driver_flash_get_version,
    .GetCapabilities = ifx_driver_flash_get_capabilities,
    .Initialize = ifx_driver_flash_initialize,
    .Uninitialize = ifx_driver_flash_uninitialize,
    .PowerControl = ifx_driver_flash_power_control,
    .ReadData = ifx_driver_flash_read_data,
    .ProgramData = ifx_driver_flash_program_data,
    .EraseSector = ifx_driver_flash_erase_sector,
    .EraseChip = ifx_driver_flash_erase_chip,
    .GetStatus = ifx_driver_flash_get_status,
    .GetInfo = ifx_driver_flash_get_info,
}
```

Definition at line 308 of file ifx_driver_flash.c.

## 7.52 platform/ext/target/infineon/common/drivers/flash/flash/ifx_driver_flash.h File Reference

```
#include "ifx_flash_driver_api.h"
#include <stdint.h>
#include "cy_flash.h"
```

Include dependency graph for ifx_driver_flash.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct ifx_driver_flash_obj_t

  *Structure containing the FLASH driver instance parameters.*

### Macros

- #define IFX_DRIVER_FLASH_ERASE_VALUE (0xFFU)

- #define IFX_DRIVER_FLASH_PROGRAM_UNIT (1U) /∗ 1 B ∗/
- #define IFX_DRIVER_FLASH_SECTOR_SIZE CY_FLASH_SIZEOF_ROW /∗ 512 B ∗/
- #define IFX_DRIVER_FLASH_CREATE_INSTANCE(instance_name, instance_obj)

## Variables

- const struct ifx_flash_driver_t ifx_driver_flash

### 7.52.1 Macro Definition Documentation

#### 7.52.1.1 IFX_DRIVER_FLASH_CREATE_INSTANCE

```
#define IFX_DRIVER_FLASH_CREATE_INSTANCE(
                instance_name,
                instance_obj )
```

**Value:**

```
    /* Use variable to validate instance_obj type */ \
    static const ifx_flash_driver_instance_t instance_name = { \
        .handler = &(instance_obj), \
        .driver = &ifx_driver_flash, \
    };
```

Definition at line 59 of file ifx_driver_flash.h.

#### 7.52.1.2 IFX_DRIVER_FLASH_ERASE_VALUE

```
#define IFX_DRIVER_FLASH_ERASE_VALUE (0xFFU)
```

Definition at line 21 of file ifx_driver_flash.h.

#### 7.52.1.3 IFX_DRIVER_FLASH_PROGRAM_UNIT

```
#define IFX_DRIVER_FLASH_PROGRAM_UNIT (1U) /* 1 B */
```

Definition at line 22 of file ifx_driver_flash.h.

#### 7.52.1.4 IFX_DRIVER_FLASH_SECTOR_SIZE

```
#define IFX_DRIVER_FLASH_SECTOR_SIZE CY_FLASH_SIZEOF_ROW /* 512 B */
```

Definition at line 23 of file ifx_driver_flash.h.

### 7.52.2 Variable Documentation

#### 7.52.2.1 ifx_driver_flash

```
const struct ifx_flash_driver_t ifx_driver_flash
```

Definition at line 308 of file ifx_driver_flash.c.

## 7.53 platform/ext/target/infineon/common/drivers/flash/ifx_driver_private.c File Reference

```
#include "ifx_driver_private.h"
```

Include dependency graph for ifx_driver_private.c:

## Functions

- int32_t ifx_flash_driver_validate_region (ARM_FLASH_INFO *flash_info, uint32_t address, uint32_t size)

  *Validate that memory block is inside of flash driver instance region.*
- int32_t ifx_flash_driver_initialize (ARM_FLASH_INFO *flash_info, uint32_t start_address, uint32_t mem_↩ block_size)

  *Initialize Flash driver instance.*

## 7.53.1 Function Documentation

### 7.53.1.1 ifx_flash_driver_initialize()

```
int32_t ifx_flash_driver_initialize (
            ARM_FLASH_INFO * flash_info,
            uint32_t start_address,
            uint32_t mem_block_size )
```

Initialize Flash driver instance.

**Parameters**

| | | |
|---|---|---|
| in | *flash_info* | Flash driver info |
| in | *start_address* | Flash region start address |
| in | *mem_block_size* | Size of flash memory block |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Success. |

**Return values**

| | |
|---|---|
| *Other* | ARM_DRIVER_ERROR_∗ Failed |

Definition at line 34 of file ifx_driver_private.c.

#### 7.53.1.2 ifx_flash_driver_validate_region()

```
int32_t ifx_flash_driver_validate_region (
            ARM_FLASH_INFO * flash_info,
            uint32_t address,
            uint32_t size )
```

Validate that memory block is inside of flash driver instance region.

**Parameters**

| | | |
|---|---|---|
| in | *flash_info* | Flash driver info |
| in | *address* | Start address of memory block |
| in | *size* | Memory block size |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Memory block is inside of flash driver instance region. |
| *ARM_DRIVER_ERROR_PARAMETER* | Memory block is outside of flash driver instance region. |

**Note**

instance settings are not validated by this function!

Definition at line 11 of file ifx_driver_private.c.

## 7.54 platform/ext/target/infineon/common/drivers/flash/ifx_driver_private.h File Reference

```
#include "ifx_flash_driver_api.h"
```

Include dependency graph for ifx_driver_private.h:

This graph shows which files directly or indirectly include this file:

## Functions

- int32_t ifx_flash_driver_validate_region (ARM_FLASH_INFO *flash_info, uint32_t address, uint32_t size)

  *Validate that memory block is inside of flash driver instance region.*
- int32_t ifx_flash_driver_initialize (ARM_FLASH_INFO *flash_info, uint32_t start_address, uint32_t mem_↩ block_size)

  *Initialize Flash driver instance.*

## 7.54.1 Function Documentation

### 7.54.1.1 ifx_flash_driver_initialize()

```
int32_t ifx_flash_driver_initialize (
            ARM_FLASH_INFO * flash_info,
            uint32_t start_address,
            uint32_t mem_block_size )
```

Initialize Flash driver instance.

**Parameters**

| | | |
|---|---|---|
| in | *flash_info* | Flash driver info |
| in | *start_address* | Flash region start address |
| in | *mem_block_size* | Size of flash memory block |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Success. |
| *Other* | ARM_DRIVER_ERROR_∗ Failed |

Definition at line 34 of file ifx_driver_private.c.

#### 7.54.1.2 ifx_flash_driver_validate_region()

```
int32_t ifx_flash_driver_validate_region (
            ARM_FLASH_INFO * flash_info,
            uint32_t address,
            uint32_t size )
```

Validate that memory block is inside of flash driver instance region.

**Parameters**

| | | |
|---|---|---|
| in | *flash_info* | Flash driver info |
| in | *address* | Start address of memory block |
| in | *size* | Memory block size |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Memory block is inside of flash driver instance region. |
| *ARM_DRIVER_ERROR_PARAMETER* | Memory block is outside of flash driver instance region. |

**Note**

instance settings are not validated by this function!

Definition at line 11 of file ifx_driver_private.c.

## 7.55 platform/ext/target/infineon/common/drivers/flash/ifx_flash_driver↩ _api.h File Reference

```
#include "Driver_Flash.h"
```

Include dependency graph for ifx_flash_driver_api.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct ifx_flash_driver_t

  *Access structure of the Flash Driver.*
- struct ifx_flash_driver_instance_t

  *Flash driver instance.*

## Macros

- #define IFX_CREATE_CMSIS_FLASH_DRIVER(instance, cmsis_driver)

## Typedefs

- typedef void(∗ ifx_flash_driver_event_t) (ifx_flash_driver_handler_t handler, uint32_t event)

  *Signal Flash event.*
- typedef struct ifx_flash_driver_t ifx_flash_driver_t

  *Access structure of the Flash Driver.*
- typedef struct ifx_flash_driver_instance_t ifx_flash_driver_instance_t

  *Flash driver instance.*

## Variables

- const typedef void ∗ ifx_flash_driver_handler_t

  *Driver instance handler.*

### 7.55.1 Macro Definition Documentation

#### 7.55.1.1 IFX_CREATE_CMSIS_FLASH_DRIVER

```
#define IFX_CREATE_CMSIS_FLASH_DRIVER(
            instance,
            cmsis_driver )
```

Create CMSIS flash driver wrapper for TF-M compatible implementation
instance - TF-M flash driver instance (ifx_flash_driver_instance_t) cmsis_driver - CMSIS flash driver that is created (ARM_DRIVER_FLASH)
Definition at line 89 of file ifx_flash_driver_api.h.

### 7.55.2 Typedef Documentation

#### 7.55.2.1 ifx_flash_driver_event_t

```
typedef void(* ifx_flash_driver_event_t) (ifx_flash_driver_handler_t handler, uint32_t event)
```

Signal Flash event.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Driver instance handler |
| in | *event* | Event notification mask, see ARM_Flash_SignalEvent_t |

Definition at line 31 of file ifx_flash_driver_api.h.

#### 7.55.2.2 ifx_flash_driver_instance_t

```
typedef struct ifx_flash_driver_instance_t ifx_flash_driver_instance_t
```

Flash driver instance.
You can create a separate driver instance for same flash chip. But each instance can handle different areas of flash memory. This allows to provide isolation of data on driver level too.

#### 7.55.2.3 ifx_flash_driver_t

```
typedef struct ifx_flash_driver_t ifx_flash_driver_t
```

Access structure of the Flash Driver.
Interface is similar to CMSIS flash driver interface (ARM_DRIVER_FLASH) except that there is an additional driver instance handler (see ifx_flash_driver_instance_t::handler) which is specific to driver implementation.

### 7.55.3 Variable Documentation

#### 7.55.3.1 ifx_flash_driver_handler_t

```
const typedef void* ifx_flash_driver_handler_t
```

Driver instance handler.
This type is used to hide internal implementation of driver instance data.
Definition at line 24 of file ifx_flash_driver_api.h.

## 7.56 platform/ext/target/infineon/common/drivers/flash/rram/ifx_driver_rram.c File Reference

```
#include "config_tfm.h"
#include "fih.h"
```

```
#include "ifx_driver_private.h"
#include "ifx_driver_rram.h"
#include "ifx_utils.h"
#include "static_checks.h"
#include <string.h>
#include "cy_device.h"
#include "cy_rram.h"
```

Include dependency graph for ifx_driver_rram.c:

## Macros

- #define IFX_DRIVER_RRAM_VERSION ARM_DRIVER_VERSION_MAJOR_MINOR(1UL, 0U)

## Functions

- TFM_COVERITY_DEVIATE_LINE (MISRA_C_2023_Rule_2_8, "Used in RRAM flash driver in the IFX_DRIVER_RRAM_CREATE_INSTANCE macro")

## Variables

- const struct ifx_flash_driver_t ifx_driver_rram

## 7.56.1 Macro Definition Documentation

### 7.56.1.1 IFX_DRIVER_RRAM_VERSION

```
#define IFX_DRIVER_RRAM_VERSION ARM_DRIVER_VERSION_MAJOR_MINOR(1UL, 0U)
```

Definition at line 26 of file ifx_driver_rram.c.

## 7.56.2 Function Documentation

### 7.56.2.1 TFM_COVERITY_DEVIATE_LINE()

```
TFM_COVERITY_DEVIATE_LINE (
            MISRA_C_2023_Rule_2_8 ,
            "Used in RRAM flash driver in the IFX_DRIVER_RRAM_CREATE_INSTANCE macro"  )
```

Here is the caller graph for this function:

## 7.56.3 Variable Documentation

### 7.56.3.1 ifx_driver_rram

```
const struct ifx_flash_driver_t ifx_driver_rram
```

**Initial value:**

```
= {
    .GetVersion = ifx_driver_rram_get_version,
    .GetCapabilities = ifx_driver_rram_get_capabilities,
    .Initialize = ifx_driver_rram_initialize,
```

```
    .Uninitialize = ifx_driver_rram_uninitialize,
    .PowerControl = ifx_driver_rram_power_control,
    .ReadData = ifx_driver_rram_read_data,
    .ProgramData = ifx_driver_rram_program_data,
    .EraseSector = ifx_driver_rram_erase_sector,
    .EraseChip = ifx_driver_rram_erase_chip,
    .GetStatus = ifx_driver_rram_get_status,
    .GetInfo = ifx_driver_rram_get_info,
}
```

Definition at line 287 of file ifx_driver_rram.c.

## 7.57 platform/ext/target/infineon/common/drivers/flash/rram/ifx_driver_rram.h File Reference

```
#include "ifx_flash_driver_api.h"
#include "ifx_s_peripherals_def.h"
#include <stdint.h>
#include "cy_rram.h"
```

Include dependency graph for ifx_driver_rram.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct ifx_driver_rram_obj_t

  *Structure containing the RRAM driver instance parameters.*

### Macros

- #define IFX_DRIVER_RRAM_ERASE_VALUE (0x0UL)
- #define IFX_DRIVER_RRAM_PROGRAM_UNIT (4U)
- #define IFX_DRIVER_RRAM_PC_LOCK_RETRIES (100)
- #define IFX_DRIVER_RRAM_CREATE_INSTANCE(instance_name, instance_obj)

## Typedefs

- typedef struct ifx_driver_rram_obj_t ifx_driver_rram_obj_t

  *Structure containing the RRAM driver instance parameters.*

## Variables

- const struct ifx_flash_driver_t ifx_driver_rram

## 7.57.1 Macro Definition Documentation

### 7.57.1.1 IFX_DRIVER_RRAM_CREATE_INSTANCE

```
#define IFX_DRIVER_RRAM_CREATE_INSTANCE(
            instance_name,
            instance_obj )
```

**Value:**

```
/* Use variable to validate instance_obj type */ \
const ifx_flash_driver_instance_t instance_name = { \
    .handler = &(instance_obj), \
    .driver = &ifx_driver_rram, \
};
```

Definition at line 59 of file ifx_driver_rram.h.

### 7.57.1.2 IFX_DRIVER_RRAM_ERASE_VALUE

```
#define IFX_DRIVER_RRAM_ERASE_VALUE (0x0UL)
```

Definition at line 25 of file ifx_driver_rram.h.

### 7.57.1.3 IFX_DRIVER_RRAM_PC_LOCK_RETRIES

```
#define IFX_DRIVER_RRAM_PC_LOCK_RETRIES (100)
```

Definition at line 29 of file ifx_driver_rram.h.

### 7.57.1.4 IFX_DRIVER_RRAM_PROGRAM_UNIT

```
#define IFX_DRIVER_RRAM_PROGRAM_UNIT (4U)
```

Definition at line 26 of file ifx_driver_rram.h.

## 7.57.2 Typedef Documentation

### 7.57.2.1 ifx_driver_rram_obj_t

```
typedef struct ifx_driver_rram_obj_t ifx_driver_rram_obj_t
```

Structure containing the RRAM driver instance parameters.
RRAM flash driver uses start_address, flash_info.sector_size and flash_info.sector_count to define a working region that is handled by active instance. Any access outside of this region is invalid and RRAM flash driver returns AR↩M_DRIVER_ERROR_PARAMETER in such cases.
ifx_driver_rram interface expects address to be relative to start_address.

**Note**

This structure can be a constant.

### 7.57.3 Variable Documentation

#### 7.57.3.1 ifx_driver_rram

```
const struct ifx_flash_driver_t ifx_driver_rram
```

Definition at line 287 of file ifx_driver_rram.c.

## 7.58 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif.c File Reference

```
#include "ifx_driver_smif.h"
#include "ifx_driver_smif_private.h"
#include <string.h>
#include "cy_device.h"
```

Include dependency graph for ifx_driver_smif.c:

### Macros

- #define IFX_DRIVER_SMIF_VERSION ARM_DRIVER_VERSION_MAJOR_MINOR(1UL, 0U)
- #define IFX_SMIF_SHORT_POLLING_RETRIES (100)

### Functions

- int32_t ifx_driver_smif_poll_block_busy (const ifx_driver_smif_obj_t *obj)

  *This function checks if the status of the SMIF block is busy.*
- int32_t ifx_driver_smif_set_mode (const ifx_driver_smif_obj_t *obj, cy_en_smif_mode_t mode)

  *Switch SMIF controller mode.*
- int32_t ifx_driver_smif_validate_region (const ifx_driver_smif_obj_t *obj, uint32_t address, uint32_t size)

  *Validate that memory block is inside of flash driver instance region.*
- ARM_DRIVER_VERSION ifx_driver_smif_get_version (ifx_flash_driver_handler_t handler)
- ARM_FLASH_CAPABILITIES ifx_driver_smif_get_capabilities (ifx_flash_driver_handler_t handler)
- int32_t ifx_driver_smif_initialize (ifx_flash_driver_handler_t handler, ifx_flash_driver_event_t cb_event)

  *Initialize SMIF flash driver instance.*
- int32_t ifx_driver_smif_uninitialize (ifx_flash_driver_handler_t handler)

- int32_t ifx_driver_smif_power_control (ifx_flash_driver_handler_t handler, ARM_POWER_STATE state)
- int32_t ifx_driver_smif_erase_sector (ifx_flash_driver_handler_t handler, uint32_t addr)

  *Erase sector of SMIF flash driver instance.*
- int32_t ifx_driver_smif_erase_chip (ifx_flash_driver_handler_t handler)

  *Erase memory region allocated for SMIF flash driver instance.*
- struct _ARM_FLASH_STATUS ifx_driver_smif_get_status (ifx_flash_driver_handler_t handler)
- ARM_FLASH_INFO * ifx_driver_smif_get_info (ifx_flash_driver_handler_t handler)

## 7.58.1 Macro Definition Documentation

### 7.58.1.1 IFX_DRIVER_SMIF_VERSION

```
#define IFX_DRIVER_SMIF_VERSION ARM_DRIVER_VERSION_MAJOR_MINOR(1UL, 0U)
```

Definition at line 22 of file ifx_driver_smif.c.

### 7.58.1.2 IFX_SMIF_SHORT_POLLING_RETRIES

```
#define IFX_SMIF_SHORT_POLLING_RETRIES (100)
```

Definition at line 26 of file ifx_driver_smif.c.

## 7.58.2 Function Documentation

### 7.58.2.1 ifx_driver_smif_erase_chip()

```
int32_t ifx_driver_smif_erase_chip (
            ifx_flash_driver_handler_t handler )
```

Erase memory region allocated for SMIF flash driver instance.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Flash driver handler data |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Success. |
| *Other* | ARM_DRIVER_ERROR_* Failed |

Definition at line 290 of file ifx_driver_smif.c.
Here is the call graph for this function:

#### 7.58.2.2 ifx_driver_smif_erase_sector()

```
int32_t ifx_driver_smif_erase_sector (
            ifx_flash_driver_handler_t handler,
            uint32_t addr )
```

Erase sector of SMIF flash driver instance.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Flash driver handler data |
| in | *addr* | Sector address, must be aligned to sector. |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Success. |
| *Other* | ARM_DRIVER_ERROR_∗ Failed |

Definition at line 241 of file ifx_driver_smif.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.58.2.3 ifx_driver_smif_get_capabilities()

```
ARM_FLASH_CAPABILITIES ifx_driver_smif_get_capabilities (
            ifx_flash_driver_handler_t handler )
```

Definition at line 128 of file ifx_driver_smif.c.

#### 7.58.2.4 ifx_driver_smif_get_info()

```
ARM_FLASH_INFO* ifx_driver_smif_get_info (
            ifx_flash_driver_handler_t handler )
```

Definition at line 323 of file ifx_driver_smif.c.
Here is the call graph for this function:

#### 7.58.2.5 ifx_driver_smif_get_status()

```
struct _ARM_FLASH_STATUS ifx_driver_smif_get_status (
            ifx_flash_driver_handler_t handler )
```

Definition at line 310 of file ifx_driver_smif.c.

#### 7.58.2.6 ifx_driver_smif_get_version()

```
ARM_DRIVER_VERSION ifx_driver_smif_get_version (
            ifx_flash_driver_handler_t handler )
```

Definition at line 111 of file ifx_driver_smif.c.
Here is the call graph for this function:

#### 7.58.2.7 ifx_driver_smif_initialize()

```
int32_t ifx_driver_smif_initialize (
            ifx_flash_driver_handler_t handler,
            ifx_flash_driver_event_t cb_event )
```

Initialize SMIF flash driver instance.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Flash driver handler data |
| in | *cb_event* | Callback event is not supported. Must be NULL. |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Success. |
| *Other* | ARM_DRIVER_ERROR_∗ Failed |

Definition at line 162 of file ifx_driver_smif.c.
Here is the call graph for this function:

#### 7.58.2.8 ifx_driver_smif_poll_block_busy()

```
int32_t ifx_driver_smif_poll_block_busy (
            const ifx_driver_smif_obj_t * obj )
```

This function checks if the status of the SMIF block is busy.

**Parameters**

| | | |
|---|---|---|
| in | *obj* | Flash driver instance data |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Memory block is not busy. |
| *ARM_DRIVER_ERROR_BUSY* | Memory block is busy. |

Definition at line 39 of file ifx_driver_smif.c.
Here is the caller graph for this function:

#### 7.58.2.9 ifx_driver_smif_power_control()

```
int32_t ifx_driver_smif_power_control (
            ifx_flash_driver_handler_t handler,
            ARM_POWER_STATE state )
```

Definition at line 233 of file ifx_driver_smif.c.

#### 7.58.2.10 ifx_driver_smif_set_mode()

```
int32_t ifx_driver_smif_set_mode (
            const ifx_driver_smif_obj_t * obj,
            cy_en_smif_mode_t mode )
```

Switch SMIF controller mode.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Flash driver handler data |
| in | *mode* | SMIF controller mode. |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Memory block is inside of flash driver instance region. |
| *ARM_DRIVER_ERROR_BUSY* | Memory block is outside of flash driver instance region. |

Definition at line 59 of file ifx_driver_smif.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.58.2.11 ifx_driver_smif_uninitialize()

```
int32_t ifx_driver_smif_uninitialize (
            ifx_flash_driver_handler_t handler )
```

Definition at line 226 of file ifx_driver_smif.c.

#### 7.58.2.12 ifx_driver_smif_validate_region()

```
int32_t ifx_driver_smif_validate_region (
            const ifx_driver_smif_obj_t * obj,
            uint32_t address,
            uint32_t size )
```

Validate that memory block is inside of flash driver instance region.

**Parameters**

| | | |
|---|---|---|
| in | *obj* | Flash driver instance data |
| in | *address* | Start address of memory block |
| in | *size* | Memory block size |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Memory block is inside of flash driver instance region. |
| *ARM_DRIVER_ERROR_PARAMETER* | Memory block is outside of flash driver instance region. |

**Note**

instance settings are not validated by this function!

Definition at line 76 of file ifx_driver_smif.c.
Here is the caller graph for this function:

## 7.59 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif.h File Reference

```
#include <stdint.h>
#include "ifx_flash_driver_api.h"
#include "cy_smif_memslot.h"
#include "static_checks.h"
```

Include dependency graph for ifx_driver_smif.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct ifx_driver_smif_mem_t

  *Structure containing the SMIF driver memory configuration.*
- struct ifx_driver_smif_obj_t

  *Structure containing the SMIF driver instance data.*

## Macros

- #define IFX_DRIVER_SMIF_MMIO_INSTANCE(instance_name, instance_obj)
- #define IFX_DRIVER_SMIF_XIP_INSTANCE(instance_name, instance_obj)

## Typedefs

- typedef struct ifx_driver_smif_mem_t ifx_driver_smif_mem_t

  *Structure containing the SMIF driver memory configuration.*
- typedef struct ifx_driver_smif_obj_t ifx_driver_smif_obj_t

  *Structure containing the SMIF driver instance data.*

## Variables

- const struct ifx_flash_driver_t ifx_driver_smif_mmio
- const struct ifx_flash_driver_t ifx_driver_smif_xip

## 7.59.1 Macro Definition Documentation

### 7.59.1.1 IFX_DRIVER_SMIF_MMIO_INSTANCE

```
#define IFX_DRIVER_SMIF_MMIO_INSTANCE(
            instance_name,
            instance_obj )
```

**Value:**

```
    /* Use variable to validate instance_obj type */ \
    const struct ifx_driver_smif_obj_t *instance_name##_validate_obj = &instance_obj; \
    const ifx_flash_driver_instance_t instance_name = { \
        .handler = &instance_obj, \
        .driver = &ifx_driver_smif_mmio, \
    };
```

Definition at line 64 of file ifx_driver_smif.h.

#### 7.59.1.2 IFX_DRIVER_SMIF_XIP_INSTANCE

```
#define IFX_DRIVER_SMIF_XIP_INSTANCE(
            instance_name,
            instance_obj )
```

**Value:**

```
/* Use variable to validate instance_obj type */ \
const struct ifx_driver_smif_obj_t *instance_name##_validate_obj = &instance_obj; \
const ifx_flash_driver_instance_t instance_name = { \
    .handler = &instance_obj, \
    .driver = &ifx_driver_smif_xip, \
};
```

Definition at line 72 of file ifx_driver_smif.h.

### 7.59.2 Typedef Documentation

#### 7.59.2.1 ifx_driver_smif_mem_t

```
typedef struct ifx_driver_smif_mem_t ifx_driver_smif_mem_t
```

Structure containing the SMIF driver memory configuration.
SMIF driver instance represent memory region with own boundaries. Such instance can share common SMIF memory configuration and context.

#### 7.59.2.2 ifx_driver_smif_obj_t

```
typedef struct ifx_driver_smif_obj_t ifx_driver_smif_obj_t
```

Structure containing the SMIF driver instance data.
SMIF flash driver uses offset, size to define a working region that is handled by active instance. Any access outside of this region is invalid and SMIF flash driver returns ARM_DRIVER_ERROR_PARAMETER in such cases.
ifx_driver_smif interface expects address to be relative to offset.

**Note**

This structure is modified by ifx_driver_smif.Initialize, so it should not be a constant.

### 7.59.3 Variable Documentation

#### 7.59.3.1 ifx_driver_smif_mmio

```
const struct ifx_flash_driver_t ifx_driver_smif_mmio
```

Definition at line 83 of file ifx_driver_smif_mmio.c.

#### 7.59.3.2 ifx_driver_smif_xip

```
const struct ifx_flash_driver_t ifx_driver_smif_xip
```

Definition at line 80 of file ifx_driver_smif_xip.c.

## 7.60 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_mmio.c File Reference

```
#include "ifx_driver_smif.h"
#include "ifx_driver_smif_private.h"
```

Include dependency graph for ifx_driver_smif_mmio.c:

## Variables

- const struct ifx_flash_driver_t ifx_driver_smif_mmio

### 7.60.1 Variable Documentation

#### 7.60.1.1 ifx_driver_smif_mmio

```
const struct ifx_flash_driver_t ifx_driver_smif_mmio
```

**Initial value:**

```
= {
    .GetVersion = ifx_driver_smif_get_version,
    .GetCapabilities = ifx_driver_smif_get_capabilities,
    .Initialize = ifx_driver_smif_initialize,
    .Uninitialize = ifx_driver_smif_uninitialize,
    .PowerControl = ifx_driver_smif_power_control,
    .ReadData = ifx_driver_smif_read_data,
    .ProgramData = ifx_driver_smif_program_data,
    .EraseSector = ifx_driver_smif_erase_sector,
    .EraseChip = ifx_driver_smif_erase_chip,
    .GetStatus = ifx_driver_smif_get_status,
    .GetInfo = ifx_driver_smif_get_info,
}
```

Definition at line 83 of file ifx_driver_smif_mmio.c.

## 7.61 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_private.h File Reference

```
#include "ifx_driver_smif.h"
```

Include dependency graph for ifx_driver_smif_private.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_DRIVER_SMIF_DATA_WIDTH 4U

## Functions

- int32_t ifx_driver_smif_initialize (ifx_flash_driver_handler_t handler, ifx_flash_driver_event_t cb_event)

  *Initialize SMIF flash driver instance.*
- int32_t ifx_driver_smif_set_mode (const ifx_driver_smif_obj_t ∗obj, cy_en_smif_mode_t mode)

  *Switch SMIF controller mode.*
- int32_t ifx_driver_smif_validate_region (const ifx_driver_smif_obj_t ∗obj, uint32_t address, uint32_t size)

  *Validate that memory block is inside of flash driver instance region.*
- ARM_DRIVER_VERSION ifx_driver_smif_get_version (ifx_flash_driver_handler_t handler)
- ARM_FLASH_CAPABILITIES ifx_driver_smif_get_capabilities (ifx_flash_driver_handler_t handler)
- int32_t ifx_driver_smif_uninitialize (ifx_flash_driver_handler_t handler)
- int32_t ifx_driver_smif_power_control (ifx_flash_driver_handler_t handler, ARM_POWER_STATE state)
- struct _ARM_FLASH_STATUS ifx_driver_smif_get_status (ifx_flash_driver_handler_t handler)
- ARM_FLASH_INFO ∗ ifx_driver_smif_get_info (ifx_flash_driver_handler_t handler)
- int32_t ifx_driver_smif_erase_sector (ifx_flash_driver_handler_t handler, uint32_t addr)

  *Erase sector of SMIF flash driver instance.*

- int32_t ifx_driver_smif_erase_chip (ifx_flash_driver_handler_t handler)

  *Erase memory region allocated for SMIF flash driver instance.*

## 7.61.1 Macro Definition Documentation

### 7.61.1.1 IFX_DRIVER_SMIF_DATA_WIDTH

```
#define IFX_DRIVER_SMIF_DATA_WIDTH 4U
```

Definition at line 14 of file ifx_driver_smif_private.h.

## 7.61.2 Function Documentation

### 7.61.2.1 ifx_driver_smif_erase_chip()

```
int32_t ifx_driver_smif_erase_chip (
            ifx_flash_driver_handler_t handler )
```

Erase memory region allocated for SMIF flash driver instance.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Flash driver handler data |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Success. |
| *Other* | ARM_DRIVER_ERROR_∗ Failed |

Definition at line 290 of file ifx_driver_smif.c.
Here is the call graph for this function:

### 7.61.2.2 ifx_driver_smif_erase_sector()

```
int32_t ifx_driver_smif_erase_sector (
            ifx_flash_driver_handler_t handler,
            uint32_t addr )
```

Erase sector of SMIF flash driver instance.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Flash driver handler data |
| in | *addr* | Sector address, must be aligned to sector. |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Success. |
| *Other* | ARM_DRIVER_ERROR_∗ Failed |

Definition at line 241 of file ifx_driver_smif.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.61.2.3 ifx_driver_smif_get_capabilities()

```
ARM_FLASH_CAPABILITIES ifx_driver_smif_get_capabilities (
            ifx_flash_driver_handler_t handler )
```

Definition at line 128 of file ifx_driver_smif.c.

#### 7.61.2.4 ifx_driver_smif_get_info()

```
ARM_FLASH_INFO* ifx_driver_smif_get_info (
            ifx_flash_driver_handler_t handler )
```

Definition at line 323 of file ifx_driver_smif.c.
Here is the call graph for this function:

#### 7.61.2.5 ifx_driver_smif_get_status()

```
struct _ARM_FLASH_STATUS ifx_driver_smif_get_status (
            ifx_flash_driver_handler_t handler )
```

Definition at line 310 of file ifx_driver_smif.c.

#### 7.61.2.6 ifx_driver_smif_get_version()

```
ARM_DRIVER_VERSION ifx_driver_smif_get_version (
            ifx_flash_driver_handler_t handler )
```

Definition at line 111 of file ifx_driver_smif.c.
Here is the call graph for this function:

#### 7.61.2.7 ifx_driver_smif_initialize()

```
int32_t ifx_driver_smif_initialize (
            ifx_flash_driver_handler_t handler,
            ifx_flash_driver_event_t cb_event )
```

Initialize SMIF flash driver instance.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Flash driver handler data |
| in | *cb_event* | Callback event is not supported. Must be NULL. |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Success. |
| *Other* | ARM_DRIVER_ERROR_∗ Failed |

Definition at line 162 of file ifx_driver_smif.c.
Here is the call graph for this function:

#### 7.61.2.8 ifx_driver_smif_power_control()

```
int32_t ifx_driver_smif_power_control (
            ifx_flash_driver_handler_t handler,
            ARM_POWER_STATE state )
```

Definition at line 233 of file ifx_driver_smif.c.

#### 7.61.2.9 ifx_driver_smif_set_mode()

```
int32_t ifx_driver_smif_set_mode (
            const ifx_driver_smif_obj_t * obj,
            cy_en_smif_mode_t mode )
```

Switch SMIF controller mode.

**Parameters**

| | | |
|---|---|---|
| in | *handler* | Flash driver handler data |
| in | *mode* | SMIF controller mode. |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Memory block is inside of flash driver instance region. |
| *ARM_DRIVER_ERROR_BUSY* | Memory block is outside of flash driver instance region. |

Definition at line 59 of file ifx_driver_smif.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.61.2.10 ifx_driver_smif_uninitialize()

```
int32_t ifx_driver_smif_uninitialize (
            ifx_flash_driver_handler_t handler )
```

Definition at line 226 of file ifx_driver_smif.c.

#### 7.61.2.11 ifx_driver_smif_validate_region()

```
int32_t ifx_driver_smif_validate_region (
            const ifx_driver_smif_obj_t * obj,
            uint32_t address,
            uint32_t size )
```

Validate that memory block is inside of flash driver instance region.

**Parameters**

| | | |
|---|---|---|
| in | *obj* | Flash driver instance data |
| in | *address* | Start address of memory block |
| in | *size* | Memory block size |

**Return values**

| | |
|---|---|
| *ARM_DRIVER_OK* | Memory block is inside of flash driver instance region. |
| *ARM_DRIVER_ERROR_PARAMETER* | Memory block is outside of flash driver instance region. |

**Note**

instance settings are not validated by this function!

Definition at line 76 of file ifx_driver_smif.c.
Here is the caller graph for this function:

## 7.62 platform/ext/target/infineon/common/drivers/flash/smif/ifx_driver_smif_xip.c File Reference

```
#include "ifx_driver_smif.h"
#include "ifx_driver_smif_private.h"
#include <string.h>
#include "cy_device.h"
```

Include dependency graph for ifx_driver_smif_xip.c:

## Variables

- const struct ifx_flash_driver_t ifx_driver_smif_xip

### 7.62.1 Variable Documentation

#### 7.62.1.1 ifx_driver_smif_xip

```
const struct ifx_flash_driver_t ifx_driver_smif_xip
```

**Initial value:**

```
= {
    .GetVersion = ifx_driver_smif_get_version,
    .GetCapabilities = ifx_driver_smif_get_capabilities,
    .Initialize = ifx_driver_smif_initialize,
    .Uninitialize = ifx_driver_smif_uninitialize,
    .PowerControl = ifx_driver_smif_power_control,
    .ReadData = ifx_driver_smif_read_data,
    .ProgramData = ifx_driver_smif_program_data,
    .EraseSector = ifx_driver_smif_erase_sector,
    .EraseChip = ifx_driver_smif_erase_chip,
    .GetStatus = ifx_driver_smif_get_status,
    .GetInfo = ifx_driver_smif_get_info,
}
```

Definition at line 80 of file ifx_driver_smif_xip.c.

## 7.63 platform/ext/target/infineon/common/drivers/protection/mpu_armv8m_drv.h File Reference

```
#include <stdint.h>
#include "cmsis.h"
```

Include dependency graph for mpu_armv8m_drv.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct mpu_armv8m_region_cfg_t

## Macros

- #define PRIVILEGED_DEFAULT_ENABLE 1
- #define HARDFAULT_NMI_ENABLE 1
- #define MPU_ARMV8M_MAIR_ATTR_DEVICE_VAL 0x04
- #define MPU_ARMV8M_MAIR_ATTR_DEVICE_IDX 0
- #define MPU_ARMV8M_MAIR_ATTR_CODE_VAL 0xAA

- #define MPU_ARMV8M_MAIR_ATTR_CODE_IDX 1
- #define MPU_ARMV8M_MAIR_ATTR_DATA_VAL 0xFF
- #define MPU_ARMV8M_MAIR_ATTR_DATA_IDX 2

## Enumerations

- enum mpu_armv8m_attr_exec_t { MPU_ARMV8M_XN_EXEC_OK, MPU_ARMV8M_XN_EXEC_NEVER }
- enum mpu_armv8m_attr_access_t { MPU_ARMV8M_AP_RW_PRIV_ONLY, MPU_ARMV8M_AP_RW_PRIV_UNPRIV, MPU_ARMV8M_AP_RO_PRIV_ONLY, MPU_ARMV8M_AP_RO_PRIV_UNPRIV }
- enum mpu_armv8m_attr_shared_t { MPU_ARMV8M_SH_NONE, MPU_ARMV8M_SH_UNUSED, MPU_ARMV8M_SH_OUTER, MPU_ARMV8M_SH_INNER }

## 7.63.1 Macro Definition Documentation

### 7.63.1.1 HARDFAULT_NMI_ENABLE

```
#define HARDFAULT_NMI_ENABLE 1
```

Definition at line 17 of file mpu_armv8m_drv.h.

### 7.63.1.2 MPU_ARMV8M_MAIR_ATTR_CODE_IDX

```
#define MPU_ARMV8M_MAIR_ATTR_CODE_IDX 1
```

Definition at line 25 of file mpu_armv8m_drv.h.

### 7.63.1.3 MPU_ARMV8M_MAIR_ATTR_CODE_VAL

```
#define MPU_ARMV8M_MAIR_ATTR_CODE_VAL 0xAA
```

Definition at line 24 of file mpu_armv8m_drv.h.

### 7.63.1.4 MPU_ARMV8M_MAIR_ATTR_DATA_IDX

```
#define MPU_ARMV8M_MAIR_ATTR_DATA_IDX 2
```

Definition at line 28 of file mpu_armv8m_drv.h.

### 7.63.1.5 MPU_ARMV8M_MAIR_ATTR_DATA_VAL

```
#define MPU_ARMV8M_MAIR_ATTR_DATA_VAL 0xFF
```

Definition at line 27 of file mpu_armv8m_drv.h.

### 7.63.1.6 MPU_ARMV8M_MAIR_ATTR_DEVICE_IDX

```
#define MPU_ARMV8M_MAIR_ATTR_DEVICE_IDX 0
```

Definition at line 22 of file mpu_armv8m_drv.h.

### 7.63.1.7 MPU_ARMV8M_MAIR_ATTR_DEVICE_VAL

```
#define MPU_ARMV8M_MAIR_ATTR_DEVICE_VAL 0x04
```

Definition at line 21 of file mpu_armv8m_drv.h.

#### 7.63.1.8 PRIVILEGED_DEFAULT_ENABLE

```
#define PRIVILEGED_DEFAULT_ENABLE 1
```

Definition at line 16 of file mpu_armv8m_drv.h.

### 7.63.2 Enumeration Type Documentation

#### 7.63.2.1 mpu_armv8m_attr_access_t

```
enum mpu_armv8m_attr_access_t
```

**Enumerator**

| | |
|---|---|
| MPU_ARMV8M_AP_RW_PRIV_ONLY | |
| MPU_ARMV8M_AP_RW_PRIV_UNPRIV | |
| MPU_ARMV8M_AP_RO_PRIV_ONLY | |
| MPU_ARMV8M_AP_RO_PRIV_UNPRIV | |

Definition at line 35 of file mpu_armv8m_drv.h.

#### 7.63.2.2 mpu_armv8m_attr_exec_t

```
enum mpu_armv8m_attr_exec_t
```

**Enumerator**

| | |
|---|---|
| MPU_ARMV8M_XN_EXEC_OK | |
| MPU_ARMV8M_XN_EXEC_NEVER | |

Definition at line 30 of file mpu_armv8m_drv.h.

#### 7.63.2.3 mpu_armv8m_attr_shared_t

```
enum mpu_armv8m_attr_shared_t
```

**Enumerator**

| | |
|---|---|
| MPU_ARMV8M_SH_NONE | |
| MPU_ARMV8M_SH_UNUSED | |
| MPU_ARMV8M_SH_OUTER | |
| MPU_ARMV8M_SH_INNER | |

Definition at line 42 of file mpu_armv8m_drv.h.

## 7.64 platform/ext/target/infineon/common/drivers/stdio/uart_pdl_stdout.c File Reference

```
#include "cy_device.h"
#include "cy_scb_uart.h"
#include "config_tfm.h"
#include "uart_stdout.h"
#include "uart_pdl_stdout.h"
```

```
#include "static_checks.h"
#include <stdio.h>
#include <limits.h>
```

Include dependency graph for uart_pdl_stdout.c:

## Functions

- int32_t stdio_output_string (const uint8_t *str, uint32_t len)
- void stdio_init (void)
- void stdio_uninit (void)

## 7.64.1 Function Documentation

### 7.64.1.1 stdio_init()

```
void stdio_init (
            void  )
```

Definition at line 182 of file uart_pdl_stdout.c.
Here is the caller graph for this function:

### 7.64.1.2 stdio_output_string()

```
int32_t stdio_output_string (
            const uint8_t * str,
            uint32_t len )
```

**Note**

Current implementation doesn't work if access to UART is done by multiple threads.

IFX_STDIO_PREFIX optional configuration can be used to provide prefix to the messages. It's useful if system has only one serial port and there is output from multiple cores/images.
Definition at line 97 of file uart_pdl_stdout.c.

Here is the call graph for this function:

#### 7.64.1.3 stdio_uninit()

```
void stdio_uninit (
            void  )
```

Definition at line 198 of file uart_pdl_stdout.c.

## 7.65 platform/ext/target/infineon/common/drivers/stdio/uart_pdl_↩ stdout.h File Reference

```
#include <stdio.h>
#include <stdint.h>
```

Include dependency graph for uart_pdl_stdout.h:

This graph shows which files directly or indirectly include this file:

## Functions

- int32_t stdio_output_string_raw (const uint8_t *str, uint32_t len)

  *Output log for SVC.*

### 7.65.1 Function Documentation

#### 7.65.1.1 stdio_output_string_raw()

```
int32_t stdio_output_string_raw (
            const uint8_t * str,
            uint32_t len )
```

Output log for SVC.
stdio_output_string_raw is intended to be used by Platform service. ifx_platform_log_msg is a client API to print message through Platform service.

## 7.66 platform/ext/target/infineon/common/interface/include/ifx_platform_api.h File Reference

```
#include "tfm_platform_api.h"
```

Include dependency graph for ifx_platform_api.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_PLATFORM_IOCTL_APP_MIN ((tfm_platform_ioctl_req_t)0x40000000)

  *Start of application specific platform IOCTL request id.*
- #define IFX_PLATFORM_IOCTL_APP_MAX ((tfm_platform_ioctl_req_t)0x7FFFFFFF)

  *Start of application specific platform IOCTL request id.*

## Functions

- uint32_t ifx_platform_log_msg (const uint8_t *msg, uint32_t msg_size)

  *Write message via Platform service to SPM UART.*

## 7.66.1 Macro Definition Documentation

### 7.66.1.1 IFX_PLATFORM_IOCTL_APP_MAX

```
#define IFX_PLATFORM_IOCTL_APP_MAX ((tfm_platform_ioctl_req_t)0x7FFFFFFF)
```

Start of application specific platform IOCTL request id.
Definition at line 30 of file ifx_platform_api.h.

### 7.66.1.2 IFX_PLATFORM_IOCTL_APP_MIN

```
#define IFX_PLATFORM_IOCTL_APP_MIN ((tfm_platform_ioctl_req_t)0x40000000)
```

Start of application specific platform IOCTL request id.
Definition at line 26 of file ifx_platform_api.h.

## 7.66.2 Function Documentation

### 7.66.2.1 ifx_platform_log_msg()

```
uint32_t ifx_platform_log_msg (
            const uint8_t * msg,
            uint32_t msg_size )
```

Write message via Platform service to SPM UART.
IFX_STDIO_PREFIX optional configuration can be used to provide prefix to the messages. It's useful if system has only one serial port and there is output from multiple cores/images.

**Parameters**

| | | |
|---|---|---|
| in | *msg* | Message to write |
| in | *msg_size* | Number of bytes to write |

**Return values**

| | |
|---|---|
| *Number* | of bytes written. |

Definition at line 14 of file ifx_platform_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.67 platform/ext/target/infineon/common/interface/include/mtb_srf_↩ ipc_custom_packet.h File Reference

```
#include "mtb_srf.h"
#include "tfm_mailbox.h"
```

Include dependency graph for mtb_srf_ipc_custom_packet.h:

### Data Structures

- struct _MTB_SRF_DATA_ALIGN

  *MTB SRF IPC request structure suitable for TFM needs. This structure overrides default mtb_srf_ipc_packet_t SRF structure.*

### Typedefs

- typedef struct _MTB_SRF_DATA_ALIGN mtb_srf_ipc_packet_t

  *MTB SRF IPC request structure suitable for TFM needs. This structure overrides default mtb_srf_ipc_packet_t SRF structure.*

### 7.67.1 Typedef Documentation

#### 7.67.1.1 mtb_srf_ipc_packet_t

```
typedef struct _MTB_SRF_DATA_ALIGN mtb_srf_ipc_packet_t
```

MTB SRF IPC request structure suitable for TFM needs. This structure overrides default mtb_srf_ipc_packet_t SRF structure.

This structure is designed to be allocated in shared memory for inter-process communication (IPC). All members are stored as values (not pointers) to ensure the entire request is self-contained and can be safely copied into and out of shared memory. Parameters required for the PSA client call are copied directly into this structure, and the structure itself is allocated in the shared memory region. This design ensures that all data needed for the request is accessible to both communicating parties without relying on process-specific pointers/memory.

## 7.68 platform/ext/target/infineon/common/interface/include/multi_core/tfm_mailbox_config.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
```

Include dependency graph for tfm_mailbox_config.h:

This graph shows which files directly or indirectly include this file:

## 7.69 platform/ext/target/infineon/common/interface/src/ifx_mtb_srf.c File Reference

```
#include <stddef.h>
#include <string.h>
```

```
#include "config_tfm.h"
#include "cmsis.h"
#include "ifx_platform_private.h"
#include "mtb_srf.h"
#include "psa_manifest/sid.h"
#include "tfm_platform_api.h"
```

Include dependency graph for ifx_mtb_srf.c:

## Functions

- cy_rslt_t mtb_srf_request_submit (mtb_srf_invec_ns_t ∗inVec_ns, uint8_t inVec_cnt_ns, mtb_srf_outvec_↩ ns_t ∗outVec_ns, uint8_t outVec_cnt_ns)

## 7.69.1 Function Documentation

### 7.69.1.1 mtb_srf_request_submit()

```
cy_rslt_t mtb_srf_request_submit (
            mtb_srf_invec_ns_t * inVec_ns,
            uint8_t inVec_cnt_ns,
            mtb_srf_outvec_ns_t * outVec_ns,
            uint8_t outVec_cnt_ns )
```

Definition at line 20 of file ifx_mtb_srf.c.

Here is the call graph for this function:

## 7.70 platform/ext/target/infineon/common/interface/src/ifx_mtb_srf_relay.c File Reference

```
#include <stddef.h>
#include <string.h>
#include "config_tfm.h"
```

Include dependency graph for ifx_mtb_srf_relay.c:

## 7.71 platform/ext/target/infineon/common/interface/src/ifx_platform_api.c File Reference

```
#include <string.h>
#include "config_tfm.h"
#include "ifx_platform_api.h"
#include "ifx_platform_private.h"
```

Include dependency graph for ifx_platform_api.c:

## Functions

- uint32_t ifx_platform_log_msg (const uint8_t *msg, uint32_t msg_size)

  *Write message via Platform service to SPM UART.*

## 7.71.1 Function Documentation

### 7.71.1.1 ifx_platform_log_msg()

```
uint32_t ifx_platform_log_msg (
            const uint8_t * msg,
            uint32_t msg_size )
```

Write message via Platform service to SPM UART.
IFX_STDIO_PREFIX optional configuration can be used to provide prefix to the messages. It's useful if system has only one serial port and there is output from multiple cores/images.

**Parameters**

| | | |
|---|---|---|
| in | *msg* | Message to write |
| in | *msg_size* | Number of bytes to write |

**Return values**

| | |
|---|---|
| *Number* | of bytes written. |

Definition at line 14 of file ifx_platform_api.c.

Here is the call graph for this function:

Here is the caller graph for this function:

## 7.72 platform/ext/target/infineon/common/interface/src/ifx_platform_private.h File Reference

`#include "tfm_platform_api.h"`

Include dependency graph for ifx_platform_private.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_PLATFORM_IOCTL_LOG_MSG 0x00000000

  *Log message to SPM UART.*
- #define IFX_PLATFORM_API_ID_MTB_SRF 2001

  *MTB SRF.*
- #define IFX_CUSTOM_PLATFORM_HAL_IOCTL 0

### 7.72.1 Macro Definition Documentation

#### 7.72.1.1 IFX_CUSTOM_PLATFORM_HAL_IOCTL

```
#define IFX_CUSTOM_PLATFORM_HAL_IOCTL 0
```

Definition at line 38 of file ifx_platform_private.h.

#### 7.72.1.2 IFX_PLATFORM_API_ID_MTB_SRF

```
#define IFX_PLATFORM_API_ID_MTB_SRF 2001
```

MTB SRF.
Definition at line 35 of file ifx_platform_private.h.

#### 7.72.1.3 IFX_PLATFORM_IOCTL_LOG_MSG

```
#define IFX_PLATFORM_IOCTL_LOG_MSG 0x00000000
```

Log message to SPM UART.
Definition at line 30 of file ifx_platform_private.h.

## 7.73 platform/ext/target/infineon/common/interface/src/multi_core/ifx_↩ mtb_mailbox.c File Reference

```
#include "cybsp.h"
#include "mtb_srf_ipc_init.h"
#include "tfm_ns_mailbox.h"
```

Include dependency graph for ifx_mtb_mailbox.c:

## Macros

- #define IFX_MTB_MAILBOX_CACHE_PRESENT (0)

## Functions

- int32_t ifx_mtb_mailbox_client_call (uint32_t call_type, const struct psa_client_params_t ∗params, int32_t client_id, struct mailbox_reply_t ∗reply)

  *Send PSA client call to on core NSPE via MTB mailbox. Wait and fetch PSA client call result.*

## 7.73.1 Macro Definition Documentation

### 7.73.1.1 IFX_MTB_MAILBOX_CACHE_PRESENT

```
#define IFX_MTB_MAILBOX_CACHE_PRESENT (0)
```

Definition at line 14 of file ifx_mtb_mailbox.c.

## 7.73.2 Function Documentation

### 7.73.2.1 ifx_mtb_mailbox_client_call()

```
int32_t ifx_mtb_mailbox_client_call (
            uint32_t call_type,
            const struct psa_client_params_t * params,
            int32_t client_id,
            struct mailbox_reply_t * reply )
```

Send PSA client call to on core NSPE via MTB mailbox. Wait and fetch PSA client call result.

**Parameters**

| | | |
|---|---|---|
| in | *call_type* | PSA client call type |
| in | *params* | Parameters used for PSA client call |
| in | *client_id* | Optional client ID of non-secure caller. It is required to identify the non-secure caller when NSPE OS enforces non-secure task isolation. |
| out | *reply* | The buffer written with PSA client call result. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | The PSA client call is completed successfully. |
| *Other* | return code Operation failed with an error code. |

Definition at line 23 of file ifx_mtb_mailbox.c.

## 7.74 platform/ext/target/infineon/common/libs/ifx_se_rt_services_utils/spe/ifx_se_tfm_utils.h File Reference

This file is a wrapper for ifx-se-rt-services-utils library for TF-M. It adds additional stuff needed to use this library from within secure partitions.

```
#include <stddef.h>
#include "ifx_se_psacrypto.h"
```

Include dependency graph for ifx_se_tfm_utils.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_SE_TFM_SYSCALL_CONTEXT NULL

  *Use this macro as context parameter to ifx_se_∗ functions.*

### 7.74.1 Detailed Description

This file is a wrapper for ifx-se-rt-services-utils library for TF-M. It adds additional stuff needed to use this library from within secure partitions.

### 7.74.2 Macro Definition Documentation

#### 7.74.2.1 IFX_SE_TFM_SYSCALL_CONTEXT

```
#define IFX_SE_TFM_SYSCALL_CONTEXT NULL
```

Use this macro as context parameter to ifx_se_∗ functions.

Example:

```
ifx_se_generate_random(output, output_size, IFX_SE_TFM_SYSCALL_CONTEXT);
```

Definition at line 33 of file ifx_se_tfm_utils.h.

## 7.75 platform/ext/target/infineon/common/nspe/mailbox/platform_ns_↩ mailbox.c File Reference

```
#include <limits.h>
#include <stdint.h>
#include <string.h>
#include "cmsis_compiler.h"
#include "cy_ipc_drv.h"
#include "cy_sysint.h"
#include "ifx_interrupt_defs.h"
#include "tfm_multi_core_api.h"
#include "tfm_ns_mailbox.h"
#include "tfm_peripherals_def.h"
#include "platform_multicore.h"
```

Include dependency graph for platform_ns_mailbox.c:

### Functions

- int32_t tfm_ns_multi_core_boot (struct ns_mailbox_queue_t ∗queue)

  *Performs multicore boot sequence in NSPE.*
- int32_t tfm_platform_ns_wait_for_s_cpu_ready (void)

  *Synchronisation with secure CPU, platform-specific implementation. Flags that the non-secure side has completed its initialization. Waits, if necessary, for the secure CPU to flag that it has completed its initialization.*
- void IFX_IRQ_NAME_TO_HANDLER() IFX_IPC_TFM_TO_NS_IPC_INTR (void)
- int32_t tfm_ns_mailbox_hal_notify_peer (void)

  *Notify SPE to deal with the PSA client call sent via mailbox.*
- int32_t tfm_ns_mailbox_hal_init (struct ns_mailbox_queue_t ∗queue)

  *Platform specific NSPE mailbox initialization. Invoked by tfm_ns_mailbox_init().*
- uint32_t tfm_ns_mailbox_hal_enter_critical (void)

  *Enter critical section of NSPE mailbox.*
- void tfm_ns_mailbox_hal_exit_critical (uint32_t state)

  *Exit critical section of NSPE mailbox.*
- uint32_t tfm_ns_mailbox_hal_enter_critical_isr (void)

  *Enter critical section of NSPE mailbox in IRQ handler.*

- void tfm_ns_mailbox_hal_exit_critical_isr (uint32_t state)

  *Enter critical section of NSPE mailbox in IRQ handler.*

## 7.75.1 Function Documentation

### 7.75.1.1 IFX_IPC_TFM_TO_NS_IPC_INTR()

```
void IFX_IRQ_NAME_TO_HANDLER() IFX_IPC_TFM_TO_NS_IPC_INTR (
            void  )
```

Definition at line 101 of file platform_ns_mailbox.c.

### 7.75.1.2 tfm_ns_mailbox_hal_enter_critical()

```
uint32_t tfm_ns_mailbox_hal_enter_critical (
            void  )
```

Enter critical section of NSPE mailbox.

**Note**

The implementation depends on platform specific hardware and use case.

**Returns**

Platform specific critical section state. Use it to exit from critical section by passing result to tfm_ns_mailbox_hal_exit_critical.

Definition at line 187 of file platform_ns_mailbox.c.
Here is the caller graph for this function:

### 7.75.1.3 tfm_ns_mailbox_hal_enter_critical_isr()

```
uint32_t tfm_ns_mailbox_hal_enter_critical_isr (
            void  )
```

Enter critical section of NSPE mailbox in IRQ handler.

**Note**

The implementation depends on platform specific hardware and use case.

**Returns**

Platform specific critical section state. Use it to exit from critical section by passing result to tfm_ns_mailbox_hal_exit_critical_isr.

Definition at line 216 of file platform_ns_mailbox.c.
Here is the call graph for this function:

#### 7.75.1.4 tfm_ns_mailbox_hal_exit_critical()

```
void tfm_ns_mailbox_hal_exit_critical (
            uint32_t state )
```

Exit critical section of NSPE mailbox.

**Note**

The implementation depends on platform specific hardware and use case.

**Parameters**

| | | |
|---|---|---|
| in | *state* | Critical section state returned by tfm_ns_mailbox_hal_enter_critical. |

Definition at line 206 of file platform_ns_mailbox.c.
Here is the caller graph for this function:

#### 7.75.1.5 tfm_ns_mailbox_hal_exit_critical_isr()

```
void tfm_ns_mailbox_hal_exit_critical_isr (
            uint32_t state )
```

Enter critical section of NSPE mailbox in IRQ handler.

**Note**

The implementation depends on platform specific hardware and use case.

**Parameters**

| | | |
|---|---|---|
| in | *state* | Critical section state returned by tfm_ns_mailbox_hal_enter_critical_isr. |

Definition at line 225 of file platform_ns_mailbox.c.
Here is the call graph for this function:

#### 7.75.1.6 tfm_ns_mailbox_hal_init()

```
int32_t tfm_ns_mailbox_hal_init (
            struct ns_mailbox_queue_t * queue )
```

Platform specific NSPE mailbox initialization. Invoked by tfm_ns_mailbox_init().

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The base address of NSPE mailbox queue to be initialized. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 133 of file platform_ns_mailbox.c.

#### 7.75.1.7 tfm_ns_mailbox_hal_notify_peer()

```
int32_t tfm_ns_mailbox_hal_notify_peer (
            void )
```

Notify SPE to deal with the PSA client call sent via mailbox.

**Note**

The implementation depends on platform specific hardware and use case.

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 118 of file platform_ns_mailbox.c.

#### 7.75.1.8 tfm_ns_multi_core_boot()

```
int32_t tfm_ns_multi_core_boot (
            struct ns_mailbox_queue_t * queue )
```

Performs multicore boot sequence in NSPE.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The base address of NSPE mailbox queue to be initialized. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 26 of file platform_ns_mailbox.c.
Here is the call graph for this function:

#### 7.75.1.9 tfm_platform_ns_wait_for_s_cpu_ready()

```
int32_t tfm_platform_ns_wait_for_s_cpu_ready (
            void )
```

Synchronisation with secure CPU, platform-specific implementation. Flags that the non-secure side has completed its initialization. Waits, if necessary, for the secure CPU to flag that it has completed its initialization.

**Return values**

| | |
|---|---|
| *Return* | 0 if succeeds. |
| *Otherwise,return* | specific error code. |

Definition at line 82 of file platform_ns_mailbox.c.

Here is the caller graph for this function:

## 7.76 platform/ext/target/infineon/common/nspe/mailbox/tfm_ns_mailbox_rtos_api.c File Reference

```
#include "cmsis_compiler.h"
#include "os_wrapper/semaphore.h"
#include "os_wrapper/kernel.h"
#include "os_wrapper/thread.h"
#include "tfm_ns_mailbox.h"
```

Include dependency graph for tfm_ns_mailbox_rtos_api.c:

### Macros

- #define MAILBOX_THREAD_FLAG 0x5FCA0000
- #define MAX_SEMAPHORE_COUNT NUM_MAILBOX_QUEUE_SLOT

### Functions

- const void * tfm_ns_mailbox_os_get_task_handle (void)
- void tfm_ns_mailbox_os_wait_reply (void)
- void tfm_ns_mailbox_os_wake_task_isr (const void *task_handle)
- void tfm_ns_mailbox_os_spin_lock (void)
- void tfm_ns_mailbox_os_spin_unlock (void)
- int32_t tfm_ns_mailbox_os_lock_init (void)
- int32_t tfm_ns_mailbox_os_lock_acquire (void)
- int32_t tfm_ns_mailbox_os_lock_release (void)

## 7.76.1 Macro Definition Documentation

### 7.76.1.1 MAILBOX_THREAD_FLAG

```
#define MAILBOX_THREAD_FLAG 0x5FCA0000
```

Definition at line 34 of file tfm_ns_mailbox_rtos_api.c.

### 7.76.1.2 MAX_SEMAPHORE_COUNT

```
#define MAX_SEMAPHORE_COUNT NUM_MAILBOX_QUEUE_SLOT
```

Definition at line 37 of file tfm_ns_mailbox_rtos_api.c.

## 7.76.2 Function Documentation

### 7.76.2.1 tfm_ns_mailbox_os_get_task_handle()

```
const void* tfm_ns_mailbox_os_get_task_handle (
            void  )
```

Definition at line 42 of file tfm_ns_mailbox_rtos_api.c.
Here is the call graph for this function:

### 7.76.2.2 tfm_ns_mailbox_os_lock_acquire()

```
int32_t tfm_ns_mailbox_os_lock_acquire (
            void  )
```

Definition at line 146 of file tfm_ns_mailbox_rtos_api.c.
Here is the call graph for this function:

#### 7.76.2.3 tfm_ns_mailbox_os_lock_init()

```
int32_t tfm_ns_mailbox_os_lock_init (
            void  )
```

Definition at line 134 of file tfm_ns_mailbox_rtos_api.c.

#### 7.76.2.4 tfm_ns_mailbox_os_lock_release()

```
int32_t tfm_ns_mailbox_os_lock_release (
            void  )
```

Definition at line 156 of file tfm_ns_mailbox_rtos_api.c.
Here is the call graph for this function:

#### 7.76.2.5 tfm_ns_mailbox_os_spin_lock()

```
void tfm_ns_mailbox_os_spin_lock (
            void  )
```

Definition at line 78 of file tfm_ns_mailbox_rtos_api.c.

#### 7.76.2.6 tfm_ns_mailbox_os_spin_unlock()

```
void tfm_ns_mailbox_os_spin_unlock (
            void  )
```

Definition at line 87 of file tfm_ns_mailbox_rtos_api.c.

#### 7.76.2.7 tfm_ns_mailbox_os_wait_reply()

```
void tfm_ns_mailbox_os_wait_reply (
            void  )
```

Definition at line 47 of file tfm_ns_mailbox_rtos_api.c.

Here is the call graph for this function:

#### 7.76.2.8 tfm_ns_mailbox_os_wake_task_isr()

```
void tfm_ns_mailbox_os_wake_task_isr (
            const void * task_handle )
```

Definition at line 65 of file tfm_ns_mailbox_rtos_api.c.
Here is the call graph for this function:

## 7.77 platform/ext/target/infineon/common/nspe/os_wrapper/os_↩ wrapper_cyabs_rtos.c File Reference

## 7.78 platform/ext/target/infineon/common/nspe/os_↩ wrapper/semaphore.h File Reference

```
#include "os_wrapper/common.h"
```

Include dependency graph for semaphore.h:

This graph shows which files directly or indirectly include this file:

## Functions

- void ∗ os_wrapper_semaphore_create (uint32_t max_count, uint32_t initial_count, const char ∗name)

  *Creates a new semaphore.*

- uint32_t os_wrapper_semaphore_acquire (void ∗handle, uint32_t timeout)

  *Acquires the semaphore.*

- uint32_t os_wrapper_semaphore_release (void ∗handle)

  *Releases the semaphore.*

- uint32_t os_wrapper_semaphore_delete (void ∗handle)

  *Deletes the semaphore.*

## 7.78.1 Function Documentation

### 7.78.1.1 os_wrapper_semaphore_acquire()

```
uint32_t os_wrapper_semaphore_acquire (
            void * handle,
            uint32_t timeout )
```

Acquires the semaphore.

**Parameters**

| | | |
|---|---|---|
| in | *hanlde* | Semaphore handle |
| in | *timeout* | Timeout value |

**Returns**

OS_WRAPPER_SUCCESS in case of successful acquision, or OS_WRAPPER_ERROR in case of error

Here is the caller graph for this function:

### 7.78.1.2 os_wrapper_semaphore_create()

```
void* os_wrapper_semaphore_create (
            uint32_t max_count,
            uint32_t initial_count,
            const char * name )
```

Creates a new semaphore.

**Parameters**

| | | |
|---|---|---|
| in | *max_count* | Highest count of the semaphore |
| in | *initial_count* | Starting count of the available semaphore |
| in | *name* | Name of the semaphore |

**Returns**

Returns handle of the semaphore created, or NULL in case of error

### 7.78.1.3 os_wrapper_semaphore_delete()

```
uint32_t os_wrapper_semaphore_delete (
            void * handle )
```

Deletes the semaphore.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Semaphore handle |

**Returns**

OS_WRAPPER_SUCCESS in case of successful release, or OS_WRAPPER_ERROR in case of error

#### 7.78.1.4 os_wrapper_semaphore_release()

```
uint32_t os_wrapper_semaphore_release (
            void * handle )
```

Releases the semaphore.

**Parameters**

| | | |
|---|---|---|
| in | *hanlde* | Semaphore handle |

**Returns**

OS_WRAPPER_SUCCESS in case of successful release, or OS_WRAPPER_ERROR in case of error

Here is the caller graph for this function:

## 7.79 platform/ext/target/infineon/common/nspe/test/ifx_fpu_ns.c File Reference

```
#include "ifx_interrupt_defs.h"
#include "tfm_peripherals_def.h"
#include "test_interrupt.h"
```

Include dependency graph for ifx_fpu_ns.c:

## Functions

- void IFX_IRQ_NAME_TO_HANDLER() TFM_FPU_NS_TEST_IRQ (void)
- void tfm_test_ns_fpu_init (void)

## 7.79.1 Function Documentation

### 7.79.1.1 TFM_FPU_NS_TEST_IRQ()

```
void IFX_IRQ_NAME_TO_HANDLER() TFM_FPU_NS_TEST_IRQ (
            void  )
```

Definition at line 16 of file ifx_fpu_ns.c.
Here is the caller graph for this function:

### 7.79.1.2 tfm_test_ns_fpu_init()

```
void tfm_test_ns_fpu_init (
            void  )
```

Definition at line 21 of file ifx_fpu_ns.c.

Here is the call graph for this function:

## 7.80 platform/ext/target/infineon/common/nspe/test/plat_test_ns.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "cy_tcpwm_counter.h"
#include "device_definition.h"
#include "tfm_plat_defs.h"
#include "tfm_plat_test.h"
#include "tfm_peripherals_def.h"
```

Include dependency graph for plat_test_ns.c:

## 7.81 platform/ext/target/infineon/common/shared/mailbox/ifx_platform_↩ mailbox.h File Reference

```
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
```

Include dependency graph for ifx_platform_mailbox.h:

This graph shows which files directly or indirectly include this file:

## 7.82 platform/ext/target/infineon/common/shared/mailbox/platform_multicore.c File Reference

```
#include <stdint.h>
#include "cmsis_compiler.h"
#include "platform_multicore.h"
#include "cy_ipc_drv.h"
#include "cy_sysint.h"
#include "tfm_peripherals_def.h"
```

Include dependency graph for platform_multicore.c:

## Macros

- #define IPC_RX_CHAN IFX_IPC_TFM_TO_NS_CHAN
- #define IPC_RX_INTR_STRUCT IFX_IPC_TFM_TO_NS_INTR_STRUCT
- #define IPC_RX_INT_MASK IFX_IPC_TFM_TO_NS_INTR_MASK
- #define IPC_TX_CHAN IFX_IPC_NS_TO_TFM_CHAN
- #define IPC_TX_NOTIFY_MASK IFX_IPC_NS_TO_TFM_NOTIFY_MASK

## Functions

- int32_t ifx_mailbox_fetch_msg_ptr (void **msg_ptr)

  *Fetch a pointer from mailbox message.*
- int32_t ifx_mailbox_fetch_msg_data (uint32_t *data_ptr)

  *Fetch a data value from mailbox message.*
- int32_t ifx_mailbox_send_msg_ptr (const void *msg_ptr)

  *Send a pointer via mailbox message.*
- int32_t ifx_mailbox_send_msg_data (uint32_t data)

  *Send a data value via mailbox message.*
- void ifx_mailbox_wait_for_notify (void)

  *Wait for a mailbox notify event.*

## 7.82.1 Macro Definition Documentation

### 7.82.1.1 IPC_RX_CHAN

```
#define IPC_RX_CHAN IFX_IPC_TFM_TO_NS_CHAN
```

Definition at line 27 of file platform_multicore.c.

### 7.82.1.2 IPC_RX_INT_MASK

```
#define IPC_RX_INT_MASK IFX_IPC_TFM_TO_NS_INTR_MASK
```

Definition at line 29 of file platform_multicore.c.

#### 7.82.1.3 IPC_RX_INTR_STRUCT

```
#define IPC_RX_INTR_STRUCT IFX_IPC_TFM_TO_NS_INTR_STRUCT
```

Definition at line 28 of file platform_multicore.c.

#### 7.82.1.4 IPC_TX_CHAN

```
#define IPC_TX_CHAN IFX_IPC_NS_TO_TFM_CHAN
```

Definition at line 31 of file platform_multicore.c.

#### 7.82.1.5 IPC_TX_NOTIFY_MASK

```
#define IPC_TX_NOTIFY_MASK IFX_IPC_NS_TO_TFM_NOTIFY_MASK
```

Definition at line 32 of file platform_multicore.c.

### 7.82.2 Function Documentation

#### 7.82.2.1 ifx_mailbox_fetch_msg_data()

```
int32_t ifx_mailbox_fetch_msg_data (
            uint32_t * data_ptr )
```

Fetch a data value from mailbox message.

**Parameters**

| | | |
|---|---|---|
| out | *data_ptr* | The address to write the pointer value to. |

**Return values**

| | |
|---|---|
| *0* | The operation succeeds. |
| *else* | The operation fails. |

Definition at line 53 of file platform_multicore.c.

#### 7.82.2.2 ifx_mailbox_fetch_msg_ptr()

```
int32_t ifx_mailbox_fetch_msg_ptr (
            void ** msg_ptr )
```

Fetch a pointer from mailbox message.

**Parameters**

| | | |
|---|---|---|
| out | *msg_ptr* | The address to write the pointer value to. |

**Return values**

| | |
|---|---|
| *0* | The operation succeeds. |
| *else* | The operation fails. |

Definition at line 35 of file platform_multicore.c.

Here is the caller graph for this function:

#### 7.82.2.3 ifx_mailbox_send_msg_data()

```
int32_t ifx_mailbox_send_msg_data (
            uint32_t data )
```

Send a data value via mailbox message.

**Parameters**

| | | |
|---|---|---|
| in | *data* | The data value to be sent |

**Return values**

| | |
|---|---|
| *0* | The operation succeeds. |
| *else* | The operation fails. |

Definition at line 89 of file platform_multicore.c.
Here is the caller graph for this function:

#### 7.82.2.4 ifx_mailbox_send_msg_ptr()

```
int32_t ifx_mailbox_send_msg_ptr (
            const void * msg_ptr )
```

Send a pointer via mailbox message.

**Parameters**

| | | |
|---|---|---|
| in | *msg_ptr* | The pointer value to be sent. |

**Return values**

| | |
|---|---|
| *0* | The operation succeeds. |

**Return values**

| | |
|---|---|
| *else* | The operation fails. |

Definition at line 71 of file platform_multicore.c.

#### 7.82.2.5 ifx_mailbox_wait_for_notify()

```
void ifx_mailbox_wait_for_notify (
            void  )
```

Wait for a mailbox notify event.
Definition at line 102 of file platform_multicore.c.
Here is the caller graph for this function:

## 7.83 platform/ext/target/infineon/common/shared/mailbox/platform_↩ multicore.h File Reference

```
#include <stdint.h>
#include "cy_device.h"
```

Include dependency graph for platform_multicore.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_PSA_CLIENT_CALL_REQ_MAGIC (0xA5CF50C6U)
- #define IFX_PSA_CLIENT_CALL_REPLY_MAGIC (0xC605FC5AU)
- #define IFX_NS_MAILBOX_INIT_ENABLE (0xAEU)
- #define IFX_S_MAILBOX_READY (0xC3U)
- #define IFX_PLATFORM_MAILBOX_SUCCESS (0x0)
- #define IFX_PLATFORM_MAILBOX_INVAL_PARAMS (INT32_MIN + 1)
- #define IFX_PLATFORM_MAILBOX_TX_ERROR (INT32_MIN + 2)
- #define IFX_PLATFORM_MAILBOX_RX_ERROR (INT32_MIN + 3)
- #define IFX_PLATFORM_MAILBOX_INIT_ERROR (INT32_MIN + 4)
- #define IFX_IPC_SYNC_MAGIC 0x7DADE011U

## Functions

- int32_t ifx_mailbox_fetch_msg_ptr (void **msg_ptr)

  *Fetch a pointer from mailbox message.*
- int32_t ifx_mailbox_fetch_msg_data (uint32_t *data_ptr)

  *Fetch a data value from mailbox message.*
- int32_t ifx_mailbox_send_msg_ptr (const void *msg_ptr)

  *Send a pointer via mailbox message.*
- int32_t ifx_mailbox_send_msg_data (uint32_t data)

  *Send a data value via mailbox message.*
- void ifx_mailbox_wait_for_notify (void)

  *Wait for a mailbox notify event.*

## 7.83.1 Macro Definition Documentation

### 7.83.1.1 IFX_IPC_SYNC_MAGIC

```
#define IFX_IPC_SYNC_MAGIC 0x7DADE011U
```

Definition at line 28 of file platform_multicore.h.

### 7.83.1.2 IFX_NS_MAILBOX_INIT_ENABLE

```
#define IFX_NS_MAILBOX_INIT_ENABLE (0xAEU)
```

Definition at line 19 of file platform_multicore.h.

### 7.83.1.3 IFX_PLATFORM_MAILBOX_INIT_ERROR

```
#define IFX_PLATFORM_MAILBOX_INIT_ERROR (INT32_MIN + 4)
```

Definition at line 26 of file platform_multicore.h.

#### 7.83.1.4 IFX_PLATFORM_MAILBOX_INVAL_PARAMS

```
#define IFX_PLATFORM_MAILBOX_INVAL_PARAMS (INT32_MIN + 1)
```

Definition at line 23 of file platform_multicore.h.

#### 7.83.1.5 IFX_PLATFORM_MAILBOX_RX_ERROR

```
#define IFX_PLATFORM_MAILBOX_RX_ERROR (INT32_MIN + 3)
```

Definition at line 25 of file platform_multicore.h.

#### 7.83.1.6 IFX_PLATFORM_MAILBOX_SUCCESS

```
#define IFX_PLATFORM_MAILBOX_SUCCESS (0x0)
```

Definition at line 22 of file platform_multicore.h.

#### 7.83.1.7 IFX_PLATFORM_MAILBOX_TX_ERROR

```
#define IFX_PLATFORM_MAILBOX_TX_ERROR (INT32_MIN + 2)
```

Definition at line 24 of file platform_multicore.h.

#### 7.83.1.8 IFX_PSA_CLIENT_CALL_REPLY_MAGIC

```
#define IFX_PSA_CLIENT_CALL_REPLY_MAGIC (0xC605FC5AU)
```

Definition at line 17 of file platform_multicore.h.

#### 7.83.1.9 IFX_PSA_CLIENT_CALL_REQ_MAGIC

```
#define IFX_PSA_CLIENT_CALL_REQ_MAGIC (0xA5CF50C6U)
```

Definition at line 16 of file platform_multicore.h.

#### 7.83.1.10 IFX_S_MAILBOX_READY

```
#define IFX_S_MAILBOX_READY (0xC3U)
```

Definition at line 20 of file platform_multicore.h.

### 7.83.2 Function Documentation

#### 7.83.2.1 ifx_mailbox_fetch_msg_data()

```
int32_t ifx_mailbox_fetch_msg_data (
            uint32_t * data_ptr )
```

Fetch a data value from mailbox message.

**Parameters**

| | | |
|---|---|---|
| out | *data_ptr* | The address to write the pointer value to. |

**Return values**

| | |
|---|---|
| *0* | The operation succeeds. |
| *else* | The operation fails. |

Definition at line 53 of file platform_multicore.c.

#### 7.83.2.2 ifx_mailbox_fetch_msg_ptr()

```
int32_t ifx_mailbox_fetch_msg_ptr (
            void ** msg_ptr )
```

Fetch a pointer from mailbox message.

**Parameters**

| | | |
|---|---|---|
| out | *msg_ptr* | The address to write the pointer value to. |

**Return values**

| | |
|---|---|
| *0* | The operation succeeds. |
| *else* | The operation fails. |

Definition at line 35 of file platform_multicore.c.
Here is the caller graph for this function:

#### 7.83.2.3 ifx_mailbox_send_msg_data()

```
int32_t ifx_mailbox_send_msg_data (
            uint32_t data )
```

Send a data value via mailbox message.

**Parameters**

| | | |
|---|---|---|
| in | *data* | The data value to be sent |

**Return values**

| | |
|---|---|
| *0* | The operation succeeds. |
| *else* | The operation fails. |

Definition at line 89 of file platform_multicore.c.

Here is the caller graph for this function:

#### 7.83.2.4 ifx_mailbox_send_msg_ptr()

```
int32_t ifx_mailbox_send_msg_ptr (
            const void * msg_ptr )
```

Send a pointer via mailbox message.

**Parameters**

| | | |
|---|---|---|
| in | *msg_ptr* | The pointer value to be sent. |

**Return values**

| | |
|---|---|
| *0* | The operation succeeds. |
| *else* | The operation fails. |

Definition at line 71 of file platform_multicore.c.

#### 7.83.2.5 ifx_mailbox_wait_for_notify()

```
void ifx_mailbox_wait_for_notify (
            void )
```

Wait for a mailbox notify event.
Definition at line 102 of file platform_multicore.c.
Here is the caller graph for this function:

## 7.84 platform/ext/target/infineon/common/spe/faults/faults.c File Reference

```
#include "config_tfm.h"
#include "cmsis.h"
```

```
#include "cy_sysfault.h"
#include "cyip_fault.h"
#include "exception_info.h"
#include "faults.h"
#include "protection_regions_cfg.h"
#include "utilities.h"
#include "static_checks.h"
```

Include dependency graph for faults.c:

## Macros

- #define SCB_SHCSR_ENABLED_FAULTS

## Functions

- FIH_RET_TYPE (enum tfm_plat_err_t) ifx_faults_cfg(void) = fih_int_encode(TFM_HAL_ERROR_GENERIC)

  *This function enables the faults interrupts (plus the isolation boundary violation interrupts)*
- void tfm_hal_platform_exception_info (void)

## 7.84.1 Macro Definition Documentation

### 7.84.1.1 SCB_SHCSR_ENABLED_FAULTS

```
#define SCB_SHCSR_ENABLED_FAULTS
```

**Value:**

```
(SCB_SHCSR_USGFAULTENA_Msk | \
SCB_SHCSR_BUSFAULTENA_Msk | \
SCB_SHCSR_MEMFAULTENA_Msk | \
SCB_SHCSR_SECUREFAULTENA_Msk)
```

Definition at line 20 of file faults.c.

## 7.84.2 Function Documentation

### 7.84.2.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_plat_err_t ) = fih_int_encode(TFM_HAL_ERROR_GENERIC)
```

This function enables the faults interrupts (plus the isolation boundary violation interrupts)
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.

Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |

**Parameters**

| | | |
|---|---|---|
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Definition at line 67 of file faults.c.
Here is the caller graph for this function:

#### 7.84.2.2 tfm_hal_platform_exception_info()

```
void tfm_hal_platform_exception_info (
            void  )
```

Definition at line 191 of file faults.c.

## 7.85 platform/ext/target/infineon/common/spe/faults/faults.h File Reference

```
#include "fih.h"
#include "tfm_plat_defs.h"
```

Include dependency graph for faults.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_plat_err_t) ifx_faults_cfg(void)

  *Configures fault handlers and sets priorities.*

## 7.85.1 Function Documentation

### 7.85.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 67 of file faults.c.

## 7.86 platform/ext/target/infineon/common/spe/faults/faults_cat1b.c File Reference

```
#include "config_tfm.h"
#include "cmsis.h"
#include "cy_sysfault.h"
#include "exception_info.h"
#include "faults.h"
#include "faults_dump.h"
#include "ifx_utils.h"
#include "tfm_spm_log.h"
#include "utilities.h"
#include "static_checks.h"
```

Include dependency graph for faults_cat1b.c:

### Functions

- void c_cpuss_interrupt_msc_IRQn_Handler (void)
- void cpuss_interrupt_msc_IRQn_Handler (void)
- void c_cpuss_interrupts_fault_0_IRQn_Handler (void)
- void cpuss_interrupts_fault_0_IRQn_Handler (void)
- FIH_RET_TYPE (enum tfm_plat_err_t) ifx_faults_platform_interrupt_enable(void)

  *Configures fault handlers and sets priorities.*

### 7.86.1 Function Documentation

#### 7.86.1.1 c_cpuss_interrupt_msc_IRQn_Handler()

```
void c_cpuss_interrupt_msc_IRQn_Handler (
            void  )
```

Definition at line 21 of file faults_cat1b.c.

Here is the call graph for this function:

#### 7.86.1.2 c_cpuss_interrupts_fault_0_IRQn_Handler()

```
void c_cpuss_interrupts_fault_0_IRQn_Handler (
            void  )
```

Definition at line 49 of file faults_cat1b.c.
Here is the call graph for this function:

#### 7.86.1.3 cpuss_interrupt_msc_IRQn_Handler()

```
void cpuss_interrupt_msc_IRQn_Handler (
            void  )
```

Definition at line 38 of file faults_cat1b.c.

#### 7.86.1.4 cpuss_interrupts_fault_0_IRQn_Handler()

```
void cpuss_interrupts_fault_0_IRQn_Handler (
            void  )
```

Definition at line 62 of file faults_cat1b.c.

#### 7.86.1.5 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_plat_err_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.

This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Definition at line 72 of file faults_cat1b.c.
Here is the call graph for this function:

## 7.87 platform/ext/target/infineon/common/spe/faults/faults_cat1d.c File Reference

```
#include "config_tfm.h"
#include "cmsis.h"
#include "cy_sysfault.h"
#include "exception_info.h"
#include "faults.h"
#include "faults_dump.h"
#include "ifx_utils.h"
#include "tfm_spm_log.h"
#include "utilities.h"
```

Include dependency graph for faults_cat1d.c:

## Functions

- void c_m33syscpuss_interrupt_msc_IRQn_Handler (void)
- void m33syscpuss_interrupt_msc_IRQn_Handler (void)
- void c_m33syscpuss_interrupts_fault_0_IRQn_Handler (void)
- void m33syscpuss_interrupts_fault_0_IRQn_Handler (void)
- FIH_RET_TYPE (enum tfm_plat_err_t) ifx_faults_platform_interrupt_enable(void)

  *Configures fault handlers and sets priorities.*

## 7.87.1 Function Documentation

### 7.87.1.1 c_m33syscpuss_interrupt_msc_IRQn_Handler()

```
void c_m33syscpuss_interrupt_msc_IRQn_Handler (
            void  )
```

Definition at line 19 of file faults_cat1d.c.
Here is the call graph for this function:

### 7.87.1.2 c_m33syscpuss_interrupts_fault_0_IRQn_Handler()

```
void c_m33syscpuss_interrupts_fault_0_IRQn_Handler (
            void  )
```

Definition at line 45 of file faults_cat1d.c.
Here is the call graph for this function:

### 7.87.1.3 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_plat_err_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Definition at line 67 of file faults_cat1d.c.
Here is the call graph for this function:

#### 7.87.1.4 m33syscpuss_interrupt_msc_IRQn_Handler()

```
void m33syscpuss_interrupt_msc_IRQn_Handler (
            void  )
```

Definition at line 35 of file faults_cat1d.c.

#### 7.87.1.5 m33syscpuss_interrupts_fault_0_IRQn_Handler()

```
void m33syscpuss_interrupts_fault_0_IRQn_Handler (
            void  )
```

Definition at line 57 of file faults_cat1d.c.

## 7.88 platform/ext/target/infineon/common/spe/faults/faults_cat1e.c File Reference

```
#include "config_tfm.h"
#include "cmsis.h"
#include "cy_sysfault.h"
#include "exception_info.h"
#include "faults.h"
#include "faults_dump.h"
#include "ifx_utils.h"
#include "tfm_spm_log.h"
```

```
#include "utilities.h"
```

Include dependency graph for faults_cat1e.c:

## Macros

- #define CY_SYSFAULT_NO_FAULT ((uint8_t)PERI_0_PERI_GP4_AHB_VIO + 1U)

## Functions

- cy_en_SysFault_status_t Cy_SysFault_Init (FAULT_STRUCT_Type *base, cy_stc_SysFault_t *config)
- void Cy_SysFault_ClearStatus (FAULT_STRUCT_Type *base)
- cy_en_SysFault_source_t Cy_SysFault_GetErrorSource (FAULT_STRUCT_Type *base)
- uint32_t Cy_SysFault_GetFaultData (FAULT_STRUCT_Type *base, cy_en_SysFault_Data_t dataSet)
- void Cy_SysFault_SetMaskByIdx (FAULT_STRUCT_Type *base, cy_en_SysFault_source_t idx)
- void Cy_SysFault_ClearInterrupt (FAULT_STRUCT_Type *base)
- void Cy_SysFault_SetInterruptMask (FAULT_STRUCT_Type *base)
- void c_m33syscpuss_interrupt_msc_IRQn_Handler (void)
- void m33syscpuss_interrupt_msc_IRQn_Handler (void)
- void c_m33syscpuss_interrupts_fault_0_IRQn_Handler (void)
- void m33syscpuss_interrupts_fault_0_IRQn_Handler (void)
- FIH_RET_TYPE (enum tfm_plat_err_t) ifx_faults_platform_interrupt_enable(void)

  *Configures fault handlers and sets priorities.*

## 7.88.1 Macro Definition Documentation

### 7.88.1.1 CY_SYSFAULT_NO_FAULT

```
#define CY_SYSFAULT_NO_FAULT ((uint8_t)PERI_0_PERI_GP4_AHB_VIO + 1U)
```

Definition at line 21 of file faults_cat1e.c.

## 7.88.2 Function Documentation

### 7.88.2.1 c_m33syscpuss_interrupt_msc_IRQn_Handler()

```
void c_m33syscpuss_interrupt_msc_IRQn_Handler (
            void  )
```

Definition at line 33 of file faults_cat1e.c.

Here is the call graph for this function:

#### 7.88.2.2 c_m33syscpuss_interrupts_fault_0_IRQn_Handler()

```
void c_m33syscpuss_interrupts_fault_0_IRQn_Handler (
            void  )
```

Definition at line 59 of file faults_cat1e.c.
Here is the call graph for this function:

#### 7.88.2.3 Cy_SysFault_ClearInterrupt()

```
void Cy_SysFault_ClearInterrupt (
            FAULT_STRUCT_Type * base )
```

Definition at line 29 of file faults_cat1e.c.
Here is the caller graph for this function:

#### 7.88.2.4 Cy_SysFault_ClearStatus()

```
void Cy_SysFault_ClearStatus (
            FAULT_STRUCT_Type * base )
```

Definition at line 25 of file faults_cat1e.c.
Here is the caller graph for this function:

#### 7.88.2.5 Cy_SysFault_GetErrorSource()

```
cy_en_SysFault_source_t Cy_SysFault_GetErrorSource (
            FAULT_STRUCT_Type * base )
```

Definition at line 26 of file faults_cat1e.c.
Here is the caller graph for this function:

#### 7.88.2.6 Cy_SysFault_GetFaultData()

```
uint32_t Cy_SysFault_GetFaultData (
            FAULT_STRUCT_Type * base,
            cy_en_SysFault_Data_t dataSet )
```

Definition at line 27 of file faults_cat1e.c.

Here is the caller graph for this function:

#### 7.88.2.7 Cy_SysFault_Init()

```
cy_en_SysFault_status_t Cy_SysFault_Init (
            FAULT_STRUCT_Type * base,
            cy_stc_SysFault_t * config )
```

Definition at line 24 of file faults_cat1e.c.

Here is the caller graph for this function:

#### 7.88.2.8 Cy_SysFault_SetInterruptMask()

```
void Cy_SysFault_SetInterruptMask (
            FAULT_STRUCT_Type * base )
```

Definition at line 30 of file faults_cat1e.c.
Here is the caller graph for this function:

#### 7.88.2.9 Cy_SysFault_SetMaskByIdx()

```
void Cy_SysFault_SetMaskByIdx (
            FAULT_STRUCT_Type * base,
            cy_en_SysFault_source_t idx )
```

Definition at line 28 of file faults_cat1e.c.
Here is the caller graph for this function:

#### 7.88.2.10 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_plat_err_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.

Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Definition at line 81 of file faults_cat1e.c.
Here is the call graph for this function:

#### 7.88.2.11 m33syscpuss_interrupt_msc_IRQn_Handler()

```
void m33syscpuss_interrupt_msc_IRQn_Handler (
            void  )
```

Definition at line 49 of file faults_cat1e.c.

#### 7.88.2.12 m33syscpuss_interrupts_fault_0_IRQn_Handler()

```
void m33syscpuss_interrupts_fault_0_IRQn_Handler (
            void  )
```

Definition at line 71 of file faults_cat1e.c.

## 7.89 platform/ext/target/infineon/common/spe/faults/faults_dump.h File Reference

```
#include "cmsis_compiler.h"
#include "cy_sysfault.h"
#include "ifx_utils.h"
#include "tfm_spm_log.h"
```

Include dependency graph for faults_dump.h:

This graph shows which files directly or indirectly include this file:

## Functions

- __STATIC_FORCEINLINE void ifx_faults_dump_peri_msx_ppc_vio_fault (FAULT_STRUCT_Type *base)

  *Print PPC violation fault information captured by fault peripheral when a locked PC tries to write to PC_MASK register.*
- __STATIC_FORCEINLINE void ifx_faults_dump_peri_ppc_pc_mask_vio_fault (FAULT_STRUCT_Type *base)

  *Print PPC MMIO access violation fault information captured by fault peripheral.*
- __STATIC_FORCEINLINE void ifx_faults_dump_peri_gpx_timeout_vio_fault (FAULT_STRUCT_Type *base)

  *Print peripheral group timeout violation fault information captured by fault peripheral.*
- __STATIC_FORCEINLINE void ifx_faults_dump_peri_gpx_ahb_vio_fault (FAULT_STRUCT_Type *base)

  *Print peripheral group AHB violation fault information captured by fault peripheral.*
- __STATIC_FORCEINLINE void ifx_faults_dump_mpc_fault (FAULT_STRUCT_Type *base)

  *Print MPC fault information captured by fault peripheral.*
- __STATIC_FORCEINLINE void ifx_faults_dump_default_fault (FAULT_STRUCT_Type *base)

  *Print fault information captured by fault peripheral.*

## 7.89.1 Function Documentation

### 7.89.1.1 ifx_faults_dump_default_fault()

```
__STATIC_FORCEINLINE void ifx_faults_dump_default_fault (
            FAULT_STRUCT_Type * base )
```

Print fault information captured by fault peripheral.

**Parameters**

| | | |
|---|---|---|
| in | *base* | Fault structure with captured fault |

Definition at line 233 of file faults_dump.h.
Here is the call graph for this function:

#### 7.89.1.2 ifx_faults_dump_mpc_fault()

```
__STATIC_FORCEINLINE void ifx_faults_dump_mpc_fault (
            FAULT_STRUCT_Type * base )
```

Print MPC fault information captured by fault peripheral.

**Parameters**

| | | |
|---|---|---|
| in | *base* | Fault structure with captured fault |

Definition at line 179 of file faults_dump.h.
Here is the call graph for this function:

#### 7.89.1.3 ifx_faults_dump_peri_gpx_ahb_vio_fault()

```
__STATIC_FORCEINLINE void ifx_faults_dump_peri_gpx_ahb_vio_fault (
            FAULT_STRUCT_Type * base )
```

Print peripheral group AHB violation fault information captured by fault peripheral.

**Parameters**

| | | |
|---|---|---|
| in | *base* | Fault structure with captured fault |

Definition at line 126 of file faults_dump.h.

Here is the call graph for this function:

#### 7.89.1.4 ifx_faults_dump_peri_gpx_timeout_vio_fault()

```
__STATIC_FORCEINLINE void ifx_faults_dump_peri_gpx_timeout_vio_fault (
            FAULT_STRUCT_Type * base )
```

Print peripheral group timeout violation fault information captured by fault peripheral.

**Parameters**

| | | |
|---|---|---|
| in | *base* | Fault structure with captured fault |

Definition at line 105 of file faults_dump.h.
Here is the call graph for this function:

#### 7.89.1.5 ifx_faults_dump_peri_msx_ppc_vio_fault()

```
__STATIC_FORCEINLINE void ifx_faults_dump_peri_msx_ppc_vio_fault (
            FAULT_STRUCT_Type * base )
```

Print PPC violation fault information captured by fault peripheral when a locked PC tries to write to PC_MASK register.

**Parameters**

| | | |
|---|---|---|
| in | *base* | Fault structure with captured fault |

Definition at line 23 of file faults_dump.h.

Here is the call graph for this function:

#### 7.89.1.6 ifx_faults_dump_peri_ppc_pc_mask_vio_fault()

```
__STATIC_FORCEINLINE void ifx_faults_dump_peri_ppc_pc_mask_vio_fault (
            FAULT_STRUCT_Type * base )
```

Print PPC MMIO access violation fault information captured by fault peripheral.
It's used to report PPC violation when a locked PC tries to write to PC_MASK register.

**Parameters**

| | | |
|---|---|---|
| in | *base* | Fault structure with captured fault |

Definition at line 70 of file faults_dump.h.
Here is the call graph for this function:

## 7.90 platform/ext/target/infineon/common/spe/fih/tfm_fih_mxcrypto.c File Reference

```
#include "tfm_fih_rng.h"
#include "cy_device.h"
#include "cy_crypto_core_trng.h"
```

Include dependency graph for tfm_fih_mxcrypto.c:

## Macros

- #define PRNG_A 13
- #define PRNG_B 17
- #define PRNG_C 5

## Functions

- fih_int tfm_fih_random_init (void)
- void tfm_fih_random_generate (uint8_t *rand)

## 7.90.1 Macro Definition Documentation

### 7.90.1.1 PRNG_A

```
#define PRNG_A 13
```

Definition at line 13 of file tfm_fih_mxcrypto.c.

### 7.90.1.2 PRNG_B

```
#define PRNG_B 17
```

Definition at line 14 of file tfm_fih_mxcrypto.c.

### 7.90.1.3 PRNG_C

```
#define PRNG_C 5
```

Definition at line 15 of file tfm_fih_mxcrypto.c.

## 7.90.2 Function Documentation

### 7.90.2.1 tfm_fih_random_generate()

```
void tfm_fih_random_generate (
            uint8_t * rand )
```

Definition at line 87 of file tfm_fih_mxcrypto.c.

#### 7.90.2.2 tfm_fih_random_init()

```
fih_int tfm_fih_random_init (
            void  )
```

Definition at line 80 of file tfm_fih_mxcrypto.c.

## 7.91 platform/ext/target/infineon/common/spe/otp/otp_flash.c File Reference

```
#include "config_tfm.h"
#include "provisioning.h"
#include "tfm_plat_otp.h"
#include "static_checks.h"
```

Include dependency graph for otp_flash.c:

### Functions

- enum tfm_plat_err_t tfm_plat_otp_init (void)
- enum tfm_plat_err_t tfm_plat_otp_read (enum tfm_otp_element_id_t id, size_t out_len, uint8_t *out)
- enum tfm_plat_err_t tfm_plat_otp_write (enum tfm_otp_element_id_t id, size_t in_len, const uint8_t *in)
- enum tfm_plat_err_t tfm_plat_otp_get_size (enum tfm_otp_element_id_t id, size_t *size)

### 7.91.1 Function Documentation

#### 7.91.1.1 tfm_plat_otp_get_size()

```
enum tfm_plat_err_t tfm_plat_otp_get_size (
            enum tfm_otp_element_id_t id,
            size_t * size )
```

Definition at line 67 of file otp_flash.c.

#### 7.91.1.2 tfm_plat_otp_init()

```
enum tfm_plat_err_t tfm_plat_otp_init (
            void  )
```

Definition at line 14 of file otp_flash.c.

#### 7.91.1.3 tfm_plat_otp_read()

```
enum tfm_plat_err_t tfm_plat_otp_read (
            enum tfm_otp_element_id_t id,
            size_t out_len,
            uint8_t * out )
```

Definition at line 19 of file otp_flash.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.91.1.4 tfm_plat_otp_write()

```
enum tfm_plat_err_t tfm_plat_otp_write (
            enum tfm_otp_element_id_t id,
            size_t in_len,
            const uint8_t * in )
```

Definition at line 57 of file otp_flash.c.

## 7.92 platform/ext/target/infineon/common/spe/protection/partition_↩ assets.h File Reference

```
#include <stddef.h>
#include "array.h"
#include "load/asset_defs.h"
#include "region_defs.h"
```

Include dependency graph for partition_assets.h:

## 7.93 platform/ext/target/infineon/common/spe/protection/partition_info.h File Reference

```
#include "load/partition_defs.h"
#include "load/asset_defs.h"
#include "protection_types.h"
```

Include dependency graph for partition_info.h:

This graph shows which files directly or indirectly include this file:

## Functions

- const ifx_partition_info_t ∗ ifx_protection_get_partition_info (const struct partition_load_info_t ∗p_ldinfo)

  *Return platform specific partition info.*

### 7.93.1 Function Documentation

#### 7.93.1.1 ifx_protection_get_partition_info()

```
const ifx_partition_info_t* ifx_protection_get_partition_info (
            const struct partition_load_info_t * p_ldinfo )
```

Return platform specific partition info.

**Parameters**

| | | |
|---|---|---|
| in | *p_ldinfo* | Partition load info. This function uses partition id to lookup for partition properties. |

It's expected that it will be used only once to bind partition boundary by tfm_hal_bind_boundary. In all other cases boundary should be converted to constant pointer to ifx_partition_info_t.

**Returns**

Pointer to platform specific partition info

## 7.94 platform/ext/target/infineon/common/spe/protection/protection_↩ mpc_api.h File Reference

This file contains MPC driver API declaration.

```
#include "fih.h"
#include "load/asset_defs.h"
#include "protection_types.h"
```

```
#include "tfm_hal_defs.h"
```

Include dependency graph for protection_mpc_api.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpc_init_cfg(void)

  *Configures the Memory Protection Controller.*

## Variables

- uintptr_t base
- uintptr_t size_t size
- uintptr_t size_t uint32_t access_type

### 7.94.1 Detailed Description

This file contains MPC driver API declaration.
It's expected that MPC driver implementation follows API declared in this file.

### 7.94.2 Function Documentation

#### 7.94.2.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures the Memory Protection Controller.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Definition at line 67 of file faults.c.

### 7.94.3 Variable Documentation

#### 7.94.3.1 access_type

```
uintptr_t size_t uint32_t access_type
```

Definition at line 82 of file protection_mpc_api.h.

#### 7.94.3.2 base

```
uintptr_t base
```

Definition at line 80 of file protection_mpc_api.h.

#### 7.94.3.3 size

```
uintptr_t size_t size
```

Definition at line 81 of file protection_mpc_api.h.

## 7.95 platform/ext/target/infineon/common/spe/protection/protection_↩mpc_hw_mpc.c File Reference

```
#include "config_impl.h"
#include "config_tfm.h"
```

```
#include "cy_mpc.h"
#include "region_defs.h"
#include "protection_types.h"
#include "protection_utils.h"
#include "protection_regions_cfg.h"
#include "region.h"
#include "ifx_regions.h"
#include "ifx_utils.h"
#include "utilities.h"
#include "static_checks.h"
```

Include dependency graph for protection_mpc_hw_mpc.c:

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpc_fill_fixed_config(void)

  *Configures fault handlers and sets priorities.*

- FIH_CALL (ifx_mpc_apply_configuration, fih_rc, &mpc_reg_cfg)
- FIH_RET (fih_rc)

## Variables

- ifx_mpc_region_config_t ifx_fixed_mpc_static_config [IFX_MAX_FIXED_CONFIGS]
- uint32_t ifx_fixed_mpc_static_config_count = 0
- const struct asset_desc_t ∗ asset
- ifx_mpc_region_config_t mpc_reg_cfg
- mpc_reg_cfg address = asset->mem.start
- mpc_reg_cfg size = asset->mem.limit - asset->mem.start
- mpc_reg_cfg ns_mask = mpc_cfg->ns_mask
- mpc_reg_cfg r_mask = mpc_cfg->pc_mask
- mpc_reg_cfg w_mask = ((asset->attr & ASSET_ATTR_READ_WRITE) != 0U) ? mpc_cfg->pc_mask : 0U

### 7.95.1 Function Documentation

#### 7.95.1.1 FIH_CALL()

```
FIH_CALL (
            ifx_mpc_apply_configuration ,
            fih_rc ,
            & mpc_reg_cfg )
```

Here is the caller graph for this function:

### 7.95.1.2 FIH_RET()

```
FIH_RET (
            fih_rc  )
```

Here is the caller graph for this function:

### 7.95.1.3 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.

Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Definition at line 28 of file protection_mpc_hw_mpc.c.
Here is the call graph for this function:

## 7.95.2 Variable Documentation

### 7.95.2.1 address

```
mpc_reg_cfg address = asset->mem.start
```

Definition at line 145 of file protection_mpc_hw_mpc.c.

#### 7.95.2.2 asset

```
const struct asset_desc_t* asset
```

**Initial value:**

```
{
    FIH_RET_TYPE(enum tfm_hal_status_t) fih_rc = fih_int_encode(TFM_HAL_ERROR_GENERIC)
```

Definition at line 141 of file protection_mpc_hw_mpc.c.

#### 7.95.2.3 ifx_fixed_mpc_static_config

```
ifx_mpc_region_config_t ifx_fixed_mpc_static_config[IFX_MAX_FIXED_CONFIGS]
```

Definition at line 25 of file protection_mpc_hw_mpc.c.

#### 7.95.2.4 ifx_fixed_mpc_static_config_count

```
uint32_t ifx_fixed_mpc_static_config_count = 0
```

Definition at line 26 of file protection_mpc_hw_mpc.c.

#### 7.95.2.5 mpc_reg_cfg

```
ifx_mpc_region_config_t mpc_reg_cfg
```

Definition at line 143 of file protection_mpc_hw_mpc.c.

#### 7.95.2.6 ns_mask

```
mpc_reg_cfg ns_mask = mpc_cfg->ns_mask
```

Definition at line 148 of file protection_mpc_hw_mpc.c.

#### 7.95.2.7 r_mask

```
mpc_reg_cfg r_mask = mpc_cfg->pc_mask
```

Definition at line 150 of file protection_mpc_hw_mpc.c.

#### 7.95.2.8 size

```
mpc_reg_cfg size = asset->mem.limit - asset->mem.start
```

Definition at line 146 of file protection_mpc_hw_mpc.c.

#### 7.95.2.9 w_mask

```
mpc_reg_cfg w_mask = ((asset->attr & ASSET_ATTR_READ_WRITE) != 0U) ? mpc_cfg->pc_mask :  0U
```

Definition at line 151 of file protection_mpc_hw_mpc.c.

## 7.96 platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc.h File Reference

This file contains types/API used by HW MPC driver, see cy_mpc.h in PDL.

```
#include <stdint.h>
#include "config_impl.h"
```

```
#include "config_tfm.h"
#include "cy_mpc.h"
#include "fih.h"
#include "ifx_platform_spe_types.h"
#include "ifx_spe_config.h"
```

Include dependency graph for protection_mpc_hw_mpc.h:

## Data Structures

- struct ifx_mpc_numbered_mmio_config_t
- struct ifx_mpc_region_config_t
- struct ifx_mpc_raw_region_config_t

  *Configuration structure for MPC raw region.*

## Macros

- #define IFX_MPC_NAMED_MMIO_SPM_ROT_CFG
- #define IFX_MPC_NAMED_MMIO_PSA_ROT_CFG
- #define IFX_MPC_NAMED_MMIO_APP_ROT_CFG
- #define IFX_MAX_FIXED_CONFIGS 5U

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpc_fill_fixed_config(void)

  *Populate the internal static MPC configuration array.*
- enum tfm_hal_status_t ifx_mpc_apply_configuration_with_mpc (const ifx_mpc_raw_region_config_t *mpc_reg_cfg)

  *Apply MPC configuration for MPC controller handled by TF-M.*

## Variables

- ifx_mpc_region_config_t ifx_fixed_mpc_static_config [5U]
- uint32_t ifx_fixed_mpc_static_config_count

### 7.96.1 Detailed Description

This file contains types/API used by HW MPC driver, see cy_mpc.h in PDL.

**Note**

Don't include this file directly, include protection_types.h instead !!! It's expected that this file is included by protection_types.h if platform is configured to use HW MPC driver via IFX_MPC_DRIVER_HW_MPC_V1 or IFX_MPC_DRIVER_HW_MPC_V2 option.

## 7.96.2 Macro Definition Documentation

### 7.96.2.1 IFX_MAX_FIXED_CONFIGS

```
#define IFX_MAX_FIXED_CONFIGS 5U
```

Definition at line 113 of file protection_mpc_hw_mpc.h.

### 7.96.2.2 IFX_MPC_NAMED_MMIO_APP_ROT_CFG

```
#define IFX_MPC_NAMED_MMIO_APP_ROT_CFG
```

**Value:**

```
    .mpc_cfg = { \
        .pc_mask        = IFX_PC_DEFAULT | IFX_PC_TFM_AROT, \
        .ns_mask        = IFX_PC_NONE, \
    },
```

Definition at line 88 of file protection_mpc_hw_mpc.h.

### 7.96.2.3 IFX_MPC_NAMED_MMIO_PSA_ROT_CFG

```
#define IFX_MPC_NAMED_MMIO_PSA_ROT_CFG
```

**Value:**

```
    .mpc_cfg = { \
        .pc_mask        = IFX_PC_DEFAULT | IFX_PC_TFM_PROT, \
        .ns_mask        = IFX_PC_NONE, \
    },
```

Definition at line 72 of file protection_mpc_hw_mpc.h.

### 7.96.2.4 IFX_MPC_NAMED_MMIO_SPM_ROT_CFG

```
#define IFX_MPC_NAMED_MMIO_SPM_ROT_CFG
```

**Value:**

```
    .mpc_cfg = { \
        .pc_mask        = IFX_PC_DEFAULT, \
        .ns_mask        = IFX_PC_NONE, \
    },
```

Definition at line 65 of file protection_mpc_hw_mpc.h.

## 7.96.3 Function Documentation

### 7.96.3.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Populate the internal static MPC configuration array.
Apply MPC configuration using raw interface.
Apply MPC configuration.
This function fills the ifx_fixed_mpc_static_config array with additional MPC region configurations that are internal to TF-M and cannot be determined by the compiler as constants. It also sets the ifx_fixed_mpc_static_config_count to the number of valid entries. The function may call tfm_core_panic() if the number of configurations exceeds IFX_MAX_FIXED_CONFIGS.

**Note**

The actual regions and permissions depend on build-time configuration and linker symbols.

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_INPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 67 of file faults.c.

### 7.96.3.2 ifx_mpc_apply_configuration_with_mpc()

```
enum tfm_hal_status_t ifx_mpc_apply_configuration_with_mpc (
            const ifx_mpc_raw_region_config_t * mpc_reg_cfg )
```

Apply MPC configuration for MPC controller handled by TF-M.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration.

Definition at line 43 of file protection_mpc_hw_mpc_v1.c.

### 7.96.4 Variable Documentation

#### 7.96.4.1 ifx_fixed_mpc_static_config

```
ifx_mpc_region_config_t ifx_fixed_mpc_static_config[5U]
```

Definition at line 25 of file protection_mpc_hw_mpc.c.

#### 7.96.4.2 ifx_fixed_mpc_static_config_count

```
uint32_t ifx_fixed_mpc_static_config_count
```

Definition at line 26 of file protection_mpc_hw_mpc.c.

## 7.97 platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc_v1.c File Reference

```
#include "config_impl.h"
#include "config_tfm.h"
#include "cy_mpc.h"
#include "fih.h"
#include "region_defs.h"
#include "protection_types.h"
#include "protection_utils.h"
#include "protection_regions_cfg.h"
#include "protection_mpc_api.h"
#include "region.h"
#include "ifx_platform_mailbox.h"
#include "ifx_regions.h"
#include "ifx_utils.h"
#include "tfm_hal_isolation.h"
#include "static_checks.h"
```

Include dependency graph for protection_mpc_hw_mpc_v1.c:

### Macros

- #define IFX_MPC_BLOCK_SIZE_TO_BYTES(mpc_size) (1UL << ((uint32_t)(mpc_size) + 5UL))

### Functions

- enum tfm_hal_status_t ifx_mpc_apply_configuration_with_mpc (const ifx_mpc_raw_region_config_t *mpc_reg_cfg)

*Apply MPC configuration for MPC controller handled by TF-M.*

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpc_init_cfg(void)

  *Configures fault handlers and sets priorities.*

- if (mem_region_cfg==NULL)
- if (fih_int_validate(pc)==0)
- if (IFX_MPC_IS_EXTERNAL(mem_region_cfg->mpc))
- for (uint32_t idx=first_block_idx;idx< last_block_idx;idx++)
- FIH_RET (fih_int_encode(TFM_HAL_SUCCESS))

## Variables

- uintptr_t base
- uintptr_t size_t size
- uintptr_t size_t uint32_t access_type
- mem_region_cfg
- uint32_t block_size = (1UL << ((uint32_t)( mem_region_cfg->mpc_block_size ) + 5UL))
- uint32_t offset = IFX_S_ADDRESS_ALIAS(base) - mem_region_cfg->s_address
- uint32_t first_block_idx = offset / block_size
- uint32_t last_block_idx = IFX_ROUND_UP_TO_MULTIPLE(offset + size, block_size) / block_size
- fih_int is_secure = ((access_type & TFM_HAL_ACCESS_NS) == 0U) ? FIH_SUCCESS : FIH_FAILURE
- fih_int pc = FIH_INVALID
- cy_en_mpc_sec_attr_t expected_sec_attr = (IS_NS_AGENT(p_info->p_ldinfo) && fih_eq(is_secure, FIH_↩ FAILURE)) ? CY_MPC_NON_SECURE : CY_MPC_SECURE

## 7.97.1 Macro Definition Documentation

### 7.97.1.1 IFX_MPC_BLOCK_SIZE_TO_BYTES

```
#define IFX_MPC_BLOCK_SIZE_TO_BYTES(
            mpc_size ) (1UL << ((uint32_t)(mpc_size) + 5UL))
```

Definition at line 28 of file protection_mpc_hw_mpc_v1.c.

## 7.97.2 Function Documentation

### 7.97.2.1 FIH_RET()

```
FIH_RET (
            fih_int_encode(TFM_HAL_SUCCESS)  )
```

Here is the caller graph for this function:

#### 7.97.2.2 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |

**Parameters**

| | | |
|---|---|---|
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_INPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_INPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

Definition at line 122 of file protection_mpc_hw_mpc_v1.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.97.2.3 for()

```
for ( )
```

Definition at line 484 of file protection_mpc_hw_mpc_v1.c.

Here is the call graph for this function:

#### 7.97.2.4 if() [1/3]

```
if (
            fih_int_validate(pc)  = = 0 )
```

Definition at line 441 of file protection_mpc_hw_mpc_v1.c.
Here is the call graph for this function:

#### 7.97.2.5 if() [2/3]

```
if (
            IFX_MPC_IS_EXTERNAL(mem_region_cfg->mpc)  )
```

Definition at line 463 of file protection_mpc_hw_mpc_v1.c.

Here is the call graph for this function:

#### 7.97.2.6 if() [3/3]

```
if (
            mem_region_cfg  = = NULL )
```

Definition at line 426 of file protection_mpc_hw_mpc_v1.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.97.2.7 ifx_mpc_apply_configuration_with_mpc()

```
enum tfm_hal_status_t ifx_mpc_apply_configuration_with_mpc (
            const ifx_mpc_raw_region_config_t * mpc_reg_cfg )
```

Apply MPC configuration for MPC controller handled by TF-M.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration.

Definition at line 43 of file protection_mpc_hw_mpc_v1.c.

### 7.97.3 Variable Documentation

#### 7.97.3.1 access_type

```
uintptr_t size_t uint32_t access_type
```

**Initial value:**

```
{
    const ifx_memory_config_t* mem_region_cfg
```

Definition at line 369 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.2 base

```
uintptr_t base
```

Definition at line 366 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.3 block_size

```
uint32_t block_size = (1UL << ((uint32_t)( mem_region_cfg->mpc_block_size ) + 5UL))
```

Definition at line 430 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.4 expected_sec_attr

```
cy_en_mpc_sec_attr_t expected_sec_attr = (IS_NS_AGENT(p_info->p_ldinfo) && fih_eq(is_secure,
FIH_FAILURE)) ? CY_MPC_NON_SECURE : CY_MPC_SECURE
```

Definition at line 446 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.5 first_block_idx

```
uint32_t first_block_idx = offset / block_size
```

Definition at line 432 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.6 is_secure

```
fih_int is_secure = ((access_type & TFM_HAL_ACCESS_NS) == 0U) ? FIH_SUCCESS : FIH_FAILURE
```

Definition at line 434 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.7 last_block_idx

```
uint32_t last_block_idx = IFX_ROUND_UP_TO_MULTIPLE(offset + size, block_size) / block_size
```

Definition at line 433 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.8 mem_region_cfg

```
mem_region_cfg
```

**Initial value:**

```
= ifx_find_memory_config(base, size,
                                                ifx_memory_cm33_config,
                                                ifx_memory_cm33_config_count)
```

Definition at line 412 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.9 offset

```
uint32_t offset = IFX_S_ADDRESS_ALIAS(base) - mem_region_cfg->s_address
```

Definition at line 431 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.10 pc

```
pc = FIH_INVALID
```

Definition at line 436 of file protection_mpc_hw_mpc_v1.c.

#### 7.97.3.11 size

```
uintptr_t size_t size
```

Definition at line 367 of file protection_mpc_hw_mpc_v1.c.

## 7.98 platform/ext/target/infineon/common/spe/protection/protection_mpc_hw_mpc_v2.c File Reference

```
#include "config_impl.h"
#include "config_tfm.h"
#include "cy_mpc.h"
#include "fih.h"
#include "region_defs.h"
#include "protection_types.h"
#include "protection_utils.h"
#include "protection_regions_cfg.h"
#include "protection_mpc_api.h"
#include "region.h"
#include "ifx_platform_mailbox.h"
#include "ifx_regions.h"
#include "ifx_utils.h"
#include "tfm_hal_isolation.h"
#include "static_checks.h"
```

Include dependency graph for protection_mpc_hw_mpc_v2.c:

### Macros

- #define IFX_MPC_BLOCK_SIZE_TO_BYTES(mpc_size) (1UL << ((uint32_t)(mpc_size) + 5UL))
- #define IFX_MPC_ROT_BLK_CFG_BLOCK_SIZE_Pos 0UL
- #define IFX_MPC_ROT_BLK_CFG_BLOCK_SIZE_Msk 0xFUL

### Functions

- enum tfm_hal_status_t ifx_mpc_apply_configuration_with_mpc (const ifx_mpc_raw_region_config_t *mpc_reg_cfg)

  *Apply MPC configuration for MPC controller handled by TF-M.*
- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpc_init_cfg(void)

  *Configures fault handlers and sets priorities.*
- if (mem_region_cfg==NULL)
- if (fih_int_validate(pc)==0)
- if (IFX_MPC_IS_EXTERNAL(mem_region_cfg->mpc))
- for (uint32_t block_offset=first_block_idx *block_size;block_offset< last_offset;block_offset+=block_size)
- FIH_RET (fih_int_encode(TFM_HAL_SUCCESS))

### Variables

- uintptr_t base
- uintptr_t size_t size
- uintptr_t size_t uint32_t access_type
- mem_region_cfg
- uint32_t block_size = (1UL << ((uint32_t)( mem_region_cfg->mpc_block_size ) + 5UL))
- uint32_t offset = IFX_S_ADDRESS_ALIAS(base) - mem_region_cfg->s_address
- uint32_t first_block_idx = offset / block_size

- uint32_t last_block_idx = IFX_ROUND_UP_TO_MULTIPLE(offset + size, block_size) / block_size
- uint32_t last_offset = last_block_idx ∗ block_size
- fih_int is_secure = ((access_type & TFM_HAL_ACCESS_NS) == 0U) ? FIH_SUCCESS : FIH_FAILURE
- fih_int pc = FIH_INVALID
- cy_en_mpc_sec_attr_t expected_sec_attr = (IS_NS_AGENT(p_info->p_ldinfo) && fih_eq(is_secure, FIH_↩ FAILURE)) ? CY_MPC_NON_SECURE : CY_MPC_SECURE

## 7.98.1 Macro Definition Documentation

### 7.98.1.1 IFX_MPC_BLOCK_SIZE_TO_BYTES

```
#define IFX_MPC_BLOCK_SIZE_TO_BYTES(
            mpc_size ) (1UL << ((uint32_t)(mpc_size) + 5UL))
```

Definition at line 50 of file protection_mpc_hw_mpc_v2.c.

### 7.98.1.2 IFX_MPC_ROT_BLK_CFG_BLOCK_SIZE_Msk

```
#define IFX_MPC_ROT_BLK_CFG_BLOCK_SIZE_Msk 0xFUL
```

Definition at line 53 of file protection_mpc_hw_mpc_v2.c.

### 7.98.1.3 IFX_MPC_ROT_BLK_CFG_BLOCK_SIZE_Pos

```
#define IFX_MPC_ROT_BLK_CFG_BLOCK_SIZE_Pos 0UL
```

Definition at line 52 of file protection_mpc_hw_mpc_v2.c.

## 7.98.2 Function Documentation

### 7.98.2.1 FIH_RET()

```
FIH_RET (
            fih_int_encode(TFM_HAL_SUCCESS)  )
```

Here is the caller graph for this function:

#### 7.98.2.2 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

Definition at line 205 of file protection_mpc_hw_mpc_v2.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.98.2.3 for()

`for ( )`

Definition at line 574 of file protection_mpc_hw_mpc_v2.c.

Here is the call graph for this function:

#### 7.98.2.4 if() [1/3]

```
if (
            fih_int_validate(pc)  = = 0 )
```

Definition at line 530 of file protection_mpc_hw_mpc_v2.c.
Here is the call graph for this function:

#### 7.98.2.5 if() [2/3]

```
if (
            IFX_MPC_IS_EXTERNAL(mem_region_cfg->mpc)  )
```

Definition at line 552 of file protection_mpc_hw_mpc_v2.c.

Here is the call graph for this function:

#### 7.98.2.6 if() [3/3]

```
if (
            mem_region_cfg  = = NULL )
```

Definition at line 514 of file protection_mpc_hw_mpc_v2.c.
Here is the call graph for this function:

#### 7.98.2.7 ifx_mpc_apply_configuration_with_mpc()

```
enum tfm_hal_status_t ifx_mpc_apply_configuration_with_mpc (
            const ifx_mpc_raw_region_config_t * mpc_reg_cfg )
```

Apply MPC configuration for MPC controller handled by TF-M.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration.

Definition at line 127 of file protection_mpc_hw_mpc_v2.c.

## 7.98.3 Variable Documentation

### 7.98.3.1 access_type

`uintptr_t size_t uint32_t access_type`

**Initial value:**

```
{
    const ifx_memory_config_t* mem_region_cfg
```

Definition at line 457 of file protection_mpc_hw_mpc_v2.c.

### 7.98.3.2 base

`uintptr_t base`

Definition at line 454 of file protection_mpc_hw_mpc_v2.c.

### 7.98.3.3 block_size

`uint32_t block_size = (1UL << ((uint32_t)( mem_region_cfg->mpc_block_size ) + 5UL))`

Definition at line 518 of file protection_mpc_hw_mpc_v2.c.

### 7.98.3.4 expected_sec_attr

`cy_en_mpc_sec_attr_t expected_sec_attr = (IS_NS_AGENT(p_info->p_ldinfo) && fih_eq(is_secure, FIH_FAILURE)) ? CY_MPC_NON_SECURE : CY_MPC_SECURE`

Definition at line 535 of file protection_mpc_hw_mpc_v2.c.

### 7.98.3.5 first_block_idx

`uint32_t first_block_idx = offset / block_size`

Definition at line 520 of file protection_mpc_hw_mpc_v2.c.

### 7.98.3.6 is_secure

`fih_int is_secure = ((access_type & TFM_HAL_ACCESS_NS) == 0U) ? FIH_SUCCESS : FIH_FAILURE`

Definition at line 523 of file protection_mpc_hw_mpc_v2.c.

### 7.98.3.7 last_block_idx

`uint32_t last_block_idx = IFX_ROUND_UP_TO_MULTIPLE(offset + size, block_size) / block_size`

Definition at line 521 of file protection_mpc_hw_mpc_v2.c.

### 7.98.3.8 last_offset

`uint32_t last_offset = last_block_idx * block_size`

Definition at line 522 of file protection_mpc_hw_mpc_v2.c.

#### 7.98.3.9 mem_region_cfg

```
mem_region_cfg
```

**Initial value:**

```
= ifx_find_memory_config(base, size,
                                        ifx_memory_cm33_config,
                                        ifx_memory_cm33_config_count)
```

Definition at line 500 of file protection_mpc_hw_mpc_v2.c.

#### 7.98.3.10 offset

```
uint32_t offset = IFX_S_ADDRESS_ALIAS(base) - mem_region_cfg->s_address
```

Definition at line 519 of file protection_mpc_hw_mpc_v2.c.

#### 7.98.3.11 pc

```
pc = FIH_INVALID
```

Definition at line 525 of file protection_mpc_hw_mpc_v2.c.

#### 7.98.3.12 size

```
uintptr_t size_t size
```

Definition at line 455 of file protection_mpc_hw_mpc_v2.c.

## 7.99 platform/ext/target/infineon/common/spe/protection/protection_mpc_sert.c File Reference

```
#include "config_impl.h"
#include "config_tfm.h"
#include "cy_mpc.h"
#include "ifx_se_platform.h"
#include "ifx_se_tfm_utils.h"
#include "protection_mpc_sert.h"
#include "region_defs.h"
#include "static_checks.h"
#include "tfm_hal_isolation.h"
```

Include dependency graph for protection_mpc_sert.c:

### Data Structures

- struct ifx_mpc_ext_cache_cfg_info_t

### Macros

- #define IFX_MPC_EXT_BLOCK_SIZE_TO_BYTES(mpc_size) (1UL << ((uint32_t)(mpc_size) + 5UL))
- #define IFX_SE_RT_RRAM_SIZE IFX_RRAM_SIZE

- #define IFX_SE_RT_RRAM_BLOCK_COUNT (IFX_SE_RT_RRAM_SIZE / IFX_SE_RT_RRAM_BLOCK_SIZE)

## Functions

- enum tfm_hal_status_t ifx_mpc_sert_apply_configuration (const ifx_mpc_raw_region_config_t *mpc_reg_cfg)

  *Apply MPC configuration for MPC controller that is handled by optional SE RT Service or other non-TF-M service/hardware.*
- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpc_sert_memory_check(const struct ifx_partition_info_t *p_info
- if (IFX_SE_RT_RRAM_BLOCK_SIZE !=block_size)
- if ((first_block_idx >=(IFX_RRAM_SIZE/IFX_SE_RT_RRAM_BLOCK_SIZE))||(last_block_idx >(IFX_RRAM_SIZE/IFX_SE_RT
- if ((access_type &TFM_HAL_ACCESS_READABLE) !=0U)
- if ((access_type &TFM_HAL_ACCESS_WRITABLE) !=0U)
- for (uint32_t block_id=first_block_idx;block_id< last_block_idx;block_id++)
- FIH_RET (fih_int_encode(TFM_HAL_SUCCESS))

## Variables

- const ifx_memory_config_t * memory_config
- const ifx_memory_config_t uintptr_t base
- const ifx_memory_config_t uintptr_t size_t size
- const ifx_memory_config_t uintptr_t size_t uint32_t access_type
- uint32_t offset = IFX_S_ADDRESS_ALIAS(base) - memory_config->s_address
- uint32_t first_block_idx = offset / IFX_SE_RT_RRAM_BLOCK_SIZE
- uint32_t last_block_idx = IFX_ROUND_UP_TO_MULTIPLE(offset + size, IFX_SE_RT_RRAM_BLOCK_SIZE) / IFX_SE_RT_RRAM_BLOCK_SIZE
- bool is_secure = (access_type & TFM_HAL_ACCESS_NS) == 0U
- uint32_t pc = (uint32_t)fih_int_decode(fih_int_encode(IFX_PC_TFM_SPM_ID))
- uint32_t mask = 1UL << ((pc * 4u) + 0u)
- uint32_t attr = is_secure ? 0u : (1UL << ((pc * 4u) + 0u))

## 7.99.1 Macro Definition Documentation

### 7.99.1.1 IFX_MPC_EXT_BLOCK_SIZE_TO_BYTES

```
#define IFX_MPC_EXT_BLOCK_SIZE_TO_BYTES(
            mpc_size ) (1UL << ((uint32_t)(mpc_size) + 5UL))
```

Definition at line 19 of file protection_mpc_sert.c.

### 7.99.1.2 IFX_SE_RT_RRAM_BLOCK_COUNT

```
#define IFX_SE_RT_RRAM_BLOCK_COUNT (IFX_SE_RT_RRAM_SIZE / IFX_SE_RT_RRAM_BLOCK_SIZE)
```

Definition at line 25 of file protection_mpc_sert.c.

### 7.99.1.3 IFX_SE_RT_RRAM_SIZE

```
#define IFX_SE_RT_RRAM_SIZE IFX_RRAM_SIZE
```

Definition at line 22 of file protection_mpc_sert.c.

## 7.99.2 Function Documentation

#### 7.99.2.1 FIH_RET()

```
FIH_RET (
            fih_int_encode(TFM_HAL_SUCCESS)  )
```

Here is the caller graph for this function:

#### 7.99.2.2 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t ) const
```

Definition at line 241 of file protection_mpc_sert.c.

#### 7.99.2.3 for()

```
for ( )
```

Definition at line 232 of file protection_mpc_sert.c.

#### 7.99.2.4 if() [1/4]

```
if (
            (access_type &TFM_HAL_ACCESS_READABLE) !  = 0U )
```

Definition at line 222 of file protection_mpc_sert.c.

#### 7.99.2.5 if() [2/4]

```
if (
            (access_type &TFM_HAL_ACCESS_WRITABLE) !  = 0U )
```

Definition at line 227 of file protection_mpc_sert.c.

#### 7.99.2.6 if() [3/4]

```
if (
            (first_block_idx >=(IFX_RRAM_SIZE/IFX_SE_RT_RRAM_BLOCK_SIZE))||(last_block_idx
>(IFX_RRAM_SIZE/IFX_SE_RT_RRAM_BLOCK_SIZE))  )
```

Definition at line 210 of file protection_mpc_sert.c.

Here is the call graph for this function:

#### 7.99.2.7 if() [4/4]

```
if (
            IFX_SE_RT_RRAM_BLOCK_SIZE !     = block_size )
```

Definition at line 203 of file protection_mpc_sert.c.
Here is the call graph for this function:

#### 7.99.2.8 ifx_mpc_sert_apply_configuration()

```
enum tfm_hal_status_t ifx_mpc_sert_apply_configuration (
            const ifx_mpc_raw_region_config_t * mpc_reg_cfg )
```

Apply MPC configuration for MPC controller that is handled by optional SE RT Service or other non-TF-M service/hardware.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not supported by external service).

Definition at line 107 of file protection_mpc_sert.c.

### 7.99.3 Variable Documentation

#### 7.99.3.1 access_type

```
const ifx_memory_config_t uintptr_t size_t uint32_t access_type
```

**Initial value:**

```
{
    if (memory_config == NULL) {
        FIH_RET(fih_int_encode(TFM_HAL_ERROR_MEM_FAULT));
    }
    uint32_t block_size = (1UL « ((uint32_t)( memory_config->mpc_block_size ) + 5UL))
```

Definition at line 197 of file protection_mpc_sert.c.

#### 7.99.3.2 attr

```
uint32_t attr = is_secure ?  0u :  (1UL << ((pc * 4u) + 0u))
```

Definition at line 220 of file protection_mpc_sert.c.

#### 7.99.3.3 base

```
const ifx_memory_config_t uintptr_t base
```

Definition at line 194 of file protection_mpc_sert.c.

#### 7.99.3.4 first_block_idx

```
uint32_t first_block_idx = offset / IFX_SE_RT_RRAM_BLOCK_SIZE
```

Definition at line 208 of file protection_mpc_sert.c.

#### 7.99.3.5 is_secure

```
bool is_secure = (access_type & TFM_HAL_ACCESS_NS) == 0U
```

Definition at line 215 of file protection_mpc_sert.c.

#### 7.99.3.6 last_block_idx

```
uint32_t last_block_idx = IFX_ROUND_UP_TO_MULTIPLE(offset + size, IFX_SE_RT_RRAM_BLOCK_SIZE) /
IFX_SE_RT_RRAM_BLOCK_SIZE
```

Definition at line 209 of file protection_mpc_sert.c.

#### 7.99.3.7 mask

```
uint32_t mask = 1UL << ((pc * 4u) + 0u)
```

Definition at line 219 of file protection_mpc_sert.c.

#### 7.99.3.8 memory_config

```
const ifx_memory_config_t* memory_config
```

Definition at line 193 of file protection_mpc_sert.c.

#### 7.99.3.9 offset

```
uint32_t offset = IFX_S_ADDRESS_ALIAS(base) - memory_config->s_address
```

Definition at line 207 of file protection_mpc_sert.c.

#### 7.99.3.10 pc

```
uint32_t pc = (uint32_t)fih_int_decode(fih_int_encode(IFX_PC_TFM_SPM_ID))
```

Definition at line 217 of file protection_mpc_sert.c.

#### 7.99.3.11 size

```
const ifx_memory_config_t uintptr_t size_t size
```

Definition at line 195 of file protection_mpc_sert.c.

## 7.100 platform/ext/target/infineon/common/spe/protection/protection_↩ mpc_sert.h File Reference

This file contains types/API used by HW MPC driver, to protect memory via SE RT Basic/Full.

```
#include "config_impl.h"
#include "config_tfm.h"
#include "fih.h"
#include "tfm_hal_defs.h"
#include "protection_types.h"
```

Include dependency graph for protection_mpc_sert.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpc_sert_init(void)

  *Initialize the MPC attribute cache for SE RT Services.*
- enum tfm_hal_status_t ifx_mpc_sert_apply_configuration (const ifx_mpc_raw_region_config_t *mpc_reg_cfg)

  *Apply MPC configuration for MPC controller that is handled by optional SE RT Service or other non-TF-M service/hardware.*

## Variables

- const ifx_memory_config_t * memory_config
- const ifx_memory_config_t uintptr_t base
- const ifx_memory_config_t uintptr_t size_t size
- const ifx_memory_config_t uintptr_t size_t uint32_t access_type

## 7.100.1 Detailed Description

This file contains types/API used by HW MPC driver, to protect memory via SE RT Basic/Full.

## 7.100.2 Function Documentation

### 7.100.2.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Initialize the MPC attribute cache for SE RT Services.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Definition at line 67 of file faults.c.

#### 7.100.2.2 ifx_mpc_sert_apply_configuration()

```
enum tfm_hal_status_t ifx_mpc_sert_apply_configuration (
            const ifx_mpc_raw_region_config_t * mpc_reg_cfg )
```

Apply MPC configuration for MPC controller that is handled by optional SE RT Service or other non-TF-M service/hardware.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not supported by external service).

Definition at line 107 of file protection_mpc_sert.c.

### 7.100.3 Variable Documentation

#### 7.100.3.1 access_type

```
const ifx_memory_config_t uintptr_t size_t uint32_t access_type
```

Definition at line 71 of file protection_mpc_sert.h.

#### 7.100.3.2 base

```
const ifx_memory_config_t uintptr_t base
```

Definition at line 69 of file protection_mpc_sert.h.

#### 7.100.3.3 memory_config

```
const ifx_memory_config_t* memory_config
```

Definition at line 68 of file protection_mpc_sert.h.

#### 7.100.3.4 size

```
const ifx_memory_config_t uintptr_t size_t size
```

Definition at line 70 of file protection_mpc_sert.h.

## 7.101 platform/ext/target/infineon/common/spe/protection/protection_mpc_sw_policy.c File Reference

```
#include "cy_device.h"
#include "cy_mpc.h"
#include "ifx_regions.h"
#include "ifx_utils.h"
#include "protection_mpc_api.h"
#include "protection_regions_cfg.h"
#include "protection_types.h"
#include "tfm_hal_isolation.h"
#include <string.h>
#include "static_checks.h"
```

Include dependency graph for protection_mpc_sw_policy.c:

### Data Structures

- struct ifx_mpc_cfg_t
- struct ifx_mpc_policy_t

### Macros

- #define IFX_MPC_BLOCK_SIZE (0x800UL) /∗ 2 KB ∗/
- #define IFX_MPC_PC_ATTR_MSK (0xFUL) /∗ Mask for MPC attributes for protection context ∗/
- #define IFX_MPC_PC_ATTR_SIZE_IN_BITS (4UL) /∗ Number of MPC attribute bits ∗/
- #define IFX_MPC_PC_ATTR_NS_MSK (1UL << 0) /∗ NS mask for MPC attributes for PC ∗/
- #define IFX_MPC_PC_ATTR_W_MSK (1UL << 1) /∗ Write mask for MPC attributes for PC ∗/
- #define IFX_MPC_PC_ATTR_R_MSK (1UL << 2) /∗ Read mask for MPC attributes for PC ∗/
- #define IFX_MPC_GET_PC_ATTR(attr, pc) (((attr) >> ((pc) ∗ IFX_MPC_PC_ATTR_SIZE_IN_BITS)) & IFX_MPC_PC_ATTR_MSK)
- #define CY_SRAM0_BASE_S (IFX_S_ADDRESS_ALIAS(CY_SRAM0_BASE))
- #define CY_FLASH_BASE_S (IFX_S_ADDRESS_ALIAS(CY_FLASH_BASE))

### Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpc_init_cfg(void)

  *Configures fault handlers and sets priorities.*

- TFM_COVERITY_DEVIATE_LINE (MISRA_C_2023_Rule_11_3, "Intentional pointer type conversion, this maps MPC policy")
- if (ifx_is_region_inside_other(base_s, base_s+size,(IFX_S_ADDRESS_ALIAS(CY_SRAM0_BASE)),(IFX_S_ADDRESS_ALIAS Y_SRAM0_BASE))+CY_SRAM0_SIZE))
- else if (ifx_is_region_inside_other(base_s, base_s+size,(IFX_S_ADDRESS_ALIAS(CY_FLASH_BA↩ SE)),(IFX_S_ADDRESS_ALIAS(CY_FLASH_BASE))+CY_FLASH_SIZE))
- for (uint32_t mpc_idx=mpc_start_idx;mpc_idx< mpc_end_idx;mpc_idx++)
- FIH_RET (fih_int_encode(TFM_HAL_ERROR_MEM_FAULT))

## Variables

- uintptr_t base
- uintptr_t size_t size
- uintptr_t size_t uint32_t access_type
- uint32_t mpc_end_idx
- uint32_t mpc_base_addr
- const ifx_mpc_policy_t *const mpc_policy = (ifx_mpc_policy_t*) &SFLASH->N_RAM_MPC
- uintptr_t base_s = IFX_S_ADDRESS_ALIAS(base)
- else

## 7.101.1 Macro Definition Documentation

### 7.101.1.1 CY_FLASH_BASE_S

```
#define CY_FLASH_BASE_S (IFX_S_ADDRESS_ALIAS(CY_FLASH_BASE))
```

Definition at line 37 of file protection_mpc_sw_policy.c.

### 7.101.1.2 CY_SRAM0_BASE_S

```
#define CY_SRAM0_BASE_S (IFX_S_ADDRESS_ALIAS(CY_SRAM0_BASE))
```

Definition at line 35 of file protection_mpc_sw_policy.c.

### 7.101.1.3 IFX_MPC_BLOCK_SIZE

```
#define IFX_MPC_BLOCK_SIZE (0x800UL) /* 2 KB */
```

Definition at line 21 of file protection_mpc_sw_policy.c.

### 7.101.1.4 IFX_MPC_GET_PC_ATTR

```
#define IFX_MPC_GET_PC_ATTR(
            attr,
            pc ) (((attr) >> ((pc) * IFX_MPC_PC_ATTR_SIZE_IN_BITS)) & IFX_MPC_PC_ATTR_MSK)
```

Definition at line 32 of file protection_mpc_sw_policy.c.

### 7.101.1.5 IFX_MPC_PC_ATTR_MSK

```
#define IFX_MPC_PC_ATTR_MSK (0xFUL) /* Mask for MPC attributes for protection context */
```

Definition at line 25 of file protection_mpc_sw_policy.c.

#### 7.101.1.6 IFX_MPC_PC_ATTR_NS_MSK

```
#define IFX_MPC_PC_ATTR_NS_MSK (1UL << 0) /* NS mask for MPC attributes for PC */
```

Definition at line 27 of file protection_mpc_sw_policy.c.

#### 7.101.1.7 IFX_MPC_PC_ATTR_R_MSK

```
#define IFX_MPC_PC_ATTR_R_MSK (1UL << 2) /* Read mask for MPC attributes for PC */
```

Definition at line 29 of file protection_mpc_sw_policy.c.

#### 7.101.1.8 IFX_MPC_PC_ATTR_SIZE_IN_BITS

```
#define IFX_MPC_PC_ATTR_SIZE_IN_BITS (4UL) /* Number of MPC attribute bits */
```

Definition at line 26 of file protection_mpc_sw_policy.c.

#### 7.101.1.9 IFX_MPC_PC_ATTR_W_MSK

```
#define IFX_MPC_PC_ATTR_W_MSK (1UL << 1) /* Write mask for MPC attributes for PC */
```

Definition at line 28 of file protection_mpc_sw_policy.c.

### 7.101.2 Function Documentation

#### 7.101.2.1 FIH_RET()

```
FIH_RET (
            fih_int_encode(TFM_HAL_ERROR_MEM_FAULT)  )
```

Here is the caller graph for this function:

#### 7.101.2.2 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Definition at line 55 of file protection_mpc_sw_policy.c.

#### 7.101.2.3 for()

`for ( )`
Definition at line 144 of file protection_mpc_sw_policy.c.

Here is the call graph for this function:

#### 7.101.2.4 if() [1/2]

```
else if (
            ifx_is_region_inside_other(base_s, base_s+size,(IFX_S_ADDRESS_ALIAS(CY_FLASH_BA↩
SE)),(IFX_S_ADDRESS_ALIAS(CY_FLASH_BASE))+CY_FLASH_SIZE)  )
```

Definition at line 133 of file protection_mpc_sw_policy.c.

#### 7.101.2.5 if() [2/2]

```
if (
            ifx_is_region_inside_other(base_s, base_s+size,(IFX_S_ADDRESS_ALIAS(CY_SRAM0_BA↩
SE)),(IFX_S_ADDRESS_ALIAS(CY_SRAM0_BASE))+CY_SRAM0_SIZE)  )
```

Definition at line 127 of file protection_mpc_sw_policy.c.

#### 7.101.2.6 TFM_COVERITY_DEVIATE_LINE()

```
TFM_COVERITY_DEVIATE_LINE (
            MISRA_C_2023_Rule_11_3 ,
            "Intentional pointer type conversion,
            this maps MPC policy"  )
```

### 7.101.3 Variable Documentation

#### 7.101.3.1 access_type

```
uintptr_t size_t uint32_t access_type
```

**Initial value:**

```
{
    uint32_t mpc_start_idx
```

Definition at line 116 of file protection_mpc_sw_policy.c.

#### 7.101.3.2 base

```
uintptr_t base
```

Definition at line 113 of file protection_mpc_sw_policy.c.

#### 7.101.3.3 base_s

```
uintptr_t base_s = IFX_S_ADDRESS_ALIAS(base)
```

Definition at line 124 of file protection_mpc_sw_policy.c.

#### 7.101.3.4 else

```
else
```

**Initial value:**

```
{
        FIH_RET(fih_int_encode(TFM_HAL_ERROR_INVALID_INPUT))
```

Definition at line 139 of file protection_mpc_sw_policy.c.

#### 7.101.3.5 mpc_base_addr

```
uint32_t mpc_base_addr
```

Definition at line 119 of file protection_mpc_sw_policy.c.

#### 7.101.3.6 mpc_end_idx

```
uint32_t mpc_end_idx
```

Definition at line 118 of file protection_mpc_sw_policy.c.

#### 7.101.3.7 mpc_policy

```
const ifx_mpc_policy_t* const mpc_policy = (ifx_mpc_policy_t*) &SFLASH->N_RAM_MPC
```

Definition at line 122 of file protection_mpc_sw_policy.c.

#### 7.101.3.8 size

```
uintptr_t size_t size
```

Definition at line 114 of file protection_mpc_sw_policy.c.

## 7.102 platform/ext/target/infineon/common/spe/protection/protection_↩ mpc_sw_policy.h File Reference

This file contains types/API used by MPC SW Policy driver.

### Macros

- #define IFX_MPC_NAMED_MMIO_SPM_ROT_CFG
- #define IFX_MPC_NAMED_MMIO_PSA_ROT_CFG
- #define IFX_MPC_NAMED_MMIO_APP_ROT_CFG

### 7.102.1 Detailed Description

This file contains types/API used by MPC SW Policy driver.

**Note**

Don't include this file directly, include protection_types.h instead !!! It's expected that this file is included by protection_types.h if platform is configured to use MPC SW Policy driver via IFX_MPC_DRIVER_SW_POLICY option.

### 7.102.2 Macro Definition Documentation

#### 7.102.2.1 IFX_MPC_NAMED_MMIO_APP_ROT_CFG

```
#define IFX_MPC_NAMED_MMIO_APP_ROT_CFG
```

Definition at line 28 of file protection_mpc_sw_policy.h.

#### 7.102.2.2 IFX_MPC_NAMED_MMIO_PSA_ROT_CFG

```
#define IFX_MPC_NAMED_MMIO_PSA_ROT_CFG
```

Definition at line 25 of file protection_mpc_sw_policy.h.

#### 7.102.2.3 IFX_MPC_NAMED_MMIO_SPM_ROT_CFG

```
#define IFX_MPC_NAMED_MMIO_SPM_ROT_CFG
```

Definition at line 22 of file protection_mpc_sw_policy.h.

## 7.103 platform/ext/target/infineon/common/spe/protection/protection_↩ mpu.c File Reference

```
#include "config_impl.h"
#include "config_tfm.h"
#include "cmsis.h"
#include "ifx_regions.h"
#include "protection_regions_cfg.h"
#include "protection_mpu.h"
#include "protection_utils.h"
#include "region.h"
#include "utilities.h"
#include "static_checks.h"
```

Include dependency graph for protection_mpu.c:

### Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpu_init_cfg(void)

  *Configures fault handlers and sets priorities.*
- void ifx_mpu_disable_asset_regions (void)

  *Disable MPU regions that are reserved for partition assets.*
- if ((p_asset->attr &ASSET_ATTR_READ_WRITE) !=0u)
- ifx_mpu_config_enable_region (ifx_mpu_used_regions_count, &mpu_config)
- FIH_RET (fih_int_encode(TFM_HAL_SUCCESS))

### Variables

- const struct asset_desc_t ∗ p_asset
- mpu_config region_base = (uint32_t)p_asset->mem.start

- mpu_config region_limit = (uint32_t)p_asset->mem.limit - 1U
- mpu_config attr_exec = MPU_ARMV8M_XN_EXEC_NEVER
- else
- mpu_config attr_access = MPU_ARMV8M_AP_RO_PRIV_UNPRIV
- mpu_config attr_sh = MPU_ARMV8M_SH_NONE

## 7.103.1 Function Documentation

### 7.103.1.1 FIH_RET()

```
FIH_RET (
            fih_int_encode(TFM_HAL_SUCCESS)  )
```

### 7.103.1.2 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Definition at line 173 of file protection_mpu.c.

#### 7.103.1.3 if()

```
if (
            (p_asset->attr &ASSET_ATTR_READ_WRITE) !     = 0u )
```

Definition at line 323 of file protection_mpu.c.

#### 7.103.1.4 ifx_mpu_config_enable_region()

```
ifx_mpu_config_enable_region (
            ifx_mpu_used_regions_count ,
            & mpu_config )
```

#### 7.103.1.5 ifx_mpu_disable_asset_regions()

```
void ifx_mpu_disable_asset_regions (
            void  )
```

Disable MPU regions that are reserved for partition assets.
Definition at line 275 of file protection_mpu.c.

### 7.103.2 Variable Documentation

#### 7.103.2.1 attr_access

```
mpu_config attr_access = MPU_ARMV8M_AP_RO_PRIV_UNPRIV
```

Definition at line 328 of file protection_mpu.c.

#### 7.103.2.2 attr_exec

```
mpu_config attr_exec = MPU_ARMV8M_XN_EXEC_NEVER
```

Definition at line 319 of file protection_mpu.c.

#### 7.103.2.3 attr_sh

```
mpu_config attr_sh = MPU_ARMV8M_SH_NONE
```

Definition at line 331 of file protection_mpu.c.

#### 7.103.2.4 else

```
else
```

**Initial value:**

```
{
        mpu_config.region_attridx = MPU_ARMV8M_MAIR_ATTR_CODE_IDX
```

Definition at line 326 of file protection_mpu.c.

#### 7.103.2.5 p_asset

```
const struct asset_desc_t* p_asset
```

**Initial value:**

```
{

    if ((fih_eq(( p_info )->ifx_ldinfo->privileged, IFX_FIH_TRUE))) {
        FIH_RET(fih_int_encode(TFM_HAL_SUCCESS));
    }
    if (ifx_mpu_used_regions_count >= CPUSS_CM33_0_MPU_S_REGION_NR) {
        FIH_RET(fih_int_encode(TFM_HAL_ERROR_BAD_STATE));
    }

    struct mpu_armv8m_region_cfg_t mpu_config
```

Definition at line 290 of file protection_mpu.c.

#### 7.103.2.6 region_base

```
mpu_config region_base = (uint32_t)p_asset->mem.start
```

Definition at line 314 of file protection_mpu.c.

#### 7.103.2.7 region_limit

```
mpu_config region_limit = (uint32_t)p_asset->mem.limit - 1U
```

Definition at line 318 of file protection_mpu.c.

## 7.104 platform/ext/target/infineon/common/spe/protection/protection_↩ mpu.h File Reference

```
#include "config_tfm.h"
#include "fih.h"
#include "load/asset_defs.h"
#include "mpu_armv8m_drv.h"
#include "partition_info.h"
#include "protection_regions_cfg.h"
#include "protection_utils.h"
#include "tfm_hal_defs.h"
```

```
#include "load/partition_defs.h"
```

Include dependency graph for protection_mpu.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_mpu_init_cfg(void)

  *Configures the Memory Protection Unit.*
- void ifx_mpu_disable_asset_regions (void)

  *Disable MPU regions that are reserved for partition assets.*

## Variables

- const struct asset_desc_t * p_asset

## 7.104.1 Function Documentation

### 7.104.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures the Memory Protection Unit.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Definition at line 67 of file faults.c.

#### 7.104.1.2 ifx_mpu_disable_asset_regions()

```
void ifx_mpu_disable_asset_regions (
            void  )
```

Disable MPU regions that are reserved for partition assets.
Definition at line 275 of file protection_mpu.c.

### 7.104.2 Variable Documentation

#### 7.104.2.1 p_asset

```
const struct asset_desc_t* p_asset
```

Definition at line 56 of file protection_mpu.h.

## 7.105 platform/ext/target/infineon/common/spe/protection/protection_↩ msc.c File Reference

```
#include "cy_ms_ctl.h"
#include "protection_msc.h"
#include "protection_regions_cfg.h"
#include "utilities.h"
```

Include dependency graph for protection_msc.c:

### Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_msc_cfg(void)

  *Configures fault handlers and sets priorities.*

## 7.105.1 Function Documentation

### 7.105.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |

**Parameters**

| | | |
|---|---|---|
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Definition at line 184 of file protection_msc.c.
Here is the call graph for this function:

## 7.106 platform/ext/target/infineon/common/spe/protection/protection_↩ msc.h File Reference

```
#include "fih.h"
#include "tfm_hal_defs.h"
```

Include dependency graph for protection_msc.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_msc_cfg(void)

  *Configures MSC.*

## 7.106.1 Function Documentation

### 7.106.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures MSC.

**Returns**

TFM_HAL_SUCCESS - MSC configured successfully TFM_HAL_ERROR_GENERIC - failed to configure M↩SC

Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.

Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.

Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.

Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 67 of file faults.c.

## 7.107 platform/ext/target/infineon/common/spe/protection/protection_↩ pc.c File Reference

```
#include "config_tfm.h"
#include "compiler_ext_defs.h"
#include "cy_ms_ctl.h"
#include "utilities.h"
#include "fih.h"
#include "memory_symbols.h"
#include "protection_pc.h"
#include "protection_regions_cfg.h"
#include "static_checks.h"
#include "tfm_platform_arch_hooks.h"
```

Include dependency graph for protection_pc.c:

### Macros

- #define IFX_VTOR_ADDRESS __vector_table
- #define IFX_RETURN_ON_EXCEPTION_BIT_MASK

### Functions

- void ifx_pc2_exception_handler (void)

  *Default secure interrupt handler which is a hardware entry point to Protection Context 2.*

- TFM_COVERITY_DEVIATE_LINE (MISRA_C_2023_Rule_2_8, "Used with assembler for restoring PC after handling exception")

  *Active protection context.*
- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_pc_init(void)

  *Configures fault handlers and sets priorities.*
- void ifx_arch_switch_protection_context (uint32_t pc)

  *Perform protection context switching.*
- void ifx_arch_switch_protection_context_from_irq (uint32_t pc)

  *Perform protection context switching from low priority IRQ.*
- void ifx_arch_switch_protection_context_thread (uint32_t pc)

  *Entry point to Protection Context switching via UsageFault/HardFault handler.*

## Variables

- const VECTOR_TABLE_Type ifx_pc2_vector_table [VECTORTABLE_SIZE]
- fih_int ifx_arch_active_protection_context = FIH_INT_INIT(IFX_PC_TFM_SPM_ID)

## 7.107.1 Macro Definition Documentation

### 7.107.1.1 IFX_RETURN_ON_EXCEPTION_BIT_MASK

```
#define IFX_RETURN_ON_EXCEPTION_BIT_MASK
```

**Value:**

```
((1 «  2) |        /*   -14 NMI Handler */ \
 (0 «  3) |        /*   -13 Hard Fault Handler */ \
 (0 «  4) |        /*   -12 MPU Fault Handler */ \
 (0 «  5) |        /*   -11 Bus Fault Handler */ \
 (0 «  6) |        /*   -10 Usage Fault Handler */ \
 (0 «  7) |        /*    -9 Secure Fault Handler */ \
 (0 «  8) |        /*    -8 Reserved */ \
 (0 «  9) |        /*    -7 Reserved */ \
 (0 « 10) |        /*    -6 Reserved */ \
 (0 « 11) |        /*    -5 SVCall Handler */ \
 (1 « 12) |        /*    -4 Debug Monitor Handler */ \
 (0 « 13) |        /*    -3 Reserved */ \
 (0 « 14) |        /*    -2 PendSV Handler */ \
 (1 « 15))       /*    -1 SysTick Handler */
```

Definition at line 43 of file protection_pc.c.

### 7.107.1.2 IFX_VTOR_ADDRESS

```
#define IFX_VTOR_ADDRESS __vector_table
```

Definition at line 31 of file protection_pc.c.

## 7.107.2 Function Documentation

### 7.107.2.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Definition at line 83 of file protection_pc.c.
Here is the call graph for this function:

#### 7.107.2.2 ifx_arch_switch_protection_context()

```
void ifx_arch_switch_protection_context (
            uint32_t pc )
```

Perform protection context switching.

**Note**

It's unsafe to switch PC to other than PC2 (used by SPM) in low priority exception, because it's possible that there will be pending IRQ and MCU will try to use MSP stack pushing exception stack within PC not equal to PC2.

IRQ must be disabled if this function is called from exception/IRQ with privilege below 0.

Definition at line 158 of file protection_pc.c.

#### 7.107.2.3 ifx_arch_switch_protection_context_from_irq()

```
void ifx_arch_switch_protection_context_from_irq (
            uint32_t pc )
```

Perform protection context switching from low priority IRQ.
This function will complete switching to PC2 (SPM) in scope of current exception. For all other protection context it will schedule switching after exit from active IRQ in thread mode by ifx_arch_switch_protection_context_thread.

**Note**

IRQ must be disabled when this function is called.

Definition at line 250 of file protection_pc.c.

#### 7.107.2.4 ifx_arch_switch_protection_context_thread()

```
void ifx_arch_switch_protection_context_thread (
            uint32_t pc )
```

Entry point to Protection Context switching via UsageFault/HardFault handler.
This function schedule UsageFault as an entry point to Protection Context switching. But UsageFault escalates HardFault because masked interrupts are disabled when ifx_arch_switch_protection_context_from_irq is called.
HardFault validates that it was called because of `udf` instruction with correct set of arguments pushed via exception stack frame that are duplicate of one prepared by ifx_arch_switch_protection_context_from_irq. HardFault calls Protection Context switching if all validation tests are passed. See PLATFORM_HARD_FAULT_HANDLER_HOOK for more information.
Definition at line 313 of file protection_pc.c.

#### 7.107.2.5 ifx_pc2_exception_handler()

```
void ifx_pc2_exception_handler (
            void )
```

Default secure interrupt handler which is a hardware entry point to Protection Context 2.
SCB ICSR VECTACTIVE is used to detect active interrupt number and select a appropriate handler from VTOR table located at IFX_VTOR_ADDRESS.
Definition at line 345 of file protection_pc.c.
Here is the caller graph for this function:

#### 7.107.2.6 TFM_COVERITY_DEVIATE_LINE()

```
TFM_COVERITY_DEVIATE_LINE (
            MISRA_C_2023_Rule_2_8 ,
            "Used with assembler for restoring PC after handling exception" )
```

Active protection context.
PC_SAVED is not used because it's inconsistent if there are multiple pending IRQs. This variable is atomically updated with PC. So, this allows to properly restore protection context after handling exception.
Note: Allocate variable in scope of SPM data to avoid access by unprivileged code.

### 7.107.3 Variable Documentation

#### 7.107.3.1 ifx_arch_active_protection_context

```
fih_int ifx_arch_active_protection_context = FIH_INT_INIT(IFX_PC_TFM_SPM_ID)
```

Definition at line 81 of file protection_pc.c.

#### 7.107.3.2 ifx_pc2_vector_table

```
const VECTOR_TABLE_Type ifx_pc2_vector_table[VECTORTABLE_SIZE]
```

**Initial value:**

```
= {
    [0] = (VECTOR_TABLE_Type)(&__INITIAL_SP),

    [1 ...  VECTORTABLE_SIZE-1] =    ifx_pc2_exception_handler,
}
```

Definition at line 63 of file protection_pc.c.

## 7.108 platform/ext/target/infineon/common/spe/protection/protection_↩ pc.h File Reference

```
#include "fih.h"
#include "tfm_hal_defs.h"
```

Include dependency graph for protection_pc.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_pc_init(void)

  *Initialize Protection Context switching.*
- void ifx_arch_switch_protection_context_thread (uint32_t pc)

  *Entry point to Protection Context switching via UsageFault/HardFault handler.*

## Variables

- fih_int ifx_arch_active_protection_context

## 7.108.1 Function Documentation

### 7.108.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Initialize Protection Context switching.
Configures the VTOR to implement switching to Protection Context used by SPM.

**Returns**

A status code as specified in tfm_hal_status_t

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |

**Parameters**

| | | |
|---|---|---|
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_INPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_INPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_INPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 67 of file faults.c.

#### 7.108.1.2 ifx_arch_switch_protection_context_thread()

```
void ifx_arch_switch_protection_context_thread (
            uint32_t pc )
```

Entry point to Protection Context switching via UsageFault/HardFault handler.
This function schedule UsageFault as an entry point to Protection Context switching. But UsageFault escalates HardFault because masked interrupts are disabled when ifx_arch_switch_protection_context_from_irq is called.
HardFault validates that it was called because of `udf` instruction with correct set of arguments pushed via exception stack frame that are duplicate of one prepared by ifx_arch_switch_protection_context_from_irq. HardFault calls Protection Context switching if all validation tests are passed. See PLATFORM_HARD_FAULT_HANDLER_HOOK for more information.
Definition at line 313 of file protection_pc.c.

### 7.108.2 Variable Documentation

#### 7.108.2.1 ifx_arch_active_protection_context

`fih_int ifx_arch_active_protection_context`
Definition at line 81 of file protection_pc.c.

## 7.109 platform/ext/target/infineon/common/spe/protection/protection_↩ ppc_api.h File Reference

This file contains PPC driver API declaration.
`#include "fih.h"`
`#include "load/asset_defs.h"`
`#include "protection_types.h"`
`#include "tfm_hal_defs.h"`
Include dependency graph for protection_ppc_api.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_ppc_init_cfg(void)

  *Configures the Peripheral Protection Controller.*

## Variables

- cy_en_prot_region_t asset

## 7.109.1 Detailed Description

This file contains PPC driver API declaration.
It's expected that PPC driver implementation follows API declared in this file.

## 7.109.2 Function Documentation

### 7.109.2.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures the Peripheral Protection Controller.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 67 of file faults.c.

### 7.109.3 Variable Documentation

#### 7.109.3.1 asset

`cy_en_prot_region_t asset`

Definition at line 56 of file protection_ppc_api.h.

## 7.110 platform/ext/target/infineon/common/spe/protection/protection_↩ ppc_v1.c File Reference

```
#include "config_tfm.h"
#include "cy_ppc.h"
#include "utilities.h"
#include "protection_types.h"
#include "protection_regions_cfg.h"
#include "protection_ppc_api.h"
#include "static_checks.h"
```

Include dependency graph for protection_ppc_v1.c:

### Functions

- ifx_check_config_validity (ppc_ptr, &ppc_attribute, &ppc_pcmask)
- if (Cy_Ppc_SetPcMask(ppc_ptr, &ppc_pcmask) !=CY_PPC_SUCCESS)
- void fih_delay ()
- if (Cy_Ppc_ConfigAttrib(ppc_ptr, &ppc_attribute) !=CY_PPC_SUCCESS)
- if ((ppc_cfg->pc_mask &IFX_PC_TFM_SPM)==0U)
- FIH_RET (fih_int_encode(TFM_HAL_SUCCESS))

### Variables

- cy_en_prot_region_t asset
- cy_stc_ppc_pc_mask_t ppc_pcmask
- cy_stc_ppc_attribute_t ppc_attribute

## 7.110.1 Function Documentation

### 7.110.1.1 fih_delay()

```
void fih_delay ( )
```

Here is the caller graph for this function:

#### 7.110.1.2 FIH_RET()

```
FIH_RET (
            fih_int_encode(TFM_HAL_SUCCESS)  )
```

Here is the caller graph for this function:

#### 7.110.1.3 if() [1/3]

```
if (
            (ppc_cfg->pc_mask &IFX_PC_TFM_SPM)  = = 0U )
```

Definition at line 347 of file protection_ppc_v1.c.

Here is the call graph for this function:

#### 7.110.1.4 if() [2/3]

```
if (
            Cy_Ppc_ConfigAttrib(ppc_ptr, &ppc_attribute) !    = CY_PPC_SUCCESS )
```

Definition at line 343 of file protection_ppc_v1.c.

Here is the call graph for this function:

#### 7.110.1.5 if() [3/3]

```
if (
            Cy_Ppc_SetPcMask(ppc_ptr, &ppc_pcmask) !     = CY_PPC_SUCCESS )
```

Definition at line 336 of file protection_ppc_v1.c.
Here is the call graph for this function:

#### 7.110.1.6 ifx_check_config_validity()

```
ifx_check_config_validity (
            ppc_ptr ,
            & ppc_attribute,
            & ppc_pcmask )
```

### 7.110.2 Variable Documentation

#### 7.110.2.1 asset

```
cy_en_prot_region_t asset
```

**Initial value:**

```
{
    PPC_Type* ppc_ptr = Cy_Ppc_GetPointerForRegion(asset)
```

Definition at line 316 of file protection_ppc_v1.c.

#### 7.110.2.2 ppc_attribute

```
cy_stc_ppc_attribute_t ppc_attribute
```

**Initial value:**

```
= {
        .startRegion      = asset,
        .endRegion        = asset,
        .secAttribute     = ppc_cfg->sec_attr,
        .secPrivAttribute = to_s_priv_attr(ppc_cfg->allow_unpriv, ppc_cfg->sec_attr),
        .nsPrivAttribute  = to_ns_priv_attr(ppc_cfg->allow_unpriv, ppc_cfg->sec_attr),
    }
```

Definition at line 326 of file protection_ppc_v1.c.

#### 7.110.2.3 ppc_pcmask

```
cy_stc_ppc_pc_mask_t ppc_pcmask
```

**Initial value:**

```
= {
        .startRegion = asset,
        .endRegion   = asset,
        .pcMask      = ppc_cfg->pc_mask | IFX_PC_TFM_SPM,
    }
```

Definition at line 320 of file protection_ppc_v1.c.

# 7.111 platform/ext/target/infineon/common/spe/protection/protection_↩ ppc_v1.h File Reference

This file contains types/API used by PPC driver.

```
#include <stdbool.h>
#include "ifx_spe_config.h"
#include "tfm_hal_defs.h"
#include "cy_ppc.h"
#include "load/partition_defs.h"
#include "load/asset_defs.h"
```

Include dependency graph for protection_ppc_v1.h:

## Data Structures

- struct ifx_ppc_named_mmio_config_t
- struct ifx_ppcx_config_t

## Macros

- #define IFX_PPC_NAMED_MMIO_SPM_ROT_CFG
- #define IFX_PPC_NAMED_MMIO_PSA_ROT_CFG
- #define IFX_PPC_NAMED_MMIO_APP_ROT_CFG

## 7.111.1 Detailed Description

This file contains types/API used by PPC driver.

**Note**

Don't include this file directly, include protection_types.h instead !!! It's expected that this file is included by protection_types.h if platform is configured to use PPC driver.

## 7.111.2 Macro Definition Documentation

### 7.111.2.1 IFX_PPC_NAMED_MMIO_APP_ROT_CFG

```
#define IFX_PPC_NAMED_MMIO_APP_ROT_CFG
```

**Value:**

```
    .ppc_cfg = { \
        .sec_attr       = CY_PPC_SECURE, \
        .allow_unpriv   = true, \
        .pc_mask        = IFX_PC_DEFAULT | IFX_PC_TFM_AROT, \
    },
```

Definition at line 79 of file protection_ppc_v1.h.

#### 7.111.2.2 IFX_PPC_NAMED_MMIO_PSA_ROT_CFG

```
#define IFX_PPC_NAMED_MMIO_PSA_ROT_CFG
```

**Value:**

```
    .ppc_cfg = { \
        .sec_attr        = CY_PPC_SECURE, \
        .allow_unpriv    = true, \
        .pc_mask         = IFX_PC_DEFAULT | IFX_PC_TFM_PROT, \
    },
```

Definition at line 61 of file protection_ppc_v1.h.

#### 7.111.2.3 IFX_PPC_NAMED_MMIO_SPM_ROT_CFG

```
#define IFX_PPC_NAMED_MMIO_SPM_ROT_CFG
```

**Value:**

```
    .ppc_cfg = { \
        .sec_attr        = CY_PPC_SECURE, \
        .allow_unpriv    = false, \
        .pc_mask         = IFX_PC_DEFAULT, \
    },
```

Definition at line 44 of file protection_ppc_v1.h.

## 7.112 platform/ext/target/infineon/common/spe/protection/protection_↩ ppc_v2.c File Reference

```
#include "config_tfm.h"
#include "cy_ppc.h"
#include "utilities.h"
#include "protection_types.h"
#include "protection_regions_cfg.h"
#include "protection_ppc_api.h"
#include "static_checks.h"
```

Include dependency graph for protection_ppc_v2.c:

### Functions

- ifx_check_config_validity (ppc_ptr, asset, &ppc_attribute, ppc_pcmask)
- if (Cy_Ppc_SetPcMask(ppc_ptr, asset, ppc_pcmask) !=CY_PPC_SUCCESS)
- void fih_delay ()
- if (Cy_Ppc_ConfigAttrib(ppc_ptr, asset, &ppc_attribute) !=CY_PPC_SUCCESS)
- if ((ppc_cfg->pc_mask &IFX_PC_TFM_SPM)==0U)
- FIH_RET (fih_int_encode(TFM_HAL_SUCCESS))

### Variables

- cy_en_prot_region_t asset
- uint32_t ppc_pcmask = ppc_cfg->pc_mask | IFX_PC_TFM_SPM
- cy_stc_ppc_attribute_t ppc_attribute

### 7.112.1 Function Documentation

#### 7.112.1.1 fih_delay()

```
void fih_delay ( )
```

#### 7.112.1.2 FIH_RET()

```
FIH_RET (
            fih_int_encode(TFM_HAL_SUCCESS)  )
```

Here is the caller graph for this function:

#### 7.112.1.3 if() [1/3]

```
if (
            (ppc_cfg->pc_mask &IFX_PC_TFM_SPM)  = = 0U )
```

Definition at line 371 of file protection_ppc_v2.c.

Here is the call graph for this function:

#### 7.112.1.4 if() [2/3]

```
if (
            Cy_Ppc_ConfigAttrib(ppc_ptr, asset, &ppc_attribute) !    = CY_PPC_SUCCESS )
```

Definition at line 367 of file protection_ppc_v2.c.

Here is the call graph for this function:

#### 7.112.1.5 if() [3/3]

```
if (
            Cy_Ppc_SetPcMask(ppc_ptr, asset, ppc_pcmask) !     = CY_PPC_SUCCESS )
```

Definition at line 360 of file protection_ppc_v2.c.
Here is the call graph for this function:

#### 7.112.1.6 ifx_check_config_validity()

```
ifx_check_config_validity (
            ppc_ptr ,
            asset ,
            & ppc_attribute,
            ppc_pcmask  )
```

### 7.112.2 Variable Documentation

#### 7.112.2.1 asset

```
cy_en_prot_region_t asset
```

**Initial value:**

```
{
    PPC_Type* ppc_ptr = Cy_Ppc_GetPointerForRegion(asset)
```

Definition at line 346 of file protection_ppc_v2.c.

#### 7.112.2.2 ppc_attribute

```
cy_stc_ppc_attribute_t ppc_attribute
```

**Initial value:**

```
= {
        .pcMask           = ppc_pcmask,
        .secAttribute     = ppc_cfg->sec_attr,
        .privAttribute    = to_priv_attr(ppc_cfg->allow_unpriv),
    }
```

Definition at line 352 of file protection_ppc_v2.c.

#### 7.112.2.3 ppc_pcmask

```
uint32_t ppc_pcmask = ppc_cfg->pc_mask | IFX_PC_TFM_SPM
```

Definition at line 350 of file protection_ppc_v2.c.

# 7.113 platform/ext/target/infineon/common/spe/protection/protection_↩ ppc_v2.h File Reference

This file contains types/API used by PPC driver.

```
#include <stdbool.h>
#include "ifx_spe_config.h"
#include "tfm_hal_defs.h"
#include "cy_ppc.h"
#include "load/partition_defs.h"
#include "load/asset_defs.h"
```

Include dependency graph for protection_ppc_v2.h:

## Data Structures

- struct ifx_ppc_named_mmio_config_t
- struct ifx_ppcx_config_t

## Macros

- #define IFX_PPC_NAMED_MMIO_SPM_ROT_CFG
- #define IFX_PPC_NAMED_MMIO_PSA_ROT_CFG
- #define IFX_PPC_NAMED_MMIO_APP_ROT_CFG

## 7.113.1 Detailed Description

This file contains types/API used by PPC driver.

**Note**

Don't include this file directly, include protection_types.h instead !!! It's expected that this file is included by protection_types.h if platform is configured to use PPC driver.

## 7.113.2 Macro Definition Documentation

### 7.113.2.1 IFX_PPC_NAMED_MMIO_APP_ROT_CFG

```
#define IFX_PPC_NAMED_MMIO_APP_ROT_CFG
```

**Value:**

```
    .ppc_cfg = { \
        .sec_attr       = CY_PPC_SECURE, \
        .allow_unpriv   = true, \
        .pc_mask        = IFX_PC_DEFAULT | IFX_PC_TFM_AROT, \
    },
```

Definition at line 88 of file protection_ppc_v2.h.

#### 7.113.2.2 IFX_PPC_NAMED_MMIO_PSA_ROT_CFG

```
#define IFX_PPC_NAMED_MMIO_PSA_ROT_CFG
```

**Value:**

```
    .ppc_cfg = { \
        .sec_attr       = CY_PPC_SECURE, \
        .allow_unpriv   = true, \
        .pc_mask        = IFX_PC_DEFAULT | IFX_PC_TFM_PROT, \
    },
```

Definition at line 70 of file protection_ppc_v2.h.

#### 7.113.2.3 IFX_PPC_NAMED_MMIO_SPM_ROT_CFG

```
#define IFX_PPC_NAMED_MMIO_SPM_ROT_CFG
```

**Value:**

```
    .ppc_cfg = { \
        .sec_attr       = CY_PPC_SECURE, \
        .allow_unpriv   = false, \
        .pc_mask        = IFX_PC_DEFAULT, \
    },
```

Definition at line 53 of file protection_ppc_v2.h.

## 7.114 platform/ext/target/infineon/common/spe/protection/protection_sau.c File Reference

```
#include <stdbool.h>
#include "config_tfm.h"
#include "ifx_regions.h"
#include "protection_sau.h"
#include "protection_regions_cfg.h"
#include "static_checks.h"
#include "region.h"
#include "utilities.h"
```

Include dependency graph for protection_sau.c:

### Macros

- #define IFX_SAU_VENEER_REGION_COUNT 0U
- #define IFX_SAU_RESERVED_REGION_COUNT IFX_SAU_VENEER_REGION_COUNT /∗ Veneer region ∗/

### Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_sau_cfg(void)

  *Configures fault handlers and sets priorities.*

### 7.114.1 Macro Definition Documentation

#### 7.114.1.1 IFX_SAU_RESERVED_REGION_COUNT

```
#define IFX_SAU_RESERVED_REGION_COUNT IFX_SAU_VENEER_REGION_COUNT /* Veneer region */
```

Definition at line 28 of file protection_sau.c.

#### 7.114.1.2 IFX_SAU_VENEER_REGION_COUNT

```
#define IFX_SAU_VENEER_REGION_COUNT 0U
```

Definition at line 24 of file protection_sau.c.

### 7.114.2 Function Documentation

#### 7.114.2.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 34 of file protection_sau.c.
Here is the call graph for this function:

## 7.115 platform/ext/target/infineon/common/spe/protection/protection_sau.h File Reference

```
#include "fih.h"
#include "tfm_hal_defs.h"
```

Include dependency graph for protection_sau.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_sau_cfg(void)

  *Configures the SAU.*

## 7.115.1 Function Documentation

### 7.115.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures the SAU.

**Returns**

TFM_HAL_SUCCESS - SAU configured successfully TFM_HAL_ERROR_GENERIC - failed to configure SAU

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.

Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)

Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |

**Parameters**

| | | |
|---|---|---|
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 67 of file faults.c.

## 7.116 platform/ext/target/infineon/common/spe/protection/protection_↩ shared_data.c File Reference

`#include "protection_shared_data.h"`
Include dependency graph for protection_shared_data.c:

### Variables

- uint32_t ifx_spm_state = 0x3498A78Bu

  *Shared state between SPM and partitions.*

### 7.116.1 Variable Documentation

#### 7.116.1.1 ifx_spm_state

```
uint32_t ifx_spm_state = 0x3498A78Bu
```

Shared state between SPM and partitions.
Partitions can access this value in read-only mode, while SPM have write access.
It's used by SE IPC Service to select a proper communication method.
Valid values are:

- IFX_SPM_STATE_INITIALIZING - allows SPM to use SE RT Services Utils. Must be used during static initialization phase only.
- IFX_SPM_STATE_RUNNING - SE IPC interface is used by SE IPC Service only.

Definition at line 11 of file protection_shared_data.c.

## 7.117 platform/ext/target/infineon/common/spe/protection/protection_shared_data.h File Reference

```
#include <stdint.h>
```

Include dependency graph for protection_shared_data.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_SPM_STATE_INITIALIZING 0x3498A78Bu
- #define IFX_SPM_STATE_RUNNING 0xA78B3498u
- #define IFX_IS_SPM_INITILIZING() (ifx_spm_state == IFX_SPM_STATE_INITIALIZING)
- #define IFX_IS_SPM_RUNNING() (ifx_spm_state == IFX_SPM_STATE_RUNNING)

## Variables

- uint32_t ifx_spm_state

  *Shared state between SPM and partitions.*

## 7.117.1 Macro Definition Documentation

### 7.117.1.1 IFX_IS_SPM_INITILIZING

```
#define IFX_IS_SPM_INITILIZING( ) (ifx_spm_state == IFX_SPM_STATE_INITIALIZING)
```

Definition at line 33 of file protection_shared_data.h.

### 7.117.1.2 IFX_IS_SPM_RUNNING

```
#define IFX_IS_SPM_RUNNING( ) (ifx_spm_state == IFX_SPM_STATE_RUNNING)
```

Definition at line 34 of file protection_shared_data.h.

### 7.117.1.3 IFX_SPM_STATE_INITIALIZING

```
#define IFX_SPM_STATE_INITIALIZING 0x3498A78Bu
```

Definition at line 15 of file protection_shared_data.h.

### 7.117.1.4 IFX_SPM_STATE_RUNNING

```
#define IFX_SPM_STATE_RUNNING 0xA78B3498u
```

Definition at line 17 of file protection_shared_data.h.

## 7.117.2 Variable Documentation

### 7.117.2.1 ifx_spm_state

```
uint32_t ifx_spm_state
```

Shared state between SPM and partitions.
Partitions can access this value in read-only mode, while SPM have write access.
It's used by SE IPC Service to select a proper communication method.
Valid values are:

- IFX_SPM_STATE_INITIALIZING - allows SPM to use SE RT Services Utils. Must be used during static initialization phase only.
- IFX_SPM_STATE_RUNNING - SE IPC interface is used by SE IPC Service only.

Definition at line 11 of file protection_shared_data.c.

## 7.118 platform/ext/target/infineon/common/spe/protection/protection_types.h File Reference

```
#include "config_tfm.h"
#include "fih.h"
#include "ifx_fih.h"
#include "ifx_spe_config.h"
#include "ifx_platform_spe_types.h"
```

Include dependency graph for protection_types.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct ifx_ppc_regions_config_t
- struct ifx_protection_region_config_t

  *Structure used to store platform specific protection properties.*
- struct ifx_partition_load_info_t

  *Structure used to store platform specific partition properties.*
- struct ifx_partition_info_t

  *Structure describing partition info.*

### Macros

- #define IFX_IS_PARTITION_PRIVILEGED(p_info) (fih_eq((p_info)->ifx_ldinfo->privileged, IFX_FIH_TRUE))
- #define IFX_SPM_BOUNDARY ((uintptr_t)0U)
- #define IFX_PROTECT_SPM_DOMAIN ((uintptr_t)0xC273706DU)
- #define IFX_PROTECT_PSA_ROT_SINGLE_DOMAIN ((uintptr_t)0xA5505341U)
- #define IFX_PROTECT_APP_ROT_SINGLE_DOMAIN ((uintptr_t)0xB6616050U)
- #define IFX_PROTECT_IS_PRIVILEGED_DOMAIN_PRIVATE(privileged, ...) ((privileged) != 0)
- #define IFX_PROTECT_IS_PRIVILEGED_DOMAIN(domain) IFX_PROTECT_IS_PRIVILEGED_DOMAIN_PRIVATE domain
- #define IFX_PROTECT_SPM_DOMAIN_CFG (1)
- #define IFX_PROTECT_PSA_ROT_DOMAIN_CFG (IFX_PSA_ROT_PRIVILEGED)

- #define IFX_PROTECT_APP_ROT_DOMAIN_CFG (IFX_APP_ROT_PRIVILEGED)
- #define IFX_PROTECT_IS_DOMAIN_EQUAL(domain1, domain2) (IFX_PROTECT_IS_PRIVILEGED_DOMAIN(domain1) == IFX_PROTECT_IS_PRIVILEGED_DOMAIN(domain2))
- #define IFX_GET_PARTITION_PC(p_info, secure) fih_int_encode(IFX_PC_TFM_SPM_ID)

  *Return partition's Protection Context using partition load info.*
- #define IFX_GET_BOUNDARY_DOMAIN(boundary) (((boundary) == IFX_SPM_BOUNDARY) ? IFX_PROTECT_SPM_DOMA : (((const ifx_partition_info_t *)boundary)->ifx_domain))

## Typedefs

- typedef struct ifx_ppc_regions_config_t ifx_ppc_regions_config_t
- typedef struct ifx_partition_info_t ifx_partition_info_t

  *Structure describing partition info.*

## 7.118.1 Macro Definition Documentation

### 7.118.1.1 IFX_GET_BOUNDARY_DOMAIN

```
#define IFX_GET_BOUNDARY_DOMAIN(
            boundary ) (((boundary) == IFX_SPM_BOUNDARY) ? IFX_PROTECT_SPM_DOMAIN : (((const
ifx_partition_info_t *)boundary)->ifx_domain))
```

Definition at line 140 of file protection_types.h.

### 7.118.1.2 IFX_GET_PARTITION_PC

```
#define IFX_GET_PARTITION_PC(
            p_info,
            secure ) fih_int_encode(IFX_PC_TFM_SPM_ID)
```

Return partition's Protection Context using partition load info.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition load info ifx_partition_info_t |
| in | *secure* | Boolean value, true for secure access |

Definition at line 135 of file protection_types.h.

### 7.118.1.3 IFX_IS_PARTITION_PRIVILEGED

```
#define IFX_IS_PARTITION_PRIVILEGED(
            p_info ) (fih_eq((p_info)->ifx_ldinfo->privileged, IFX_FIH_TRUE))
```

Definition at line 36 of file protection_types.h.

### 7.118.1.4 IFX_PROTECT_APP_ROT_DOMAIN_CFG

```
#define IFX_PROTECT_APP_ROT_DOMAIN_CFG (IFX_APP_ROT_PRIVILEGED)
```

Definition at line 107 of file protection_types.h.

### 7.118.1.5 IFX_PROTECT_APP_ROT_SINGLE_DOMAIN

```
#define IFX_PROTECT_APP_ROT_SINGLE_DOMAIN ((uintptr_t)0xB6616050U)
```

Definition at line 47 of file protection_types.h.

#### 7.118.1.6 IFX_PROTECT_IS_DOMAIN_EQUAL

```
#define IFX_PROTECT_IS_DOMAIN_EQUAL(
            domain1,
            domain2 ) (IFX_PROTECT_IS_PRIVILEGED_DOMAIN(domain1) == IFX_PROTECT_IS_PRIVILEGED_DOMAIN(domain2
```

Definition at line 120 of file protection_types.h.

#### 7.118.1.7 IFX_PROTECT_IS_PRIVILEGED_DOMAIN

```
#define IFX_PROTECT_IS_PRIVILEGED_DOMAIN(
            domain ) IFX_PROTECT_IS_PRIVILEGED_DOMAIN_PRIVATE domain
```

Definition at line 58 of file protection_types.h.

#### 7.118.1.8 IFX_PROTECT_IS_PRIVILEGED_DOMAIN_PRIVATE

```
#define IFX_PROTECT_IS_PRIVILEGED_DOMAIN_PRIVATE(
            privileged,
            ...  ) ((privileged) != 0)
```

Definition at line 50 of file protection_types.h.

#### 7.118.1.9 IFX_PROTECT_PSA_ROT_DOMAIN_CFG

```
#define IFX_PROTECT_PSA_ROT_DOMAIN_CFG (IFX_PSA_ROT_PRIVILEGED)
```

Definition at line 104 of file protection_types.h.

#### 7.118.1.10 IFX_PROTECT_PSA_ROT_SINGLE_DOMAIN

```
#define IFX_PROTECT_PSA_ROT_SINGLE_DOMAIN ((uintptr_t)0xA5505341U)
```

Definition at line 44 of file protection_types.h.

#### 7.118.1.11 IFX_PROTECT_SPM_DOMAIN

```
#define IFX_PROTECT_SPM_DOMAIN ((uintptr_t)0xC273706DU)
```

Definition at line 41 of file protection_types.h.

#### 7.118.1.12 IFX_PROTECT_SPM_DOMAIN_CFG

```
#define IFX_PROTECT_SPM_DOMAIN_CFG (1)
```

Definition at line 101 of file protection_types.h.

#### 7.118.1.13 IFX_SPM_BOUNDARY

```
#define IFX_SPM_BOUNDARY ((uintptr_t)0U)
```

Definition at line 38 of file protection_types.h.

### 7.118.2 Typedef Documentation

#### 7.118.2.1 ifx_partition_info_t

```
typedef struct ifx_partition_info_t ifx_partition_info_t
```

Structure describing partition info.
Pointer to this structure is used to provide boundary for isolation HAL.

#### 7.118.2.2 ifx_ppc_regions_config_t

```
typedef struct ifx_ppc_regions_config_t ifx_ppc_regions_config_t
```

## 7.119 platform/ext/target/infineon/common/spe/protection/protection_tz.c File Reference

```
#include <arm_cmse.h>
#include <stdint.h>
#include <stdbool.h>
#include "config_impl.h"
#include "config_tfm.h"
#include "protection_tz.h"
#include "static_checks.h"
#include "tfm_hal_isolation.h"
```

Include dependency graph for protection_tz.c:

### Functions

- if ((UINTPTR_MAX - p)< size)
- if (flags &(~known))
- switch (flags &known_secure_level)
- if (permb.value !=perme.value)
- switch (flags &(~known_secure_level))
- FIH_RET (fih_int_encode(TFM_HAL_ERROR_MEM_FAULT))
- if (IS_NS_AGENT_MAILBOX(p_info->p_ldinfo) &&fih_eq(is_non_secure, FIH_SUCCESS))
- void fih_delay ()
- if ((access_type &TFM_HAL_ACCESS_WRITABLE)==TFM_HAL_ACCESS_WRITABLE)
- else if ((access_type &TFM_HAL_ACCESS_READABLE) !=0u)
- if (IS_NS_AGENT_TZ(p_info->p_ldinfo))
- if (fih_not_eq(fih_rc, fih_int_encode(TFM_HAL_SUCCESS)))
- FIH_RET (fih_rc)

## Variables

- static size_t size
- static size_t uint32_t flags
- char ∗ pb = (char ∗) p
- char ∗ pe = pb + size - 1
- uint32_t known = ((uint32_t)CMSE_MPU_UNPRIV) | ((uint32_t)CMSE_MPU_READWRITE) | ((uint32_t)C↩MSE_MPU_READ)
- uint32_t known_secure_level = CMSE_MPU_UNPRIV
- const bool singleCheck = (((uintptr_t)pb ^ (uintptr_t)pe) < 32U)
- void perme
- uintptr_t base
- uintptr_t size_t uint32_t access_type
- fih_int is_non_secure = ((access_type & TFM_HAL_ACCESS_NS) != 0u) ? FIH_SUCCESS : FIH_FAILURE
- else

## 7.119.1 Function Documentation

### 7.119.1.1 fih_delay()

```
void fih_delay ( )
```

### 7.119.1.2 FIH_RET() [1/2]

```
FIH_RET (
            fih_int_encode(TFM_HAL_ERROR_MEM_FAULT)  )
```

Here is the caller graph for this function:

### 7.119.1.3 FIH_RET() [2/2]

```
FIH_RET (
            fih_rc  )
```

### 7.119.1.4 if() [1/8]

```
else if (
            (access_type &TFM_HAL_ACCESS_READABLE) !  = 0u )
```

Definition at line 158 of file protection_tz.c.

#### 7.119.1.5 if() [2/8]

```
if (
            (access_type &TFM_HAL_ACCESS_WRITABLE)  = = TFM_HAL_ACCESS_WRITABLE )
```

Definition at line 156 of file protection_tz.c.

#### 7.119.1.6 if() [3/8]

```
if ( )
```

Definition at line 38 of file protection_tz.c.
Here is the call graph for this function:

#### 7.119.1.7 if() [4/8]

```
if (
            fih_not_eq(fih_rc, fih_int_encode(TFM_HAL_SUCCESS))  )
```

Definition at line 191 of file protection_tz.c.
Here is the call graph for this function:

#### 7.119.1.8 if() [5/8]

```
if (
            flags & ~known )
```

Definition at line 49 of file protection_tz.c.

Here is the call graph for this function:

#### 7.119.1.9 if() [6/8]

```
if (
            IS_NS_AGENT_MAILBOX(p_info->p_ldinfo) &&fih_eq(is_non_secure, FIH_SUCCESS)  )
```

Definition at line 146 of file protection_tz.c.
Here is the call graph for this function:

#### 7.119.1.10 if() [7/8]

```
if (
            IS_NS_AGENT_TZ(p_info->p_ldinfo)  )
```

Definition at line 166 of file protection_tz.c.

#### 7.119.1.11 if() [8/8]

```
if (
            permb.value !    = perme.value )
```

Definition at line 85 of file protection_tz.c.
Here is the call graph for this function:

#### 7.119.1.12 switch() [1/2]

```
switch (
            flags & ~known_secure_level )
```

Definition at line 95 of file protection_tz.c.
Here is the call graph for this function:

#### 7.119.1.13 switch() [2/2]

```
switch (
            flags & known_secure_level )
```

Definition at line 58 of file protection_tz.c.
Here is the call graph for this function:

### 7.119.2 Variable Documentation

#### 7.119.2.1 access_type

```
uintptr_t size_t uint32_t access_type
```

**Initial value:**

```
{
    FIH_RET_TYPE(enum tfm_hal_status_t) fih_rc = fih_int_encode(TFM_HAL_ERROR_GENERIC)
```

Definition at line 139 of file protection_tz.c.

#### 7.119.2.2 base

```
uintptr_t base
```

Definition at line 136 of file protection_tz.c.

#### 7.119.2.3 else

```
else
```

**Initial value:**

```
{
        FIH_RET(fih_int_encode(TFM_HAL_ERROR_INVALID_INPUT))
```

Definition at line 160 of file protection_tz.c.

#### 7.119.2.4 flags

```
uint32_t flags
```

**Initial value:**

```
{
    cmse_address_info_t permb, perme
```

Definition at line 33 of file protection_tz.c.

#### 7.119.2.5 is_non_secure

```
fih_int is_non_secure = ((access_type & TFM_HAL_ACCESS_NS) != 0u) ?  FIH_SUCCESS : FIH_FAILURE
```

Definition at line 141 of file protection_tz.c.

#### 7.119.2.6 known

```
uint32_t known = ((uint32_t)CMSE_MPU_UNPRIV) | ((uint32_t)CMSE_MPU_READWRITE) | ((uint32_t)CM↩
SE_MPU_READ)
```

Definition at line 43 of file protection_tz.c.

#### 7.119.2.7 known_secure_level

```
uint32_t known_secure_level = CMSE_MPU_UNPRIV
```

Definition at line 44 of file protection_tz.c.

#### 7.119.2.8 pb

```
char* pb = (char *) p
```

Definition at line 35 of file protection_tz.c.

#### 7.119.2.9 pe

```
pe = pb + size - 1
```

Definition at line 35 of file protection_tz.c.

#### 7.119.2.10 perme

```
void perme
```

Definition at line 90 of file protection_tz.c.

#### 7.119.2.11 singleCheck

```
const bool singleCheck = (((uintptr_t)pb ^ (uintptr_t)pe) < 32U)
```

Definition at line 55 of file protection_tz.c.

#### 7.119.2.12 size

```
uintptr_t size_t size
```

Definition at line 32 of file protection_tz.c.

## 7.120 platform/ext/target/infineon/common/spe/protection/protection_tz.h File Reference

```
#include "fih.h"
#include "protection_types.h"
#include "tfm_hal_defs.h"
```

Include dependency graph for protection_tz.h:

This graph shows which files directly or indirectly include this file:

### Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_tz_memory_check(const ifx_partition_info_t *p_info

  *Check access to memory region by TrustZone (SAU, IDAU, MPU).*

### Variables

- uintptr_t base
- uintptr_t size_t size
- uintptr_t size_t uint32_t access_type

### 7.120.1 Function Documentation

#### 7.120.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t ) const
```

Check access to memory region by TrustZone (SAU, IDAU, MPU).

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |
| in | *privileged* | Is partition privileged. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by TrustZone. TFM_HAL_ERRO↩ R_MEM_FAULT - The memory region has not the access permissions by TrustZone. TFM_HAL_ERROR_↩ INVALID_INPUT - Invalid inputs.

### 7.120.2 Variable Documentation

#### 7.120.2.1 access_type

```
uintptr_t size_t uint32_t access_type
```

Definition at line 34 of file protection_tz.h.

#### 7.120.2.2 base

```
uintptr_t base
```

Definition at line 32 of file protection_tz.h.

#### 7.120.2.3 size

```
uintptr_t size_t size
```

Definition at line 33 of file protection_tz.h.

## 7.121 platform/ext/target/infineon/common/spe/protection/protection_↩ utils.c File Reference

```
#include "protection_utils.h"
#include "ifx_regions.h"
#include "mxs22.h"
```

Include dependency graph for protection_utils.c:

## Functions

- const ifx_memory_config_t ∗ ifx_find_memory_config (uint32_t address, uint32_t size, const ifx_memory_config_t ∗const configs[ ], const size_t config_count)
- enum tfm_hal_status_t ifx_get_all_memory_configs (const ifx_memory_config_t ∗memory_config[ ], uint32↩_t ∗list_size, uint32_t mem_address, uint32_t mem_size)

## 7.121.1 Function Documentation

### 7.121.1.1 ifx_find_memory_config()

```
const ifx_memory_config_t* ifx_find_memory_config (
            uint32_t address,
            uint32_t size,
            const ifx_memory_config_t *const configs[],
            const size_t config_count )
```

Definition at line 14 of file protection_utils.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.121.1.2 ifx_get_all_memory_configs()

```
enum tfm_hal_status_t ifx_get_all_memory_configs (
            const ifx_memory_config_t * memory_config[],
            uint32_t * list_size,
            uint32_t mem_address,
            uint32_t mem_size )
```

Definition at line 43 of file protection_utils.c.
Here is the call graph for this function:

## 7.122 platform/ext/target/infineon/common/spe/protection/protection_utils.h File Reference

```
#include "protection_types.h"
```

Include dependency graph for protection_utils.h:

This graph shows which files directly or indirectly include this file:

## 7.123 platform/ext/target/infineon/common/spe/protection/tfm_hal_↩ isolation.c File Reference

```
#include <arm_cmse.h>
#include "cmsis.h"
#include "config_tfm.h"
#include "faults.h"
#include "protection_types.h"
#include "protection_mpu.h"
#include "protection_msc.h"
#include "protection_sau.h"
#include "protection_tz.h"
#include "partition_info.h"
#include "ifx_regions.h"
#include "tfm_hal_isolation.h"
#include "tfm_peripherals_def.h"
#include "load/spm_load_api.h"
#include "load/asset_defs.h"
#include "protection_shared_data.h"
#include "protection_utils.h"
#include "ifx_spm_init.h"
#include "static_checks.h"
```

Include dependency graph for tfm_hal_isolation.c:

### Functions

- if ((p_info==NULL)||(cfg==NULL))
- for (asset_idx=0;asset_idx< asset_cnt;asset_idx++)
- FIH_RET (fih_int_encode(TFM_HAL_SUCCESS))
- if (p_ldinf->nassets !=0U)
- if (p_info->asset_cnt !=0U)
- for (uint32_t i=0;i< p_info->peri_region_count;i++)
- if (size==0U)
- if (base==0U)
- if (base >(UINTPTR_MAX - size))

- if ((access_type &∼(TFM_HAL_ACCESS_READABLE|TFM_HAL_ACCESS_WRITABLE|TFM_HAL_AC↩ CESS_NS)) !=0U)
- FIH_CALL (ifx_tz_memory_check, fih_rc, p_info, base, size, access_type)
- if (fih_not_eq(fih_rc, fih_int_encode(TFM_HAL_SUCCESS)))
- FIH_CALL (ifx_mpc_memory_check, fih_rc, p_info, base, size, access_type)
- FIH_RET (fih_rc)
- if ((p_ldinf==NULL)||(p_boundary==NULL))
- FIH_CALL (ifx_protect_partition_assets, result, p_info, p_info->ifx_ldinfo->protect_cfg_enabled)
- if (fih_not_eq(result, fih_int_encode(TFM_HAL_SUCCESS)))
- FIH_RET_TYPE (bool)
- __DMB ()
- if (boundary !=((uintptr_t) 0U))
- __set_CONTROL (ctrl.w)
- __DSB ()
- __ISB ()
- FIH_RET (result)

## Variables

- static cy_en_prot_region_t asset
- static const ifx_protection_region_config_t * cfg
- static const ifx_protection_region_config_t const struct asset_desc_t * p_assets
- static const ifx_protection_region_config_t const struct asset_desc_t uint32_t asset_cnt
- const struct partition_load_info_t * p_ldinf = p_info->p_ldinfo
- uintptr_t base
- uintptr_t size_t size
- uintptr_t size_t uint32_t access_type
- uintptr_t * p_boundary
- const ifx_partition_info_t * p_info = ifx_protection_get_partition_info(p_ldinf)
- uintptr_t boundary
- CONTROL_Type ctrl
- result = fih_int_encode(TFM_HAL_ERROR_GENERIC)
- ctrl w = __get_CONTROL()
- else

## 7.123.1 Function Documentation

### 7.123.1.1 __DMB()

```
__DMB ( )
```

Here is the caller graph for this function:

#### 7.123.1.2 __DSB()

```
__DSB ( )
```

Here is the caller graph for this function:

#### 7.123.1.3 __ISB()

```
__ISB ( )
```

Here is the caller graph for this function:

#### 7.123.1.4 __set_CONTROL()

```
__set_CONTROL (
            ctrl.  w )
```

Here is the caller graph for this function:

#### 7.123.1.5 FIH_CALL() [1/3]

```
FIH_CALL (
            ifx_mpc_memory_check ,
            fih_rc ,
            p_info ,
            base ,
            size ,
            access_type  )
```

#### 7.123.1.6 FIH_CALL() [2/3]

```
FIH_CALL (
            ifx_protect_partition_assets ,
            result ,
            p_info ,
            p_info->ifx_ldinfo-> protect_cfg_enabled )
```

#### 7.123.1.7 FIH_CALL() [3/3]

```
FIH_CALL (
            ifx_tz_memory_check ,
            fih_rc ,
            p_info ,
            base ,
            size ,
            access_type  )
```

Here is the caller graph for this function:

#### 7.123.1.8 FIH_RET() [1/3]

```
FIH_RET (
            fih_int_encode(TFM_HAL_SUCCESS)  )
```

Here is the caller graph for this function:

#### 7.123.1.9 FIH_RET() [2/3]

```
FIH_RET (
            fih_rc  )
```

#### 7.123.1.10 FIH_RET() [3/3]

```
FIH_RET (
            result  )
```

#### 7.123.1.11 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            bool  )
```

Definition at line 629 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.12 for() [1/2]

```
for ( )
```

Definition at line 130 of file tfm_hal_isolation.c.

Here is the call graph for this function:

#### 7.123.1.13 for() [2/2]

```
for (
            uint32_t i = 0; i < p_info->peri_region_count; i++ )
```

Definition at line 300 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.14 if() [1/11]

```
if (
            (access_type &~(TFM_HAL_ACCESS_READABLE|TFM_HAL_ACCESS_WRITABLE|TFM_HAL_ACCESS_↩
NS)) !  = 0U )
```

Definition at line 485 of file tfm_hal_isolation.c.

Here is the call graph for this function:

#### 7.123.1.15 if() [2/11]

```
if (
            (p_info==NULL)||(cfg==NULL)  )
```

Definition at line 126 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.16 if() [3/11]

```
if (
            (p_ldinf==NULL)||(p_boundary==NULL)  )
```

Definition at line 533 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.17 if() [4/11]

```
if (
            base ,
            (UINTPTR_MAX - size)  )
```

Definition at line 480 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.18 if() [5/11]

```
if (
            base  = = 0U )
```

Definition at line 475 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.19 if() [6/11]

```
if (
            boundary !    = ((uintptr_t)0U) )
```

Definition at line 749 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.20 if() [7/11]

```
if (
            fih_not_eq(fih_rc, fih_int_encode(TFM_HAL_SUCCESS))  )
```

Definition at line 495 of file tfm_hal_isolation.c.

Here is the call graph for this function:

#### 7.123.1.21 if() [8/11]

```
if (
            fih_not_eq(result, fih_int_encode(TFM_HAL_SUCCESS))  )
```

Definition at line 574 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.22 if() [9/11]

```
if (
            p_info->asset_cnt !    = 0U )
```

Definition at line 246 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.23 if() [10/11]

```
if (
            p_ldinf->nassets !     = 0U )
```

Definition at line 233 of file tfm_hal_isolation.c.
Here is the call graph for this function:

#### 7.123.1.24 if() [11/11]

```
if (
            size   = = 0U )
```

Definition at line 471 of file tfm_hal_isolation.c.
Here is the call graph for this function:

### 7.123.2 Variable Documentation

#### 7.123.2.1 access_type

```
uintptr_t size_t uint32_t access_type
```

**Initial value:**

```
{
    const ifx_partition_info_t *p_info = (const ifx_partition_info_t *)boundary
```

Definition at line 455 of file tfm_hal_isolation.c.

#### 7.123.2.2 asset

```
cy_en_prot_region_t asset
```

Definition at line 63 of file tfm_hal_isolation.c.

#### 7.123.2.3 asset_cnt

```
const ifx_protection_region_config_t const struct asset_desc_t uint32_t asset_cnt
```

**Initial value:**

```
{
    size_t  asset_idx
```

Definition at line 122 of file tfm_hal_isolation.c.

#### 7.123.2.4 base

```
uintptr_t base
```

Definition at line 453 of file tfm_hal_isolation.c.

#### 7.123.2.5 boundary

```
uintptr_t boundary
```

**Initial value:**

```
{
    FIH_RET_TYPE(enum tfm_hal_status_t) result
```

Definition at line 675 of file tfm_hal_isolation.c.

#### 7.123.2.6 cfg

```
static const ifx_protection_region_config_t * cfg
```

**Initial value:**

```
{
    FIH_RET_TYPE(enum tfm_hal_status_t) result
```

Definition at line 119 of file tfm_hal_isolation.c.

#### 7.123.2.7 ctrl

```
CONTROL_Type ctrl
```

Definition at line 677 of file tfm_hal_isolation.c.

#### 7.123.2.8 else

```
else
```

**Initial value:**

```
{

        ctrl.b.nPRIV = 0
```

Definition at line 756 of file tfm_hal_isolation.c.

#### 7.123.2.9 p_assets

```
const ifx_protection_region_config_t const struct asset_desc_t* p_assets
```

Definition at line 120 of file tfm_hal_isolation.c.

#### 7.123.2.10 p_boundary

```
* p_boundary
```

**Initial value:**

```
{
    FIH_RET_TYPE(enum tfm_hal_status_t) result
```

Definition at line 530 of file tfm_hal_isolation.c.

#### 7.123.2.11 p_info

```
const ifx_partition_info_t* p_info = ifx_protection_get_partition_info(p_ldinf)
```

Definition at line 538 of file tfm_hal_isolation.c.

#### 7.123.2.12 p_ldinf

```
const struct partition_load_info_t* p_ldinf = p_info->p_ldinfo
```

Definition at line 232 of file tfm_hal_isolation.c.

#### 7.123.2.13 result

```
result = fih_int_encode(TFM_HAL_ERROR_GENERIC)
```

Definition at line 685 of file tfm_hal_isolation.c.

#### 7.123.2.14 size

```
uintptr_t size_t size
```

Definition at line 454 of file tfm_hal_isolation.c.

#### 7.123.2.15 w

```
ctrl w = __get_CONTROL()
```

Definition at line 748 of file tfm_hal_isolation.c.

## 7.124 platform/ext/target/infineon/common/spe/protection/tfm_platform↩ _arch_hooks.c File Reference

```
#include "config_tfm.h"
#include "tfm_platform_arch_hooks.h"
#include "spm.h"
#include "static_checks.h"
#include "protection_regions_cfg.h"
```

Include dependency graph for tfm_platform_arch_hooks.c:

### Functions

- ifx_aapcs_fih_int ifx_arch_get_context_protection_context (const struct context_ctrl_t *prev_ctx, const struct context_ctrl_t *cur_ctx, uint32_t exc_return)
- ifx_aapcs_fih_int ifx_arch_get_current_component_protection_context (uint32_t exc_return)

### 7.124.1 Function Documentation

#### 7.124.1.1 ifx_arch_get_context_protection_context()

```
ifx_aapcs_fih_int ifx_arch_get_context_protection_context (
            const struct context_ctrl_t * prev_ctx,
            const struct context_ctrl_t * cur_ctx,
            uint32_t exc_return )
```

Definition at line 36 of file tfm_platform_arch_hooks.c.

#### 7.124.1.2 ifx_arch_get_current_component_protection_context()

```
ifx_aapcs_fih_int ifx_arch_get_current_component_protection_context (
            uint32_t exc_return )
```

Definition at line 44 of file tfm_platform_arch_hooks.c.
Here is the call graph for this function:

## 7.125 platform/ext/target/infineon/common/spe/protection/tfm_platform↩ _arch_hooks.h File Reference

```
#include "ifx_fih.h"
#include "ifx_regions.h"
#include "tfm_arch.h"
#include "protection_pc.h"
#include "protection_regions_cfg.h"
```

Include dependency graph for tfm_platform_arch_hooks.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_PLATFORM_EXIT_FROM_INTERRUPT_WITH_PC_RESTORE_HOOK
- #define PLATFORM_SVC_HANDLER_GET_MSP_HOOK(reg)
- #define PLATFORM_SVC_HANDLER_EXIT_HOOK
- #define PLATFORM_INIT_SPM_FUNC_CONTEXT_HOOK(msp, ctx)

  *Hook to update isolation before switching to SPM thread function.*
- #define PLATFORM_THREAD_MODE_SPM_RETURN_HOOK(exc_return, msp)

  *Hook to update isolation to handle exit from SPM thread function.*
- #define PLATFORM_SVC_HANDLER_HOOK_IAR_REQUIRED
- #define PLATFORM_SVC_HANDLER_TO_FLIH_FUNC_ENTER_HOOK
- #define PLATFORM_SVC_HANDLER_TO_FLIH_FUNC_EXIT_HOOK
- #define PLATFORM_SVC_HANDLER_FROM_FLIH_FUNC_ENTER_HOOK
- #define PLATFORM_SVC_HANDLER_FROM_FLIH_FUNC_EXIT_HOOK
- #define PLATFORM_PENDSV_HANDLER_HOOK_IAR_REQUIRED
- #define PLATFORM_PENDSV_HANDLER_EXIT_HOOK
- #define PLATFORM_HARD_FAULT_HANDLER_HOOK_VALIDATE_PC()
- #define PLATFORM_HARD_FAULT_HANDLER_HOOK_IAR_REQUIRED _Pragma("required = ifx_arch_switch_protection_co
- #define PLATFORM_HARD_FAULT_HANDLER_HOOK()

## Functions

- ifx_aapcs_fih_int ifx_arch_get_context_protection_context (const struct context_ctrl_t *prev_ctx, const struct context_ctrl_t *cur_ctx, uint32_t exc_return)
- ifx_aapcs_fih_int ifx_arch_get_current_component_protection_context (uint32_t exc_return)
- void ifx_arch_switch_protection_context (uint32_t pc)

  *Perform protection context switching.*
- void ifx_arch_switch_protection_context_from_irq (uint32_t pc)

  *Perform protection context switching from low priority IRQ.*

### 7.125.1 Macro Definition Documentation

#### 7.125.1.1 IFX_PLATFORM_EXIT_FROM_INTERRUPT_WITH_PC_RESTORE_HOOK

```
#define IFX_PLATFORM_EXIT_FROM_INTERRUPT_WITH_PC_RESTORE_HOOK
```

**Value:**

```
        "cpsid  i                                   \n" \
        /* Pop Protection Context from stack */ \
        "pop    {r0, r1}                            \n" \
        /* Switch Protection Context */ \
        "b.w    ifx_arch_switch_protection_context_from_irq \n"
```

Definition at line 36 of file tfm_platform_arch_hooks.h.

#### 7.125.1.2 PLATFORM_HARD_FAULT_HANDLER_HOOK

```
#define PLATFORM_HARD_FAULT_HANDLER_HOOK( )
```

Definition at line 239 of file tfm_platform_arch_hooks.h.

#### 7.125.1.3 PLATFORM_HARD_FAULT_HANDLER_HOOK_IAR_REQUIRED

```
#define PLATFORM_HARD_FAULT_HANDLER_HOOK_IAR_REQUIRED _Pragma("required = ifx_arch_switch_protection_context")
```

Definition at line 210 of file tfm_platform_arch_hooks.h.

#### 7.125.1.4 PLATFORM_HARD_FAULT_HANDLER_HOOK_VALIDATE_PC

```
#define PLATFORM_HARD_FAULT_HANDLER_HOOK_VALIDATE_PC( )
```

Definition at line 205 of file tfm_platform_arch_hooks.h.

#### 7.125.1.5 PLATFORM_INIT_SPM_FUNC_CONTEXT_HOOK

```
#define PLATFORM_INIT_SPM_FUNC_CONTEXT_HOOK(
            msp,
            ctx )
```

**Value:**

```
        { \
            /* Switch to protection context used by SPM. */ \
            /* Update saved PC stored by \ref ifx_pc2_exception_handler */ \
            ((uint32_t *)msp)[-1] = IFX_PC_TFM_SPM_ID; \
            ((uint32_t *)msp)[-2] = IFX_PC_TFM_SPM_ID; \
        }
```

Hook to update isolation before switching to SPM thread function.

**Parameters**

| | |
|---|---|
| *msp* | Address of MSP stack frame after adjusting it by PLATFORM_SVC_HANDLER_GET_MSP_HOOK |
| *ctx* | Address of SPM PSP stack used to handle SVC request in thread mode |

Definition at line 77 of file tfm_platform_arch_hooks.h.

#### 7.125.1.6 PLATFORM_PENDSV_HANDLER_EXIT_HOOK

```
#define PLATFORM_PENDSV_HANDLER_EXIT_HOOK
```

**Value:**

```
        "cpsid  i                                       \n" \
        /* Pop Saved Protection Context */ \
        "pop    {r2, r3}                                \n" \
        "push   {lr}                                    \n" \
        "mov    r2, lr                                  \n" \
        "bl     ifx_arch_get_context_protection_context \n" \
        "pop    {lr}                                    \n" \
        "b.w    ifx_arch_switch_protection_context_from_irq \n"
```

Definition at line 171 of file tfm_platform_arch_hooks.h.

#### 7.125.1.7 PLATFORM_PENDSV_HANDLER_HOOK_IAR_REQUIRED

```
#define PLATFORM_PENDSV_HANDLER_HOOK_IAR_REQUIRED
```

**Value:**

```
_Pragma("required = ifx_arch_get_context_protection_context") \
_Pragma("required = ifx_arch_switch_protection_context_from_irq")
```

Definition at line 162 of file tfm_platform_arch_hooks.h.

#### 7.125.1.8 PLATFORM_SVC_HANDLER_EXIT_HOOK

```
#define PLATFORM_SVC_HANDLER_EXIT_HOOK
```

**Value:**

```
        "cpsid  i                                   \n" \
        /* Pop Protection Context from stack */ \
        "pop    {r0, r1}                            \n" \
        /* Switch Protection Context */ \
        "b.w    ifx_arch_switch_protection_context \n"
```

Definition at line 54 of file tfm_platform_arch_hooks.h.

#### 7.125.1.9 PLATFORM_SVC_HANDLER_FROM_FLIH_FUNC_ENTER_HOOK

```
#define PLATFORM_SVC_HANDLER_FROM_FLIH_FUNC_ENTER_HOOK
```

**Value:**

```
        /* Remove Saved PC for \ref TFM_SVC_FLIH_FUNC_RETURN SVC call from stack */ \
        "pop     {r0, r1}                                          \n"
```

Definition at line 149 of file tfm_platform_arch_hooks.h.

#### 7.125.1.10 PLATFORM_SVC_HANDLER_FROM_FLIH_FUNC_EXIT_HOOK

```
#define PLATFORM_SVC_HANDLER_FROM_FLIH_FUNC_EXIT_HOOK
```

**Value:**

```
        /* There is no need to switch PC, because \ref TFM_SVC_FLIH_FUNC_RETURN should
* return to \ref spm_handle_interrupt which is running within privileged
* PC2/SPM context */ \
        "bx      lr                                          \n"
```

Definition at line 157 of file tfm_platform_arch_hooks.h.

#### 7.125.1.11 PLATFORM_SVC_HANDLER_GET_MSP_HOOK

```
#define PLATFORM_SVC_HANDLER_GET_MSP_HOOK(
               reg )
```

**Value:**

```
        /* 8 bytes used to store Saved PC */ \
        "mrs "  reg ", msp                      \n" \
        "add "  reg ", #8                       \n"
```

Definition at line 46 of file tfm_platform_arch_hooks.h.

#### 7.125.1.12 PLATFORM_SVC_HANDLER_HOOK_IAR_REQUIRED

```
#define PLATFORM_SVC_HANDLER_HOOK_IAR_REQUIRED
```

**Value:**

```
_Pragma("required = ifx_arch_get_current_component_protection_context") \
_Pragma("required = ifx_arch_switch_protection_context")
```

Definition at line 108 of file tfm_platform_arch_hooks.h.

#### 7.125.1.13 PLATFORM_SVC_HANDLER_TO_FLIH_FUNC_ENTER_HOOK

```
#define PLATFORM_SVC_HANDLER_TO_FLIH_FUNC_ENTER_HOOK
```

**Value:**

```
        /* "Unstack" orig_exc_return, dummy, PSP, PSPLIM pushed by current handler */ \
        "pop     {r0-r3}                                          \n" \
        /* Remove Saved PC from stack */ \
        "add     sp, #8                                           \n" \
        /* "Stack" orig_exc_return, dummy, PSP, PSPLIM pushed by current handler */ \
        "push     {r0-r3}                                         \n"
```

Definition at line 122 of file tfm_platform_arch_hooks.h.

#### 7.125.1.14 PLATFORM_SVC_HANDLER_TO_FLIH_FUNC_EXIT_HOOK

```
#define PLATFORM_SVC_HANDLER_TO_FLIH_FUNC_EXIT_HOOK
```

**Value:**

```
        "cpsid  i                                                \n" \
        "push   {lr}                                             \n" \
        "mov    r0, lr                                           \n" \
        "bl     ifx_arch_get_current_component_protection_context \n" \
        "pop    {lr}                                             \n" \
        "b.w    ifx_arch_switch_protection_context               \n"
```

Definition at line 134 of file tfm_platform_arch_hooks.h.

#### 7.125.1.15 PLATFORM_THREAD_MODE_SPM_RETURN_HOOK

```
#define PLATFORM_THREAD_MODE_SPM_RETURN_HOOK(
            exc_return,
            msp )
```

**Value:**

```
        { \
            uint32_t component_pc = ifx_arch_get_current_component_protection_context(exc_return); \
            uint32_t return_pc = component_pc; \
            ((uint32_t *)msp)[-1] = return_pc; \
            ((uint32_t *)msp)[-2] = return_pc; \
        }
```

Hook to update isolation to handle exit from SPM thread function.
Definition at line 98 of file tfm_platform_arch_hooks.h.

### 7.125.2 Function Documentation

#### 7.125.2.1 ifx_arch_get_context_protection_context()

```
ifx_aapcs_fih_int ifx_arch_get_context_protection_context (
            const struct context_ctrl_t * prev_ctx,
            const struct context_ctrl_t * cur_ctx,
            uint32_t exc_return )
```

Definition at line 36 of file tfm_platform_arch_hooks.c.

#### 7.125.2.2 ifx_arch_get_current_component_protection_context()

```
ifx_aapcs_fih_int ifx_arch_get_current_component_protection_context (
            uint32_t exc_return )
```

Definition at line 44 of file tfm_platform_arch_hooks.c.

Here is the call graph for this function:

#### 7.125.2.3 ifx_arch_switch_protection_context()

```
void ifx_arch_switch_protection_context (
            uint32_t pc )
```

Perform protection context switching.

**Note**

It's unsafe to switch PC to other than PC2 (used by SPM) in low priority exception, because it's possible that there will be pending IRQ and MCU will try to use MSP stack pushing exception stack within PC not equal to PC2.

IRQ must be disabled if this function is called from exception/IRQ with privilege below 0.

Definition at line 158 of file protection_pc.c.

#### 7.125.2.4 ifx_arch_switch_protection_context_from_irq()

```
void ifx_arch_switch_protection_context_from_irq (
            uint32_t pc )
```

Perform protection context switching from low priority IRQ.
This function will complete switching to PC2 (SPM) in scope of current exception. For all other protection context it will schedule switching after exit from active IRQ in thread mode by ifx_arch_switch_protection_context_thread.

**Note**

IRQ must be disabled when this function is called.

Definition at line 250 of file protection_pc.c.

## 7.126 platform/ext/target/infineon/common/spe/provisioning/provisioning.c File Reference

```
#include "config_tfm.h"
#include "ifx_s_peripherals_def.h"
#include "mxs22.h"
#include "provisioning.h"
#include "tfm_plat_provisioning.h"
#include "cy_device.h"
#include "cy_syslib.h"
#include "static_checks.h"
```

Include dependency graph for provisioning.c:

## Functions

- enum tfm_plat_err_t tfm_plat_provisioning_perform (void)
- void tfm_plat_provisioning_check_for_dummy_keys (void)
- int tfm_plat_provisioning_is_required (void)
- uint32_t ifx_get_security_lifecycle (void)

  *Retrieve the security lifecycle of the device.*

## 7.126.1 Function Documentation

### 7.126.1.1 ifx_get_security_lifecycle()

```
uint32_t ifx_get_security_lifecycle (
            void  )
```

Retrieve the security lifecycle of the device.

**Returns**

Security lifecycle state, see PSA_LIFECYCLE_XXX in psa/lifecycle.h

Definition at line 44 of file provisioning.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.126.1.2 tfm_plat_provisioning_check_for_dummy_keys()

```
void tfm_plat_provisioning_check_for_dummy_keys (
            void  )
```

Definition at line 25 of file provisioning.c.

#### 7.126.1.3 tfm_plat_provisioning_is_required()

```
int tfm_plat_provisioning_is_required (
            void  )
```

Definition at line 32 of file provisioning.c.

#### 7.126.1.4 tfm_plat_provisioning_perform()

```
enum tfm_plat_err_t tfm_plat_provisioning_perform (
            void  )
```

Definition at line 18 of file provisioning.c.

## 7.127 platform/ext/target/infineon/common/spe/provisioning/provisioning.h File Reference

```
#include <stdint.h>
#include "psa/lifecycle.h"
#include "static_checks.h"
#include "cy_device.h"
```

Include dependency graph for provisioning.h:

This graph shows which files directly or indirectly include this file:

## Enumerations

- enum ifx_dap_state_t { IFX_DAP_DISABLED, IFX_DAP_ENABLED_NS_ONLY, IFX_DAP_ENABLED_S_NS, IFX_DAP_UNKNOWN }

  *Enumeration used to define debug access port state.*

## Functions

- uint32_t ifx_get_security_lifecycle (void)

  *Retrieve the security lifecycle of the device.*
- ifx_dap_state_t ifx_get_dap_state (void)

  *Retrieve the state of debug access port.*

## 7.127.1 Enumeration Type Documentation

### 7.127.1.1 ifx_dap_state_t

`enum ifx_dap_state_t`

Enumeration used to define debug access port state.

**Enumerator**

| | |
|---|---|
| IFX_DAP_DISABLED | |
| IFX_DAP_ENABLED_NS_ONLY | |
| IFX_DAP_ENABLED_S_NS | |
| IFX_DAP_UNKNOWN | |

Definition at line 21 of file provisioning.h.

## 7.127.2 Function Documentation

### 7.127.2.1 ifx_get_dap_state()

```
ifx_dap_state_t ifx_get_dap_state (
            void  )
```

Retrieve the state of debug access port.

**Returns**

debug access port state, see ifx_dap_state_t

Definition at line 63 of file ifx_platform_provisioning.c.
Here is the caller graph for this function:

#### 7.127.2.2 ifx_get_security_lifecycle()

```
uint32_t ifx_get_security_lifecycle (
            void  )
```

Retrieve the security lifecycle of the device.

**Returns**

Security lifecycle state, see PSA_LIFECYCLE_XXX in psa/lifecycle.h

Definition at line 44 of file provisioning.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.128 platform/ext/target/infineon/common/spe/services/attestation/ifx↩ _attest_hal.c File Reference

```
#include <stdbool.h>
#include <string.h>
#include "cmsis_compiler.h"
#include "config_tfm.h"
#include "provisioning.h"
#include "tfm_attest_hal.h"
#include "tfm_plat_boot_seed.h"
#include "psa/crypto.h"
```

Include dependency graph for ifx_attest_hal.c:

### Macros

- #define IFX_ATTESTATION_KEY_ID ((psa_key_id_t)(PSA_KEY_ID_VENDOR_MIN + 1U))

- #define IFX_ATTESTATION_PROFILE_DEFINITION "tag:psacertified.org,2023:psa#tfm"
- #define IFX_ATTESTATION_VERIFICATION_SERVICE "www.trustedfirmware.org"
- #define BOOL_SEED_INITIALIZED 0x3Bu
- #define BOOL_SEED_UNINITIALIZED 0xACu

## Functions

- enum tfm_plat_err_t tfm_plat_get_boot_seed (uint32_t size, uint8_t *buf)
- enum tfm_security_lifecycle_t tfm_attest_hal_get_security_lifecycle (void)
- enum tfm_plat_err_t tfm_attest_hal_get_verification_service (uint32_t *size, uint8_t *buf)
- enum tfm_plat_err_t tfm_attest_hal_get_profile_definition (uint32_t *size, uint8_t *buf)

## 7.128.1 Macro Definition Documentation

### 7.128.1.1 BOOL_SEED_INITIALIZED

```
#define BOOL_SEED_INITIALIZED 0x3Bu
```

Definition at line 35 of file ifx_attest_hal.c.

### 7.128.1.2 BOOL_SEED_UNINITIALIZED

```
#define BOOL_SEED_UNINITIALIZED 0xACu
```

Definition at line 36 of file ifx_attest_hal.c.

### 7.128.1.3 IFX_ATTESTATION_KEY_ID

```
#define IFX_ATTESTATION_KEY_ID ((psa_key_id_t)(PSA_KEY_ID_VENDOR_MIN + 1U))
```

Definition at line 19 of file ifx_attest_hal.c.

### 7.128.1.4 IFX_ATTESTATION_PROFILE_DEFINITION

```
#define IFX_ATTESTATION_PROFILE_DEFINITION "tag:psacertified.org,2023:psa#tfm"
```

Default value of profile definition for Initial Attestation service.
Definition at line 25 of file ifx_attest_hal.c.

### 7.128.1.5 IFX_ATTESTATION_VERIFICATION_SERVICE

```
#define IFX_ATTESTATION_VERIFICATION_SERVICE "www.trustedfirmware.org"
```

Default value of verification service for Initial Attestation.
Definition at line 32 of file ifx_attest_hal.c.

## 7.128.2 Function Documentation

### 7.128.2.1 tfm_attest_hal_get_profile_definition()

```
enum tfm_plat_err_t tfm_attest_hal_get_profile_definition (
            uint32_t * size,
            uint8_t * buf )
```

Definition at line 114 of file ifx_attest_hal.c.

Here is the call graph for this function:

#### 7.128.2.2 tfm_attest_hal_get_security_lifecycle()

```
enum tfm_security_lifecycle_t tfm_attest_hal_get_security_lifecycle (
            void  )
```

Definition at line 63 of file ifx_attest_hal.c.
Here is the call graph for this function:

#### 7.128.2.3 tfm_attest_hal_get_verification_service()

```
enum tfm_plat_err_t tfm_attest_hal_get_verification_service (
            uint32_t * size,
            uint8_t * buf )
```

Definition at line 94 of file ifx_attest_hal.c.
Here is the call graph for this function:

#### 7.128.2.4 tfm_plat_get_boot_seed()

```
enum tfm_plat_err_t tfm_plat_get_boot_seed (
            uint32_t size,
            uint8_t * buf )
```

Definition at line 38 of file ifx_attest_hal.c.
Here is the call graph for this function:

## 7.129 platform/ext/target/infineon/common/spe/services/attestation/ifx_attest_hal_se_rt.c File Reference

```
#include "attest.h"
#include "config_tfm.h"
#include "ifx_se_fih.h"
#include "ifx_se_platform.h"
#include "ifx_se_tfm_utils.h"
#include "psa/initial_attestation.h"
#include "tfm_attest_hal.h"
#include "tfm_plat_device_id.h"
#include "static_checks.h"
#include <string.h>
```

Include dependency graph for ifx_attest_hal_se_rt.c:

### Functions

- psa_status_t tfm_attest_hal_get_token (const void *challenge_buf, size_t challenge_size, void *token_buf, size_t token_buf_size, size_t *token_size)
- psa_status_t tfm_attest_hal_get_token_size (size_t challenge_size, size_t *token_size)

### 7.129.1 Function Documentation

#### 7.129.1.1 tfm_attest_hal_get_token()

```
psa_status_t tfm_attest_hal_get_token (
            const void * challenge_buf,
            size_t challenge_size,
            void * token_buf,
```

```
            size_t token_buf_size,
            size_t * token_size )
```

Definition at line 78 of file ifx_attest_hal_se_rt.c.
Here is the call graph for this function:

#### 7.129.1.2 tfm_attest_hal_get_token_size()

```
psa_status_t tfm_attest_hal_get_token_size (
            size_t challenge_size,
            size_t * token_size )
```

Definition at line 127 of file ifx_attest_hal_se_rt.c.

## 7.130 platform/ext/target/infineon/common/spe/services/attestation/ifx_plat_device_id.c File Reference

```
#include <string.h>
#include "config_tfm.h"
#include "cmsis.h"
#include "ifx_s_peripherals_def.h"
#include "tfm_plat_device_id.h"
#include "static_checks.h"
```

Include dependency graph for ifx_plat_device_id.c:

### Macros

- #define IFX_ADD_TO_IMPLEMENTATION_ID(buffer, val)
- #define IFX_ATTESTATION_HW_VERSION "0123456789012-12345"

### Functions

- enum tfm_plat_err_t tfm_plat_get_implementation_id (uint32_t *size, uint8_t *buf)
- enum tfm_plat_err_t tfm_plat_get_cert_ref (uint32_t *size, uint8_t *buf)

## 7.130.1 Macro Definition Documentation

### 7.130.1.1 IFX_ADD_TO_IMPLEMENTATION_ID

```
#define IFX_ADD_TO_IMPLEMENTATION_ID(
            buffer,
            val )
```

**Value:**

```
(void)memcpy((buffer), (const void*)(&(val)),
    sizeof((val))); \
(buffer) += sizeof((val));
```

Definition at line 19 of file ifx_plat_device_id.c.

### 7.130.1.2 IFX_ATTESTATION_HW_VERSION

```
#define IFX_ATTESTATION_HW_VERSION "0123456789012-12345"
```

Default value of H/W version for Initial Attestation service.
Definition at line 26 of file ifx_plat_device_id.c.

## 7.130.2 Function Documentation

### 7.130.2.1 tfm_plat_get_cert_ref()

```
enum tfm_plat_err_t tfm_plat_get_cert_ref (
            uint32_t * size,
            uint8_t * buf )
```

Definition at line 109 of file ifx_plat_device_id.c.
Here is the call graph for this function:

### 7.130.2.2 tfm_plat_get_implementation_id()

```
enum tfm_plat_err_t tfm_plat_get_implementation_id (
            uint32_t * size,
            uint8_t * buf )
```

Definition at line 29 of file ifx_plat_device_id.c.

Here is the call graph for this function:

## 7.131 platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_flash.c File Reference

```
#include "array.h"
#include "crypto_keys_flash.h"
#include "psa_manifest/pid.h"
#include "tfm_builtin_key_ids.h"
#include "tfm_builtin_key_loader.h"
#include "tfm_plat_crypto_keys.h"
#include "static_checks.h"
#include "ifx_spe_config.h"
#include <string.h>
#include <stdbool.h>
```

Include dependency graph for crypto_keys_flash.c:

### Functions

- size_t tfm_plat_builtin_key_get_policy_table_ptr (const tfm_plat_builtin_key_policy_t ∗desc_ptr[ ])
- size_t tfm_plat_builtin_key_get_desc_table_ptr (const tfm_plat_builtin_key_descriptor_t ∗desc_ptr[ ])

### 7.131.1 Function Documentation

#### 7.131.1.1 tfm_plat_builtin_key_get_desc_table_ptr()

```
size_t tfm_plat_builtin_key_get_desc_table_ptr (
            const tfm_plat_builtin_key_descriptor_t * desc_ptr[] )
```

Definition at line 162 of file crypto_keys_flash.c.

Here is the caller graph for this function:

#### 7.131.1.2 tfm_plat_builtin_key_get_policy_table_ptr()

```
size_t tfm_plat_builtin_key_get_policy_table_ptr (
            const tfm_plat_builtin_key_policy_t * desc_ptr[] )
```

Definition at line 156 of file crypto_keys_flash.c.

## 7.132 platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_rram.c File Reference

```
#include <string.h>
#include "crypto_keys_rram.h"
#include "ifx_assets_rram.h"
#include "psa_manifest/pid.h"
#include "tfm_builtin_key_ids.h"
#include "tfm_builtin_key_loader.h"
#include "tfm_plat_crypto_keys.h"
#include "static_checks.h"
```

Include dependency graph for crypto_keys_rram.c:

### Macros

- #define NUMBER_OF_ELEMENTS_OF(x) sizeof(x)/sizeof(∗x)
- #define MAPPED_TZ_NS_AGENT_DEFAULT_CLIENT_ID 0xaaaa7fffU
- #define MAPPED_MAILBOX_NS_AGENT_DEFAULT_CLIENT_ID 0xaaaaffffU
- #define TFM_TZ_NS_PARTITION_ID MAPPED_TZ_NS_AGENT_DEFAULT_CLIENT_ID
- #define TFM_MBOX_NS_PARTITION_ID MAPPED_MAILBOX_NS_AGENT_DEFAULT_CLIENT_ID

### Functions

- size_t tfm_plat_builtin_key_get_policy_table_ptr (const tfm_plat_builtin_key_policy_t ∗desc_ptr[ ])
- size_t tfm_plat_builtin_key_get_desc_table_ptr (const tfm_plat_builtin_key_descriptor_t ∗desc_ptr[ ])

### 7.132.1 Macro Definition Documentation

#### 7.132.1.1 MAPPED_MAILBOX_NS_AGENT_DEFAULT_CLIENT_ID

```
#define MAPPED_MAILBOX_NS_AGENT_DEFAULT_CLIENT_ID 0xaaaaffffU
```

Definition at line 33 of file crypto_keys_rram.c.

#### 7.132.1.2 MAPPED_TZ_NS_AGENT_DEFAULT_CLIENT_ID

```
#define MAPPED_TZ_NS_AGENT_DEFAULT_CLIENT_ID 0xaaaa7fffU
```

Definition at line 32 of file crypto_keys_rram.c.

#### 7.132.1.3 NUMBER_OF_ELEMENTS_OF

```
#define NUMBER_OF_ELEMENTS_OF(
              x ) sizeof(x)/sizeof(*x)
```

Definition at line 30 of file crypto_keys_rram.c.

#### 7.132.1.4 TFM_MBOX_NS_PARTITION_ID

```
#define TFM_MBOX_NS_PARTITION_ID MAPPED_MAILBOX_NS_AGENT_DEFAULT_CLIENT_ID
```

Definition at line 35 of file crypto_keys_rram.c.

#### 7.132.1.5 TFM_TZ_NS_PARTITION_ID

```
#define TFM_TZ_NS_PARTITION_ID MAPPED_TZ_NS_AGENT_DEFAULT_CLIENT_ID
```

Definition at line 34 of file crypto_keys_rram.c.

### 7.132.2 Function Documentation

#### 7.132.2.1 tfm_plat_builtin_key_get_desc_table_ptr()

```
size_t tfm_plat_builtin_key_get_desc_table_ptr (
             const tfm_plat_builtin_key_descriptor_t * desc_ptr[] )
```

Definition at line 181 of file crypto_keys_rram.c.

#### 7.132.2.2 tfm_plat_builtin_key_get_policy_table_ptr()

```
size_t tfm_plat_builtin_key_get_policy_table_ptr (
             const tfm_plat_builtin_key_policy_t * desc_ptr[] )
```

Definition at line 175 of file crypto_keys_rram.c.

## 7.133 platform/ext/target/infineon/common/spe/services/crypto/crypto_keys_se_rt.c File Reference

```
#include "crypto_platform.h"
#include "ifx_se_psacrypto.h"
#include "psa_manifest/pid.h"
#include "tfm_builtin_key_ids.h"
```

```
#include "tfm_builtin_key_loader.h"
#include "tfm_plat_crypto_keys.h"
#include "static_checks.h"
```

Include dependency graph for crypto_keys_se_rt.c:

## Macros

- #define NUMBER_OF_ELEMENTS_OF(x) sizeof(x)/sizeof(∗x)

## Functions

- size_t tfm_plat_builtin_key_get_desc_table_ptr (const tfm_plat_builtin_key_descriptor_t ∗desc_ptr[ ])

### 7.133.1 Macro Definition Documentation

#### 7.133.1.1 NUMBER_OF_ELEMENTS_OF

```
#define NUMBER_OF_ELEMENTS_OF(
              x ) sizeof(x)/sizeof(*x)
```

Definition at line 21 of file crypto_keys_se_rt.c.

### 7.133.2 Function Documentation

#### 7.133.2.1 tfm_plat_builtin_key_get_desc_table_ptr()

```
size_t tfm_plat_builtin_key_get_desc_table_ptr (
            const tfm_plat_builtin_key_descriptor_t * desc_ptr[] )
```

Definition at line 316 of file crypto_keys_se_rt.c.

## 7.134 platform/ext/target/infineon/common/spe/services/crypto/crypto_↩ nv_seed.c File Reference

```
#include "cy_device.h"
#include "tfm_plat_crypto_nv_seed.h"
#include <string.h>
#include "static_checks.h"
```

Include dependency graph for crypto_nv_seed.c:

## Functions

- int tfm_plat_crypto_provision_entropy_seed (void)
- int tfm_plat_crypto_nv_seed_read (unsigned char ∗buf, size_t buf_len)
- int tfm_plat_crypto_nv_seed_write (const unsigned char ∗buf, size_t buf_len)

## 7.134.1 Function Documentation

### 7.134.1.1 tfm_plat_crypto_nv_seed_read()

```
int tfm_plat_crypto_nv_seed_read (
            unsigned char * buf,
            size_t buf_len )
```

Definition at line 26 of file crypto_nv_seed.c.
Here is the call graph for this function:

### 7.134.1.2 tfm_plat_crypto_nv_seed_write()

```
int tfm_plat_crypto_nv_seed_write (
            const unsigned char * buf,
            size_t buf_len )
```

Definition at line 107 of file crypto_nv_seed.c.

#### 7.134.1.3 tfm_plat_crypto_provision_entropy_seed()

```
int tfm_plat_crypto_provision_entropy_seed (
            void  )
```

Definition at line 20 of file crypto_nv_seed.c.

## 7.135 platform/ext/target/infineon/common/spe/services/crypto/crypto_rnd_se_rt.c File Reference

```
#include "config_tfm.h"
#include "crypto_platform.h"
#include "ifx_se_psacrypto.h"
#include "ifx_utils.h"
#include "psa/crypto.h"
#include "mbedtls/platform.h"
#include <string.h>
```

Include dependency graph for crypto_rnd_se_rt.c:

## 7.136 platform/ext/target/infineon/common/spe/services/crypto/mbedtls_accel_configs/crypto_hw_cryptolite_config.h File Reference

## 7.137 platform/ext/target/infineon/common/spe/services/crypto/mbedtls_accel_configs/crypto_hw_mxcrypto_config.h File Reference

## 7.138 platform/ext/target/infineon/common/spe/services/its/drivers/its_flash_driver.c File Reference

```
#include "config_tfm.h"
#include "ifx_driver_flash.h"
#include "partition/flash_layout.h"
```

Include dependency graph for its_flash_driver.c:

## 7.139 platform/ext/target/infineon/common/spe/services/its/drivers/its_rram_driver.c File Reference

```
#include "config_tfm.h"
#include "ifx_driver_rram.h"
#include "partition/flash_layout.h"
#include "cy_device.h"
```

Include dependency graph for its_rram_driver.c:

## 7.140 platform/ext/target/infineon/common/spe/services/its/its_hal.c File Reference

```
#include "config_tfm.h"
#include "partition/flash_layout.h"
#include "static_checks.h"
#include "tfm_hal_its.h"
```

Include dependency graph for its_hal.c:

### Functions

- enum tfm_hal_status_t tfm_hal_its_fs_info (struct tfm_hal_its_fs_info_t *fs_info)

### 7.140.1 Function Documentation

#### 7.140.1.1 tfm_hal_its_fs_info()

```
enum tfm_hal_status_t tfm_hal_its_fs_info (
            struct tfm_hal_its_fs_info_t * fs_info )
```

Definition at line 15 of file its_hal.c.

# 7.141 platform/ext/target/infineon/common/spe/services/mailbox/platform_hal_multi_core_cm55.c File Reference

```
#include <stdint.h>
#include <stdio.h>
#include "config_tfm.h"
#include "cy_ipc_drv.h"
#include "fih.h"
#include "ifx_platform_mailbox.h"
#include "ifx_regions.h"
#include "tfm_hal_multi_core.h"
#include "tfm_peripherals_def.h"
```

Include dependency graph for platform_hal_multi_core_cm55.c:

## Macros

- #define IFX_IS_REGION_IN_ITCM_MEMORY(base, size)
- #define IFX_IS_REGION_IN_DTCM_MEMORY(base, size)
- #define IFX_IS_REGION_IN_ITCM_MEMORY_REMAPPED(base, size)
- #define IFX_IS_REGION_IN_DTCM_MEMORY_REMAPPED(base, size)

## 7.141.1 Macro Definition Documentation

### 7.141.1.1 IFX_IS_REGION_IN_DTCM_MEMORY

```
#define IFX_IS_REGION_IN_DTCM_MEMORY(
            base,
            size )
```

**Value:**

```
        (ifx_is_region_inside_other(base, base + size, \
                                    CY_CM55_DTCM_INTERNAL_NS_SBUS_BASE, \
                                    (CY_CM55_DTCM_INTERNAL_NS_SBUS_BASE + CY_CM55_DTCM_INTERNAL_SIZE))
    == true)
```

Definition at line 30 of file platform_hal_multi_core_cm55.c.

### 7.141.1.2 IFX_IS_REGION_IN_DTCM_MEMORY_REMAPPED

```
#define IFX_IS_REGION_IN_DTCM_MEMORY_REMAPPED(
            base,
            size )
```

**Value:**

```
        (ifx_is_region_inside_other(base, base + size, \
                                    CY_CM55_DTCM_NS_SBUS_BASE, \
                                    (CY_CM55_DTCM_NS_SBUS_BASE + CY_CM55_DTCM_INTERNAL_SIZE)) == true)
```

Definition at line 40 of file platform_hal_multi_core_cm55.c.

#### 7.141.1.3 IFX_IS_REGION_IN_ITCM_MEMORY

```
#define IFX_IS_REGION_IN_ITCM_MEMORY(
            base,
            size )
```

**Value:**

```
        (ifx_is_region_inside_other(base, base + size, \
                                    CY_CM55_ITCM_INTERNAL_NS_SBUS_BASE, \
                                    (CY_CM55_ITCM_INTERNAL_NS_SBUS_BASE + CY_CM55_ITCM_INTERNAL_SIZE))
    == true)
```

Definition at line 25 of file platform_hal_multi_core_cm55.c.

#### 7.141.1.4 IFX_IS_REGION_IN_ITCM_MEMORY_REMAPPED

```
#define IFX_IS_REGION_IN_ITCM_MEMORY_REMAPPED(
            base,
            size )
```

**Value:**

```
        (ifx_is_region_inside_other(base, base + size, \
                                    CY_CM55_ITCM_NS_SBUS_BASE, \
                                    (CY_CM55_ITCM_NS_SBUS_BASE + CY_CM55_ITCM_INTERNAL_SIZE)) == true)
```

Definition at line 35 of file platform_hal_multi_core_cm55.c.

## 7.142 platform/ext/target/infineon/common/spe/services/mailbox/platform_spe_mailbox.c File Reference

```
#include "config_tfm.h"
#include "cmsis.h"
#include "cmsis_compiler.h"
#include "cy_device.h"
#include "cy_ipc_drv.h"
#include "cy_sysint.h"
#include "current.h"
#include "fih.h"
#include "ifx_utils.h"
#include "tfm_hal_isolation.h"
#include "tfm_hal_multi_core.h"
#include "tfm_peripherals_def.h"
#include "tfm_spe_mailbox.h"
#include "platform_multicore.h"
#include "utilities.h"
#include "static_checks.h"
```

Include dependency graph for platform_spe_mailbox.c:

### Macros

- #define MAILBOX_CLEAN_CACHE(addr, size) { __DSB(); }
- #define MAILBOX_INVALIDATE_CACHE(addr, size) do {} while (0)

## Functions

- int32_t tfm_mailbox_hal_notify_peer (void)
- int32_t tfm_mailbox_hal_init (struct secure_mailbox_queue_t *s_queue)
- int32_t tfm_mailbox_hal_deinit (struct secure_mailbox_queue_t *s_queue)
- uint32_t tfm_mailbox_hal_enter_critical (void)
- void tfm_mailbox_hal_exit_critical (uint32_t state)

## 7.142.1 Macro Definition Documentation

### 7.142.1.1 MAILBOX_CLEAN_CACHE

```
#define MAILBOX_CLEAN_CACHE(
            addr,
            size ) { __DSB(); }
```

Definition at line 32 of file platform_spe_mailbox.c.

### 7.142.1.2 MAILBOX_INVALIDATE_CACHE

```
#define MAILBOX_INVALIDATE_CACHE(
            addr,
            size ) do {} while (0)
```

Definition at line 33 of file platform_spe_mailbox.c.

## 7.142.2 Function Documentation

### 7.142.2.1 tfm_mailbox_hal_deinit()

```
int32_t tfm_mailbox_hal_deinit (
            struct secure_mailbox_queue_t * s_queue )
```

Definition at line 135 of file platform_spe_mailbox.c.

### 7.142.2.2 tfm_mailbox_hal_enter_critical()

```
uint32_t tfm_mailbox_hal_enter_critical (
            void )
```

Definition at line 146 of file platform_spe_mailbox.c.

### 7.142.2.3 tfm_mailbox_hal_exit_critical()

```
void tfm_mailbox_hal_exit_critical (
            uint32_t state )
```

Definition at line 171 of file platform_spe_mailbox.c.

### 7.142.2.4 tfm_mailbox_hal_init()

```
int32_t tfm_mailbox_hal_init (
            struct secure_mailbox_queue_t * s_queue )
```

Definition at line 67 of file platform_spe_mailbox.c.

Here is the call graph for this function:

#### 7.142.2.5 tfm_mailbox_hal_notify_peer()

```
int32_t tfm_mailbox_hal_notify_peer (
            void  )
```

Definition at line 52 of file platform_spe_mailbox.c.

## 7.143 platform/ext/target/infineon/common/spe/services/mailbox/tfm_hal_multi_core.c File Reference

```
#include "config_tfm.h"
#include "cmsis.h"
#include "cy_ipc_drv.h"
#include "ifx_regions.h"
#include "platform_multicore.h"
#include "tfm_hal_multi_core.h"
#include "tfm_peripherals_def.h"
#include "tfm_sp_log.h"
```

Include dependency graph for tfm_hal_multi_core.c:

### Functions

- void tfm_hal_wait_for_ns_cpu_ready (void)

### 7.143.1 Function Documentation

#### 7.143.1.1 tfm_hal_wait_for_ns_cpu_ready()

```
void tfm_hal_wait_for_ns_cpu_ready (
            void  )
```

Definition at line 19 of file tfm_hal_multi_core.c.

## 7.144 platform/ext/target/infineon/common/spe/services/mailbox/tfm_interrupts.c File Reference

```
#include <stdint.h>
#include "config_tfm.h"
#include "cmsis.h"
#include "cy_ipc_drv.h"
#include "ifx_interrupt_defs.h"
#include "interrupt.h"
#include "platform_multicore.h"
#include "spm.h"
#include "tfm_multi_core.h"
#include "tfm_peripherals_def.h"
#include "load/interrupt_defs.h"
#include "static_checks.h"
```

Include dependency graph for tfm_interrupts.c:

### Functions

- void IFX_IRQ_NAME_TO_HANDLER() IFX_IPC_NS_TO_TFM_IPC_INTR (void)
- enum tfm_hal_status_t mailbox_irq_init (void *p_pt, const struct irq_load_info_t *p_ildi)

### 7.144.1 Function Documentation

#### 7.144.1.1 IFX_IPC_NS_TO_TFM_IPC_INTR()

```
void IFX_IRQ_NAME_TO_HANDLER() IFX_IPC_NS_TO_TFM_IPC_INTR (
            void  )
```

Definition at line 41 of file tfm_interrupts.c.

#### 7.144.1.2 mailbox_irq_init()

```
enum tfm_hal_status_t mailbox_irq_init (
            void * p_pt,
            const struct irq_load_info_t * p_ildi )
```

Definition at line 54 of file tfm_interrupts.c.

## 7.145 platform/ext/target/infineon/common/spe/services/platform/drivers/nv↩ _counters_flash_driver.c File Reference

```
#include "nv_counters_flash_driver.h"
#include "region_defs.h"
#include "static_checks.h"
```

Include dependency graph for nv_counters_flash_driver.c:

### Macros

- #define IFX_TFM_NV_COUNTERS_ERASE_VALUE IFX_DRIVER_FLASH_ERASE_VALUE

### 7.145.1 Macro Definition Documentation

#### 7.145.1.1 IFX_TFM_NV_COUNTERS_ERASE_VALUE

```
#define IFX_TFM_NV_COUNTERS_ERASE_VALUE IFX_DRIVER_FLASH_ERASE_VALUE
```

Definition at line 13 of file nv_counters_flash_driver.c.

## 7.146 platform/ext/target/infineon/common/spe/services/platform/drivers/nv↩ _counters_flash_driver.h File Reference

```
#include "ifx_driver_flash.h"
```

Include dependency graph for nv_counters_flash_driver.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_TFM_NV_COUNTERS_PROGRAM_UNIT IFX_DRIVER_FLASH_PROGRAM_UNIT
- #define IFX_TFM_NV_COUNTERS_SECTOR_SIZE IFX_DRIVER_FLASH_SECTOR_SIZE
- #define IFX_NV_COUNTERS_CMSIS_FLASH_INSTANCE (ifx_nv_counters_cmsis_flash_instance)

## Variables

- ARM_DRIVER_FLASH ifx_nv_counters_cmsis_flash_instance

### 7.146.1 Macro Definition Documentation

#### 7.146.1.1 IFX_NV_COUNTERS_CMSIS_FLASH_INSTANCE

```
#define IFX_NV_COUNTERS_CMSIS_FLASH_INSTANCE (ifx_nv_counters_cmsis_flash_instance)
```

Definition at line 23 of file nv_counters_flash_driver.h.

#### 7.146.1.2 IFX_TFM_NV_COUNTERS_PROGRAM_UNIT

```
#define IFX_TFM_NV_COUNTERS_PROGRAM_UNIT IFX_DRIVER_FLASH_PROGRAM_UNIT
```

Definition at line 14 of file nv_counters_flash_driver.h.

#### 7.146.1.3 IFX_TFM_NV_COUNTERS_SECTOR_SIZE

```
#define IFX_TFM_NV_COUNTERS_SECTOR_SIZE IFX_DRIVER_FLASH_SECTOR_SIZE
```

Definition at line 16 of file nv_counters_flash_driver.h.

### 7.146.2 Variable Documentation

#### 7.146.2.1 ifx_nv_counters_cmsis_flash_instance

```
ARM_DRIVER_FLASH ifx_nv_counters_cmsis_flash_instance
```

## 7.147 platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_rram_driver.c File Reference

```
#include "nv_counters_rram_driver.h"
#include "region_defs.h"
```

Include dependency graph for nv_counters_rram_driver.c:

### Macros

- #define IFX_TFM_NV_COUNTERS_ERASE_VALUE IFX_DRIVER_RRAM_ERASE_VALUE

### 7.147.1 Macro Definition Documentation

#### 7.147.1.1 IFX_TFM_NV_COUNTERS_ERASE_VALUE

```
#define IFX_TFM_NV_COUNTERS_ERASE_VALUE IFX_DRIVER_RRAM_ERASE_VALUE
```

Definition at line 12 of file nv_counters_rram_driver.c.

## 7.148 platform/ext/target/infineon/common/spe/services/platform/drivers/nv_counters_rram_driver.h File Reference

```
#include "ifx_driver_rram.h"
```

Include dependency graph for nv_counters_rram_driver.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_TFM_NV_COUNTERS_PROGRAM_UNIT IFX_DRIVER_RRAM_PROGRAM_UNIT
- #define IFX_TFM_NV_COUNTERS_SECTOR_SIZE CY_RRAM_BLOCK_SIZE_BYTES
- #define IFX_NV_COUNTERS_CMSIS_FLASH_INSTANCE (ifx_nv_counters_cmsis_flash_instance)

### Variables

- ARM_DRIVER_FLASH ifx_nv_counters_cmsis_flash_instance

### 7.148.1 Macro Definition Documentation

#### 7.148.1.1 IFX_NV_COUNTERS_CMSIS_FLASH_INSTANCE

```
#define IFX_NV_COUNTERS_CMSIS_FLASH_INSTANCE (ifx_nv_counters_cmsis_flash_instance)
```

Definition at line 18 of file nv_counters_rram_driver.h.

#### 7.148.1.2 IFX_TFM_NV_COUNTERS_PROGRAM_UNIT

```
#define IFX_TFM_NV_COUNTERS_PROGRAM_UNIT IFX_DRIVER_RRAM_PROGRAM_UNIT
```

Definition at line 14 of file nv_counters_rram_driver.h.

#### 7.148.1.3 IFX_TFM_NV_COUNTERS_SECTOR_SIZE

```
#define IFX_TFM_NV_COUNTERS_SECTOR_SIZE CY_RRAM_BLOCK_SIZE_BYTES
```

Definition at line 15 of file nv_counters_rram_driver.h.

### 7.148.2 Variable Documentation

#### 7.148.2.1 ifx_nv_counters_cmsis_flash_instance

```
ARM_DRIVER_FLASH ifx_nv_counters_cmsis_flash_instance
```

## 7.149 platform/ext/target/infineon/common/spe/services/platform/nv_counters_flash.c File Reference

```
#include "ifx_se_crc32.h"
#include "ifx_utils.h"
#include "nv_counters_flash_driver.h"
#include "static_checks.h"
#include "tfm_plat_nv_counters.h"
#include <limits.h>
#include <stddef.h>
```

Include dependency graph for nv_counters_flash.c:

### Data Structures

- struct ifx_nv_counters

  *Struct representing the NV counter data in flash.*

## Macros

- #define IFX_NV_COUNTER_SIZE sizeof(uint32_t)
- #define IFX_INIT_VALUE_SIZE sizeof(uint32_t)
- #define IFX_CHECKSUM_SIZE sizeof(uint32_t)
- #define IFX_NUM_NV_COUNTERS ((IFX_TFM_NV_COUNTERS_SECTOR_SIZE - IFX_INIT_VALUE_SIZE - IFX_CHECKSUM_SIZE) / IFX_NV_COUNTER_SIZE)
- #define IFX_NV_COUNTERS_INITIALIZED 0xC0DE0042U
- #define IFX_TFM_NV_COUNTERS_AREA_OFFSET (0U)
- #define IFX_TFM_NV_COUNTERS_BACKUP_OFFSET (IFX_TFM_NV_COUNTERS_AREA_OFFSET + IFX_TFM_NV_COUNTERS_SECTOR_SIZE)
- #define IFX_NV_COUNTERS_CRC_INIT (0)

## Typedefs

- typedef struct ifx_nv_counters ifx_nv_counters_t

  *Struct representing the NV counter data in flash.*

## Functions

- enum tfm_plat_err_t tfm_plat_init_nv_counter (void)
- enum tfm_plat_err_t tfm_plat_read_nv_counter (enum tfm_nv_counter_t counter_id, uint32_t size, uint8_t *val)
- enum tfm_plat_err_t tfm_plat_set_nv_counter (enum tfm_nv_counter_t counter_id, uint32_t value)
- enum tfm_plat_err_t tfm_plat_increment_nv_counter (enum tfm_nv_counter_t counter_id)

## 7.149.1 Macro Definition Documentation

### 7.149.1.1 IFX_CHECKSUM_SIZE

```
#define IFX_CHECKSUM_SIZE sizeof(uint32_t)
```

Definition at line 30 of file nv_counters_flash.c.

### 7.149.1.2 IFX_INIT_VALUE_SIZE

```
#define IFX_INIT_VALUE_SIZE sizeof(uint32_t)
```

Definition at line 29 of file nv_counters_flash.c.

### 7.149.1.3 IFX_NUM_NV_COUNTERS

```
#define IFX_NUM_NV_COUNTERS ((IFX_TFM_NV_COUNTERS_SECTOR_SIZE - IFX_INIT_VALUE_SIZE - IFX_CHECKSUM_SIZE)
/ IFX_NV_COUNTER_SIZE)
```

Definition at line 31 of file nv_counters_flash.c.

### 7.149.1.4 IFX_NV_COUNTER_SIZE

```
#define IFX_NV_COUNTER_SIZE sizeof(uint32_t)
```

Definition at line 28 of file nv_counters_flash.c.

#### 7.149.1.5 IFX_NV_COUNTERS_CRC_INIT

```
#define IFX_NV_COUNTERS_CRC_INIT (0)
```

Definition at line 38 of file nv_counters_flash.c.

#### 7.149.1.6 IFX_NV_COUNTERS_INITIALIZED

```
#define IFX_NV_COUNTERS_INITIALIZED 0xC0DE0042U
```

Definition at line 33 of file nv_counters_flash.c.

#### 7.149.1.7 IFX_TFM_NV_COUNTERS_AREA_OFFSET

```
#define IFX_TFM_NV_COUNTERS_AREA_OFFSET (0U)
```

Definition at line 35 of file nv_counters_flash.c.

#### 7.149.1.8 IFX_TFM_NV_COUNTERS_BACKUP_OFFSET

```
#define IFX_TFM_NV_COUNTERS_BACKUP_OFFSET (IFX_TFM_NV_COUNTERS_AREA_OFFSET + IFX_TFM_NV_COUNTERS_SECTOR_SIZE)
```

Definition at line 36 of file nv_counters_flash.c.

### 7.149.2 Typedef Documentation

#### 7.149.2.1 ifx_nv_counters_t

```
typedef struct ifx_nv_counters ifx_nv_counters_t
```

Struct representing the NV counter data in flash.

### 7.149.3 Function Documentation

#### 7.149.3.1 tfm_plat_increment_nv_counter()

```
enum tfm_plat_err_t tfm_plat_increment_nv_counter (
            enum tfm_nv_counter_t counter_id )
```

Definition at line 231 of file nv_counters_flash.c.

Here is the call graph for this function:

#### 7.149.3.2 tfm_plat_init_nv_counter()

```
enum tfm_plat_err_t tfm_plat_init_nv_counter (
            void  )
```

Definition at line 81 of file nv_counters_flash.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.149.3.3 tfm_plat_read_nv_counter()

```
enum tfm_plat_err_t tfm_plat_read_nv_counter (
            enum tfm_nv_counter_t counter_id,
            uint32_t size,
            uint8_t * val )
```

Definition at line 146 of file nv_counters_flash.c.
Here is the caller graph for this function:

#### 7.149.3.4 tfm_plat_set_nv_counter()

```
enum tfm_plat_err_t tfm_plat_set_nv_counter (
            enum tfm_nv_counter_t counter_id,
            uint32_t value )
```

Definition at line 170 of file nv_counters_flash.c.

Here is the caller graph for this function:

## 7.150 platform/ext/target/infineon/common/spe/services/platform/nv_counters_rram.c File Reference

```
#include "ifx_flash_driver_api.h"
#include "ifx_se_crc32.h"
#include "ifx_utils.h"
#include "nv_counters_rram_driver.h"
#include "project_memory_layout.h"
#include "tfm_plat_nv_counters.h"
#include "static_checks.h"
```

Include dependency graph for nv_counters_rram.c:

### Data Structures

- struct ifx_rram_counter_t

### Macros

- #define IFX_TFM_NV_COUNTERS_BASE (0UL) /∗ Address of NV counters block, it is equal 0 as relative are used ∗/
- #define IFX_TFM_NV_COUNTERS_COUNTER_AREA_OFFSET (0UL) /∗ Relative offset of counters area in NV counters area ∗/
- #define IFX_TFM_NV_COUNTERS_BACKUP_AREA_OFFSET (IFX_TFM_NV_COUNTERS_COUNTER_AREA_OFFSET + ((PLAT_NV_COUNTER_MAX) ∗ sizeof(ifx_rram_counter_t)))
- #define IFX_TFM_NV_COUNTERS_BACKUP_AREA_SIZE (PLAT_NV_COUNTER_MAX ∗ sizeof(ifx_rram_counter_t))
- #define IFX_NVM_INIT_DONE_FLAG (0xAA5533CCUL)
- #define IFX_TFM_NV_COUNTERS_NVM_INIT_DONE_FLAG_OFFSET
- #define IFX_TFM_NV_COUNTERS_EXPECTED_AREA_SIZE

### Typedefs

- typedef uint32_t ifx_rram_counter_value_t

## Functions

- enum tfm_plat_err_t ifx_rram_clear_counters_and_backups (void)
- enum tfm_plat_err_t tfm_plat_init_nv_counter (void)
- enum tfm_plat_err_t tfm_plat_read_nv_counter (enum tfm_nv_counter_t counter_id, uint32_t size, uint8_t *val)
- enum tfm_plat_err_t tfm_plat_set_nv_counter (enum tfm_nv_counter_t counter_id, uint32_t value)
- enum tfm_plat_err_t tfm_plat_increment_nv_counter (enum tfm_nv_counter_t counter_id)

## 7.150.1 Macro Definition Documentation

### 7.150.1.1 IFX_NVM_INIT_DONE_FLAG

```
#define IFX_NVM_INIT_DONE_FLAG (0xAA5533CCUL)
```

Definition at line 39 of file nv_counters_rram.c.

### 7.150.1.2 IFX_TFM_NV_COUNTERS_BACKUP_AREA_OFFSET

```
#define IFX_TFM_NV_COUNTERS_BACKUP_AREA_OFFSET (IFX_TFM_NV_COUNTERS_COUNTER_AREA_OFFSET +
((PLAT_NV_COUNTER_MAX) * sizeof(ifx_rram_counter_t)))
```

Definition at line 36 of file nv_counters_rram.c.

### 7.150.1.3 IFX_TFM_NV_COUNTERS_BACKUP_AREA_SIZE

```
#define IFX_TFM_NV_COUNTERS_BACKUP_AREA_SIZE (PLAT_NV_COUNTER_MAX * sizeof(ifx_rram_counter_t))
```

Definition at line 37 of file nv_counters_rram.c.

### 7.150.1.4 IFX_TFM_NV_COUNTERS_BASE

```
#define IFX_TFM_NV_COUNTERS_BASE (0UL) /* Address of NV counters block, it is equal 0 as relative
are used */
```

Definition at line 33 of file nv_counters_rram.c.

### 7.150.1.5 IFX_TFM_NV_COUNTERS_COUNTER_AREA_OFFSET

```
#define IFX_TFM_NV_COUNTERS_COUNTER_AREA_OFFSET (0UL) /* Relative offset of counters area in
NV counters area */
```

Definition at line 34 of file nv_counters_rram.c.

### 7.150.1.6 IFX_TFM_NV_COUNTERS_EXPECTED_AREA_SIZE

```
#define IFX_TFM_NV_COUNTERS_EXPECTED_AREA_SIZE
```

**Value:**

```
(IFX_ALIGN_UP_TO((sizeof(ifx_rram_counter_t)
* (2U * ((uint32_t)PLAT_NV_COUNTER_MAX))), \
IFX_TFM_NV_COUNTERS_SECTOR_SIZE) +
IFX_TFM_NV_COUNTERS_SECTOR_SIZE)
```

Definition at line 51 of file nv_counters_rram.c.

#### 7.150.1.7 IFX_TFM_NV_COUNTERS_NVM_INIT_DONE_FLAG_OFFSET

```
#define IFX_TFM_NV_COUNTERS_NVM_INIT_DONE_FLAG_OFFSET
```

**Value:**

```
(IFX_ALIGN_UP_TO((IFX_TFM_NV_COUNTERS_BACKUP_AREA_OFFSET
    + \
IFX_TFM_NV_COUNTERS_BACKUP_AREA_SIZE),
    IFX_TFM_NV_COUNTERS_SECTOR_SIZE))
```

Definition at line 42 of file nv_counters_rram.c.

### 7.150.2 Typedef Documentation

#### 7.150.2.1 ifx_rram_counter_value_t

```
typedef uint32_t ifx_rram_counter_value_t
```

Definition at line 21 of file nv_counters_rram.c.

### 7.150.3 Function Documentation

#### 7.150.3.1 ifx_rram_clear_counters_and_backups()

```
enum tfm_plat_err_t ifx_rram_clear_counters_and_backups (
            void  )
```

Definition at line 298 of file nv_counters_rram.c.

#### 7.150.3.2 tfm_plat_increment_nv_counter()

```
enum tfm_plat_err_t tfm_plat_increment_nv_counter (
            enum tfm_nv_counter_t counter_id )
```

Definition at line 418 of file nv_counters_rram.c.
Here is the call graph for this function:

#### 7.150.3.3 tfm_plat_init_nv_counter()

```
enum tfm_plat_err_t tfm_plat_init_nv_counter (
            void  )
```

Definition at line 320 of file nv_counters_rram.c.

Here is the call graph for this function:

#### 7.150.3.4 tfm_plat_read_nv_counter()

```
enum tfm_plat_err_t tfm_plat_read_nv_counter (
            enum tfm_nv_counter_t counter_id,
            uint32_t size,
            uint8_t * val )
```

Definition at line 366 of file nv_counters_rram.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.150.3.5 tfm_plat_set_nv_counter()

```
enum tfm_plat_err_t tfm_plat_set_nv_counter (
            enum tfm_nv_counter_t counter_id,
            uint32_t value )
```

Definition at line 378 of file nv_counters_rram.c.

Here is the caller graph for this function:

## 7.151 platform/ext/target/infineon/common/spe/services/platform/nv_counters_se_rt.c File Reference

```
#include "config_tfm.h"
#include "tfm_plat_nv_counters.h"
#include "ifx_se_fih.h"
#include "ifx_se_platform.h"
#include "ifx_se_tfm_utils.h"
#include <string.h>
```

Include dependency graph for nv_counters_se_rt.c:

### Macros

- #define IFX_PLAT_NV_COUNTER_PS_0 IFX_SE_RRAM_ROLLBACK_COUNTER_NUM_MIN

  *RRAM counters 0..2 are used for Protected Storage.*
- #define IFX_PLAT_NV_COUNTER_NS_0

  *RRAM counters 3..5 are used for NS.*
- #define IFX_PLAT_RRAM_COUNTER_NUMBER

  *Number of RRAM counters use by Platform Service.*

### Functions

- enum tfm_plat_err_t tfm_plat_init_nv_counter (void)
- enum tfm_plat_err_t tfm_plat_read_nv_counter (enum tfm_nv_counter_t counter_id, uint32_t size, uint8_t *val)
- enum tfm_plat_err_t tfm_plat_increment_nv_counter (enum tfm_nv_counter_t counter_id)

### 7.151.1 Macro Definition Documentation

#### 7.151.1.1 IFX_PLAT_NV_COUNTER_NS_0

```
#define IFX_PLAT_NV_COUNTER_NS_0
```

**Value:**

```
(IFX_PLAT_NV_COUNTER_PS_0 + \
((uint32_t)PLAT_NV_COUNTER_PS_2 - (uint32_t)PLAT_NV_COUNTER_PS_0 +
    1U))
```

RRAM counters 3..5 are used for NS.
Definition at line 21 of file nv_counters_se_rt.c.

#### 7.151.1.2 IFX_PLAT_NV_COUNTER_PS_0

```
#define IFX_PLAT_NV_COUNTER_PS_0 IFX_SE_RRAM_ROLLBACK_COUNTER_NUM_MIN
```

RRAM counters 0..2 are used for Protected Storage.
Definition at line 18 of file nv_counters_se_rt.c.

#### 7.151.1.3 IFX_PLAT_RRAM_COUNTER_NUMBER

```
#define IFX_PLAT_RRAM_COUNTER_NUMBER
```

**Value:**

```
((PLAT_NV_COUNTER_PS_2 - PLAT_NV_COUNTER_PS_0 + 1U) + \
(PLAT_NV_COUNTER_NS_2 - PLAT_NV_COUNTER_NS_0 + 1U))
```

Number of RRAM counters use by Platform Service.
Definition at line 25 of file nv_counters_se_rt.c.

### 7.151.2 Function Documentation

#### 7.151.2.1 tfm_plat_increment_nv_counter()

```
enum tfm_plat_err_t tfm_plat_increment_nv_counter (
            enum tfm_nv_counter_t counter_id )
```

Definition at line 98 of file nv_counters_se_rt.c.

#### 7.151.2.2 tfm_plat_init_nv_counter()

```
enum tfm_plat_err_t tfm_plat_init_nv_counter (
            void  )
```

Definition at line 61 of file nv_counters_se_rt.c.

#### 7.151.2.3 tfm_plat_read_nv_counter()

```
enum tfm_plat_err_t tfm_plat_read_nv_counter (
            enum tfm_nv_counter_t counter_id,
            uint32_t size,
            uint8_t * val )
```

Definition at line 66 of file nv_counters_se_rt.c.

## 7.152 platform/ext/target/infineon/common/spe/services/platform/tfm_platform_system.c File Reference

```
#include "cmsis.h"
#include "cy_pdl.h"
#include "ifx_platform_api.h"
#include "ifx_platform_private.h"
```

```
#include "platform_svc_api.h"
#include "psa/service.h"
#include "tfm_platform_system.h"
```

Include dependency graph for tfm_platform_system.c:

## Functions

- void tfm_platform_hal_system_reset (void)
- psa_status_t ifx_mtb_srf_handler (const psa_msg_t *msg)
- psa_status_t tfm_platform_hal_additional_services (const psa_msg_t *msg)
- enum tfm_platform_err_t tfm_platform_hal_ioctl (tfm_platform_ioctl_req_t request, psa_invec *in_vec, psa_outvec *out_vec)

## 7.152.1 Function Documentation

### 7.152.1.1 ifx_mtb_srf_handler()

```
psa_status_t ifx_mtb_srf_handler (
            const psa_msg_t * msg )
```

Definition at line 58 of file tfm_platform_system.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.152.1.2 tfm_platform_hal_additional_services()

```
psa_status_t tfm_platform_hal_additional_services (
            const psa_msg_t * msg )
```

Definition at line 78 of file tfm_platform_system.c.
Here is the call graph for this function:

#### 7.152.1.3 tfm_platform_hal_ioctl()

```
enum tfm_platform_err_t tfm_platform_hal_ioctl (
            tfm_platform_ioctl_req_t request,
            psa_invec * in_vec,
            psa_outvec * out_vec )
```

Definition at line 87 of file tfm_platform_system.c.
Here is the call graph for this function:

#### 7.152.1.4 tfm_platform_hal_system_reset()

```
void tfm_platform_hal_system_reset (
            void  )
```

Definition at line 25 of file tfm_platform_system.c.

Here is the call graph for this function:

Here is the caller graph for this function:

## 7.153 platform/ext/target/infineon/common/spe/services/ps/drivers/ps_↩ rram_driver.c File Reference

```
#include "config_tfm.h"
#include "region_defs.h"
#include "ifx_driver_rram.h"
#include "tfm_peripherals_def.h"
```

Include dependency graph for ps_rram_driver.c:

## 7.154 platform/ext/target/infineon/common/spe/services/ps/drivers/ps_↩ smif_driver.c File Reference

```
#include "config_tfm.h"
#include "region_defs.h"
```

Include dependency graph for ps_smif_driver.c:

## 7.155 platform/ext/target/infineon/common/spe/services/ps/ps_hal.c File Reference

```
#include "config_tfm.h"
#include "static_checks.h"
#include "tfm_hal_ps.h"
```

Include dependency graph for ps_hal.c:

### Functions

- enum tfm_hal_status_t tfm_hal_ps_fs_info (struct tfm_hal_ps_fs_info_t *fs_info)

### 7.155.1 Function Documentation

#### 7.155.1.1 tfm_hal_ps_fs_info()

```
enum tfm_hal_status_t tfm_hal_ps_fs_info (
            struct tfm_hal_ps_fs_info_t * fs_info )
```

Definition at line 14 of file ps_hal.c.

## 7.156 platform/ext/target/infineon/common/spe/services/ps/ps_keys_se_rt.c File Reference

```
#include "config_tfm.h"
#include "ifx_utils.h"
#include "tfm_crypto_defs.h"
#include "tfm_plat_ps_crypto.h"
```

Include dependency graph for ps_keys_se_rt.c:

### Functions

- psa_status_t tfm_platform_ps_set_key (psa_key_id_t *ps_key, const uint8_t *key_label, size_t key_label_len)

### 7.156.1 Function Documentation

#### 7.156.1.1 tfm_platform_ps_set_key()

```
psa_status_t tfm_platform_ps_set_key (
            psa_key_id_t * ps_key,
            const uint8_t * key_label,
            size_t key_label_len )
```

Definition at line 16 of file ps_keys_se_rt.c.

Here is the call graph for this function:

## 7.157 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_ipc.c File Reference

```
#include "ifx_se_ipc_service_syscall.h"
#include "cy_syslib.h"
```

Include dependency graph for ifx_se_ipc_service_ipc.c:

### Functions

- ifx_se_status_t ifx_se_ipc_service_spm_syscall (ifx_se_fih_ptr_t ipc_packet, ifx_se_fih_t ipc_packet_size, void *ctx)

  *Handle SPM's syscall to SE IPC Service.*

### 7.157.1 Function Documentation

#### 7.157.1.1 ifx_se_ipc_service_spm_syscall()

```
ifx_se_status_t ifx_se_ipc_service_spm_syscall (
            ifx_se_fih_ptr_t ipc_packet,
            ifx_se_fih_t ipc_packet_size,
            void * ctx )
```

Handle SPM's syscall to SE IPC Service.

**Returns**

Returns values as specified by the ifx_se_status_t

Definition at line 13 of file ifx_se_ipc_service_ipc.c.
Here is the caller graph for this function:

## 7.158 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_req_mngr.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include "ifx_se_syscall.h"
#include "static_checks.h"
#include "psa/service.h"
#include "psa_manifest/pid.h"
#include "psa_manifest/ifx_se_ipc_service.h"
```

Include dependency graph for ifx_se_ipc_service_req_mngr.c:

## Functions

- psa_status_t ifx_se_ipc_service_sfn (const psa_msg_t *msg)
- psa_status_t ifx_se_ipc_service_entry (void)

### 7.158.1 Function Documentation

#### 7.158.1.1 ifx_se_ipc_service_entry()

```
psa_status_t ifx_se_ipc_service_entry (
            void  )
```

Definition at line 112 of file ifx_se_ipc_service_req_mngr.c.

#### 7.158.1.2 ifx_se_ipc_service_sfn()

```
psa_status_t ifx_se_ipc_service_sfn (
            const psa_msg_t * msg )
```

Definition at line 20 of file ifx_se_ipc_service_req_mngr.c.
Here is the call graph for this function:

## 7.159 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_spm.c File Reference

```
#include "ifx_se_platform.h"
#include "ifx_se_tfm_utils.h"
#include "tfm_spm_log.h"
#include "ifx_se_ipc_service_spm.h"
```

Include dependency graph for ifx_se_ipc_service_spm.c:

## Functions

- enum tfm_plat_err_t ifx_se_ipc_service_spm_init (void)

  *Lets SPM to use SE IPC channel.*

- enum tfm_plat_err_t ifx_se_ipc_service_spm_shutdown (void)

  *Shutdown SE IPC Service.*

### 7.159.1 Function Documentation

#### 7.159.1.1 ifx_se_ipc_service_spm_init()

```
enum tfm_plat_err_t ifx_se_ipc_service_spm_init (
            void  )
```

Lets SPM to use SE IPC channel.

**Note**

SE IPC Service interface (calls to se-rt-services-utils) should be used during initialization phase only before calling ifx_se_ipc_service_spm_shutdown. It's not possible to use SE IPC interface within SPM and secure/non-secure partition at the same time. So it's important to shutdown this interface by calling ifx_se_ipc_service_spm_shutdown before BACKEND_SPM_INIT.

**Returns**

Returns values as specified by the tfm_plat_err_t

Definition at line 14 of file ifx_se_ipc_service_spm.c.

#### 7.159.1.2 ifx_se_ipc_service_spm_shutdown()

```
enum tfm_plat_err_t ifx_se_ipc_service_spm_shutdown (
            void  )
```

Shutdown SE IPC Service.

**Returns**

Returns values as specified by the tfm_plat_err_t

Definition at line 29 of file ifx_se_ipc_service_spm.c.

## 7.160 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_spm.h File Reference

```
#include "tfm_plat_defs.h"
```

Include dependency graph for ifx_se_ipc_service_spm.h:

This graph shows which files directly or indirectly include this file:

## Functions

- enum tfm_plat_err_t ifx_se_ipc_service_spm_init (void)

  *Lets SPM to use SE IPC channel.*
- enum tfm_plat_err_t ifx_se_ipc_service_spm_shutdown (void)

  *Shutdown SE IPC Service.*

## 7.160.1 Function Documentation

### 7.160.1.1 ifx_se_ipc_service_spm_init()

```
enum tfm_plat_err_t ifx_se_ipc_service_spm_init (
            void  )
```

Lets SPM to use SE IPC channel.

**Note**

SE IPC Service interface (calls to se-rt-services-utils) should be used during initialization phase only before calling ifx_se_ipc_service_spm_shutdown. It's not possible to use SE IPC interface within SPM and secure/non-secure partition at the same time. So it's important to shutdown this interface by calling ifx_se_ipc_service_spm_shutdown before BACKEND_SPM_INIT.

**Returns**

Returns values as specified by the tfm_plat_err_t

Definition at line 14 of file ifx_se_ipc_service_spm.c.

#### 7.160.1.2 ifx_se_ipc_service_spm_shutdown()

```
enum tfm_plat_err_t ifx_se_ipc_service_spm_shutdown (
            void  )
```

Shutdown SE IPC Service.

**Returns**

Returns values as specified by the tfm_plat_err_t

Definition at line 29 of file ifx_se_ipc_service_spm.c.

## 7.161 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall.h File Reference

```
#include "ifx_se_syscall.h"
```

Include dependency graph for ifx_se_ipc_service_syscall.h:

This graph shows which files directly or indirectly include this file:

## Functions

- ifx_se_status_t ifx_se_ipc_service_spm_syscall (ifx_se_fih_ptr_t ipc_packet, ifx_se_fih_t ipc_packet_size, void *ctx)

  *Handle SPM's syscall to SE IPC Service.*

## 7.161.1 Function Documentation

### 7.161.1.1 ifx_se_ipc_service_spm_syscall()

```
ifx_se_status_t ifx_se_ipc_service_spm_syscall (
            ifx_se_fih_ptr_t ipc_packet,
            ifx_se_fih_t ipc_packet_size,
            void * ctx )
```

Handle SPM's syscall to SE IPC Service.

**Returns**

Returns values as specified by the ifx_se_status_t

Definition at line 13 of file ifx_se_ipc_service_ipc.c.
Here is the caller graph for this function:

## 7.162 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall_direct.c File Reference

```
#include "ifx_se_fih.h"
#include "ifx_se_syscall.h"
```

Include dependency graph for ifx_se_ipc_service_syscall_direct.c:

### Functions

- ifx_se_status_t ifx_se_syscall (ifx_se_fih_ptr_t ipc_packet, ifx_se_fih_t ipc_packet_size, void *ctx)

### 7.162.1 Function Documentation

#### 7.162.1.1 ifx_se_syscall()

```
ifx_se_status_t ifx_se_syscall (
            ifx_se_fih_ptr_t ipc_packet,
            ifx_se_fih_t ipc_packet_size,
            void * ctx )
```

Definition at line 29 of file ifx_se_ipc_service_syscall_direct.c.

## 7.163 platform/ext/target/infineon/common/spe/services/se_ipc_service/ifx_se_ipc_service_syscall_partition.c File Reference

```
#include "ifx_se_syscall.h"
#include "psa/client.h"
#include "psa_manifest/sid.h"
#include "static_checks.h"
#include "ifx_se_ipc_service_syscall.h"
#include "protection_shared_data.h"
```

Include dependency graph for ifx_se_ipc_service_syscall_partition.c:

## Functions

- void tfm_core_panic (void)
- ifx_se_status_t ifx_se_syscall (ifx_se_fih_ptr_t ipc_packet, ifx_se_fih_t ipc_packet_size, void *ctx)

## 7.163.1 Function Documentation

### 7.163.1.1 ifx_se_syscall()

```
ifx_se_status_t ifx_se_syscall (
            ifx_se_fih_ptr_t ipc_packet,
            ifx_se_fih_t ipc_packet_size,
            void * ctx )
```

Definition at line 38 of file ifx_se_ipc_service_syscall_partition.c.
Here is the call graph for this function:

### 7.163.1.2 tfm_core_panic()

```
void tfm_core_panic (
            void  )
```

Definition at line 18 of file utilities.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.164 platform/ext/target/infineon/common/spe/services/tfm_tests/plat_test.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "config_tfm.h"
#include "cy_tcpwm_counter.h"
```

```
#include "device_definition.h"
#include "tfm_peripherals_def.h"
#include "tfm_plat_defs.h"
#include "tfm_plat_test.h"
#include "plat_test.h"
```

Include dependency graph for plat_test.c:

## 7.165 platform/ext/target/infineon/common/spe/services/tfm_tests/plat↩ _test.h File Reference

```
#include "device_definition.h"
#include "tfm_hal_defs.h"
```

Include dependency graph for plat_test.h:

This graph shows which files directly or indirectly include this file:

## 7.166 platform/ext/target/infineon/common/spe/services/tfm_tests/plat↩ _test_non_secure_timer.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "config_tfm.h"
#include "cy_tcpwm_counter.h"
#include "tfm_plat_defs.h"
#include "tfm_plat_test.h"
#include "tfm_peripherals_def.h"
#include "plat_test.h"
```

Include dependency graph for plat_test_non_secure_timer.c:

## 7.167 platform/ext/target/infineon/common/spe/services/tfm_tests/plat↩ _test_secure_timer.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "config_tfm.h"
#include "cy_tcpwm_counter.h"
#include "tfm_plat_defs.h"
#include "tfm_plat_test.h"
#include "tfm_peripherals_def.h"
#include "plat_test.h"
#include "platform_svc_api.h"
```

Include dependency graph for plat_test_secure_timer.c:

## Functions

- void tfm_plat_test_secure_timer_start (void)
- void tfm_plat_test_secure_timer_clear_intr (void)
- void tfm_plat_test_secure_timer_stop (void)

## 7.167.1 Function Documentation

### 7.167.1.1 tfm_plat_test_secure_timer_clear_intr()

```
void tfm_plat_test_secure_timer_clear_intr (
            void  )
```

Definition at line 51 of file plat_test_secure_timer.c.

### 7.167.1.2 tfm_plat_test_secure_timer_start()

```
void tfm_plat_test_secure_timer_start (
            void  )
```

Definition at line 32 of file plat_test_secure_timer.c.
Here is the call graph for this function:

### 7.167.1.3 tfm_plat_test_secure_timer_stop()

```
void tfm_plat_test_secure_timer_stop (
            void  )
```

Definition at line 59 of file plat_test_secure_timer.c.

Here is the call graph for this function:

## 7.168 platform/ext/target/infineon/common/spe/services/tfm_tests/tfm_↩ interrupts_test_s.c File Reference

```
#include <stdint.h>
#include "config_tfm.h"
#include "cmsis.h"
#include "device_definition.h"
#include "interrupt.h"
#include "tfm_peripherals_def.h"
#include "load/interrupt_defs.h"
#include "ifx_interrupt_defs.h"
#include "static_checks.h"
```

Include dependency graph for tfm_interrupts_test_s.c:

## 7.169 platform/ext/target/infineon/common/spe/svc/platform_svc_api.c File Reference

```
#include <stdint.h>
#include "config_tfm.h"
#include "compiler_ext_defs.h"
#include "utilities.h"
#include "config_impl.h"
#include "platform_svc_api.h"
```

Include dependency graph for platform_svc_api.c:

## Functions

- __naked int32_t ifx_call_platform_uart_log (const uint8_t *str, uint32_t len)

  *Write string to SPM UART log.*
- __naked int32_t ifx_call_platform_enable_systick (uint32_t enable)

  *Enable/disable SysTick for FLIH/SLIH tf-m-tests.*
- __naked void ifx_call_platform_system_reset (void)

  *Resets the system.*

## 7.169.1 Function Documentation

### 7.169.1.1 ifx_call_platform_enable_systick()

```
__naked int32_t ifx_call_platform_enable_systick (
            uint32_t enable )
```

Enable/disable SysTick for FLIH/SLIH tf-m-tests.

**Parameters**

| | | |
|---|---|---|
| in | *enable* | 0 - disable SysTick IRQ, != 0 - enable IRQ |

**Return values**

| | |
|---|---|
| *Platform* | error code, see tfm_platform_err_t |

Definition at line 23 of file platform_svc_api.c.

Here is the caller graph for this function:

#### 7.169.1.2 ifx_call_platform_system_reset()

```
__naked void ifx_call_platform_system_reset (
            void  )
```

Resets the system.
Requests a system reset to reset the MCU.
Definition at line 34 of file platform_svc_api.c.
Here is the caller graph for this function:

#### 7.169.1.3 ifx_call_platform_uart_log()

```
__naked int32_t ifx_call_platform_uart_log (
            const uint8_t * str,
            uint32_t len )
```

Write string to SPM UART log.

**Parameters**

| | | |
|---|---|---|
| in | *str* | String to output |
| in | *len* | String size |

**Return values**

| | |
|---|---|
| *Platform* | error code, see tfm_platform_err_t |

Definition at line 17 of file platform_svc_api.c.

Here is the caller graph for this function:

## 7.170 platform/ext/target/infineon/common/spe/svc/platform_svc_api.h File Reference

`#include "svc_num.h"`
Include dependency graph for platform_svc_api.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_SVC_PLATFORM_UART_LOG TFM_SVC_NUM_PLATFORM_THREAD(0x0U)
- #define IFX_SVC_PLATFORM_ENABLE_SYSTICK TFM_SVC_NUM_PLATFORM_THREAD(0x1U)
- #define IFX_SVC_PLATFORM_SYSTEM_RESET TFM_SVC_NUM_PLATFORM_THREAD(0x2U)

## Functions

- int32_t ifx_call_platform_enable_systick (uint32_t enable)

  *Enable/disable SysTick for FLIH/SLIH tf-m-tests.*
- int32_t ifx_call_platform_uart_log (const uint8_t *str, uint32_t len)

  *Write string to SPM UART log.*
- void ifx_call_platform_system_reset (void)

  *Resets the system.*

## 7.170.1 Macro Definition Documentation

### 7.170.1.1 IFX_SVC_PLATFORM_ENABLE_SYSTICK

```
#define IFX_SVC_PLATFORM_ENABLE_SYSTICK TFM_SVC_NUM_PLATFORM_THREAD(0x1U)
```

Definition at line 18 of file platform_svc_api.h.

### 7.170.1.2 IFX_SVC_PLATFORM_SYSTEM_RESET

```
#define IFX_SVC_PLATFORM_SYSTEM_RESET TFM_SVC_NUM_PLATFORM_THREAD(0x2U)
```

Definition at line 21 of file platform_svc_api.h.

### 7.170.1.3 IFX_SVC_PLATFORM_UART_LOG

```
#define IFX_SVC_PLATFORM_UART_LOG TFM_SVC_NUM_PLATFORM_THREAD(0x0U)
```

Definition at line 15 of file platform_svc_api.h.

## 7.170.2 Function Documentation

### 7.170.2.1 ifx_call_platform_enable_systick()

```
int32_t ifx_call_platform_enable_systick (
            uint32_t enable )
```

Enable/disable SysTick for FLIH/SLIH tf-m-tests.

**Parameters**

| | | |
|---|---|---|
| in | *enable* | 0 - disable SysTick IRQ, != 0 - enable IRQ |

**Return values**

| | |
|---|---|
| *Platform* | error code, see tfm_platform_err_t |

Definition at line 23 of file platform_svc_api.c.

Here is the caller graph for this function:

#### 7.170.2.2 ifx_call_platform_system_reset()

```
void ifx_call_platform_system_reset (
            void  )
```

Resets the system.
Requests a system reset to reset the MCU.
Definition at line 34 of file platform_svc_api.c.
Here is the caller graph for this function:

#### 7.170.2.3 ifx_call_platform_uart_log()

```
int32_t ifx_call_platform_uart_log (
            const uint8_t * str,
            uint32_t len )
```

Write string to SPM UART log.

**Parameters**

| | | |
|---|---|---|
| in | *str* | String to output |
| in | *len* | String size |

**Return values**

| | |
|---|---|
| *Platform* | error code, see tfm_platform_err_t |

Definition at line 17 of file platform_svc_api.c.

Here is the caller graph for this function:

## 7.171 platform/ext/target/infineon/common/spe/svc/platform_svc_handler.c File Reference

```
#include "config_tfm.h"
#include "cmsis.h"
#include "cy_device_headers.h"
#include "platform_svc_api.h"
#include "spm.h"
#include "static_checks.h"
#include "target_cfg.h"
#include "tfm_hal_isolation.h"
#include "tfm_peripherals_def.h"
```

Include dependency graph for platform_svc_handler.c:

### Functions

- int32_t platform_svc_handlers (uint8_t svc_num, uint32_t *svc_args, uint32_t lr)

### 7.171.1 Function Documentation

#### 7.171.1.1 platform_svc_handlers()

```
int32_t platform_svc_handlers (
            uint8_t svc_num,
            uint32_t * svc_args,
            uint32_t lr )
```

Definition at line 87 of file platform_svc_handler.c.

## 7.172 platform/ext/target/infineon/common/spe/test/ifx_fpu_s.c File Reference

```
#include "ifx_interrupt_defs.h"
#include "tfm_peripherals_def.h"
#include "test_interrupt.h"
```

Include dependency graph for ifx_fpu_s.c:

### Functions

- void IFX_IRQ_NAME_TO_HANDLER() TFM_FPU_S_TEST_IRQ (void)
- void tfm_test_s_fpu_init (void)

  *Enabling FPU secure interrupts for tests.*

### 7.172.1 Function Documentation

#### 7.172.1.1 TFM_FPU_S_TEST_IRQ()

```
void IFX_IRQ_NAME_TO_HANDLER() TFM_FPU_S_TEST_IRQ (
            void  )
```

Definition at line 16 of file ifx_fpu_s.c.

Here is the caller graph for this function:

#### 7.172.1.2 tfm_test_s_fpu_init()

```
void tfm_test_s_fpu_init (
            void  )
```

Enabling FPU secure interrupts for tests.
Definition at line 21 of file ifx_fpu_s.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.173 platform/ext/target/infineon/common/spe/test/ifx_fpu_s.h File Reference

### Functions

- void tfm_test_s_fpu_init (void)

  *Enabling FPU secure interrupts for tests.*

### 7.173.1 Function Documentation

#### 7.173.1.1 tfm_test_s_fpu_init()

```
void tfm_test_s_fpu_init (
            void  )
```

Enabling FPU secure interrupts for tests.
Definition at line 21 of file ifx_fpu_s.c.

Here is the call graph for this function:

Here is the caller graph for this function:

## 7.174 platform/ext/target/infineon/common/spe/v80m/target_cfg.c File Reference

```
#include "config_tfm.h"
#include "target_cfg.h"
#include "cmsis.h"
#include "protection_regions_cfg.h"
```

Include dependency graph for target_cfg.c:

### Macros

- #define SCB_AIRCR_WRITE_MASK ((0x5FAUL << SCB_AIRCR_VECTKEY_Pos))

### Functions

- FIH_RET_TYPE (enum tfm_plat_err_t) ifx_system_reset_cfg(void)

  *Configures fault handlers and sets priorities.*
- enum tfm_plat_err_t ifx_init_debug (void)

  *Configures the system debug properties.*
- enum tfm_plat_err_t ifx_init_system_control_block (void)

  *Configures Sleep and Exception Handling (System Control Block).*

## 7.174.1 Macro Definition Documentation

### 7.174.1.1 SCB_AIRCR_WRITE_MASK

```
#define SCB_AIRCR_WRITE_MASK ((0x5FAUL << SCB_AIRCR_VECTKEY_Pos))
```

Definition at line 17 of file target_cfg.c.

## 7.174.2 Function Documentation

### 7.174.2.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_plat_err_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 19 of file target_cfg.c.
Here is the call graph for this function:

#### 7.174.2.2 ifx_init_debug()

```
enum tfm_plat_err_t ifx_init_debug (
            void  )
```

Configures the system debug properties.

**Returns**

Returns values as specified by the tfm_plat_err_t

Definition at line 43 of file target_cfg.c.
Here is the caller graph for this function:

#### 7.174.2.3 ifx_init_system_control_block()

```
enum tfm_plat_err_t ifx_init_system_control_block (
            void  )
```

Configures Sleep and Exception Handling (System Control Block).

**Returns**

Returns values as specified by the tfm_plat_err_t

Definition at line 98 of file target_cfg.c.
Here is the caller graph for this function:

## 7.175 platform/ext/target/infineon/common/spe/v80m/target_cfg.h File Reference

```
#include "fih.h"
#include "tfm_plat_defs.h"
```

Include dependency graph for target_cfg.h:

This graph shows which files directly or indirectly include this file:

## Functions

- FIH_RET_TYPE (enum tfm_plat_err_t) ifx_system_reset_cfg(void)

  *Configures the system reset request properties.*
- enum tfm_plat_err_t ifx_init_debug (void)

  *Configures the system debug properties.*
- enum tfm_plat_err_t ifx_init_system_control_block (void)

  *Configures Sleep and Exception Handling (System Control Block).*

## 7.175.1 Function Documentation

### 7.175.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures the system reset request properties.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩ M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩ NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩ NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_MEM_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_INPUT - Invalid inputs.

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |

**Parameters**

| | | |
|---|---|---|
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HAL_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 67 of file faults.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.175.1.2 ifx_init_debug()

```
enum tfm_plat_err_t ifx_init_debug (
            void  )
```

Configures the system debug properties.

**Returns**

Returns values as specified by the tfm_plat_err_t

Definition at line 43 of file target_cfg.c.
Here is the caller graph for this function:

#### 7.175.1.3 ifx_init_system_control_block()

```
enum tfm_plat_err_t ifx_init_system_control_block (
            void  )
```

Configures Sleep and Exception Handling (System Control Block).

**Returns**

Returns values as specified by the tfm_plat_err_t

Definition at line 98 of file target_cfg.c.
Here is the caller graph for this function:

## 7.176 platform/ext/target/infineon/common/spe/v80m/tfm_hal_platform.c File Reference

```
#include "config_tfm.h"
#include "faults.h"
#include "fih.h"
#include "ifx_boot_shared_data.h"
#include "tfm_hal_platform.h"
#include "region_defs.h"
#include "cmsis.h"
#include "target_cfg.h"
#include "uart_stdout.h"
#include "static_checks.h"
```

Include dependency graph for tfm_hal_platform.c:

### Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) tfm_hal_platform_init(void)

  *Configures fault handlers and sets priorities.*

### 7.176.1 Function Documentation

#### 7.176.1.1 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t )
```

Configures fault handlers and sets priorities.
Configures the system reset request properties.
Configures the SAU.
Initialize Protection Context switching.
Configures MSC.
Populate the internal static MPC configuration array.
This function enables platform specific faults interrupts.
This function enables the faults interrupts (plus the isolation boundary violation interrupts)

**Returns**

Returns values as specified by the tfm_plat_err_t

Configures fault handlers and sets priorities.
Configures all external interrupts to target the NS state, apart for the ones associated to secure peripherals (plus MPC and PPC)
Apply a configuration to assets of a partition.
Apply a configuration to specified assets of a partition.
Apply PPC protection to named MMIO asset.
Isolate memory region (numbered MMIO) by MPU.
Check memory access permissions using MPC attribute cache.
Apply MPC configuration using raw interface.
Apply MPC configuration.
Check access to memory region by MPC.
This function enables platform specific faults interrupts.

**Returns**

Returns values as specified by the tfm_plat_err_t

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

**Parameters**

| | | |
|---|---|---|
| in | *mpc_reg_cfg* | MPC configuration structure which will be applied. See ifx_mpc_raw_region_config_t for more details. |

**Returns**

TFM_HAL_SUCCESS - MPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - configuration can not be applied. Can occur when MPC that should protect the memory is not controlled by TFM. TFM_HAL_ERROR_GENERIC - failed to apply MPC configuration. TFM_HAL_ERROR_INVALID_I↩NPUT - invalid input (e.g. provided memory region is not fully located in any known memory).

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ATE - Failed to isolate numbered MMIO by MPU

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩L_ERROR_GENERIC - Failed to apply PPC configuration.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |
| in | *p_assets* | Array of partition's assets |
| in | *asset_cnt* | Number of items in p_assets |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t |
| in | *cfg* | Pointer to configuration ifx_protection_region_config_t |

**Returns**

TFM_HAL_SUCCESS - config successfully updated TFM_HAL_ERROR_GENERIC - failed to apply config TFM_HAL_ERROR_INVALID_INPUT - invalid parameter

Configures fault handlers and sets priorities.
Check access to memory region by MPC.

**Returns**

TFM_HAL_SUCCESS - static MPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static MPC

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t. |
| in | *base* | The base address of the region. |
| in | *size* | The size of the region. |
| in | *access_type* | The memory access types to be checked between given memory and boundaries. |

**Returns**

TFM_HAL_SUCCESS - The memory region has the access permissions by MPC. TFM_HAL_ERROR_ME↩M_FAULT - The memory region has not the access permissions by MPC. TFM_HAL_ERROR_INVALID_I↩NPUT - Invalid inputs.

Configures fault handlers and sets priorities.
Check memory access permissions using MPC attribute cache.
This function sets up the RRAM MPC attribute cache used for memory protection checks.

**Returns**

TFM_HAL_SUCCESS on success, error code otherwise.

This function validates access permissions for a memory region using the MPC cache.

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Partition information structure. |
| in | *memory_config* | Memory configuration structure. |
| in | *base* | Base address of the memory region. |
| in | *size* | Size of the memory region. |
| in | *access_type* | Requested access type (read/write/secure/non-secure). |

**Returns**

TFM_HAL_SUCCESS - Access is permitted. TFM_HAL_ERROR_MEM_FAULT - Access is not permitted or invalid input.

Configures fault handlers and sets priorities.
Isolate memory region (numbered MMIO) by MPU.

**Returns**

TFM_HAL_SUCCESS - Static MPU initialized successfully TFM_HAL_ERROR_INVALID_INPUT - Failed to init static MPU

**Parameters**

| | | |
|---|---|---|
| in | *p_info* | Pointer to partition info ifx_partition_info_t that is owner of the p_asset |
| in | *p_asset* | Asset that must be protected by MPU |

**Returns**

TFM_HAL_SUCCESS - Numbered MMIO was isolated successfully by MPU TFM_HAL_ERROR_BAD_ST↩ ATE - Failed to isolate numbered MMIO by MPU

Configures fault handlers and sets priorities.
Apply PPC protection to named MMIO asset.

**Returns**

TFM_HAL_SUCCESS - static PPC initialized successfully TFM_HAL_ERROR_GENERIC - failed to init static PPC

**Parameters**

| | | |
|---|---|---|
| in | *ppc_cfg* | PPC configuration settings that should be used. See ifx_ppc_named_mmio_config_t structure for details. |
| in | *asset* | Named MMIO asset that should be protected. |

**Returns**

TFM_HAL_SUCCESS - PPC configuration applied successfully. TFM_HAL_ERROR_NOT_SUPPORTED - Asset attributes are not supported. TFM_HAL_ERROR_INVALID_INPUT - Invalid named asset. TFM_HA↩ L_ERROR_GENERIC - Failed to apply PPC configuration.

Definition at line 23 of file tfm_hal_platform.c.

Here is the call graph for this function:

## 7.177 platform/ext/target/infineon/common/utils/ifx_boot_shared_data.c File Reference

```
#include "ifx_boot_shared_data.h"
#include "region_defs.h"
#include "tfm_boot_status.h"
#include "tfm_spm_log.h"
#include "static_checks.h"
#include <string.h>
```

Include dependency graph for ifx_boot_shared_data.c:

## 7.178 platform/ext/target/infineon/common/utils/ifx_boot_shared_data.h File Reference

This graph shows which files directly or indirectly include this file:

## 7.179 platform/ext/target/infineon/common/utils/ifx_fih.h File Reference

```
#include "aapcs_local.h"
#include "fih.h"
```

Include dependency graph for ifx_fih.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_FIH_TRUE true
- #define IFX_FIH_FALSE false
- #define IFX_FIH_BOOL bool
- #define ifx_fih_to_aapcs_fih(x) (x)

## Typedefs

- typedef uint32_t ifx_aapcs_fih_int

## 7.179.1 Macro Definition Documentation

### 7.179.1.1 IFX_FIH_BOOL

```
#define IFX_FIH_BOOL bool
```

Definition at line 40 of file ifx_fih.h.

### 7.179.1.2 IFX_FIH_FALSE

```
#define IFX_FIH_FALSE false
```

Definition at line 39 of file ifx_fih.h.

#### 7.179.1.3 ifx_fih_to_aapcs_fih

```
#define ifx_fih_to_aapcs_fih(
            x ) (x)
```

Definition at line 46 of file ifx_fih.h.

#### 7.179.1.4 IFX_FIH_TRUE

```
#define IFX_FIH_TRUE true
```

Definition at line 38 of file ifx_fih.h.

### 7.179.2 Typedef Documentation

#### 7.179.2.1 ifx_aapcs_fih_int

```
typedef uint32_t ifx_aapcs_fih_int
```

Type used to return uint32_t to assembler code via registers for AAPCS ABI
Definition at line 43 of file ifx_fih.h.

## 7.180 platform/ext/target/infineon/common/utils/ifx_interrupt_defs.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_CONCAT(x, y) x##y
- #define IFX_IRQ_NAME_TO_HANDLER(irq) IFX_CONCAT(irq, _Handler)

### 7.180.1 Macro Definition Documentation

#### 7.180.1.1 IFX_CONCAT

```
#define IFX_CONCAT(
            x,
            y ) x##y
```

Definition at line 16 of file ifx_interrupt_defs.h.

#### 7.180.1.2 IFX_IRQ_NAME_TO_HANDLER

```
#define IFX_IRQ_NAME_TO_HANDLER(
            irq ) IFX_CONCAT(irq, _Handler)
```

Definition at line 17 of file ifx_interrupt_defs.h.

## 7.181 platform/ext/target/infineon/common/utils/ifx_regions.c File Reference

`#include "ifx_regions.h"`
Include dependency graph for ifx_regions.c:

### Functions

- bool ifx_is_region_inside_other (uintptr_t first_region_start, uintptr_t first_region_limit, uintptr_t second_region_start, uintptr_t second_region_limit)

  *Check whether a first memory region is inside second memory region.*
- bool ifx_is_region_overlap_other (uintptr_t first_region_start, uintptr_t first_region_limit, uintptr_t second_region_start, uintptr_t second_region_limit)

  *Check whether a memory region overlaps other memory region.*

### 7.181.1 Function Documentation

#### 7.181.1.1 ifx_is_region_inside_other()

```
bool ifx_is_region_inside_other (
            uintptr_t first_region_start,
            uintptr_t first_region_limit,
            uintptr_t second_region_start,
            uintptr_t second_region_limit )
```

Check whether a first memory region is inside second memory region.

**Parameters**

| | | |
|---|---|---|
| in | *first_region_start* | The start address of the first region |
| in | *first_region_limit* | The end address of the first region |
| in | *second_region_start* | The start address of the second region, which should contain the first region |
| in | *second_region_limit* | The end address of the second region, which should contain the first region |

**Returns**

true if the second region contains the first region, false otherwise.

Definition at line 11 of file ifx_regions.c.
Here is the caller graph for this function:

#### 7.181.1.2 ifx_is_region_overlap_other()

```
bool ifx_is_region_overlap_other (
            uintptr_t first_region_start,
            uintptr_t first_region_limit,
            uintptr_t second_region_start,
            uintptr_t second_region_limit )
```

Check whether a memory region overlaps other memory region.

**Parameters**

| | | |
|---|---|---|
| in | *first_region_start* | The start address of the first region |
| in | *first_region_limit* | The end address of the first region |
| in | *second_region_start* | The start address of the second region, which should overlap the first region |
| in | *second_region_limit* | The end address of the second region, which should overlap the first region |

**Returns**

true if the second region overlaps the first region, false otherwise.

Definition at line 30 of file ifx_regions.c.

## 7.182 platform/ext/target/infineon/common/utils/ifx_regions.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "region.h"
#include "static_checks.h"
```

Include dependency graph for ifx_regions.h:

This graph shows which files directly or indirectly include this file:

## Functions

- bool ifx_is_region_inside_other (uintptr_t first_region_start, uintptr_t first_region_limit, uintptr_t second_region_start, uintptr_t second_region_limit)

  *Check whether a first memory region is inside second memory region.*
- bool ifx_is_region_overlap_other (uintptr_t first_region_start, uintptr_t first_region_limit, uintptr_t second_region_start, uintptr_t second_region_limit)

  *Check whether a memory region overlaps other memory region.*
- REGION_DECLARE (Image$$, PT_RO_START, $ $Base)
- REGION_DECLARE (Image$$, PT_RO_END, $ $Base)
- REGION_DECLARE (Image$$, TFM_NS_AGENT_TZ_CODE_START, $ $RO$ $Base)
- REGION_DECLARE (Image$$, TFM_NS_AGENT_TZ_CODE_END, $ $RO$ $Limit)
- REGION_DECLARE (Image$$, TFM_UNPRIV_BASE_CODE_START, $ $RO$ $Base)
- REGION_DECLARE (Image$$, TFM_UNPRIV_BASE_CODE_END, $ $RO$ $Limit)
- REGION_DECLARE (Image$$, TFM_UNPRIV_CODE_START, $ $RO$ $Base)
- REGION_DECLARE (Image$$, TFM_UNPRIV_CODE_END, $ $RO$ $Limit)
- REGION_DECLARE (Image$$, ARM_LIB_STACK, $ $Base)

## 7.182.1 Function Documentation

### 7.182.1.1 ifx_is_region_inside_other()

```
bool ifx_is_region_inside_other (
            uintptr_t first_region_start,
            uintptr_t first_region_limit,
            uintptr_t second_region_start,
            uintptr_t second_region_limit )
```

Check whether a first memory region is inside second memory region.

**Parameters**

| | | |
|---|---|---|
| in | *first_region_start* | The start address of the first region |
| in | *first_region_limit* | The end address of the first region |
| in | *second_region_start* | The start address of the second region, which should contain the first region |
| in | *second_region_limit* | The end address of the second region, which should contain the first region |

**Returns**

true if the second region contains the first region, false otherwise.

Definition at line 11 of file ifx_regions.c.
Here is the caller graph for this function:

### 7.182.1.2 ifx_is_region_overlap_other()

```
bool ifx_is_region_overlap_other (
            uintptr_t first_region_start,
            uintptr_t first_region_limit,
            uintptr_t second_region_start,
            uintptr_t second_region_limit )
```

Check whether a memory region overlaps other memory region.

**Parameters**

| | | |
|---|---|---|
| in | *first_region_start* | The start address of the first region |
| in | *first_region_limit* | The end address of the first region |
| in | *second_region_start* | The start address of the second region, which should overlap the first region |
| in | *second_region_limit* | The end address of the second region, which should overlap the first region |

**Returns**

true if the second region overlaps the first region, false otherwise.

Definition at line 30 of file ifx_regions.c.

### 7.182.1.3 REGION_DECLARE() [1/9]

```
REGION_DECLARE (
            Image$$ ,
            ARM_LIB_STACK ,
             $ )
```

#### 7.182.1.4 REGION_DECLARE() [2/9]

```
REGION_DECLARE (
            Image$$ ,
            PT_RO_END ,
             $ )
```

#### 7.182.1.5 REGION_DECLARE() [3/9]

```
REGION_DECLARE (
            Image$$ ,
            PT_RO_START ,
             $ )
```

#### 7.182.1.6 REGION_DECLARE() [4/9]

```
REGION_DECLARE (
            Image$$ ,
            TFM_NS_AGENT_TZ_CODE_END ,
             $ $RO$ )
```

#### 7.182.1.7 REGION_DECLARE() [5/9]

```
REGION_DECLARE (
            Image$$ ,
            TFM_NS_AGENT_TZ_CODE_START ,
             $ $RO$ )
```

#### 7.182.1.8 REGION_DECLARE() [6/9]

```
REGION_DECLARE (
            Image$$ ,
            TFM_UNPRIV_BASE_CODE_END ,
             $ $RO$ )
```

#### 7.182.1.9 REGION_DECLARE() [7/9]

```
REGION_DECLARE (
            Image$$ ,
            TFM_UNPRIV_BASE_CODE_START ,
             $ $RO$ )
```

#### 7.182.1.10 REGION_DECLARE() [8/9]

```
REGION_DECLARE (
            Image$$ ,
            TFM_UNPRIV_CODE_END ,
             $ $RO$ )
```

#### 7.182.1.11 REGION_DECLARE() [9/9]

```
REGION_DECLARE (
            Image$$ ,
```

```
TFM_UNPRIV_CODE_START ,
 $ $RO$ )
```

## 7.183 platform/ext/target/infineon/common/utils/ifx_utils.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_ALIGN_UP_TO(x, align) ((((x) % (align)) == 0u) ? (x) : ((((x) / (align)) + 1u) ∗ (align)))
- #define IFX_ASSERT(condition) sizeof(char[!!(condition) ? 1 : -1])
- #define IFX_UNSIGNED(x) (x ## U)
- #define IFX_UNSIGNED_TO_VALUE(x) IFX_UNSIGNED(x)
- #define IFX_ROUND_UP_TO_MULTIPLE(number, multiple) ((((number) + (multiple) - 1U) / (multiple)) ∗ (multiple))
- #define IFX_ROUND_DOWN_TO_MULTIPLE(number, multiple) (((number) / (multiple)) ∗ (multiple))
- #define IFX_UNUSED(x) ((void)x)

### 7.183.1 Macro Definition Documentation

#### 7.183.1.1 IFX_ALIGN_UP_TO

```
#define IFX_ALIGN_UP_TO(
            x,
            align ) ((((x) % (align)) == 0u) ? (x) : ((((x) / (align)) + 1u) * (align)))
```

Definition at line 15 of file ifx_utils.h.

#### 7.183.1.2 IFX_ASSERT

```
#define IFX_ASSERT(
            condition ) sizeof(char[!!(condition) ? 1 : -1])
```

Definition at line 25 of file ifx_utils.h.

#### 7.183.1.3 IFX_ROUND_DOWN_TO_MULTIPLE

```
#define IFX_ROUND_DOWN_TO_MULTIPLE(
            number,
            multiple ) (((number) / (multiple)) * (multiple))
```

Definition at line 42 of file ifx_utils.h.

#### 7.183.1.4 IFX_ROUND_UP_TO_MULTIPLE

```
#define IFX_ROUND_UP_TO_MULTIPLE(
            number,
            multiple ) ((((number) + (multiple) - 1U) / (multiple)) * (multiple))
```

Definition at line 40 of file ifx_utils.h.

#### 7.183.1.5 IFX_UNSIGNED

```
#define IFX_UNSIGNED(
            x ) (x ## U)
```

Definition at line 33 of file ifx_utils.h.

#### 7.183.1.6 IFX_UNSIGNED_TO_VALUE

```
#define IFX_UNSIGNED_TO_VALUE(
            x ) IFX_UNSIGNED(x)
```

Definition at line 37 of file ifx_utils.h.

#### 7.183.1.7 IFX_UNUSED

```
#define IFX_UNUSED(
            x ) ((void)x)
```

Definition at line 44 of file ifx_utils.h.

## 7.184 platform/ext/target/infineon/common/utils/mxs22.h File Reference

```
#include "ifx_utils.h"
```

Include dependency graph for mxs22.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_IDAU_SECURE_ADDRESS_MASK (IFX_UNSIGNED(0x10000000))
- #define IFX_S_ADDRESS_ALIAS(x) ((x) | IFX_IDAU_SECURE_ADDRESS_MASK)
- #define IFX_NS_ADDRESS_ALIAS(x) ((x) & ~IFX_IDAU_SECURE_ADDRESS_MASK)
- #define IFX_S_ADDRESS_ALIAS_T(t, x) ((t)(IFX_S_ADDRESS_ALIAS((uint32_t)(x))))
- #define IFX_NS_ADDRESS_ALIAS_T(t, x) ((t)(IFX_NS_ADDRESS_ALIAS((uint32_t)(x))))

### 7.184.1 Macro Definition Documentation

#### 7.184.1.1 IFX_IDAU_SECURE_ADDRESS_MASK

```
#define IFX_IDAU_SECURE_ADDRESS_MASK (IFX_UNSIGNED(0x10000000))
```

Definition at line 16 of file mxs22.h.

#### 7.184.1.2 IFX_NS_ADDRESS_ALIAS

```
#define IFX_NS_ADDRESS_ALIAS(
            x ) ((x) & ~IFX_IDAU_SECURE_ADDRESS_MASK)
```

Definition at line 30 of file mxs22.h.

#### 7.184.1.3 IFX_NS_ADDRESS_ALIAS_T

```
#define IFX_NS_ADDRESS_ALIAS_T(
            t,
            x ) ((t)(IFX_NS_ADDRESS_ALIAS((uint32_t)(x))))
```

Definition at line 37 of file mxs22.h.

#### 7.184.1.4 IFX_S_ADDRESS_ALIAS

```
#define IFX_S_ADDRESS_ALIAS(
            x ) ((x) | IFX_IDAU_SECURE_ADDRESS_MASK)
```

Definition at line 27 of file mxs22.h.

#### 7.184.1.5 IFX_S_ADDRESS_ALIAS_T

```
#define IFX_S_ADDRESS_ALIAS_T(
            t,
            x ) ((t)(IFX_S_ADDRESS_ALIAS((uint32_t)(x))))
```

Definition at line 34 of file mxs22.h.

## 7.185 platform/ext/target/infineon/common/utils/se/ifx_se_crc32.c File Reference

Fast high-distance 32-bit CRC for data integrity protection.

```
#include "ifx_se_crc32.h"
```

Include dependency graph for ifx_se_crc32.c:

## Functions

- void ifx_se_crc32d6_open (uint32_t ∗P, uint32_t init)
- uint32_t ifx_se_crc32d6_close (uint32_t ∗P)
- void ifx_se_crc32d6a_update (uint32_t ∗P, uint8_t Q)
- uint32_t ifx_se_crc32d6a (size_t n, uint8_t const ∗Q, uint32_t init)
- void ifx_se_crc32d6b_update (uint32_t ∗P, uint16_t Q)
- uint32_t ifx_se_crc32d6b (size_t n, uint16_t const ∗Q, uint32_t init)

### 7.185.1 Detailed Description

Fast high-distance 32-bit CRC for data integrity protection.

**Version**

1.0

### 7.185.2 Function Documentation

#### 7.185.2.1 ifx_se_crc32d6_close()

```
uint32_t ifx_se_crc32d6_close (
            uint32_t * P )
```

Used to finalize CRC state P and return 32 bit digest.

**Parameters**

| | | |
|---|---|---|
| in | *P* | The pointer to the CRC32 state variable. |

**Returns**

32 bit digest

Definition at line 24 of file ifx_se_crc32.c.
Here is the caller graph for this function:

### 7.185.2.2 ifx_se_crc32d6_open()

```
void ifx_se_crc32d6_open (
            uint32_t * P,
            uint32_t init )
```

Used to initialize CRC state P.

**Parameters**

| | | |
|---|---|---|
| out | *P* | The pointer to the CRC32 state variable. |
| in | *init* | Initial value for CRC32 calculation |

Definition at line 19 of file ifx_se_crc32.c.
Here is the caller graph for this function:

### 7.185.2.3 ifx_se_crc32d6a()

```
uint32_t ifx_se_crc32d6a (
            size_t n,
            uint8_t const * Q,
            uint32_t init )
```

Used to calculate 32 bit digest from the message.

**Parameters**

| | | |
|---|---|---|
| in | *n* | The length of the message. |
| in | *Q* | The pointer to the message. |
| in | *init* | Initial value for CRC32 calculation. |

**Returns**

32 bit digest

Definition at line 37 of file ifx_se_crc32.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.185.2.4 ifx_se_crc32d6a_update()

```
void ifx_se_crc32d6a_update (
            uint32_t * P,
            uint8_t Q )
```

Used to update CRC state P with next message symbol Q.

**Parameters**

| | | |
|---|---|---|
| in,out | *P* | The pointer to the CRC32 state variable. |
| in | *Q* | message symbol Q |

Definition at line 29 of file ifx_se_crc32.c.
Here is the caller graph for this function:

#### 7.185.2.5 ifx_se_crc32d6b()

```
uint32_t ifx_se_crc32d6b (
            size_t n,
            uint16_t const * Q,
            uint32_t init )
```

Used to calculate 32 bit digest from the 16-bit symbol message.

**Parameters**

| | | |
|---|---|---|
| in | *n* | The length of the message. |
| in | *Q* | The pointer to the message. |
| in | *init* | Initial value for CRC32 calculation. |

**Returns**

32 bit digest

Definition at line 62 of file ifx_se_crc32.c.
Here is the call graph for this function:

#### 7.185.2.6 ifx_se_crc32d6b_update()

```
void ifx_se_crc32d6b_update (
            uint32_t * P,
            uint16_t Q )
```

Used to update CRC state P with next 16-bit message symbol Q.

**Parameters**

| | | |
|---|---|---|
| in,out | *P* | The pointer to the CRC32 state variable. |
| in | *Q* | message symbol Q (16-bit) |

Definition at line 50 of file ifx_se_crc32.c.

Here is the caller graph for this function:

## 7.186 platform/ext/target/infineon/common/utils/se/ifx_se_crc32.h File Reference

Fast high-distance 32-bit CRC for data integrity protection.

```
#include <stdint.h>
#include <stddef.h>
```

Include dependency graph for ifx_se_crc32.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_CRC32_CRC_WIDTH 32u
- #define IFX_CRC32_BYTE_WIDTH (sizeof(uint8_t) ∗ 8u)
- #define IFX_CRC32_WORD_WIDTH (sizeof(uint16_t) ∗ 8u)
- #define IFX_CRC32_CRC_SIZE (IFX_CRC32_CRC_WIDTH / 8u)

- #define IFX_CRC32_INIT (0xFFFFFFFFuL)
- #define IFX_CRC32_CALC(Q, n) ifx_se_crc32d6a(n, Q, IFX_CRC32_INIT)
- #define IFX_CRC32_CALC_APPEND(data, data_size)

## Functions

- void ifx_se_crc32d6_open (uint32_t *P, uint32_t init)
- uint32_t ifx_se_crc32d6_close (uint32_t *P)
- void ifx_se_crc32d6a_update (uint32_t *P, uint8_t Q)
- uint32_t ifx_se_crc32d6a (size_t n, uint8_t const *Q, uint32_t init)
- void ifx_se_crc32d6b_update (uint32_t *P, uint16_t Q)
- uint32_t ifx_se_crc32d6b (size_t n, uint16_t const *Q, uint32_t init)

## 7.186.1 Detailed Description

Fast high-distance 32-bit CRC for data integrity protection.

**Version**

1.0

## 7.186.2 Macro Definition Documentation

### 7.186.2.1 IFX_CRC32_BYTE_WIDTH

```
#define IFX_CRC32_BYTE_WIDTH (sizeof(uint8_t) * 8u)
```

Definition at line 40 of file ifx_se_crc32.h.

### 7.186.2.2 IFX_CRC32_CALC

```
#define IFX_CRC32_CALC(
            Q,
            n ) ifx_se_crc32d6a(n, Q, IFX_CRC32_INIT)
```

Definition at line 47 of file ifx_se_crc32.h.

### 7.186.2.3 IFX_CRC32_CALC_APPEND

```
#define IFX_CRC32_CALC_APPEND(
            data,
            data_size )
```

**Value:**

```
do {                                                                    \
    uint32_t data_crc = IFX_CRC32_CALC((const uint8_t*)(data), (data_size)); \
    memcpy((uint8_t *)(data) + (data_size), &data_crc, sizeof(data_crc));    \
} while(0)
```

Calculates and appends CRC32 value to the and of the data block

**Parameters**

| | | |
|---|---|---|
| in | *data* | The pointer to the input data block |
| in | *data_size* | The size of actual data to calculate CRC32 |

**Note**

The data buffer MUST be larger than data_size by at least IFX_CRC32_CRC_SIZE (4 bytes) to save the CRC32 value at the end of the data block

Definition at line 59 of file ifx_se_crc32.h.

#### 7.186.2.4 IFX_CRC32_CRC_SIZE

```
#define IFX_CRC32_CRC_SIZE (IFX_CRC32_CRC_WIDTH / 8u)
```

Definition at line 43 of file ifx_se_crc32.h.

#### 7.186.2.5 IFX_CRC32_CRC_WIDTH

```
#define IFX_CRC32_CRC_WIDTH 32u
```

Definition at line 39 of file ifx_se_crc32.h.

#### 7.186.2.6 IFX_CRC32_INIT

```
#define IFX_CRC32_INIT (0xFFFFFFFFuL)
```

Definition at line 45 of file ifx_se_crc32.h.

#### 7.186.2.7 IFX_CRC32_WORD_WIDTH

```
#define IFX_CRC32_WORD_WIDTH (sizeof(uint16_t) * 8u)
```

Definition at line 41 of file ifx_se_crc32.h.

### 7.186.3 Function Documentation

#### 7.186.3.1 ifx_se_crc32d6_close()

```
uint32_t ifx_se_crc32d6_close (
            uint32_t * P )
```

Used to finalize CRC state P and return 32 bit digest.

**Parameters**

| | | |
|---|---|---|
| in | *P* | The pointer to the CRC32 state variable. |

**Returns**

32 bit digest

Definition at line 24 of file ifx_se_crc32.c.
Here is the caller graph for this function:

#### 7.186.3.2 ifx_se_crc32d6_open()

```
void ifx_se_crc32d6_open (
```

```
            uint32_t * P,
            uint32_t init )
```

Used to initialize CRC state P.

**Parameters**

| | | |
|---|---|---|
| out | *P* | The pointer to the CRC32 state variable. |
| in | *init* | Initial value for CRC32 calculation |

Definition at line 19 of file ifx_se_crc32.c.
Here is the caller graph for this function:

### 7.186.3.3 ifx_se_crc32d6a()

```
uint32_t ifx_se_crc32d6a (
            size_t n,
            uint8_t const * Q,
            uint32_t init )
```

Used to calculate 32 bit digest from the message.

**Parameters**

| | | |
|---|---|---|
| in | *n* | The length of the message. |
| in | *Q* | The pointer to the message. |
| in | *init* | Initial value for CRC32 calculation. |

**Returns**

32 bit digest

Definition at line 37 of file ifx_se_crc32.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.186.3.4 ifx_se_crc32d6a_update()

```
void ifx_se_crc32d6a_update (
            uint32_t * P,
            uint8_t Q )
```

Used to update CRC state P with next message symbol Q.

**Parameters**

| | | |
|---|---|---|
| in,out | *P* | The pointer to the CRC32 state variable. |
| in | *Q* | message symbol Q |

Definition at line 29 of file ifx_se_crc32.c.
Here is the caller graph for this function:

#### 7.186.3.5 ifx_se_crc32d6b()

```
uint32_t ifx_se_crc32d6b (
            size_t n,
            uint16_t const * Q,
            uint32_t init )
```

Used to calculate 32 bit digest from the 16-bit symbol message.

**Parameters**

| | | |
|---|---|---|
| in | *n* | The length of the message. |
| in | *Q* | The pointer to the message. |
| in | *init* | Initial value for CRC32 calculation. |

**Returns**

32 bit digest

Definition at line 62 of file ifx_se_crc32.c.
Here is the call graph for this function:

#### 7.186.3.6 ifx_se_crc32d6b_update()

```
void ifx_se_crc32d6b_update (
            uint32_t * P,
            uint16_t Q )
```

Used to update CRC state P with next 16-bit message symbol Q.

**Parameters**

| | | |
|---|---|---|
| in,out | *P* | The pointer to the CRC32 state variable. |
| in | *Q* | message symbol Q (16-bit) |

Definition at line 50 of file ifx_se_crc32.c.
Here is the caller graph for this function:

## 7.187 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/device/include/project_memory_layout.h File Reference

```
#include "cymem_memory_locations.h"
#include "ifx_tfm_image_config.h"
#include "mxs22.h"
#include "cymem_CM33_0_S.h"
```

```
#include "cymem_CM33_0.h"
#include "cymem_CM55_0.h"
```

Include dependency graph for project_memory_layout.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_RRAM_SBUS_BASE IFX_UNSIGNED(0x22000000)
- #define IFX_RRAM_CBUS_BASE IFX_UNSIGNED(0x02000000)
- #define IFX_RRAM_SIZE IFX_UNSIGNED(0x0006A000)
- #define IFX_SOCMEM_RAM_SBUS_BASE IFX_UNSIGNED(0x26000000)
- #define IFX_SOCMEM_RAM_CBUS_BASE IFX_UNSIGNED(0x06000000)
- #define IFX_SOCMEM_RAM_SIZE IFX_UNSIGNED(0x00500000)
- #define IFX_XIP_PORT0_SBUS_BASE IFX_UNSIGNED(0x60000000)
- #define IFX_XIP_PORT0_CBUS_BASE IFX_UNSIGNED(0x08000000)
- #define IFX_XIP_PORT0_SIZE IFX_UNSIGNED(0x04000000)
- #define IFX_XIP_PORT1_SBUS_BASE IFX_UNSIGNED(0x64000000)
- #define IFX_XIP_PORT1_CBUS_BASE IFX_UNSIGNED(0x0C000000)
- #define IFX_XIP_PORT1_SIZE IFX_UNSIGNED(0x04000000)
- #define IFX_SRAM0_SBUS_BASE IFX_UNSIGNED(0x24000000)
- #define IFX_SRAM0_CBUS_BASE IFX_UNSIGNED(0x04000000)
- #define IFX_SRAM0_SIZE IFX_UNSIGNED(0x00080000)
- #define IFX_SRAM1_SBUS_BASE IFX_UNSIGNED(0x24080000)
- #define IFX_SRAM1_CBUS_BASE IFX_UNSIGNED(0x04080000)
- #define IFX_SRAM1_SIZE IFX_UNSIGNED(0x00080000)
- #define IFX_TFM_ITS_OFFSET CYMEM_CM33_0_S_TFM_ITS_OFFSET
- #define IFX_TFM_ITS_ADDR CYMEM_CM33_0_S_TFM_ITS_S_START
- #define IFX_TFM_ITS_SIZE CYMEM_CM33_0_S_TFM_ITS_SIZE
- #define IFX_TFM_ITS_LOCATION CYMEM_CM33_0_S_TFM_ITS_LOCATION
- #define IFX_TFM_PS_OFFSET CYMEM_CM33_0_S_TFM_PS_OFFSET
- #define IFX_TFM_PS_ADDR CYMEM_CM33_0_S_TFM_PS_S_START
- #define IFX_TFM_PS_SIZE CYMEM_CM33_0_S_TFM_PS_SIZE
- #define IFX_TFM_PS_LOCATION CYMEM_CM33_0_S_TFM_PS_LOCATION
- #define IFX_TFM_NV_COUNTERS_AREA_OFFSET CYMEM_CM33_0_S_TFM_NV_COUNTERS_OFF↩ SET
- #define IFX_TFM_NV_COUNTERS_AREA_ADDR CYMEM_CM33_0_S_TFM_NV_COUNTERS_S_START
- #define IFX_TFM_NV_COUNTERS_AREA_SIZE CYMEM_CM33_0_S_TFM_NV_COUNTERS_SIZE
- #define IFX_TFM_NV_COUNTERS_AREA_LOCATION CYMEM_CM33_0_S_TFM_NV_COUNTERS_L↩ OCATION

- #define IFX_CM33_NS_IMAGE_EXECUTE_OFFSET CYMEM_CM33_0_m33_nvm_OFFSET
- #define IFX_CM33_NS_IMAGE_EXECUTE_ADDR CYMEM_CM33_0_m33_nvm_C_START
- #define IFX_CM33_NS_IMAGE_EXECUTE_SIZE CYMEM_CM33_0_m33_nvm_SIZE
- #define IFX_CM33_NS_IMAGE_EXECUTE_LOCATION CYMEM_CM33_0_m33_nvm_LOCATION
- #define IFX_CM55_NS_IMAGE_EXECUTE_OFFSET CYMEM_CM55_0_m55_nvm_OFFSET
- #define IFX_CM55_NS_IMAGE_EXECUTE_ADDR CYMEM_CM55_0_m55_nvm_START
- #define IFX_CM55_NS_IMAGE_EXECUTE_SIZE CYMEM_CM55_0_m55_nvm_SIZE
- #define IFX_CM55_NS_IMAGE_EXECUTE_LOCATION CYMEM_CM55_0_m55_nvm_LOCATION
- #define IFX_TFM_DATA_OFFSET CYMEM_CM33_0_S_m33s_data_OFFSET
- #define IFX_TFM_DATA_ADDR CYMEM_CM33_0_S_m33s_data_S_START
- #define IFX_TFM_DATA_SIZE CYMEM_CM33_0_S_m33s_data_SIZE
- #define IFX_TFM_DATA_LOCATION CYMEM_CM33_0_S_m33s_data_LOCATION
- #define IFX_CM33_NS_DATA_OFFSET CYMEM_CM33_0_m33_data_OFFSET
- #define IFX_CM33_NS_DATA_ADDR CYMEM_CM33_0_m33_data_START
- #define IFX_CM33_NS_DATA_SIZE CYMEM_CM33_0_m33_data_SIZE
- #define IFX_CM33_NS_DATA_LOCATION CYMEM_CM33_0_m33_data_LOCATION
- #define IFX_CM55_NS_DATA_SIZE CYMEM_CM55_0_m55_data_SIZE
- #define IFX_CM55_NS_DATA_LOCATION CYMEM_CM55_0_m55_data_LOCATION
- #define IFX_CM55_NS_DATA_OFFSET CYMEM_CM55_0_m55_data_INTERNAL_OFFSET
- #define IFX_CM55_NS_DATA_ADDR CYMEM_CM55_0_m55_data_INTERNAL_START
- #define IFX_TFM_BOOT_SHARED_DATA_OFFSET CYMEM_CM33_0_S_m33s_shared_OFFSET
- #define IFX_TFM_BOOT_SHARED_DATA_ADDR CYMEM_CM33_0_S_m33s_shared_S_START
- #define IFX_TFM_BOOT_SHARED_DATA_SIZE CYMEM_CM33_0_S_m33s_shared_SIZE
- #define IFX_TFM_BOOT_SHARED_DATA_LOCATION CYMEM_CM33_0_S_m33s_shared_LOCATION
- #define IFX_CM33_NS_SHARED_MEMORY_OFFSET CYMEM_CM33_0_m33_allocatable_shared_OFF↩
SET
- #define IFX_CM33_NS_SHARED_MEMORY_ADDR CYMEM_CM33_0_m33_allocatable_shared_START
- #define IFX_CM33_NS_SHARED_MEMORY_SIZE CYMEM_CM33_0_m33_allocatable_shared_SIZE
- #define IFX_CM33_NS_SHARED_MEMORY_LOCATION CYMEM_CM33_0_m33_allocatable_shared_L↩
OCATION
- #define IFX_CM55_NS_SHARED_MEMORY_OFFSET CYMEM_CM55_0_m55_allocatable_shared_OFF↩
SET
- #define IFX_CM55_NS_SHARED_MEMORY_ADDR CYMEM_CM55_0_m55_allocatable_shared_START
- #define IFX_CM55_NS_SHARED_MEMORY_SIZE CYMEM_CM55_0_m55_allocatable_shared_SIZE
- #define IFX_CM55_NS_SHARED_MEMORY_LOCATION CYMEM_CM55_0_m55_allocatable_shared_L↩
OCATION
- #define IFX_FF_TEST_SERVER_PARTITION_MMIO_OFFSET CYMEM_CM33_0_S_FF_TEST_SERVE↩
R_PARTITION_MMIO_OFFSET
- #define IFX_FF_TEST_SERVER_PARTITION_MMIO_ADDR CYMEM_CM33_0_S_FF_TEST_SERVER_↩
PARTITION_MMIO_S_START
- #define IFX_FF_TEST_SERVER_PARTITION_MMIO_SIZE CYMEM_CM33_0_S_FF_TEST_SERVER_↩
PARTITION_MMIO_SIZE
- #define IFX_FF_TEST_SERVER_PARTITION_MMIO_LOCATION CYMEM_CM33_0_S_FF_TEST_SER↩
VER_PARTITION_MMIO_LOCATION
- #define IFX_FF_TEST_DRIVER_PARTITION_MMIO_OFFSET CYMEM_CM33_0_S_FF_TEST_DRIVER↩
_PARTITION_MMIO_OFFSET
- #define IFX_FF_TEST_DRIVER_PARTITION_MMIO_ADDR CYMEM_CM33_0_S_FF_TEST_DRIVER_P↩
ARTITION_MMIO_S_START
- #define IFX_FF_TEST_DRIVER_PARTITION_MMIO_SIZE CYMEM_CM33_0_S_FF_TEST_DRIVER_P↩
ARTITION_MMIO_SIZE
- #define IFX_FF_TEST_DRIVER_PARTITION_MMIO_LOCATION CYMEM_CM33_0_S_FF_TEST_DRIV↩
ER_PARTITION_MMIO_LOCATION
- #define IFX_TEST_PROT_PARTITION_MEMORY_OFFSET CYMEM_CM33_0_S_TEST_PROT_PARTI↩
TION_MEMORY_OFFSET
- #define IFX_TEST_PROT_PARTITION_MEMORY_ADDR CYMEM_CM33_0_S_TEST_PROT_PARTITI↩
ON_MEMORY_S_START

- #define IFX_TEST_PROT_PARTITION_MEMORY_SIZE CYMEM_CM33_0_S_TEST_PROT_PARTITIO↩N_MEMORY_SIZE
- #define IFX_TEST_PROT_PARTITION_MEMORY_LOCATION CYMEM_CM33_0_S_TEST_PROT_PAR↩TITION_MEMORY_LOCATION
- #define IFX_TEST_AROT_PARTITION_MEMORY_OFFSET CYMEM_CM33_0_S_TEST_AROT_PARTI↩TION_MEMORY_OFFSET
- #define IFX_TEST_AROT_PARTITION_MEMORY_ADDR CYMEM_CM33_0_S_TEST_AROT_PARTITI↩ON_MEMORY_S_START
- #define IFX_TEST_AROT_PARTITION_MEMORY_SIZE CYMEM_CM33_0_S_TEST_AROT_PARTITIO↩N_MEMORY_SIZE
- #define IFX_TEST_AROT_PARTITION_MEMORY_LOCATION CYMEM_CM33_0_S_TEST_AROT_PAR↩TITION_MEMORY_LOCATION

## 7.187.1 Macro Definition Documentation

### 7.187.1.1 IFX_CM33_NS_DATA_ADDR

```
#define IFX_CM33_NS_DATA_ADDR CYMEM_CM33_0_m33_data_START
```

Definition at line 181 of file project_memory_layout.h.

### 7.187.1.2 IFX_CM33_NS_DATA_LOCATION

```
#define IFX_CM33_NS_DATA_LOCATION CYMEM_CM33_0_m33_data_LOCATION
```

Definition at line 183 of file project_memory_layout.h.

### 7.187.1.3 IFX_CM33_NS_DATA_OFFSET

```
#define IFX_CM33_NS_DATA_OFFSET CYMEM_CM33_0_m33_data_OFFSET
```

Definition at line 180 of file project_memory_layout.h.

### 7.187.1.4 IFX_CM33_NS_DATA_SIZE

```
#define IFX_CM33_NS_DATA_SIZE CYMEM_CM33_0_m33_data_SIZE
```

Definition at line 182 of file project_memory_layout.h.

### 7.187.1.5 IFX_CM33_NS_IMAGE_EXECUTE_ADDR

```
#define IFX_CM33_NS_IMAGE_EXECUTE_ADDR CYMEM_CM33_0_m33_nvm_C_START
```

Definition at line 154 of file project_memory_layout.h.

### 7.187.1.6 IFX_CM33_NS_IMAGE_EXECUTE_LOCATION

```
#define IFX_CM33_NS_IMAGE_EXECUTE_LOCATION CYMEM_CM33_0_m33_nvm_LOCATION
```

Definition at line 156 of file project_memory_layout.h.

### 7.187.1.7 IFX_CM33_NS_IMAGE_EXECUTE_OFFSET

```
#define IFX_CM33_NS_IMAGE_EXECUTE_OFFSET CYMEM_CM33_0_m33_nvm_OFFSET
```

Following macro are generated by Edge Protect solution personality:

- IFX_TFM_IMAGE_EXECUTE_OFFSET

- IFX_TFM_IMAGE_EXECUTE_ADDR
- IFX_TFM_IMAGE_EXECUTE_SIZE
- IFX_TFM_IMAGE_EXECUTE_LOCATION

Definition at line 153 of file project_memory_layout.h.

#### 7.187.1.8 IFX_CM33_NS_IMAGE_EXECUTE_SIZE

```
#define IFX_CM33_NS_IMAGE_EXECUTE_SIZE CYMEM_CM33_0_m33_nvm_SIZE
```

Definition at line 155 of file project_memory_layout.h.

#### 7.187.1.9 IFX_CM33_NS_SHARED_MEMORY_ADDR

```
#define IFX_CM33_NS_SHARED_MEMORY_ADDR CYMEM_CM33_0_m33_allocatable_shared_START
```

Definition at line 218 of file project_memory_layout.h.

#### 7.187.1.10 IFX_CM33_NS_SHARED_MEMORY_LOCATION

```
#define IFX_CM33_NS_SHARED_MEMORY_LOCATION CYMEM_CM33_0_m33_allocatable_shared_LOCATION
```

Definition at line 220 of file project_memory_layout.h.

#### 7.187.1.11 IFX_CM33_NS_SHARED_MEMORY_OFFSET

```
#define IFX_CM33_NS_SHARED_MEMORY_OFFSET CYMEM_CM33_0_m33_allocatable_shared_OFFSET
```

Definition at line 217 of file project_memory_layout.h.

#### 7.187.1.12 IFX_CM33_NS_SHARED_MEMORY_SIZE

```
#define IFX_CM33_NS_SHARED_MEMORY_SIZE CYMEM_CM33_0_m33_allocatable_shared_SIZE
```

Definition at line 219 of file project_memory_layout.h.

#### 7.187.1.13 IFX_CM55_NS_DATA_ADDR

```
#define IFX_CM55_NS_DATA_ADDR CYMEM_CM55_0_m55_data_INTERNAL_START
```

Definition at line 192 of file project_memory_layout.h.

#### 7.187.1.14 IFX_CM55_NS_DATA_LOCATION

```
#define IFX_CM55_NS_DATA_LOCATION CYMEM_CM55_0_m55_data_LOCATION
```

Definition at line 188 of file project_memory_layout.h.

#### 7.187.1.15 IFX_CM55_NS_DATA_OFFSET

```
#define IFX_CM55_NS_DATA_OFFSET CYMEM_CM55_0_m55_data_INTERNAL_OFFSET
```

Definition at line 191 of file project_memory_layout.h.

#### 7.187.1.16 IFX_CM55_NS_DATA_SIZE

```
#define IFX_CM55_NS_DATA_SIZE CYMEM_CM55_0_m55_data_SIZE
```

Definition at line 187 of file project_memory_layout.h.

#### 7.187.1.17 IFX_CM55_NS_IMAGE_EXECUTE_ADDR

```
#define IFX_CM55_NS_IMAGE_EXECUTE_ADDR CYMEM_CM55_0_m55_nvm_START
```

Definition at line 162 of file project_memory_layout.h.

#### 7.187.1.18 IFX_CM55_NS_IMAGE_EXECUTE_LOCATION

```
#define IFX_CM55_NS_IMAGE_EXECUTE_LOCATION CYMEM_CM55_0_m55_nvm_LOCATION
```

Definition at line 164 of file project_memory_layout.h.

#### 7.187.1.19 IFX_CM55_NS_IMAGE_EXECUTE_OFFSET

```
#define IFX_CM55_NS_IMAGE_EXECUTE_OFFSET CYMEM_CM55_0_m55_nvm_OFFSET
```

Definition at line 161 of file project_memory_layout.h.

#### 7.187.1.20 IFX_CM55_NS_IMAGE_EXECUTE_SIZE

```
#define IFX_CM55_NS_IMAGE_EXECUTE_SIZE CYMEM_CM55_0_m55_nvm_SIZE
```

Definition at line 163 of file project_memory_layout.h.

#### 7.187.1.21 IFX_CM55_NS_SHARED_MEMORY_ADDR

```
#define IFX_CM55_NS_SHARED_MEMORY_ADDR CYMEM_CM55_0_m55_allocatable_shared_START
```

Definition at line 224 of file project_memory_layout.h.

#### 7.187.1.22 IFX_CM55_NS_SHARED_MEMORY_LOCATION

```
#define IFX_CM55_NS_SHARED_MEMORY_LOCATION CYMEM_CM55_0_m55_allocatable_shared_LOCATION
```

Definition at line 226 of file project_memory_layout.h.

#### 7.187.1.23 IFX_CM55_NS_SHARED_MEMORY_OFFSET

```
#define IFX_CM55_NS_SHARED_MEMORY_OFFSET CYMEM_CM55_0_m55_allocatable_shared_OFFSET
```

Definition at line 223 of file project_memory_layout.h.

#### 7.187.1.24 IFX_CM55_NS_SHARED_MEMORY_SIZE

```
#define IFX_CM55_NS_SHARED_MEMORY_SIZE CYMEM_CM55_0_m55_allocatable_shared_SIZE
```

Definition at line 225 of file project_memory_layout.h.

#### 7.187.1.25 IFX_FF_TEST_DRIVER_PARTITION_MMIO_ADDR

```
#define IFX_FF_TEST_DRIVER_PARTITION_MMIO_ADDR CYMEM_CM33_0_S_FF_TEST_DRIVER_PARTITION_MMIO_↩
S_START
```

Definition at line 243 of file project_memory_layout.h.

#### 7.187.1.26 IFX_FF_TEST_DRIVER_PARTITION_MMIO_LOCATION

```
#define IFX_FF_TEST_DRIVER_PARTITION_MMIO_LOCATION CYMEM_CM33_0_S_FF_TEST_DRIVER_PARTITION_M↩
MIO_LOCATION
```

Definition at line 245 of file project_memory_layout.h.

#### 7.187.1.27 IFX_FF_TEST_DRIVER_PARTITION_MMIO_OFFSET

```
#define IFX_FF_TEST_DRIVER_PARTITION_MMIO_OFFSET CYMEM_CM33_0_S_FF_TEST_DRIVER_PARTITION_MMI↩
O_OFFSET
```

Definition at line 242 of file project_memory_layout.h.

#### 7.187.1.28 IFX_FF_TEST_DRIVER_PARTITION_MMIO_SIZE

```
#define IFX_FF_TEST_DRIVER_PARTITION_MMIO_SIZE CYMEM_CM33_0_S_FF_TEST_DRIVER_PARTITION_MMIO_↩
SIZE
```

Definition at line 244 of file project_memory_layout.h.

#### 7.187.1.29 IFX_FF_TEST_SERVER_PARTITION_MMIO_ADDR

```
#define IFX_FF_TEST_SERVER_PARTITION_MMIO_ADDR CYMEM_CM33_0_S_FF_TEST_SERVER_PARTITION_MMIO_↩
S_START
```

Definition at line 236 of file project_memory_layout.h.

#### 7.187.1.30 IFX_FF_TEST_SERVER_PARTITION_MMIO_LOCATION

```
#define IFX_FF_TEST_SERVER_PARTITION_MMIO_LOCATION CYMEM_CM33_0_S_FF_TEST_SERVER_PARTITION_M↩
MIO_LOCATION
```

Definition at line 238 of file project_memory_layout.h.

#### 7.187.1.31 IFX_FF_TEST_SERVER_PARTITION_MMIO_OFFSET

```
#define IFX_FF_TEST_SERVER_PARTITION_MMIO_OFFSET CYMEM_CM33_0_S_FF_TEST_SERVER_PARTITION_MMI↩
O_OFFSET
```

Definition at line 235 of file project_memory_layout.h.

#### 7.187.1.32 IFX_FF_TEST_SERVER_PARTITION_MMIO_SIZE

```
#define IFX_FF_TEST_SERVER_PARTITION_MMIO_SIZE CYMEM_CM33_0_S_FF_TEST_SERVER_PARTITION_MMIO_↩
SIZE
```

Definition at line 237 of file project_memory_layout.h.

#### 7.187.1.33 IFX_RRAM_CBUS_BASE

```
#define IFX_RRAM_CBUS_BASE IFX_UNSIGNED(0x02000000)
```

Definition at line 70 of file project_memory_layout.h.

#### 7.187.1.34 IFX_RRAM_SBUS_BASE

```
#define IFX_RRAM_SBUS_BASE IFX_UNSIGNED(0x22000000)
```

Definition at line 69 of file project_memory_layout.h.

#### 7.187.1.35 IFX_RRAM_SIZE

```
#define IFX_RRAM_SIZE IFX_UNSIGNED(0x0006A000)
```

Definition at line 75 of file project_memory_layout.h.

#### 7.187.1.36 IFX_SOCMEM_RAM_CBUS_BASE

```
#define IFX_SOCMEM_RAM_CBUS_BASE IFX_UNSIGNED(0x06000000)
```

Definition at line 78 of file project_memory_layout.h.

#### 7.187.1.37 IFX_SOCMEM_RAM_SBUS_BASE

```
#define IFX_SOCMEM_RAM_SBUS_BASE IFX_UNSIGNED(0x26000000)
```

Definition at line 77 of file project_memory_layout.h.

#### 7.187.1.38 IFX_SOCMEM_RAM_SIZE

```
#define IFX_SOCMEM_RAM_SIZE IFX_UNSIGNED(0x00500000)
```

Definition at line 79 of file project_memory_layout.h.

#### 7.187.1.39 IFX_SRAM0_CBUS_BASE

```
#define IFX_SRAM0_CBUS_BASE IFX_UNSIGNED(0x04000000)
```

Definition at line 90 of file project_memory_layout.h.

#### 7.187.1.40 IFX_SRAM0_SBUS_BASE

```
#define IFX_SRAM0_SBUS_BASE IFX_UNSIGNED(0x24000000)
```

Definition at line 89 of file project_memory_layout.h.

#### 7.187.1.41 IFX_SRAM0_SIZE

```
#define IFX_SRAM0_SIZE IFX_UNSIGNED(0x00080000)
```

Definition at line 91 of file project_memory_layout.h.

#### 7.187.1.42 IFX_SRAM1_CBUS_BASE

```
#define IFX_SRAM1_CBUS_BASE IFX_UNSIGNED(0x04080000)
```

Definition at line 94 of file project_memory_layout.h.

#### 7.187.1.43 IFX_SRAM1_SBUS_BASE

```
#define IFX_SRAM1_SBUS_BASE IFX_UNSIGNED(0x24080000)
```

Definition at line 93 of file project_memory_layout.h.

#### 7.187.1.44 IFX_SRAM1_SIZE

```
#define IFX_SRAM1_SIZE IFX_UNSIGNED(0x00080000)
```

Definition at line 95 of file project_memory_layout.h.

#### 7.187.1.45 IFX_TEST_AROT_PARTITION_MEMORY_ADDR

```
#define IFX_TEST_AROT_PARTITION_MEMORY_ADDR CYMEM_CM33_0_S_TEST_AROT_PARTITION_MEMORY_S_START
```

Definition at line 257 of file project_memory_layout.h.

#### 7.187.1.46 IFX_TEST_AROT_PARTITION_MEMORY_LOCATION

```
#define IFX_TEST_AROT_PARTITION_MEMORY_LOCATION CYMEM_CM33_0_S_TEST_AROT_PARTITION_MEMORY_LO↩
CATION
```

Definition at line 259 of file project_memory_layout.h.

#### 7.187.1.47 IFX_TEST_AROT_PARTITION_MEMORY_OFFSET

```
#define IFX_TEST_AROT_PARTITION_MEMORY_OFFSET CYMEM_CM33_0_S_TEST_AROT_PARTITION_MEMORY_OFFSET
```

Definition at line 256 of file project_memory_layout.h.

#### 7.187.1.48 IFX_TEST_AROT_PARTITION_MEMORY_SIZE

```
#define IFX_TEST_AROT_PARTITION_MEMORY_SIZE CYMEM_CM33_0_S_TEST_AROT_PARTITION_MEMORY_SIZE
```

Definition at line 258 of file project_memory_layout.h.

#### 7.187.1.49 IFX_TEST_PROT_PARTITION_MEMORY_ADDR

```
#define IFX_TEST_PROT_PARTITION_MEMORY_ADDR CYMEM_CM33_0_S_TEST_PROT_PARTITION_MEMORY_S_START
```

Definition at line 250 of file project_memory_layout.h.

#### 7.187.1.50 IFX_TEST_PROT_PARTITION_MEMORY_LOCATION

```
#define IFX_TEST_PROT_PARTITION_MEMORY_LOCATION CYMEM_CM33_0_S_TEST_PROT_PARTITION_MEMORY_LO↩
CATION
```

Definition at line 252 of file project_memory_layout.h.

#### 7.187.1.51 IFX_TEST_PROT_PARTITION_MEMORY_OFFSET

```
#define IFX_TEST_PROT_PARTITION_MEMORY_OFFSET CYMEM_CM33_0_S_TEST_PROT_PARTITION_MEMORY_OFFSET
```

Definition at line 249 of file project_memory_layout.h.

#### 7.187.1.52 IFX_TEST_PROT_PARTITION_MEMORY_SIZE

```
#define IFX_TEST_PROT_PARTITION_MEMORY_SIZE CYMEM_CM33_0_S_TEST_PROT_PARTITION_MEMORY_SIZE
```

Definition at line 251 of file project_memory_layout.h.

#### 7.187.1.53 IFX_TFM_BOOT_SHARED_DATA_ADDR

```
#define IFX_TFM_BOOT_SHARED_DATA_ADDR CYMEM_CM33_0_S_m33s_shared_S_START
```

Definition at line 207 of file project_memory_layout.h.

#### 7.187.1.54 IFX_TFM_BOOT_SHARED_DATA_LOCATION

```
#define IFX_TFM_BOOT_SHARED_DATA_LOCATION CYMEM_CM33_0_S_m33s_shared_LOCATION
```

Definition at line 209 of file project_memory_layout.h.

#### 7.187.1.55 IFX_TFM_BOOT_SHARED_DATA_OFFSET

```
#define IFX_TFM_BOOT_SHARED_DATA_OFFSET CYMEM_CM33_0_S_m33s_shared_OFFSET
```

Definition at line 206 of file project_memory_layout.h.

#### 7.187.1.56 IFX_TFM_BOOT_SHARED_DATA_SIZE

```
#define IFX_TFM_BOOT_SHARED_DATA_SIZE CYMEM_CM33_0_S_m33s_shared_SIZE
```

Definition at line 208 of file project_memory_layout.h.

#### 7.187.1.57 IFX_TFM_DATA_ADDR

```
#define IFX_TFM_DATA_ADDR CYMEM_CM33_0_S_m33s_data_S_START
```

Definition at line 174 of file project_memory_layout.h.

#### 7.187.1.58 IFX_TFM_DATA_LOCATION

```
#define IFX_TFM_DATA_LOCATION CYMEM_CM33_0_S_m33s_data_LOCATION
```

Definition at line 176 of file project_memory_layout.h.

#### 7.187.1.59 IFX_TFM_DATA_OFFSET

```
#define IFX_TFM_DATA_OFFSET CYMEM_CM33_0_S_m33s_data_OFFSET
```

Definition at line 173 of file project_memory_layout.h.

#### 7.187.1.60 IFX_TFM_DATA_SIZE

```
#define IFX_TFM_DATA_SIZE CYMEM_CM33_0_S_m33s_data_SIZE
```

Definition at line 175 of file project_memory_layout.h.

#### 7.187.1.61 IFX_TFM_ITS_ADDR

```
#define IFX_TFM_ITS_ADDR CYMEM_CM33_0_S_TFM_ITS_S_START
```

Definition at line 117 of file project_memory_layout.h.

#### 7.187.1.62 IFX_TFM_ITS_LOCATION

```
#define IFX_TFM_ITS_LOCATION CYMEM_CM33_0_S_TFM_ITS_LOCATION
```

Definition at line 119 of file project_memory_layout.h.

#### 7.187.1.63 IFX_TFM_ITS_OFFSET

```
#define IFX_TFM_ITS_OFFSET CYMEM_CM33_0_S_TFM_ITS_OFFSET
```

Definition at line 116 of file project_memory_layout.h.

#### 7.187.1.64 IFX_TFM_ITS_SIZE

```
#define IFX_TFM_ITS_SIZE CYMEM_CM33_0_S_TFM_ITS_SIZE
```

Definition at line 118 of file project_memory_layout.h.

#### 7.187.1.65 IFX_TFM_NV_COUNTERS_AREA_ADDR

```
#define IFX_TFM_NV_COUNTERS_AREA_ADDR CYMEM_CM33_0_S_TFM_NV_COUNTERS_S_START
```

Definition at line 131 of file project_memory_layout.h.

#### 7.187.1.66 IFX_TFM_NV_COUNTERS_AREA_LOCATION

```
#define IFX_TFM_NV_COUNTERS_AREA_LOCATION CYMEM_CM33_0_S_TFM_NV_COUNTERS_LOCATION
```

Definition at line 133 of file project_memory_layout.h.

#### 7.187.1.67 IFX_TFM_NV_COUNTERS_AREA_OFFSET

```
#define IFX_TFM_NV_COUNTERS_AREA_OFFSET CYMEM_CM33_0_S_TFM_NV_COUNTERS_OFFSET
```

Definition at line 130 of file project_memory_layout.h.

#### 7.187.1.68 IFX_TFM_NV_COUNTERS_AREA_SIZE

```
#define IFX_TFM_NV_COUNTERS_AREA_SIZE CYMEM_CM33_0_S_TFM_NV_COUNTERS_SIZE
```

Definition at line 132 of file project_memory_layout.h.

#### 7.187.1.69 IFX_TFM_PS_ADDR

```
#define IFX_TFM_PS_ADDR CYMEM_CM33_0_S_TFM_PS_S_START
```

Definition at line 124 of file project_memory_layout.h.

#### 7.187.1.70 IFX_TFM_PS_LOCATION

```
#define IFX_TFM_PS_LOCATION CYMEM_CM33_0_S_TFM_PS_LOCATION
```

Definition at line 126 of file project_memory_layout.h.

#### 7.187.1.71 IFX_TFM_PS_OFFSET

```
#define IFX_TFM_PS_OFFSET CYMEM_CM33_0_S_TFM_PS_OFFSET
```

Definition at line 123 of file project_memory_layout.h.

#### 7.187.1.72 IFX_TFM_PS_SIZE

```
#define IFX_TFM_PS_SIZE CYMEM_CM33_0_S_TFM_PS_SIZE
```

Definition at line 125 of file project_memory_layout.h.

#### 7.187.1.73 IFX_XIP_PORT0_CBUS_BASE

```
#define IFX_XIP_PORT0_CBUS_BASE IFX_UNSIGNED(0x08000000)
```

Definition at line 82 of file project_memory_layout.h.

#### 7.187.1.74 IFX_XIP_PORT0_SBUS_BASE

```
#define IFX_XIP_PORT0_SBUS_BASE IFX_UNSIGNED(0x60000000)
```

Definition at line 81 of file project_memory_layout.h.

#### 7.187.1.75 IFX_XIP_PORT0_SIZE

```
#define IFX_XIP_PORT0_SIZE IFX_UNSIGNED(0x04000000)
```

Definition at line 83 of file project_memory_layout.h.

#### 7.187.1.76 IFX_XIP_PORT1_CBUS_BASE

```
#define IFX_XIP_PORT1_CBUS_BASE IFX_UNSIGNED(0x0C000000)
```

Definition at line 86 of file project_memory_layout.h.

#### 7.187.1.77 IFX_XIP_PORT1_SBUS_BASE

```
#define IFX_XIP_PORT1_SBUS_BASE IFX_UNSIGNED(0x64000000)
```

Definition at line 85 of file project_memory_layout.h.

#### 7.187.1.78 IFX_XIP_PORT1_SIZE

```
#define IFX_XIP_PORT1_SIZE IFX_UNSIGNED(0x04000000)
```

Definition at line 87 of file project_memory_layout.h.

## 7.188 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EV↩ K/shared/device/include/startup_pse84.h File Reference

```
#include <stddef.h>
#include "cmsis.h"
#include "cy_device.h"
#include "ifx_startup.h"
```

Include dependency graph for startup_pse84.h:

This graph shows which files directly or indirectly include this file:

## 7.189 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/device/source/startup_pse84.c File Reference

```
#include "startup_pse84.h"
#include "pse84_core_interrupts.h"
#include "static_checks.h"
```

Include dependency graph for startup_pse84.c:

### Functions

- __NO_RETURN void __PROGRAM_START (void)
- __NO_RETURN void Reset_Handler (void)
- __NO_RETURN void Default_Handler (void)

## Variables

- uint32_t __INITIAL_SP
- uint32_t __STACK_LIMIT
- IFX_CORE_DEFINE_EXCEPTIONS_LIST const IFX_CORE_DEFINE_INTERRUPTS_LIST VECTOR_TABLE_Type __vector_table[VECTORTABLE_SIZE] IFX_VECTOR_TABLE_ATTRIBUTE

## 7.189.1 Function Documentation

### 7.189.1.1 __PROGRAM_START()

```
__NO_RETURN void __PROGRAM_START (
            void  )
```

Here is the caller graph for this function:

### 7.189.1.2 Default_Handler()

```
__NO_RETURN void Default_Handler (
            void  )
```

Definition at line 134 of file startup_pse84.c.

### 7.189.1.3 Reset_Handler()

```
__NO_RETURN void Reset_Handler (
            void  )
```

Definition at line 74 of file startup_pse84.c.
Here is the call graph for this function:

## 7.189.2 Variable Documentation

#### 7.189.2.1 __INITIAL_SP

```
uint32_t __INITIAL_SP
```

#### 7.189.2.2 __STACK_LIMIT

```
uint32_t __STACK_LIMIT
```

#### 7.189.2.3 IFX_VECTOR_TABLE_ATTRIBUTE

```
IFX_CORE_DEFINE_EXCEPTIONS_LIST const IFX_CORE_DEFINE_INTERRUPTS_LIST VECTOR_TABLE_Type __vector_table
[VECTORTABLE_SIZE] IFX_VECTOR_TABLE_ATTRIBUTE
```

**Initial value:**

```
= {
    IFX_CORE_EXCEPTIONS_LIST
    IFX_CORE_INTERRUPTS_LIST
}
```

Definition at line 60 of file startup_pse84.c.

## 7.190 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/shared/ifx_peripherals_def.h File Reference

```
#include "cy_device.h"
#include "cy_tcpwm_counter.h"
#include "cycfg_pins.h"
#include "mxs22.h"
```

Include dependency graph for ifx_peripherals_def.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_IPC_TFM_TO_NS_IRQ_PRIORITY 3U
- #define IFX_IPC_NS_TO_TFM_IRQ_PRIORITY 3U

- #define IFX_IRQ_TEST_NS_INTERRUPT CYBSP_USER_LED1_IRQ
- #define IFX_IRQ_TEST_NS_INTERRUPT_GPIO_PORT IFX_NS_ADDRESS_ALIAS_T(GPIO_PRT_Type∗, CYBSP_USER_LED1_PORT)
- #define IFX_IRQ_TEST_NS_INTERRUPT_GPIO_PIN CYBSP_USER_LED1_PIN
- #define IFX_IRQ_TEST_FLIH_INTERRUPT CYBSP_DEBUG_UART_TX_IRQ
- #define IFX_IRQ_TEST_FLIH_INTERRUPT_GPIO_PORT CYBSP_DEBUG_UART_TX_PORT
- #define IFX_IRQ_TEST_FLIH_INTERRUPT_GPIO_PIN CYBSP_DEBUG_UART_TX_PIN
- #define TFM_FPU_NS_TEST_IRQ ioss_interrupts_gpio_2_IRQn
- #define IFX_MXCM55 IFX_NS_ADDRESS_ALIAS_T(MXCM55_Type∗, MXCM55)

## 7.190.1 Macro Definition Documentation

### 7.190.1.1 IFX_IPC_NS_TO_TFM_IRQ_PRIORITY

```
#define IFX_IPC_NS_TO_TFM_IRQ_PRIORITY 3U
```

Definition at line 24 of file ifx_peripherals_def.h.

### 7.190.1.2 IFX_IPC_TFM_TO_NS_IRQ_PRIORITY

```
#define IFX_IPC_TFM_TO_NS_IRQ_PRIORITY 3U
```

Definition at line 23 of file ifx_peripherals_def.h.

### 7.190.1.3 IFX_IRQ_TEST_FLIH_INTERRUPT

```
#define IFX_IRQ_TEST_FLIH_INTERRUPT CYBSP_DEBUG_UART_TX_IRQ
```

Definition at line 34 of file ifx_peripherals_def.h.

### 7.190.1.4 IFX_IRQ_TEST_FLIH_INTERRUPT_GPIO_PIN

```
#define IFX_IRQ_TEST_FLIH_INTERRUPT_GPIO_PIN CYBSP_DEBUG_UART_TX_PIN
```

Definition at line 36 of file ifx_peripherals_def.h.

### 7.190.1.5 IFX_IRQ_TEST_FLIH_INTERRUPT_GPIO_PORT

```
#define IFX_IRQ_TEST_FLIH_INTERRUPT_GPIO_PORT CYBSP_DEBUG_UART_TX_PORT
```

Definition at line 35 of file ifx_peripherals_def.h.

### 7.190.1.6 IFX_IRQ_TEST_NS_INTERRUPT

```
#define IFX_IRQ_TEST_NS_INTERRUPT CYBSP_USER_LED1_IRQ
```

Definition at line 28 of file ifx_peripherals_def.h.

### 7.190.1.7 IFX_IRQ_TEST_NS_INTERRUPT_GPIO_PIN

```
#define IFX_IRQ_TEST_NS_INTERRUPT_GPIO_PIN CYBSP_USER_LED1_PIN
```

Definition at line 30 of file ifx_peripherals_def.h.

#### 7.190.1.8 IFX_IRQ_TEST_NS_INTERRUPT_GPIO_PORT

```
#define IFX_IRQ_TEST_NS_INTERRUPT_GPIO_PORT IFX_NS_ADDRESS_ALIAS_T(GPIO_PRT_Type*, CYBSP_USER↩
_LED1_PORT)
```

Definition at line 29 of file ifx_peripherals_def.h.

#### 7.190.1.9 IFX_MXCM55

```
#define IFX_MXCM55 IFX_NS_ADDRESS_ALIAS_T(MXCM55_Type*, MXCM55)
```

Definition at line 41 of file ifx_peripherals_def.h.

#### 7.190.1.10 TFM_FPU_NS_TEST_IRQ

```
#define TFM_FPU_NS_TEST_IRQ ioss_interrupts_gpio_2_IRQn
```

Definition at line 38 of file ifx_peripherals_def.h.

## 7.191 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EV↩ K/spe/ifx_s_peripherals_def.h File Reference

```
#include "cy_device.h"
#include "cy_tcpwm_counter.h"
#include "cy_scb_uart.h"
#include "cycfg_peripherals.h"
#include "mxs22.h"
```

Include dependency graph for ifx_s_peripherals_def.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_TFM_PS_SMIF_MEMORY_CONFIG ifx_smif_0_memory_config
- #define IFX_SMIF_0_MEMORY_CONFIG S25HS512T_SMIF0_SlaveSlot_0
- #define TFM_FPU_S_TEST_IRQ ioss_interrupts_gpio_1_IRQn
- #define IFX_RRAMC0 IFX_NS_ADDRESS_ALIAS_T(RRAMC_Type∗, RRAMC0)
- #define IFX_SMIF_HW IFX_NS_ADDRESS_ALIAS_T(SMIF_Type∗, CYBSP_SMIF_CORE_0_XSPI_FLA↩ SH_HW)
- #define IFX_NON_SECURE_SFLASH_BASE IFX_NS_ADDRESS_ALIAS_T(NON_SECURE_SFLASH_↩ Type∗, NON_SECURE_SFLASH_BASE)
- #define IFX_LCS_BASE IFX_NS_ADDRESS_ALIAS_T(SRSS_Type∗, SRSS)

## Variables

- struct ifx_driver_smif_mem_t ifx_smif_0_memory_config

## 7.191.1 Macro Definition Documentation

### 7.191.1.1 IFX_LCS_BASE

```
#define IFX_LCS_BASE IFX_NS_ADDRESS_ALIAS_T(SRSS_Type*, SRSS)
```

Definition at line 55 of file ifx_s_peripherals_def.h.

### 7.191.1.2 IFX_NON_SECURE_SFLASH_BASE

```
#define IFX_NON_SECURE_SFLASH_BASE IFX_NS_ADDRESS_ALIAS_T(NON_SECURE_SFLASH_Type*, NON_SECURE↩
_SFLASH_BASE)
```

Definition at line 48 of file ifx_s_peripherals_def.h.

### 7.191.1.3 IFX_RRAMC0

```
#define IFX_RRAMC0 IFX_NS_ADDRESS_ALIAS_T(RRAMC_Type*, RRAMC0)
```

Definition at line 42 of file ifx_s_peripherals_def.h.

### 7.191.1.4 IFX_SMIF_0_MEMORY_CONFIG

```
#define IFX_SMIF_0_MEMORY_CONFIG S25HS512T_SMIF0_SlaveSlot_0
```

Definition at line 35 of file ifx_s_peripherals_def.h.

### 7.191.1.5 IFX_SMIF_HW

```
#define IFX_SMIF_HW IFX_NS_ADDRESS_ALIAS_T(SMIF_Type*, CYBSP_SMIF_CORE_0_XSPI_FLASH_HW)
```

Definition at line 45 of file ifx_s_peripherals_def.h.

### 7.191.1.6 IFX_TFM_PS_SMIF_MEMORY_CONFIG

```
#define IFX_TFM_PS_SMIF_MEMORY_CONFIG ifx_smif_0_memory_config
```

Definition at line 22 of file ifx_s_peripherals_def.h.

#### 7.191.1.7 TFM_FPU_S_TEST_IRQ

```
#define TFM_FPU_S_TEST_IRQ ioss_interrupts_gpio_1_IRQn
```

Definition at line 39 of file ifx_s_peripherals_def.h.

### 7.191.2 Variable Documentation

#### 7.191.2.1 ifx_smif_0_memory_config

```
struct ifx_driver_smif_mem_t ifx_smif_0_memory_config
```

## 7.192 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EV↩ K/spe/ifx_spm_init.c File Reference

```
#include "config_tfm.h"
#include "region_defs.h"
#include "ifx_spm_init.h"
#include "tfm_peripherals_def.h"
#include "tfm_spm_log.h"
#include "utilities.h"
#include "cybsp.h"
#include "cycfg.h"
#include "cy_device.h"
#include "cy_gpio.h"
#include "cy_scb_uart.h"
#include "cy_sysclk.h"
```

Include dependency graph for ifx_spm_init.c:

### Functions

- void ifx_init_spm_peripherals (void)

### 7.192.1 Function Documentation

#### 7.192.1.1 ifx_init_spm_peripherals()

```
void ifx_init_spm_peripherals (
            void  )
```

Initialize SPM peripherals.

Definition at line 23 of file ifx_spm_init.c.

Here is the call graph for this function:

## 7.193 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/ifx_spm_init.h File Reference

This graph shows which files directly or indirectly include this file:

### Functions

- void ifx_init_spm_peripherals (void)

### 7.193.1 Function Documentation

#### 7.193.1.1 ifx_init_spm_peripherals()

```
void ifx_init_spm_peripherals (
            void  )
```

Initialize SPM peripherals.
Definition at line 23 of file ifx_spm_init.c.

Here is the call graph for this function:

## 7.194 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection_regions_cfg.c File Reference

```
#include "array.h"
#include "project_memory_layout.h"
#include "cy_device.h"
#include "protection_regions_cfg.h"
```

Include dependency graph for protection_regions_cfg.c:

### Variables

- PPC_Type *const ifx_ppcx_region_ptrs [ ]
- const size_t ifx_ppcx_region_ptrs_count = ARRAY_SIZE(ifx_ppcx_region_ptrs)
- const ifx_ppcx_config_t ifx_ppcx_static_config [ ]
- const size_t ifx_ppcx_static_config_count = ARRAY_SIZE(ifx_ppcx_static_config)
- const ifx_msc_agc_resp_config_t ifx_msc_agc_resp_config [ ]
- const size_t ifx_msc_agc_resp_config_count = ARRAY_SIZE(ifx_msc_agc_resp_config)
- const ifx_msc_agc_resp_config_v1_t ifx_msc_agc_resp_config_v1 [ ]
- const size_t ifx_msc_agc_resp_config_v1_count = ARRAY_SIZE(ifx_msc_agc_resp_config_v1)
- const IRQn_Type ifx_secure_interrupts_config [ ]
- const size_t ifx_secure_interrupts_config_count = ARRAY_SIZE(ifx_secure_interrupts_config)
- const cy_en_SysFault_source_t ifx_fault_sources_fault_struct0 [ ]
- const size_t ifx_fault_sources_fault_struct0_count = ARRAY_SIZE(ifx_fault_sources_fault_struct0)

### 7.194.1 Variable Documentation

#### 7.194.1.1 ifx_fault_sources_fault_struct0

```
const cy_en_SysFault_source_t ifx_fault_sources_fault_struct0[]
```

Definition at line 181 of file protection_regions_cfg.c.

#### 7.194.1.2 ifx_fault_sources_fault_struct0_count

```
const size_t ifx_fault_sources_fault_struct0_count = ARRAY_SIZE(ifx_fault_sources_fault_struct0)
```

Definition at line 217 of file protection_regions_cfg.c.

#### 7.194.1.3 ifx_msc_agc_resp_config

```
const ifx_msc_agc_resp_config_t ifx_msc_agc_resp_config[]
```

**Initial value:**

```
= {
}
```

Definition at line 100 of file protection_regions_cfg.c.

#### 7.194.1.4 ifx_msc_agc_resp_config_count

```
const size_t ifx_msc_agc_resp_config_count = ARRAY_SIZE(ifx_msc_agc_resp_config)
```

Definition at line 122 of file protection_regions_cfg.c.

#### 7.194.1.5 ifx_msc_agc_resp_config_v1

```
const ifx_msc_agc_resp_config_v1_t ifx_msc_agc_resp_config_v1[]
```

Definition at line 125 of file protection_regions_cfg.c.

#### 7.194.1.6 ifx_msc_agc_resp_config_v1_count

```
const size_t ifx_msc_agc_resp_config_v1_count = ARRAY_SIZE(ifx_msc_agc_resp_config_v1)
```

Definition at line 165 of file protection_regions_cfg.c.

#### 7.194.1.7 ifx_ppcx_region_ptrs

```
PPC_Type* const ifx_ppcx_region_ptrs[]
```

**Initial value:**

```
= {
    PPC0,
    PPC1,
}
```

Definition at line 24 of file protection_regions_cfg.c.

#### 7.194.1.8 ifx_ppcx_region_ptrs_count

```
const size_t ifx_ppcx_region_ptrs_count = ARRAY_SIZE(ifx_ppcx_region_ptrs)
```

Definition at line 30 of file protection_regions_cfg.c.

#### 7.194.1.9 ifx_ppcx_static_config

```
const ifx_ppcx_config_t ifx_ppcx_static_config[]
```

**Initial value:**

```
= {
    {
        .configs = cycfg_ppc_0_domains_config,
        .config_count = &cycfg_ppc_0_domains_count,
        .ppc_base = PPC0,
    },
    {
        .configs = cycfg_ppc_1_domains_config,
        .config_count = &cycfg_ppc_1_domains_count,
        .ppc_base = PPC1,
    },
}
```

Definition at line 83 of file protection_regions_cfg.c.

#### 7.194.1.10 ifx_ppcx_static_config_count

```
const size_t ifx_ppcx_static_config_count = ARRAY_SIZE(ifx_ppcx_static_config)
```

Definition at line 97 of file protection_regions_cfg.c.

#### 7.194.1.11 ifx_secure_interrupts_config

```
const IRQn_Type ifx_secure_interrupts_config[]
```

**Initial value:**

```
= {
      m33syscpuss_interrupts_fault_0_IRQn,
      m33syscpuss_interrupt_msc_IRQn,
}
```

Definition at line 169 of file protection_regions_cfg.c.

#### 7.194.1.12 ifx_secure_interrupts_config_count

```
const size_t ifx_secure_interrupts_config_count = ARRAY_SIZE(ifx_secure_interrupts_config)
```

Definition at line 178 of file protection_regions_cfg.c.

## 7.195 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EVK/spe/protection_regions_cfg.h File Reference

```
#include "cy_mpc.h"
#include "cy_ms_ctl.h"
#include "cy_ppc.h"
#include "cy_sysfault.h"
#include "cycfg_protection.h"
#include "cycfg_qspi_memslot.h"
#include "cycfg_ppc.h"
#include "cycfg_system.h"
#include "partition_edge.h"
#include "project_memory_layout.h"
#include "protection_types.h"
```

Include dependency graph for protection_regions_cfg.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct ifx_msc_agc_resp_config_t
- struct ifx_msc_agc_resp_config_v1_t

## Macros

- #define IFX_SMIF_XIP0_ENABLED (CY_SMIF_DEVICE_NUM0 != 0U)
- #define IFX_SMIF_XIP1_ENABLED (CY_SMIF_DEVICE_NUM1 != 0U)
- #define IFX_SOCMEM_ENABLED (socmem_0_mpc_0_REGION_COUNT != 0U)
- #define CY_SAU_REGION_CNT ((sizeof(SAU_config)/sizeof(SAU_config[0])) - 1u)

## Typedefs

- typedef cy_stc_sau_config_t ifx_sau_config_t

## Variables

- PPC_Type *const ifx_ppcx_region_ptrs [ ]
- const size_t ifx_ppcx_region_ptrs_count
- const ifx_ppcx_config_t ifx_ppcx_static_config [ ]
- const size_t ifx_ppcx_static_config_count
- const ifx_msc_agc_resp_config_t ifx_msc_agc_resp_config [ ]
- const size_t ifx_msc_agc_resp_config_count
- const ifx_msc_agc_resp_config_v1_t ifx_msc_agc_resp_config_v1 [ ]
- const size_t ifx_msc_agc_resp_config_v1_count
- const IRQn_Type ifx_secure_interrupts_config [ ]
- const size_t ifx_secure_interrupts_config_count
- const cy_en_SysFault_source_t ifx_fault_sources_fault_struct0 [ ]
- const size_t ifx_fault_sources_fault_struct0_count

## 7.195.1 Macro Definition Documentation

### 7.195.1.1 CY_SAU_REGION_CNT

```
#define CY_SAU_REGION_CNT ((sizeof(SAU_config)/sizeof(SAU_config[0])) - 1u)
```

Definition at line 50 of file protection_regions_cfg.h.

### 7.195.1.2 IFX_SMIF_XIP0_ENABLED

```
#define IFX_SMIF_XIP0_ENABLED (CY_SMIF_DEVICE_NUM0 != 0U)
```

Definition at line 25 of file protection_regions_cfg.h.

### 7.195.1.3 IFX_SMIF_XIP1_ENABLED

```
#define IFX_SMIF_XIP1_ENABLED (CY_SMIF_DEVICE_NUM1 != 0U)
```

Definition at line 28 of file protection_regions_cfg.h.

### 7.195.1.4 IFX_SOCMEM_ENABLED

```
#define IFX_SOCMEM_ENABLED (socmem_0_mpc_0_REGION_COUNT != 0U)
```

Definition at line 31 of file protection_regions_cfg.h.

## 7.195.2 Typedef Documentation

#### 7.195.2.1 ifx_sau_config_t

`typedef cy_stc_sau_config_t ifx_sau_config_t`
Definition at line 45 of file protection_regions_cfg.h.

### 7.195.3 Variable Documentation

#### 7.195.3.1 ifx_fault_sources_fault_struct0

`const cy_en_SysFault_source_t ifx_fault_sources_fault_struct0[]`
Definition at line 181 of file protection_regions_cfg.c.

#### 7.195.3.2 ifx_fault_sources_fault_struct0_count

`const size_t ifx_fault_sources_fault_struct0_count`
Definition at line 217 of file protection_regions_cfg.c.

#### 7.195.3.3 ifx_msc_agc_resp_config

`const ifx_msc_agc_resp_config_t ifx_msc_agc_resp_config[]`
Definition at line 100 of file protection_regions_cfg.c.

#### 7.195.3.4 ifx_msc_agc_resp_config_count

`const size_t ifx_msc_agc_resp_config_count`
Definition at line 122 of file protection_regions_cfg.c.

#### 7.195.3.5 ifx_msc_agc_resp_config_v1

`const ifx_msc_agc_resp_config_v1_t ifx_msc_agc_resp_config_v1[]`
Definition at line 125 of file protection_regions_cfg.c.

#### 7.195.3.6 ifx_msc_agc_resp_config_v1_count

`const size_t ifx_msc_agc_resp_config_v1_count`
Definition at line 165 of file protection_regions_cfg.c.

#### 7.195.3.7 ifx_ppcx_region_ptrs

`PPC_Type* const ifx_ppcx_region_ptrs[]`
Definition at line 24 of file protection_regions_cfg.c.

#### 7.195.3.8 ifx_ppcx_region_ptrs_count

`const size_t ifx_ppcx_region_ptrs_count`
Definition at line 30 of file protection_regions_cfg.c.

#### 7.195.3.9 ifx_ppcx_static_config

```
const ifx_ppcx_config_t ifx_ppcx_static_config[]
```
Definition at line 83 of file protection_regions_cfg.c.

#### 7.195.3.10 ifx_ppcx_static_config_count

```
const size_t ifx_ppcx_static_config_count
```
Definition at line 97 of file protection_regions_cfg.c.

#### 7.195.3.11 ifx_secure_interrupts_config

```
const IRQn_Type ifx_secure_interrupts_config[]
```
Definition at line 169 of file protection_regions_cfg.c.

#### 7.195.3.12 ifx_secure_interrupts_config_count

```
const size_t ifx_secure_interrupts_config_count
```
Definition at line 178 of file protection_regions_cfg.c.

## 7.196 platform/ext/target/infineon/pse84/board/KIT_PSOCE84_EV↩ K/spe/shared_ro_data.c File Reference

This file contains shared data that can be read/write within SPM and read-only from any secure partition.

```
#include "ifx_driver_smif.h"
#include "cycfg_qspi_memslot.h"
#include "tfm_peripherals_def.h"
#include "static_checks.h"
```
Include dependency graph for shared_ro_data.c:

### 7.196.1 Detailed Description

This file contains shared data that can be read/write within SPM and read-only from any secure partition.

## 7.197 platform/ext/target/infineon/pse84/config/ifx_platform_spe_types.h File Reference

This file contains platform specific types and data declaration used to build secure image.

```
#include <stdint.h>
#include "cy_mpc.h"
```

```
#include "partition_edge.h"
```

Include dependency graph for ifx_platform_spe_types.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct ifx_memory_config_t

## Macros

- #define IFX_GET_PC(mask, pc) (((mask) >> ((uint32_t)(pc))) & 1UL)
- #define IFX_PROTECTION_MPU_PERIPHERAL_REGION_START ((uint32_t)MMIO_NS_M33SYS_STA↩RT)
- #define IFX_PROTECTION_MPU_PERIPHERAL_REGION_LIMIT ((uint32_t)(MMIO_S_M33SYS_START + MMIO_M33SYS_SIZE + MMIO_M55APP_SIZE - 1))
- #define IFX_MPC_IS_EXTERNAL(base)
- #define VECTOR_FIXED_EXP_NR ARMV8M_FIXED_EXP_NR

## Typedefs

- typedef uint32_t ifx_pc_mask_t
- typedef cy_stc_mpc_unified_t ifx_mem_domain_cfg_t
- typedef cy_stc_mpc_regions_t ifx_mem_domain_region_cfg_t

## Variables

- const ifx_memory_config_t *const ifx_memory_cm33_config [ ]
- const size_t ifx_memory_cm33_config_count
- const ifx_memory_config_t *const ifx_memory_cm55_config [ ]
- const size_t ifx_memory_cm55_config_count

### 7.197.1 Detailed Description

This file contains platform specific types and data declaration used to build secure image.
This file is part of Infineon platform configuration files. It's expected that this file provides platform dependent types and data declaration used by Infineon common code.

### 7.197.2 Macro Definition Documentation

#### 7.197.2.1 IFX_GET_PC

```
#define IFX_GET_PC(
            mask,
            pc ) (((mask) >> ((uint32_t)(pc))) & 1UL)
```

Definition at line 24 of file ifx_platform_spe_types.h.

#### 7.197.2.2 IFX_MPC_IS_EXTERNAL

```
#define IFX_MPC_IS_EXTERNAL(
            base )
```

**Value:**

```
    (((void *)base == IFX_MPC_NOT_CONTROLLED_BY_TFM) || \
     ((void *)base == RRAMC0_MPC0) || \
     ((void *)base == RRAMC0_MPC1))
```

Definition at line 33 of file ifx_platform_spe_types.h.

#### 7.197.2.3 IFX_PROTECTION_MPU_PERIPHERAL_REGION_LIMIT

```
#define IFX_PROTECTION_MPU_PERIPHERAL_REGION_LIMIT ((uint32_t)(MMIO_S_M33SYS_START + MMIO_↩
M33SYS_SIZE + MMIO_M55APP_SIZE - 1))
```

Definition at line 31 of file ifx_platform_spe_types.h.

#### 7.197.2.4 IFX_PROTECTION_MPU_PERIPHERAL_REGION_START

```
#define IFX_PROTECTION_MPU_PERIPHERAL_REGION_START ((uint32_t)MMIO_NS_M33SYS_START)
```

Definition at line 30 of file ifx_platform_spe_types.h.

#### 7.197.2.5 VECTOR_FIXED_EXP_NR

```
#define VECTOR_FIXED_EXP_NR ARMV8M_FIXED_EXP_NR
```

Definition at line 40 of file ifx_platform_spe_types.h.

### 7.197.3 Typedef Documentation

#### 7.197.3.1 ifx_mem_domain_cfg_t

```
typedef cy_stc_mpc_unified_t ifx_mem_domain_cfg_t
```

Definition at line 57 of file ifx_platform_spe_types.h.

#### 7.197.3.2 ifx_mem_domain_region_cfg_t

```
typedef cy_stc_mpc_regions_t ifx_mem_domain_region_cfg_t
```

Definition at line 60 of file ifx_platform_spe_types.h.

#### 7.197.3.3 ifx_pc_mask_t

```
typedef uint32_t ifx_pc_mask_t
```

Definition at line 46 of file ifx_platform_spe_types.h.

### 7.197.4 Variable Documentation

#### 7.197.4.1 ifx_memory_cm33_config

```
const ifx_memory_config_t* const ifx_memory_cm33_config[]
```

Definition at line 130 of file protection_data.c.

#### 7.197.4.2 ifx_memory_cm33_config_count

```
const size_t ifx_memory_cm33_config_count
```

Definition at line 154 of file protection_data.c.

#### 7.197.4.3 ifx_memory_cm55_config

```
const ifx_memory_config_t* const ifx_memory_cm55_config[]
```

Definition at line 156 of file protection_data.c.

#### 7.197.4.4 ifx_memory_cm55_config_count

```
const size_t ifx_memory_cm55_config_count
```

Definition at line 166 of file protection_data.c.

## 7.198 platform/ext/target/infineon/pse84/config/pse84_spe_config.h File Reference

This file contains PSE84 family specific configuration used to build secure image.

This graph shows which files directly or indirectly include this file:

## Macros

- #define IFX_MPC_CONFIGURED_BY_TFM 1
- #define IFX_PLATFORM_MPC_PRESENT 1
- #define IFX_MPC_DRIVER_HW_MPC_WITH_ROT 1
- #define IFX_MPC_DRIVER_HW_MPC_WITHOUT_ROT 0
- #define IFX_MPC_CM55_MPC 1
- #define IFX_REGION_MAX_MPC_COUNT (2U)
- #define IFX_MPC_NOT_CONTROLLED_BY_TFM (NULL)
- #define IFX_SE_RT_RRAM_BLOCK_SIZE 4096u
- #define IFX_SE_RT_RRAM_MPC_DEFAULT_ATTRIBUTES 0x00000400u
- #define IFX_PLATFORM_PPC_PRESENT 1
- #define IFX_MSC_ACG_RESP_CONFIG 1
- #define IFX_MSC_ACG_RESP_CONFIG_V1 1
- #define IFX_MSC_TFM_CORE_BUS_MASTER_ID CY_MS_CTL_ID_CM33_0

### 7.198.1 Detailed Description

This file contains PSE84 family specific configuration used to build secure image.
This file is part of Infineon platform configuration files. It's expected that this file provides platform dependent configuration used by Infineon common code.

### 7.198.2 Macro Definition Documentation

#### 7.198.2.1 IFX_MPC_CM55_MPC

```
#define IFX_MPC_CM55_MPC 1
```

Definition at line 33 of file pse84_spe_config.h.

#### 7.198.2.2 IFX_MPC_CONFIGURED_BY_TFM

```
#define IFX_MPC_CONFIGURED_BY_TFM 1
```

Definition at line 21 of file pse84_spe_config.h.

#### 7.198.2.3 IFX_MPC_DRIVER_HW_MPC_WITH_ROT

```
#define IFX_MPC_DRIVER_HW_MPC_WITH_ROT 1
```

Definition at line 27 of file pse84_spe_config.h.

#### 7.198.2.4 IFX_MPC_DRIVER_HW_MPC_WITHOUT_ROT

```
#define IFX_MPC_DRIVER_HW_MPC_WITHOUT_ROT 0
```

Definition at line 30 of file pse84_spe_config.h.

#### 7.198.2.5 IFX_MPC_NOT_CONTROLLED_BY_TFM

```
#define IFX_MPC_NOT_CONTROLLED_BY_TFM (NULL)
```

Definition at line 39 of file pse84_spe_config.h.

#### 7.198.2.6 IFX_MSC_ACG_RESP_CONFIG

```
#define IFX_MSC_ACG_RESP_CONFIG 1
```

Definition at line 51 of file pse84_spe_config.h.

#### 7.198.2.7 IFX_MSC_ACG_RESP_CONFIG_V1

```
#define IFX_MSC_ACG_RESP_CONFIG_V1 1
```

Definition at line 54 of file pse84_spe_config.h.

#### 7.198.2.8 IFX_MSC_TFM_CORE_BUS_MASTER_ID

```
#define IFX_MSC_TFM_CORE_BUS_MASTER_ID CY_MS_CTL_ID_CM33_0
```

Definition at line 57 of file pse84_spe_config.h.

#### 7.198.2.9 IFX_PLATFORM_MPC_PRESENT

```
#define IFX_PLATFORM_MPC_PRESENT 1
```

Definition at line 24 of file pse84_spe_config.h.

#### 7.198.2.10 IFX_PLATFORM_PPC_PRESENT

```
#define IFX_PLATFORM_PPC_PRESENT 1
```

Definition at line 48 of file pse84_spe_config.h.

#### 7.198.2.11 IFX_REGION_MAX_MPC_COUNT

```
#define IFX_REGION_MAX_MPC_COUNT (2U)
```

Definition at line 38 of file pse84_spe_config.h.

#### 7.198.2.12 IFX_SE_RT_RRAM_BLOCK_SIZE

```
#define IFX_SE_RT_RRAM_BLOCK_SIZE 4096u
```

Definition at line 42 of file pse84_spe_config.h.

#### 7.198.2.13 IFX_SE_RT_RRAM_MPC_DEFAULT_ATTRIBUTES

```
#define IFX_SE_RT_RRAM_MPC_DEFAULT_ATTRIBUTES 0x00000400u
```

Definition at line 45 of file pse84_spe_config.h.

## 7.199 platform/ext/target/infineon/pse84/config_tfm_target.h File Reference

```
#include "ifx_platform_config.h"
```

Include dependency graph for config_tfm_target.h:

### Macros

- #define IFX_MEMORY_CONFIGURATOR_MPC_CONFIG
- #define ITS_BUF_SIZE ITS_MAX_ASSET_SIZE
- #define CRYPTO_STACK_SIZE 0x2000
- #define PS_STACK_SIZE 0x1000
- #define PS_NUM_ASSETS 20
- #define IFX_SE_IPC_SERVICE_STACK_SIZE 0x400
- #define PLATFORM_SERVICE_INPUT_BUFFER_SIZE 128U

### 7.199.1 Macro Definition Documentation

#### 7.199.1.1 CRYPTO_STACK_SIZE

```
#define CRYPTO_STACK_SIZE 0x2000
```

Definition at line 58 of file config_tfm_target.h.

#### 7.199.1.2 IFX_MEMORY_CONFIGURATOR_MPC_CONFIG

```
#define IFX_MEMORY_CONFIGURATOR_MPC_CONFIG
```

Definition at line 31 of file config_tfm_target.h.

#### 7.199.1.3 IFX_SE_IPC_SERVICE_STACK_SIZE

```
#define IFX_SE_IPC_SERVICE_STACK_SIZE 0x400
```

Definition at line 114 of file config_tfm_target.h.

#### 7.199.1.4 ITS_BUF_SIZE

```
#define ITS_BUF_SIZE ITS_MAX_ASSET_SIZE
```

Definition at line 39 of file config_tfm_target.h.

#### 7.199.1.5 PLATFORM_SERVICE_INPUT_BUFFER_SIZE

```
#define PLATFORM_SERVICE_INPUT_BUFFER_SIZE 128U
```

Definition at line 159 of file config_tfm_target.h.

#### 7.199.1.6 PS_NUM_ASSETS

```
#define PS_NUM_ASSETS 20
```

Definition at line 72 of file config_tfm_target.h.

#### 7.199.1.7 PS_STACK_SIZE

```
#define PS_STACK_SIZE 0x1000
```

Definition at line 63 of file config_tfm_target.h.

## 7.200 platform/ext/target/infineon/pse84/epc2/board/KIT_PSOCE84_EVK/config_bsp.h File Reference

### Macros

- #define IFX_STARTUP_HEADER_FILE "startup_pse84.h"

### 7.200.1 Macro Definition Documentation

#### 7.200.1.1 IFX_STARTUP_HEADER_FILE

```
#define IFX_STARTUP_HEADER_FILE "startup_pse84.h"
```

Definition at line 12 of file config_bsp.h.

## 7.201 platform/ext/target/infineon/pse84/epc2/ifx_platform_config.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_PSE84_EPC2
- #define PS_DEFAULT_CRYPTO_CONFIG 1
- #define CONFIG_TFM_SCHEDULE_WHEN_NS_INTERRUPTED 0
- #define CONFIG_TFM_SECURE_THREAD_MASK_NS_INTERRUPT 0

### 7.201.1 Macro Definition Documentation

#### 7.201.1.1 CONFIG_TFM_SCHEDULE_WHEN_NS_INTERRUPTED

```
#define CONFIG_TFM_SCHEDULE_WHEN_NS_INTERRUPTED 0
```

Definition at line 30 of file ifx_platform_config.h.

#### 7.201.1.2 CONFIG_TFM_SECURE_THREAD_MASK_NS_INTERRUPT

```
#define CONFIG_TFM_SECURE_THREAD_MASK_NS_INTERRUPT 0
```

Definition at line 32 of file ifx_platform_config.h.

#### 7.201.1.3 IFX_PSE84_EPC2

```
#define IFX_PSE84_EPC2
```

Definition at line 14 of file ifx_platform_config.h.

#### 7.201.1.4 PS_DEFAULT_CRYPTO_CONFIG

```
#define PS_DEFAULT_CRYPTO_CONFIG 1
```

Definition at line 20 of file ifx_platform_config.h.

## 7.202 platform/ext/target/infineon/pse84/epc2/ifx_platform_spe_config.h File Reference

This file contains platform specific configuration used to build secure image.
`#include "pse84_spe_config.h"`
Include dependency graph for ifx_platform_spe_config.h:

### Macros

- #define IFX_PC_SE_RT_ID 0x01UL /∗ Secure Enclave ∗/
- #define IFX_PC_TFM_SPM_ID 0x02UL /∗ TFM SPM ∗/
- #define IFX_PC_TFM_PROT_ID 0x02UL /∗ TFM PSA RoT ∗/
- #define IFX_PC_TFM_AROT_ID 0x02UL /∗ TFM Application RoT ∗/
- #define IFX_PC_CM33_NSPE_ID 0x02UL /∗ CM33 NSPE Application ∗/
- #define IFX_PC_TZ_NSPE_ID IFX_PC_CM33_NSPE_ID
- #define IFX_PC_CM55_NSPE_ID 0x06UL /∗ CM55 NSPE Application ∗/
- #define IFX_PC_MAILBOX_NSPE_ID IFX_PC_CM55_NSPE_ID
- #define IFX_PC_DEBUGGER_ID 0x07UL /∗ Debugger ∗/
- #define IFX_PC_NONE (0x00) /∗ No PC, empty PC mask ∗/
- #define IFX_PC_SE_RT (1UL << IFX_PC_SE_RT_ID)
- #define IFX_PC_TFM_SPM (1UL << IFX_PC_TFM_SPM_ID)
- #define IFX_PC_TFM_PROT (1UL << IFX_PC_TFM_PROT_ID)
- #define IFX_PC_TFM_AROT (1UL << IFX_PC_TFM_AROT_ID)
- #define IFX_PC_CM33_NSPE (1UL << IFX_PC_CM33_NSPE_ID)
- #define IFX_PC_TZ_NSPE (1UL << IFX_PC_TZ_NSPE_ID)
- #define IFX_PC_CM55_NSPE (1UL << IFX_PC_CM55_NSPE_ID)
- #define IFX_PC_MAILBOX_NSPE (1UL << IFX_PC_MAILBOX_NSPE_ID)
- #define IFX_PC_DEBUGGER (1UL << IFX_PC_DEBUGGER_ID)
- #define IFX_PC_DEFAULT (IFX_PC_SE_RT | IFX_PC_TFM_SPM)
- #define IFX_PSA_ROT_PRIVILEGED 1
- #define IFX_APP_ROT_PRIVILEGED 0
- #define IFX_NS_AGENT_TZ_PRIVILEGED 1
- #define IFX_APP_ROT_PPC_DYNAMIC_ISOLATION 1
- #define IFX_MULTIPLE_IAK_KEY_TYPES 0

### 7.202.1 Detailed Description

This file contains platform specific configuration used to build secure image.
This file is part of Infineon platform configuration files. It's expected that this file provides platform dependent configuration used by Infineon common code.

## 7.202.2 Macro Definition Documentation

### 7.202.2.1 IFX_APP_ROT_PPC_DYNAMIC_ISOLATION

```
#define IFX_APP_ROT_PPC_DYNAMIC_ISOLATION 1
```

Definition at line 58 of file ifx_platform_spe_config.h.

### 7.202.2.2 IFX_APP_ROT_PRIVILEGED

```
#define IFX_APP_ROT_PRIVILEGED 0
```

Definition at line 52 of file ifx_platform_spe_config.h.

### 7.202.2.3 IFX_MULTIPLE_IAK_KEY_TYPES

```
#define IFX_MULTIPLE_IAK_KEY_TYPES 0
```

Definition at line 61 of file ifx_platform_spe_config.h.

### 7.202.2.4 IFX_NS_AGENT_TZ_PRIVILEGED

```
#define IFX_NS_AGENT_TZ_PRIVILEGED 1
```

Definition at line 55 of file ifx_platform_spe_config.h.

### 7.202.2.5 IFX_PC_CM33_NSPE

```
#define IFX_PC_CM33_NSPE (1UL << IFX_PC_CM33_NSPE_ID)
```

Definition at line 38 of file ifx_platform_spe_config.h.

### 7.202.2.6 IFX_PC_CM33_NSPE_ID

```
#define IFX_PC_CM33_NSPE_ID 0x02UL /* CM33 NSPE Application */
```

Definition at line 27 of file ifx_platform_spe_config.h.

### 7.202.2.7 IFX_PC_CM55_NSPE

```
#define IFX_PC_CM55_NSPE (1UL << IFX_PC_CM55_NSPE_ID)
```

Definition at line 40 of file ifx_platform_spe_config.h.

### 7.202.2.8 IFX_PC_CM55_NSPE_ID

```
#define IFX_PC_CM55_NSPE_ID 0x06UL /* CM55 NSPE Application */
```

Definition at line 29 of file ifx_platform_spe_config.h.

### 7.202.2.9 IFX_PC_DEBUGGER

```
#define IFX_PC_DEBUGGER (1UL << IFX_PC_DEBUGGER_ID)
```

Definition at line 42 of file ifx_platform_spe_config.h.

#### 7.202.2.10 IFX_PC_DEBUGGER_ID

```
#define IFX_PC_DEBUGGER_ID 0x07UL /* Debugger */
```

Definition at line 31 of file ifx_platform_spe_config.h.

#### 7.202.2.11 IFX_PC_DEFAULT

```
#define IFX_PC_DEFAULT (IFX_PC_SE_RT | IFX_PC_TFM_SPM)
```

Definition at line 46 of file ifx_platform_spe_config.h.

#### 7.202.2.12 IFX_PC_MAILBOX_NSPE

```
#define IFX_PC_MAILBOX_NSPE (1UL << IFX_PC_MAILBOX_NSPE_ID)
```

Definition at line 41 of file ifx_platform_spe_config.h.

#### 7.202.2.13 IFX_PC_MAILBOX_NSPE_ID

```
#define IFX_PC_MAILBOX_NSPE_ID IFX_PC_CM55_NSPE_ID
```

Definition at line 30 of file ifx_platform_spe_config.h.

#### 7.202.2.14 IFX_PC_NONE

```
#define IFX_PC_NONE (0x00) /* No PC, empty PC mask */
```

Definition at line 33 of file ifx_platform_spe_config.h.

#### 7.202.2.15 IFX_PC_SE_RT

```
#define IFX_PC_SE_RT (1UL << IFX_PC_SE_RT_ID)
```

Definition at line 34 of file ifx_platform_spe_config.h.

#### 7.202.2.16 IFX_PC_SE_RT_ID

```
#define IFX_PC_SE_RT_ID 0x01UL /* Secure Enclave */
```

Definition at line 23 of file ifx_platform_spe_config.h.

#### 7.202.2.17 IFX_PC_TFM_AROT

```
#define IFX_PC_TFM_AROT (1UL << IFX_PC_TFM_AROT_ID)
```

Definition at line 37 of file ifx_platform_spe_config.h.

#### 7.202.2.18 IFX_PC_TFM_AROT_ID

```
#define IFX_PC_TFM_AROT_ID 0x02UL /* TFM Application RoT */
```

Definition at line 26 of file ifx_platform_spe_config.h.

#### 7.202.2.19 IFX_PC_TFM_PROT

```
#define IFX_PC_TFM_PROT (1UL << IFX_PC_TFM_PROT_ID)
```

Definition at line 36 of file ifx_platform_spe_config.h.

#### 7.202.2.20 IFX_PC_TFM_PROT_ID

```
#define IFX_PC_TFM_PROT_ID 0x02UL /* TFM PSA RoT */
```

Definition at line 25 of file ifx_platform_spe_config.h.

#### 7.202.2.21 IFX_PC_TFM_SPM

```
#define IFX_PC_TFM_SPM (1UL << IFX_PC_TFM_SPM_ID)
```

Definition at line 35 of file ifx_platform_spe_config.h.

#### 7.202.2.22 IFX_PC_TFM_SPM_ID

```
#define IFX_PC_TFM_SPM_ID 0x02UL /* TFM SPM */
```

Definition at line 24 of file ifx_platform_spe_config.h.

#### 7.202.2.23 IFX_PC_TZ_NSPE

```
#define IFX_PC_TZ_NSPE (1UL << IFX_PC_TZ_NSPE_ID)
```

Definition at line 39 of file ifx_platform_spe_config.h.

#### 7.202.2.24 IFX_PC_TZ_NSPE_ID

```
#define IFX_PC_TZ_NSPE_ID IFX_PC_CM33_NSPE_ID
```

Definition at line 28 of file ifx_platform_spe_config.h.

#### 7.202.2.25 IFX_PSA_ROT_PRIVILEGED

```
#define IFX_PSA_ROT_PRIVILEGED 1
```

Definition at line 49 of file ifx_platform_spe_config.h.

## 7.203 platform/ext/target/infineon/pse84/epc2/spe/services/crypto/crypto_keys_rram.h File Reference

```
#include <stdint.h>
```

Include dependency graph for crypto_keys_rram.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct ifx_plat_epc2_keys_t

## Macros

- #define IFX_CRYPTO_KEYS_PTR ((ifx_plat_epc2_keys_t *)0x1200132EU)
- #define IFX_CRYPTO_KEYS_HUK_KEY_BITS (256)
- #define IFX_CRYPTO_KEYS_HUK_KEY_LEN (PSA_KEY_EXPORT_ECC_KEY_PAIR_MAX_SIZE(256U))
- #define IFX_CRYPTO_KEYS_HUK_ALGORITHM (PSA_ALG_HKDF(PSA_ALG_SHA_256))
- #define IFX_CRYPTO_KEYS_HUK_TYPE (PSA_KEY_TYPE_DERIVE)
- #define IFX_CRYPTO_KEYS_IAK_PRIVATE_KEY_BITS (256)
- #define IFX_CRYPTO_KEYS_IAK_PRIVATE_KEY_LEN (PSA_KEY_EXPORT_ECC_KEY_PAIR_MAX_↩ SIZE(256U))
- #define IFX_CRYPTO_KEYS_IAK_PRIVATE_ALGORITHM (PSA_ALG_ECDSA(PSA_ALG_SHA_256))
- #define IFX_CRYPTO_KEYS_IAK_PRIVATE_TYPE (PSA_KEY_TYPE_ECC_KEY_PAIR(PSA_ECC_FA↩ MILY_SECP_R1))
- #define IFX_CRYPTO_KEYS_IAK_PUBLIC_KEY_BITS (256)
- #define IFX_CRYPTO_KEYS_IAK_PUBLIC_KEY_LEN (PSA_KEY_EXPORT_ECC_PUBLIC_KEY_MAX↩ _SIZE(256U))
- #define IFX_CRYPTO_KEYS_IAK_PUBLIC_ALGORITHM (PSA_ALG_ECDSA(PSA_ALG_SHA_256))
- #define IFX_CRYPTO_KEYS_IAK_PUBLIC_TYPE (PSA_KEY_TYPE_ECC_PUBLIC_KEY(PSA_ECC_F↩ AMILY_SECP_R1))

## 7.203.1 Macro Definition Documentation

### 7.203.1.1 IFX_CRYPTO_KEYS_HUK_ALGORITHM

```
#define IFX_CRYPTO_KEYS_HUK_ALGORITHM (PSA_ALG_HKDF(PSA_ALG_SHA_256))
```

Definition at line 28 of file crypto_keys_rram.h.

#### 7.203.1.2 IFX_CRYPTO_KEYS_HUK_KEY_BITS

```
#define IFX_CRYPTO_KEYS_HUK_KEY_BITS (256)
```

Definition at line 26 of file crypto_keys_rram.h.

#### 7.203.1.3 IFX_CRYPTO_KEYS_HUK_KEY_LEN

```
#define IFX_CRYPTO_KEYS_HUK_KEY_LEN (PSA_KEY_EXPORT_ECC_KEY_PAIR_MAX_SIZE(256U))
```

Definition at line 27 of file crypto_keys_rram.h.

#### 7.203.1.4 IFX_CRYPTO_KEYS_HUK_TYPE

```
#define IFX_CRYPTO_KEYS_HUK_TYPE (PSA_KEY_TYPE_DERIVE)
```

Definition at line 29 of file crypto_keys_rram.h.

#### 7.203.1.5 IFX_CRYPTO_KEYS_IAK_PRIVATE_ALGORITHM

```
#define IFX_CRYPTO_KEYS_IAK_PRIVATE_ALGORITHM (PSA_ALG_ECDSA(PSA_ALG_SHA_256))
```

Definition at line 33 of file crypto_keys_rram.h.

#### 7.203.1.6 IFX_CRYPTO_KEYS_IAK_PRIVATE_KEY_BITS

```
#define IFX_CRYPTO_KEYS_IAK_PRIVATE_KEY_BITS (256)
```

Definition at line 31 of file crypto_keys_rram.h.

#### 7.203.1.7 IFX_CRYPTO_KEYS_IAK_PRIVATE_KEY_LEN

```
#define IFX_CRYPTO_KEYS_IAK_PRIVATE_KEY_LEN (PSA_KEY_EXPORT_ECC_KEY_PAIR_MAX_SIZE(256U))
```

Definition at line 32 of file crypto_keys_rram.h.

#### 7.203.1.8 IFX_CRYPTO_KEYS_IAK_PRIVATE_TYPE

```
#define IFX_CRYPTO_KEYS_IAK_PRIVATE_TYPE (PSA_KEY_TYPE_ECC_KEY_PAIR(PSA_ECC_FAMILY_SECP_R1))
```

Definition at line 34 of file crypto_keys_rram.h.

#### 7.203.1.9 IFX_CRYPTO_KEYS_IAK_PUBLIC_ALGORITHM

```
#define IFX_CRYPTO_KEYS_IAK_PUBLIC_ALGORITHM (PSA_ALG_ECDSA(PSA_ALG_SHA_256))
```

Definition at line 38 of file crypto_keys_rram.h.

#### 7.203.1.10 IFX_CRYPTO_KEYS_IAK_PUBLIC_KEY_BITS

```
#define IFX_CRYPTO_KEYS_IAK_PUBLIC_KEY_BITS (256)
```

Definition at line 36 of file crypto_keys_rram.h.

#### 7.203.1.11 IFX_CRYPTO_KEYS_IAK_PUBLIC_KEY_LEN

```
#define IFX_CRYPTO_KEYS_IAK_PUBLIC_KEY_LEN (PSA_KEY_EXPORT_ECC_PUBLIC_KEY_MAX_SIZE(256U))
```

Definition at line 37 of file crypto_keys_rram.h.

#### 7.203.1.12 IFX_CRYPTO_KEYS_IAK_PUBLIC_TYPE

```
#define IFX_CRYPTO_KEYS_IAK_PUBLIC_TYPE (PSA_KEY_TYPE_ECC_PUBLIC_KEY(PSA_ECC_FAMILY_SECP_R1))
```

Definition at line 39 of file crypto_keys_rram.h.

#### 7.203.1.13 IFX_CRYPTO_KEYS_PTR

```
#define IFX_CRYPTO_KEYS_PTR ((ifx_plat_epc2_keys_t *)0x1200132EU)
```

Definition at line 24 of file crypto_keys_rram.h.

## 7.204 platform/ext/target/infineon/pse84/epc2/spe/services/crypto/mbedtls_target_config_pse84.h File Reference

### Macros

- #define MBEDTLS_PSA_ASSUME_EXCLUSIVE_BUFFERS

### 7.204.1 Macro Definition Documentation

#### 7.204.1.1 MBEDTLS_PSA_ASSUME_EXCLUSIVE_BUFFERS

```
#define MBEDTLS_PSA_ASSUME_EXCLUSIVE_BUFFERS
```

Definition at line 33 of file mbedtls_target_config_pse84.h.

## 7.205 platform/ext/target/infineon/pse84/epc2/spe/services/crypto/platform_builtin_key_loader_ids.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define TFM_BUILTIN_MAX_KEY_LEN 96
- #define TFM_BUILTIN_KEY_SLOT_IAK TFM_BUILTIN_KEY_SLOT_ATTEST_PRIV

### Enumerations

- enum psa_drv_slot_number_t { TFM_BUILTIN_KEY_SLOT_HUK = 0, TFM_BUILTIN_KEY_SLOT_ATTEST_PRIV, TFM_BUILTIN_KEY_SLOT_ATTEST_PUB, TFM_BUILTIN_KEY_SLOT_MAX }

### 7.205.1 Macro Definition Documentation

#### 7.205.1.1 TFM_BUILTIN_KEY_SLOT_IAK

```
#define TFM_BUILTIN_KEY_SLOT_IAK TFM_BUILTIN_KEY_SLOT_ATTEST_PRIV
```

Definition at line 25 of file platform_builtin_key_loader_ids.h.

#### 7.205.1.2 TFM_BUILTIN_MAX_KEY_LEN

```
#define TFM_BUILTIN_MAX_KEY_LEN 96
```

Definition at line 16 of file platform_builtin_key_loader_ids.h.

### 7.205.2 Enumeration Type Documentation

#### 7.205.2.1 psa_drv_slot_number_t

```
enum psa_drv_slot_number_t
```

**Enumerator**

| | |
|---|---|
| TFM_BUILTIN_KEY_SLOT_HUK | |
| TFM_BUILTIN_KEY_SLOT_ATTEST_PRIV | |
| TFM_BUILTIN_KEY_SLOT_ATTEST_PUB | |
| TFM_BUILTIN_KEY_SLOT_MAX | |

Definition at line 18 of file platform_builtin_key_loader_ids.h.

## 7.206 platform/ext/target/infineon/pse84/epc2/tests/ifx_isolation_test_data.c File Reference

```
#include <stdint.h>
#include "ifx_test_arot_master_api.h"
#include "ifx_test_prot_master_api.h"
#include "ifx_test_arot_slave_api.h"
#include "ifx_test_prot_slave_api.h"
#include "ifx_test_service_api.h"
#include "ifx_test_common.h"
```

Include dependency graph for ifx_isolation_test_data.c:

### Variables

- const ifx_test_case_t test_cases [ ]
- const uint32_t num_of_cases = sizeof(test_cases) / sizeof(test_cases[0])

### 7.206.1 Variable Documentation

#### 7.206.1.1 num_of_cases

```
const uint32_t num_of_cases = sizeof(test_cases) / sizeof(test_cases[0])
```

Definition at line 44 of file ifx_isolation_test_data.c.

#### 7.206.1.2 test_cases

```
const ifx_test_case_t test_cases[]
```

**Initial value:**

```
= {
    {ifx_access_to_partition_memory,     ifx_arot_slave_get_partition_data, IFX_HAS_ACCESS_FALSE, "NS has no
      access to ARoT\r\n"},
    {ifx_access_to_partition_memory,     ifx_prot_slave_get_partition_data, IFX_HAS_ACCESS_FALSE, "NS has no
      access to PRoT\r\n"},
    {ifx_access_to_partition_memory,     ifx_test_get_spm_assets,           IFX_HAS_ACCESS_FALSE, "NS has no
      access to SPM\r\n"},
}
```

Definition at line 17 of file ifx_isolation_test_data.c.

## 7.207 platform/ext/target/infineon/pse84/epc2/tests/ifx_platform_tests_config.h File Reference

### Macros

- #define IFX_PC_TZ_NSPE_ID 0x02UL /∗ CM33 NSPE Application ∗/
- #define IFX_MAILBOX_ITCM_ADDRESS_1 0x00000010u
- #define IFX_MAILBOX_ITCM_ADDRESS_2 0x48000010u
- #define IFX_MAILBOX_DTCM_ADDRESS_1 0x20000010u
- #define IFX_MAILBOX_DTCM_ADDRESS_2 0x48040010u

### 7.207.1 Macro Definition Documentation

#### 7.207.1.1 IFX_MAILBOX_DTCM_ADDRESS_1

```
#define IFX_MAILBOX_DTCM_ADDRESS_1 0x20000010u
```

Definition at line 21 of file ifx_platform_tests_config.h.

#### 7.207.1.2 IFX_MAILBOX_DTCM_ADDRESS_2

```
#define IFX_MAILBOX_DTCM_ADDRESS_2 0x48040010u
```

Definition at line 24 of file ifx_platform_tests_config.h.

#### 7.207.1.3 IFX_MAILBOX_ITCM_ADDRESS_1

```
#define IFX_MAILBOX_ITCM_ADDRESS_1 0x00000010u
```

Definition at line 15 of file ifx_platform_tests_config.h.

#### 7.207.1.4 IFX_MAILBOX_ITCM_ADDRESS_2

```
#define IFX_MAILBOX_ITCM_ADDRESS_2 0x48000010u
```

Definition at line 18 of file ifx_platform_tests_config.h.

#### 7.207.1.5 IFX_PC_TZ_NSPE_ID

```
#define IFX_PC_TZ_NSPE_ID 0x02UL /* CM33 NSPE Application */
```

Definition at line 12 of file ifx_platform_tests_config.h.

## 7.208 platform/ext/target/infineon/pse84/shared/device/config/device_cfg.h File Reference

Configuration file native driver re-targeting.
This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_IRQ_TEST_TIMER_S

### 7.208.1 Detailed Description

Configuration file native driver re-targeting.
This file can be used to add native driver specific macro definitions to select which peripherals are available in the build.
This is a default device configuration file with all peripherals enabled.

### 7.208.2 Macro Definition Documentation

#### 7.208.2.1 IFX_IRQ_TEST_TIMER_S

```
#define IFX_IRQ_TEST_TIMER_S
```

Definition at line 29 of file device_cfg.h.

## 7.209 platform/ext/target/infineon/pse84/shared/device/include/cmsis.h File Reference

```
#include "cmsis_compiler.h"
#include "cy_device.h"
```

Include dependency graph for cmsis.h:

This graph shows which files directly or indirectly include this file:

## 7.210 platform/ext/target/infineon/pse84/shared/device/include/pse84_core_interrupts.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define IFX_CORE_DEFINE_EXCEPTIONS_LIST
- #define IFX_CORE_DEFINE_INTERRUPTS_LIST
- #define IFX_CORE_EXCEPTIONS_LIST
- #define IFX_CORE_INTERRUPTS_LIST

### 7.210.1 Macro Definition Documentation

#### 7.210.1.1 IFX_CORE_DEFINE_EXCEPTIONS_LIST

```
#define IFX_CORE_DEFINE_EXCEPTIONS_LIST
```

**Value:**

```
    DEFAULT_IRQ_HANDLER(NMI_Handler) \
    DEFAULT_IRQ_HANDLER(HardFault_Handler) \
    DEFAULT_IRQ_HANDLER(MemManage_Handler) \
    DEFAULT_IRQ_HANDLER(BusFault_Handler) \
    DEFAULT_IRQ_HANDLER(UsageFault_Handler) \
    DEFAULT_IRQ_HANDLER(SecureFault_Handler) \
    DEFAULT_IRQ_HANDLER(SVC_Handler) \
    DEFAULT_IRQ_HANDLER(DebugMon_Handler) \
    DEFAULT_IRQ_HANDLER(PendSV_Handler) \
    DEFAULT_IRQ_HANDLER(SysTick_Handler)
```

Definition at line 15 of file pse84_core_interrupts.h.

#### 7.210.1.2 IFX_CORE_DEFINE_INTERRUPTS_LIST

```
#define IFX_CORE_DEFINE_INTERRUPTS_LIST
```

Definition at line 27 of file pse84_core_interrupts.h.

#### 7.210.1.3 IFX_CORE_EXCEPTIONS_LIST

```
#define IFX_CORE_EXCEPTIONS_LIST
```

**Value:**

```
    (VECTOR_TABLE_Type)(&__INITIAL_SP),         /*       Initial Stack Pointer */ \
    Reset_Handler,                              /*   -15 Reset Handler */ \
    NMI_Handler,                                /*   -14 NMI Handler */ \
    HardFault_Handler,                          /*   -13 Hard Fault Handler */ \
    MemManage_Handler,                          /*   -12 MPU Fault Handler */ \
    BusFault_Handler,                           /*   -11 Bus Fault Handler */ \
    UsageFault_Handler,                         /*   -10 Usage Fault Handler */ \
    SecureFault_Handler,                        /*    -9 Secure Fault Handler */ \
    0,                                          /*    -8 Reserved */ \
    0,                                          /*    -7 Reserved */ \
    0,                                          /*    -6 Reserved */ \
    SVC_Handler,                                /*    -5 SVCall Handler */ \
    DebugMon_Handler,                           /*    -4 Debug Monitor Handler */ \
    0,                                          /*    -3 Reserved */ \
    PendSV_Handler,                             /*    -2 PendSV Handler */ \
    SysTick_Handler,                            /*    -1 SysTick Handler */
```

Definition at line 224 of file pse84_core_interrupts.h.

#### 7.210.1.4 IFX_CORE_INTERRUPTS_LIST

```
#define IFX_CORE_INTERRUPTS_LIST
```

196 [Active] cpuss[0] DataWire #1, Channel #15

Definition at line 242 of file pse84_core_interrupts.h.

## 7.211 platform/ext/target/infineon/pse84/shared/partition/flash_layout.h File Reference

```
#include "config_tfm.h"
#include "project_memory_layout.h"
```

Include dependency graph for flash_layout.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define FLASH_LAYOUT_H

### 7.211.1 Macro Definition Documentation

#### 7.211.1.1 FLASH_LAYOUT_H

```
#define FLASH_LAYOUT_H
```

Definition at line 29 of file flash_layout.h.

# 7.212 platform/ext/target/infineon/pse84/shared/partition/pse84_s_linker_alignments.h File Reference

## Macros

- #define IFX_LINKER_RRAM_ALIGNMENT 0x1000 /∗ RRAM has 4 KBs MPC blocks ∗/
- #define IFX_LINKER_SMIF_ALIGNMENT 0x20000 /∗ SMIF has 128 KBs MPC blocks ∗/
- #define IFX_LINKER_SRAM_ALIGNMENT 0x1000 /∗ SRAM has 4 KBs MPC blocks ∗/
- #define IFX_LINKER_SOCMEM_ALIGNMENT 0x2000 /∗ SOCMEM has 8 KBs MPC blocks ∗/
- #define IFX_LINKER_CODE_ALIGNMENT IFX_LINKER_SRAM_ALIGNMENT
- #define TFM_LINKER_VENEERS_ALIGNMENT IFX_LINKER_CODE_ALIGNMENT
- #define TFM_LINKER_UNPRIV_CODE_ALIGNMENT IFX_LINKER_CODE_ALIGNMENT
- #define TFM_LINKER_NS_AGENT_TZ_CODE_ALIGNMENT IFX_LINKER_CODE_ALIGNMENT
- #define IFX_LINKER_DATA_ALIGNMENT IFX_LINKER_SRAM_ALIGNMENT
- #define TFM_LINKER_APP_ROT_LINKER_DATA_ALIGNMENT IFX_LINKER_DATA_ALIGNMENT
- #define TFM_LINKER_PSA_ROT_LINKER_DATA_ALIGNMENT IFX_LINKER_DATA_ALIGNMENT
- #define TFM_LINKER_SP_META_PTR_ALIGNMENT IFX_LINKER_DATA_ALIGNMENT
- #define TFM_LINKER_IFX_CODE_COVERAGE_ALIGNMENT IFX_LINKER_DATA_ALIGNMENT

### 7.212.1 Macro Definition Documentation

#### 7.212.1.1 IFX_LINKER_CODE_ALIGNMENT

```
#define IFX_LINKER_CODE_ALIGNMENT IFX_LINKER_SRAM_ALIGNMENT
```

Definition at line 22 of file pse84_s_linker_alignments.h.

#### 7.212.1.2 IFX_LINKER_DATA_ALIGNMENT

```
#define IFX_LINKER_DATA_ALIGNMENT IFX_LINKER_SRAM_ALIGNMENT
```

Definition at line 42 of file pse84_s_linker_alignments.h.

#### 7.212.1.3 IFX_LINKER_RRAM_ALIGNMENT

```
#define IFX_LINKER_RRAM_ALIGNMENT 0x1000 /* RRAM has 4 KBs MPC blocks */
```

Definition at line 14 of file pse84_s_linker_alignments.h.

#### 7.212.1.4 IFX_LINKER_SMIF_ALIGNMENT

```
#define IFX_LINKER_SMIF_ALIGNMENT 0x20000 /* SMIF has 128 KBs MPC blocks */
```

Definition at line 15 of file pse84_s_linker_alignments.h.

#### 7.212.1.5 IFX_LINKER_SOCMEM_ALIGNMENT

```
#define IFX_LINKER_SOCMEM_ALIGNMENT 0x2000 /* SOCMEM has 8 KBs MPC blocks */
```

Definition at line 17 of file pse84_s_linker_alignments.h.

#### 7.212.1.6 IFX_LINKER_SRAM_ALIGNMENT

```
#define IFX_LINKER_SRAM_ALIGNMENT 0x1000 /* SRAM has 4 KBs MPC blocks */
```

Definition at line 16 of file pse84_s_linker_alignments.h.

#### 7.212.1.7 TFM_LINKER_APP_ROT_LINKER_DATA_ALIGNMENT

```
#define TFM_LINKER_APP_ROT_LINKER_DATA_ALIGNMENT IFX_LINKER_DATA_ALIGNMENT
```

Definition at line 49 of file pse84_s_linker_alignments.h.

#### 7.212.1.8 TFM_LINKER_IFX_CODE_COVERAGE_ALIGNMENT

```
#define TFM_LINKER_IFX_CODE_COVERAGE_ALIGNMENT IFX_LINKER_DATA_ALIGNMENT
```

Definition at line 56 of file pse84_s_linker_alignments.h.

#### 7.212.1.9 TFM_LINKER_NS_AGENT_TZ_CODE_ALIGNMENT

```
#define TFM_LINKER_NS_AGENT_TZ_CODE_ALIGNMENT IFX_LINKER_CODE_ALIGNMENT
```

Definition at line 37 of file pse84_s_linker_alignments.h.

#### 7.212.1.10 TFM_LINKER_PSA_ROT_LINKER_DATA_ALIGNMENT

```
#define TFM_LINKER_PSA_ROT_LINKER_DATA_ALIGNMENT IFX_LINKER_DATA_ALIGNMENT
```

Definition at line 51 of file pse84_s_linker_alignments.h.

#### 7.212.1.11 TFM_LINKER_SP_META_PTR_ALIGNMENT

```
#define TFM_LINKER_SP_META_PTR_ALIGNMENT IFX_LINKER_DATA_ALIGNMENT
```

Definition at line 53 of file pse84_s_linker_alignments.h.

#### 7.212.1.12 TFM_LINKER_UNPRIV_CODE_ALIGNMENT

```
#define TFM_LINKER_UNPRIV_CODE_ALIGNMENT IFX_LINKER_CODE_ALIGNMENT
```

Definition at line 35 of file pse84_s_linker_alignments.h.

#### 7.212.1.13 TFM_LINKER_VENEERS_ALIGNMENT

```
#define TFM_LINKER_VENEERS_ALIGNMENT IFX_LINKER_CODE_ALIGNMENT
```

Definition at line 33 of file pse84_s_linker_alignments.h.

## 7.213 platform/ext/target/infineon/pse84/shared/partition/region_defs.h File Reference

```
#include "flash_layout.h"
```

Include dependency graph for region_defs.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define S_CODE_VECTOR_TABLE_SIZE (0x354)
- #define S_MSP_STACK_SIZE (0x0002000)
- #define IFX_CM33_NS_MSP_STACK_SIZE (0x0001000)
- #define IFX_CM55_NS_MSP_STACK_SIZE (0x0001000)
- #define S_CODE_START (IFX_TFM_IMAGE_EXECUTE_ADDR + IFX_UNSIGNED_TO_VALUE(BL2_H↩EADER_SIZE))
- #define S_CODE_SIZE
- #define S_CODE_LIMIT (S_CODE_START + S_CODE_SIZE - 1)
- #define S_DATA_START (IFX_TFM_DATA_ADDR)
- #define S_DATA_SIZE (IFX_TFM_DATA_SIZE)
- #define S_DATA_LIMIT (S_DATA_START + S_DATA_SIZE - 1)
- #define IFX_OFF_CORE_NSPE_BOOT_ADDR IFX_CM55_NS_CODE_START
- #define BOOT_TFM_SHARED_DATA_BASE IFX_TFM_BOOT_SHARED_DATA_ADDR
- #define BOOT_TFM_SHARED_DATA_SIZE IFX_TFM_BOOT_SHARED_DATA_SIZE
- #define BOOT_TFM_SHARED_DATA_LIMIT

### 7.213.1 Macro Definition Documentation

#### 7.213.1.1 BOOT_TFM_SHARED_DATA_BASE

```
#define BOOT_TFM_SHARED_DATA_BASE IFX_TFM_BOOT_SHARED_DATA_ADDR
```

Definition at line 114 of file region_defs.h.

#### 7.213.1.2 BOOT_TFM_SHARED_DATA_LIMIT

```
#define BOOT_TFM_SHARED_DATA_LIMIT
```

**Value:**

```
(BOOT_TFM_SHARED_DATA_BASE + \
BOOT_TFM_SHARED_DATA_SIZE - 1)
```

Definition at line 116 of file region_defs.h.

#### 7.213.1.3 BOOT_TFM_SHARED_DATA_SIZE

```
#define BOOT_TFM_SHARED_DATA_SIZE IFX_TFM_BOOT_SHARED_DATA_SIZE
```

Definition at line 115 of file region_defs.h.

#### 7.213.1.4 IFX_CM33_NS_MSP_STACK_SIZE

```
#define IFX_CM33_NS_MSP_STACK_SIZE (0x0001000)
```

Definition at line 30 of file region_defs.h.

#### 7.213.1.5 IFX_CM55_NS_MSP_STACK_SIZE

```
#define IFX_CM55_NS_MSP_STACK_SIZE (0x0001000)
```

Definition at line 31 of file region_defs.h.

#### 7.213.1.6 IFX_OFF_CORE_NSPE_BOOT_ADDR

```
#define IFX_OFF_CORE_NSPE_BOOT_ADDR IFX_CM55_NS_CODE_START
```

Definition at line 111 of file region_defs.h.

#### 7.213.1.7 S_CODE_LIMIT

```
#define S_CODE_LIMIT (S_CODE_START + S_CODE_SIZE - 1)
```

Definition at line 49 of file region_defs.h.

#### 7.213.1.8 S_CODE_SIZE

```
#define S_CODE_SIZE
```

**Value:**

```
(IFX_TFM_IMAGE_EXECUTE_SIZE -
IFX_UNSIGNED_TO_VALUE(BL2_HEADER_SIZE) - \
IFX_UNSIGNED_TO_VALUE(BL2_TRAILER_SIZE))
```

Definition at line 47 of file region_defs.h.

#### 7.213.1.9 S_CODE_START

```
#define S_CODE_START (IFX_TFM_IMAGE_EXECUTE_ADDR + IFX_UNSIGNED_TO_VALUE(BL2_HEADER_SIZE))
```

Definition at line 46 of file region_defs.h.

#### 7.213.1.10 S_CODE_VECTOR_TABLE_SIZE

```
#define S_CODE_VECTOR_TABLE_SIZE (0x354)
```

Definition at line 24 of file region_defs.h.

#### 7.213.1.11 S_DATA_LIMIT

```
#define S_DATA_LIMIT (S_DATA_START + S_DATA_SIZE - 1)
```

Definition at line 53 of file region_defs.h.

#### 7.213.1.12 S_DATA_SIZE

```
#define S_DATA_SIZE (IFX_TFM_DATA_SIZE)
```

Definition at line 52 of file region_defs.h.

#### 7.213.1.13 S_DATA_START

```
#define S_DATA_START (IFX_TFM_DATA_ADDR)
```

Definition at line 51 of file region_defs.h.

#### 7.213.1.14 S_MSP_STACK_SIZE

```
#define S_MSP_STACK_SIZE (0x0002000)
```

Definition at line 26 of file region_defs.h.

## 7.214 platform/ext/target/infineon/pse84/shared/platform_nv_counters_↩ ids.h File Reference

```
#include <stdint.h>
```

Include dependency graph for platform_nv_counters_ids.h:

## Enumerations

- enum tfm_nv_counter_t {
  PLAT_NV_COUNTER_PS_0 = 0, PLAT_NV_COUNTER_PS_1, PLAT_NV_COUNTER_PS_2, PLAT_NV_COUNTER_NS_0, PLAT_NV_COUNTER_NS_1, PLAT_NV_COUNTER_NS_2, PLAT_NV_COUNTER_MAX, PLAT_NV_COUNTER_BOUNDAR
  = UINT32_MAX }

### 7.214.1 Enumeration Type Documentation

#### 7.214.1.1 tfm_nv_counter_t

```
enum tfm_nv_counter_t
```

**Enumerator**

| | |
|---|---|
| PLAT_NV_COUNTER_PS_0 | |
| PLAT_NV_COUNTER_PS_1 | |
| PLAT_NV_COUNTER_PS_2 | |
| PLAT_NV_COUNTER_NS_0 | |
| PLAT_NV_COUNTER_NS_1 | |
| PLAT_NV_COUNTER_NS_2 | |
| PLAT_NV_COUNTER_MAX | |
| PLAT_NV_COUNTER_BOUNDARY | |

Definition at line 19 of file platform_nv_counters_ids.h.

## 7.215 platform/ext/target/infineon/pse84/shared/tfm_peripherals_def.h File Reference

```
#include "cycfg.h"
#include "ifx_peripherals_def.h"
```

Include dependency graph for tfm_peripherals_def.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define DEFAULT_IRQ_PRIORITY (1UL << (__NVIC_PRIO_BITS - 2))
- #define IFX_IPC_NS_TO_TFM_CHAN (9U)
- #define IFX_IPC_NS_TO_TFM_INTR_MASK (1UL << IFX_IPC_NS_TO_TFM_CHAN)
- #define IFX_IPC_NS_TO_TFM_INTR_STRUCT (6U)
- #define IFX_IPC_NS_TO_TFM_NOTIFY_MASK (1UL << IFX_IPC_NS_TO_TFM_INTR_STRUCT)
- #define IFX_IPC_NS_TO_TFM_IPC_INTR m33syscpuss_interrupts_ipc_dpslp_6_IRQn
- #define IFX_IPC_TFM_TO_NS_CHAN (8U)
- #define IFX_IPC_TFM_TO_NS_INTR_MASK (1UL << IFX_IPC_TFM_TO_NS_CHAN)
- #define IFX_IPC_TFM_TO_NS_INTR_STRUCT (7U)
- #define IFX_IPC_TFM_TO_NS_NOTIFY_MASK (1UL << IFX_IPC_TFM_TO_NS_INTR_STRUCT)
- #define IFX_IPC_TFM_TO_NS_IPC_INTR m33syscpuss_interrupts_ipc_dpslp_7_IRQn
- #define IFX_IPC_MAILBOX_LOCK_CHAN (10U)
- #define IFX_IPC_TFM_TO_NS_IRQ_PRIORITY 3U
- #define IFX_IPC_NS_TO_TFM_IRQ_PRIORITY 3U
- #define MAILBOX_IRQ IFX_IPC_TFM_TO_NS_IPC_INTR
- #define IFX_TEST_PERIPHERAL_1 PROT_PERI0_GPIO_PRT0_PRT
- #define IFX_TEST_PERIPHERAL_1_BASE GPIO_PRT0
- #define IFX_TEST_PERIPHERAL_1_SIZE 0x40U
- #define IFX_TEST_PERIPHERAL_2 PROT_PERI0_GPIO_PRT1_PRT
- #define IFX_TEST_PERIPHERAL_2_BASE GPIO_PRT1
- #define IFX_TEST_PERIPHERAL_2_SIZE 0x40U

## 7.215.1 Macro Definition Documentation

### 7.215.1.1 DEFAULT_IRQ_PRIORITY

```
#define DEFAULT_IRQ_PRIORITY (1UL << (__NVIC_PRIO_BITS - 2))
```

Definition at line 45 of file tfm_peripherals_def.h.

### 7.215.1.2 IFX_IPC_MAILBOX_LOCK_CHAN

```
#define IFX_IPC_MAILBOX_LOCK_CHAN (10U)
```

Definition at line 64 of file tfm_peripherals_def.h.

### 7.215.1.3 IFX_IPC_NS_TO_TFM_CHAN

```
#define IFX_IPC_NS_TO_TFM_CHAN (9U)
```

Definition at line 48 of file tfm_peripherals_def.h.

### 7.215.1.4 IFX_IPC_NS_TO_TFM_INTR_MASK

```
#define IFX_IPC_NS_TO_TFM_INTR_MASK (1UL << IFX_IPC_NS_TO_TFM_CHAN)
```

Definition at line 49 of file tfm_peripherals_def.h.

#### 7.215.1.5 IFX_IPC_NS_TO_TFM_INTR_STRUCT

```
#define IFX_IPC_NS_TO_TFM_INTR_STRUCT (6U)
```

Definition at line 51 of file tfm_peripherals_def.h.

#### 7.215.1.6 IFX_IPC_NS_TO_TFM_IPC_INTR

```
#define IFX_IPC_NS_TO_TFM_IPC_INTR m33syscpuss_interrupts_ipc_dpslp_6_IRQn
```

Definition at line 53 of file tfm_peripherals_def.h.

#### 7.215.1.7 IFX_IPC_NS_TO_TFM_IRQ_PRIORITY

```
#define IFX_IPC_NS_TO_TFM_IRQ_PRIORITY 3U
```

Definition at line 71 of file tfm_peripherals_def.h.

#### 7.215.1.8 IFX_IPC_NS_TO_TFM_NOTIFY_MASK

```
#define IFX_IPC_NS_TO_TFM_NOTIFY_MASK (1UL << IFX_IPC_NS_TO_TFM_INTR_STRUCT)
```

Definition at line 52 of file tfm_peripherals_def.h.

#### 7.215.1.9 IFX_IPC_TFM_TO_NS_CHAN

```
#define IFX_IPC_TFM_TO_NS_CHAN (8U)
```

Definition at line 56 of file tfm_peripherals_def.h.

#### 7.215.1.10 IFX_IPC_TFM_TO_NS_INTR_MASK

```
#define IFX_IPC_TFM_TO_NS_INTR_MASK (1UL << IFX_IPC_TFM_TO_NS_CHAN)
```

Definition at line 57 of file tfm_peripherals_def.h.

#### 7.215.1.11 IFX_IPC_TFM_TO_NS_INTR_STRUCT

```
#define IFX_IPC_TFM_TO_NS_INTR_STRUCT (7U)
```

Definition at line 59 of file tfm_peripherals_def.h.

#### 7.215.1.12 IFX_IPC_TFM_TO_NS_IPC_INTR

```
#define IFX_IPC_TFM_TO_NS_IPC_INTR m33syscpuss_interrupts_ipc_dpslp_7_IRQn
```

Definition at line 61 of file tfm_peripherals_def.h.

#### 7.215.1.13 IFX_IPC_TFM_TO_NS_IRQ_PRIORITY

```
#define IFX_IPC_TFM_TO_NS_IRQ_PRIORITY 3U
```

Definition at line 68 of file tfm_peripherals_def.h.

#### 7.215.1.14 IFX_IPC_TFM_TO_NS_NOTIFY_MASK

```
#define IFX_IPC_TFM_TO_NS_NOTIFY_MASK (1UL << IFX_IPC_TFM_TO_NS_INTR_STRUCT)
```

Definition at line 60 of file tfm_peripherals_def.h.

#### 7.215.1.15 IFX_TEST_PERIPHERAL_1

```
#define IFX_TEST_PERIPHERAL_1 PROT_PERI0_GPIO_PRT0_PRT
```

Definition at line 118 of file tfm_peripherals_def.h.

#### 7.215.1.16 IFX_TEST_PERIPHERAL_1_BASE

```
#define IFX_TEST_PERIPHERAL_1_BASE GPIO_PRT0
```

Definition at line 119 of file tfm_peripherals_def.h.

#### 7.215.1.17 IFX_TEST_PERIPHERAL_1_SIZE

```
#define IFX_TEST_PERIPHERAL_1_SIZE 0x40U
```

Definition at line 120 of file tfm_peripherals_def.h.

#### 7.215.1.18 IFX_TEST_PERIPHERAL_2

```
#define IFX_TEST_PERIPHERAL_2 PROT_PERI0_GPIO_PRT1_PRT
```

Definition at line 123 of file tfm_peripherals_def.h.

#### 7.215.1.19 IFX_TEST_PERIPHERAL_2_BASE

```
#define IFX_TEST_PERIPHERAL_2_BASE GPIO_PRT1
```

Definition at line 124 of file tfm_peripherals_def.h.

#### 7.215.1.20 IFX_TEST_PERIPHERAL_2_SIZE

```
#define IFX_TEST_PERIPHERAL_2_SIZE 0x40U
```

Definition at line 125 of file tfm_peripherals_def.h.

#### 7.215.1.21 MAILBOX_IRQ

```
#define MAILBOX_IRQ IFX_IPC_TFM_TO_NS_IPC_INTR
```

Definition at line 79 of file tfm_peripherals_def.h.

## 7.216 platform/ext/target/infineon/pse84/spe/platform_otp_ids.h File Reference

```
#include <stdint.h>
```

Include dependency graph for platform_otp_ids.h:

## Enumerations

- enum tfm_otp_element_id_t { PLAT_OTP_ID_LCS = 1, PLAT_OTP_ID_MAX = UINT32_MAX }

### 7.216.1 Enumeration Type Documentation

#### 7.216.1.1 tfm_otp_element_id_t

```
enum tfm_otp_element_id_t
```

**Enumerator**

| | |
|---|---|
| PLAT_OTP_ID_LCS | |
| PLAT_OTP_ID_MAX | |

Definition at line 20 of file platform_otp_ids.h.

## 7.217 platform/ext/target/infineon/pse84/spe/protection/platform_partition_assets.h File Reference

```
#include <stddef.h>
#include "array.h"
#include "load/asset_defs.h"
#include "region_defs.h"
```

Include dependency graph for platform_partition_assets.h:

## 7.218 platform/ext/target/infineon/pse84/spe/protection/protection_data.c File Reference

```
#include "config_impl.h"
#include "config_tfm.h"
#include "array.h"
#include "cy_device.h"
#include "project_memory_layout.h"
#include "protection_regions_cfg.h"
#include "protection_types.h"
```

Include dependency graph for protection_data.c:

### Functions

- FIH_RET_TYPE (enum tfm_hal_status_t) ifx_domain_mem_get_asset(const ifx_mem_domain_region_cfg_t *p_region
- if ((void *) p_region->base==(void *) RRAMC0_MPC0)
- FIH_RET (fih_int_encode(TFM_HAL_SUCCESS))

### Variables

- const ifx_memory_config_t *const ifx_memory_cm33_config [ ]
- const size_t ifx_memory_cm33_config_count = ARRAY_SIZE(ifx_memory_cm33_config)
- const ifx_memory_config_t *const ifx_memory_cm55_config [ ]
- const size_t ifx_memory_cm55_config_count = ARRAY_SIZE(ifx_memory_cm55_config)
- struct asset_desc_t * p_asset
- struct asset_desc_t bool * valid
- else
- p_asset mem start = mem_base + p_region->offset
- p_asset mem limit = p_asset->mem.start + p_region->size
- p_asset attr = ASSET_ATTR_NUMBERED_MMIO | ASSET_ATTR_READ_WRITE

## 7.218.1 Function Documentation

### 7.218.1.1 FIH_RET()

```
FIH_RET (
            fih_int_encode(TFM_HAL_SUCCESS)  )
```

### 7.218.1.2 FIH_RET_TYPE()

```
FIH_RET_TYPE (
            enum tfm_hal_status_t ) const
```

### 7.218.1.3 if()

```
else if (
            (void *) p_region-> base = = (void*)RRAMC0_MPC0 )
```

Definition at line 175 of file protection_data.c.

## 7.218.2 Variable Documentation

### 7.218.2.1 attr

```
p_asset attr = ASSET_ATTR_NUMBERED_MMIO | ASSET_ATTR_READ_WRITE
```

Definition at line 206 of file protection_data.c.

### 7.218.2.2 else

```
else
```

**Initial value:**

```
{
        FIH_RET(fih_int_encode(TFM_HAL_ERROR_INVALID_INPUT))
```

Definition at line 199 of file protection_data.c.

### 7.218.2.3 ifx_memory_cm33_config

```
const ifx_memory_config_t* const ifx_memory_cm33_config[]
```

**Initial value:**

```
= {
    &ifx_rram0_sbus_config,
    &ifx_sram0_sbus_config,
    &ifx_sram1_sbus_config,
    &ifx_socmem_sbus_config,
    &ifx_rram0_cbus_config,
    &ifx_sram0_cbus_config,
    &ifx_sram1_cbus_config,
    &ifx_socmem_cbus_config,
}
```

Definition at line 130 of file protection_data.c.

### 7.218.2.4 ifx_memory_cm33_config_count

```
const size_t ifx_memory_cm33_config_count = ARRAY_SIZE(ifx_memory_cm33_config)
```

Definition at line 154 of file protection_data.c.

#### 7.218.2.5 ifx_memory_cm55_config

```
const ifx_memory_config_t* const ifx_memory_cm55_config[]
```

**Initial value:**

```
= {
}
```

Definition at line 156 of file protection_data.c.

#### 7.218.2.6 ifx_memory_cm55_config_count

```
const size_t ifx_memory_cm55_config_count = ARRAY_SIZE(ifx_memory_cm55_config)
```

Definition at line 166 of file protection_data.c.

#### 7.218.2.7 limit

```
p_asset mem limit = p_asset->mem.start + p_region->size
```

Definition at line 205 of file protection_data.c.

#### 7.218.2.8 p_asset

```
struct asset_desc_t* p_asset
```

Definition at line 169 of file protection_data.c.

#### 7.218.2.9 start

```
p_asset mem start = mem_base + p_region->offset
```

Definition at line 204 of file protection_data.c.

#### 7.218.2.10 valid

```
* valid
```

**Initial value:**

```
{
    uint32_t mem_base = 0UL
```

Definition at line 171 of file protection_data.c.

## 7.219 platform/ext/target/infineon/pse84/spe/provisioning/ifx_platform_provisioning.c File Reference

```
#include "cmsis.h"
#include "provisioning.h"
#include "cy_device.h"
```

Include dependency graph for ifx_platform_provisioning.c:

## Data Structures

- struct ifx_debug_policy_t
- struct ifx_oem_policy_t

## Macros

- #define IFX_DEBUG_POLICY_CM33_AP_CTL_Pos (0UL)
- #define IFX_DEBUG_POLICY_CM33_AP_CTL_Msk (0x007UL)
- #define IFX_DEBUG_POLICY_CM55_AP_CTL_Pos (3UL)
- #define IFX_DEBUG_POLICY_CM55_AP_CTL_Msk (0x038UL)
- #define IFX_DEBUG_POLICY_SYS_AP_CTL_Pos (6UL)
- #define IFX_DEBUG_POLICY_SYS_AP_CTL_Msk (0x1C0UL)
- #define IFX_DEBUG_POLICY_AP_DISABLE (0U)
- #define IFX_DEBUG_POLICY_AP_ENABLE (1U)
- #define IFX_DEBUG_POLICY_AP_ALLOW_FW (2U)
- #define IFX_DEBUG_POLICY_AP_ALLOW_CERT (3U)
- #define IFX_DEBUG_POLICY_AP_ALLOW_OPEN (4U)
- #define IFX_OEM_POLICY_PUB_KEY_SIZE (65U)
- #define IFX_OEM_POLICY_ADDR (0x12001100u)

## Functions

- ifx_dap_state_t ifx_get_dap_state (void)

  *Retrieve the state of debug access port.*

## 7.219.1 Macro Definition Documentation

### 7.219.1.1 IFX_DEBUG_POLICY_AP_ALLOW_CERT

```
#define IFX_DEBUG_POLICY_AP_ALLOW_CERT (3U)
```

Definition at line 25 of file ifx_platform_provisioning.c.

### 7.219.1.2 IFX_DEBUG_POLICY_AP_ALLOW_FW

```
#define IFX_DEBUG_POLICY_AP_ALLOW_FW (2U)
```

Definition at line 24 of file ifx_platform_provisioning.c.

#### 7.219.1.3 IFX_DEBUG_POLICY_AP_ALLOW_OPEN

```
#define IFX_DEBUG_POLICY_AP_ALLOW_OPEN (4U)
```

Definition at line 26 of file ifx_platform_provisioning.c.

#### 7.219.1.4 IFX_DEBUG_POLICY_AP_DISABLE

```
#define IFX_DEBUG_POLICY_AP_DISABLE (0U)
```

Definition at line 22 of file ifx_platform_provisioning.c.

#### 7.219.1.5 IFX_DEBUG_POLICY_AP_ENABLE

```
#define IFX_DEBUG_POLICY_AP_ENABLE (1U)
```

Definition at line 23 of file ifx_platform_provisioning.c.

#### 7.219.1.6 IFX_DEBUG_POLICY_CM33_AP_CTL_Msk

```
#define IFX_DEBUG_POLICY_CM33_AP_CTL_Msk (0x007UL)
```

Definition at line 15 of file ifx_platform_provisioning.c.

#### 7.219.1.7 IFX_DEBUG_POLICY_CM33_AP_CTL_Pos

```
#define IFX_DEBUG_POLICY_CM33_AP_CTL_Pos (0UL)
```

Definition at line 14 of file ifx_platform_provisioning.c.

#### 7.219.1.8 IFX_DEBUG_POLICY_CM55_AP_CTL_Msk

```
#define IFX_DEBUG_POLICY_CM55_AP_CTL_Msk (0x038UL)
```

Definition at line 17 of file ifx_platform_provisioning.c.

#### 7.219.1.9 IFX_DEBUG_POLICY_CM55_AP_CTL_Pos

```
#define IFX_DEBUG_POLICY_CM55_AP_CTL_Pos (3UL)
```

Definition at line 16 of file ifx_platform_provisioning.c.

#### 7.219.1.10 IFX_DEBUG_POLICY_SYS_AP_CTL_Msk

```
#define IFX_DEBUG_POLICY_SYS_AP_CTL_Msk (0x1C0UL)
```

Definition at line 19 of file ifx_platform_provisioning.c.

#### 7.219.1.11 IFX_DEBUG_POLICY_SYS_AP_CTL_Pos

```
#define IFX_DEBUG_POLICY_SYS_AP_CTL_Pos (6UL)
```

Definition at line 18 of file ifx_platform_provisioning.c.

#### 7.219.1.12 IFX_OEM_POLICY_ADDR

```
#define IFX_OEM_POLICY_ADDR (0x12001100u)
```

Definition at line 36 of file ifx_platform_provisioning.c.

#### 7.219.1.13 IFX_OEM_POLICY_PUB_KEY_SIZE

```
#define IFX_OEM_POLICY_PUB_KEY_SIZE (65U)
```

Definition at line 33 of file ifx_platform_provisioning.c.

### 7.219.2 Function Documentation

#### 7.219.2.1 ifx_get_dap_state()

```
ifx_dap_state_t ifx_get_dap_state (
            void  )
```

Retrieve the state of debug access port.

**Returns**

debug access port state, see ifx_dap_state_t

Definition at line 63 of file ifx_platform_provisioning.c.
Here is the caller graph for this function:

## 7.220 platform/ext/target/infineon/pse84/tests/secure/s_test.c File Reference

```
#include "extra_s_tests.h"
```

Include dependency graph for s_test.c:

### Functions

- void register_testsuite_extra_s_interface (struct test_suite_t *p_test_suite)

### 7.220.1 Function Documentation

#### 7.220.1.1 register_testsuite_extra_s_interface()

```
void register_testsuite_extra_s_interface (
            struct test_suite_t * p_test_suite )
```

Definition at line 22 of file s_test.c.

## 7.221 secure_fw/include/array.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define ARRAY_SIZE(arr) (sizeof(arr)/sizeof(arr[0]))
- #define IOVEC_LEN(x) (uint32_t)ARRAY_SIZE(x)

### 7.221.1 Macro Definition Documentation

#### 7.221.1.1 ARRAY_SIZE

```
#define ARRAY_SIZE(
            arr ) (sizeof(arr)/sizeof(arr[0]))
```

Definition at line 16 of file array.h.

#### 7.221.1.2 IOVEC_LEN

```
#define IOVEC_LEN(
            x ) (uint32_t)ARRAY_SIZE(x)
```

Definition at line 20 of file array.h.

## 7.222 secure_fw/include/async.h File Reference

```
#include "psa/error.h"
```

Include dependency graph for async.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define ASYNC_MSG_REPLY (0x00000004u)

### 7.222.1 Macro Definition Documentation

#### 7.222.1.1 ASYNC_MSG_REPLY

```
#define ASYNC_MSG_REPLY (0x00000004u)
```

The signal number for the Secure Partition message acknowledgment.
Definition at line 17 of file async.h.

## 7.223 secure_fw/include/build_config_check.h File Reference

```
#include "load/spm_load_api.h"
#include "thread.h"
```

Include dependency graph for build_config_check.h:

This graph shows which files directly or indirectly include this file:

## 7.224 secure_fw/include/compiler_ext_defs.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define SYNTAX_UNIFIED ".syntax unified \n"

### 7.224.1 Macro Definition Documentation

#### 7.224.1.1 SYNTAX_UNIFIED

```
#define SYNTAX_UNIFIED ".syntax unified \n"
```

Definition at line 32 of file compiler_ext_defs.h.

## 7.225 secure_fw/include/config_tfm.h File Reference

```
#include "config_base.h"
```

Include dependency graph for config_tfm.h:

## 7.226 secure_fw/include/crt_impl_private.h File Reference

```
#include <stddef.h>
#include <stdint.h>
```

Include dependency graph for crt_impl_private.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- union composite_addr_t

### Macros

- #define ADDR_WORD_UNALIGNED(x) ((x) & 0x3)

### 7.226.1 Macro Definition Documentation

#### 7.226.1.1 ADDR_WORD_UNALIGNED

```
#define ADDR_WORD_UNALIGNED(
            x ) ((x) & 0x3)
```

Definition at line 14 of file crt_impl_private.h.

## 7.227 secure_fw/include/security_defs.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define STACK_SEAL_PATTERN 0xFEF5EDA5

### 7.227.1 Macro Definition Documentation

#### 7.227.1.1 STACK_SEAL_PATTERN

```
#define STACK_SEAL_PATTERN 0xFEF5EDA5
```

Definition at line 17 of file security_defs.h.

## 7.228 secure_fw/partitions/crypto/config_crypto_check.h File Reference

```
#include "config_tfm.h"
```

Include dependency graph for config_crypto_check.h:

This graph shows which files directly or indirectly include this file:

## 7.229 secure_fw/partitions/crypto/config_engine_buf.h File Reference

This graph shows which files directly or indirectly include this file:

## 7.230 secure_fw/partitions/crypto/crypto_aead.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "config_tfm.h"
#include "static_checks.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_key.h"
#include "tfm_crypto_defs.h"
#include "crypto_library.h"
```

Include dependency graph for crypto_aead.c:

### Functions

- psa_status_t tfm_crypto_aead_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the AEAD module.*

## 7.231 secure_fw/partitions/crypto/crypto_alloc.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <string.h>
```

```
#include "config_tfm.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_defs.h"
```

Include dependency graph for crypto_alloc.c:

## Data Structures

- struct tfm_crypto_operation_s

  *A type describing the context stored in Secure memory by the TF-M Crypto service to support multipart calls on secure side.*

## Macros

- #define TFM_CRYPTO_INVALID_HANDLE (0x0u)

  *This value is used to mark an handle for multipart operations as invalid.*

## Enumerations

- enum { TFM_CRYPTO_NOT_IN_USE = 0, TFM_CRYPTO_IN_USE = 1 }

  *Define miscellaneous literal constants that are used in the service.*

## Functions

- psa_status_t tfm_crypto_init_alloc (void)

  *Initialise the Alloc module.*
- psa_status_t tfm_crypto_operation_alloc (enum tfm_crypto_operation_type type, uint32_t *handle, void **ctx)

  *Allocate an operation context in the backend.*
- psa_status_t tfm_crypto_operation_release (uint32_t *handle)

  *Release an operation context in the backend.*
- psa_status_t tfm_crypto_operation_lookup (enum tfm_crypto_operation_type type, uint32_t handle, void **ctx)

  *Look up an operation context in the backend for the corresponding frontend operation.*

## 7.231.1 Macro Definition Documentation

### 7.231.1.1 TFM_CRYPTO_INVALID_HANDLE

```
#define TFM_CRYPTO_INVALID_HANDLE (0x0u)
```

This value is used to mark an handle for multipart operations as invalid.
Definition at line 29 of file crypto_alloc.c.

### 7.231.2 Enumeration Type Documentation

#### 7.231.2.1 anonymous enum

`anonymous enum`
Define miscellaneous literal constants that are used in the service.

**Enumerator**

| | |
|---|---|
| TFM_CRYPTO_NOT_IN_USE | |
| TFM_CRYPTO_IN_USE | |

Definition at line 21 of file crypto_alloc.c.

## 7.232 secure_fw/partitions/crypto/crypto_asymmetric.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "config_tfm.h"
#include "static_checks.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_key.h"
#include "tfm_crypto_defs.h"
#include "crypto_library.h"
```

Include dependency graph for crypto_asymmetric.c:

### Functions

- psa_status_t tfm_crypto_asymmetric_sign_interface (psa_invec in_vec[], psa_outvec out_vec[], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Asymmetric signing module.*

- psa_status_t tfm_crypto_asymmetric_encrypt_interface (psa_invec in_vec[], psa_outvec out_vec[], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Asymmetric encryption module.*

## 7.233 secure_fw/partitions/crypto/crypto_check_config.h File Reference

```
#include "config_tfm.h"
```

Include dependency graph for crypto_check_config.h:

This graph shows which files directly or indirectly include this file:

## 7.234 secure_fw/partitions/crypto/crypto_cipher.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "config_tfm.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_key.h"
#include "tfm_crypto_defs.h"
#include "crypto_library.h"
```

Include dependency graph for crypto_cipher.c:

## Functions

- psa_status_t tfm_crypto_cipher_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Cipher module.*

## 7.235 secure_fw/partitions/crypto/crypto_hash.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "config_tfm.h"
#include "static_checks.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_defs.h"
```

Include dependency graph for crypto_hash.c:

## Functions

- psa_status_t tfm_crypto_hash_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ])

  *This function acts as interface for the Hash module.*

## 7.236 secure_fw/partitions/crypto/crypto_init.c File Reference

```
#include <stdbool.h>
#include "config_tfm.h"
#include "config_crypto_check.h"
```

```
#include "static_checks.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_key.h"
#include "tfm_crypto_defs.h"
#include "tfm_sp_log.h"
#include "crypto_check_config.h"
#include "tfm_plat_crypto_keys.h"
#include "crypto_library.h"
#include <string.h>
#include "psa/framework_feature.h"
#include "psa/service.h"
#include "psa_manifest/tfm_crypto.h"
```

Include dependency graph for crypto_init.c:

## Data Structures

- struct **tfm_crypto_scratch**

  *Internal scratch used for IOVec allocations.*

## Macros

- #define ALIGN(x, a) (((x) + ((a) - 1)) & ∼((a) - 1))

  *Aligns a value x up to an alignment a.*
- #define TFM_CRYPTO_IOVEC_ALIGNMENT (4u)

  *Maximum alignment required by any iovec parameters to the TF-M Crypto partition.*

## Functions

- psa_status_t tfm_crypto_get_caller_id (int32_t ∗id)

  *Returns the ID of the caller.*
- psa_status_t tfm_crypto_init (void)

  *Initialise the service.*
- psa_status_t tfm_crypto_sfn (const psa_msg_t ∗msg)
- psa_status_t tfm_crypto_api_dispatcher (psa_invec in_vec[ ], size_t in_len, psa_outvec out_vec[ ], size_↩ t out_len)

  *This function acts as interface from the framework dispatching calls to the set of functions that implement the PSA Crypto APIs. It is based on the Uniform Signatures prototype.*

## 7.236.1 Macro Definition Documentation

### 7.236.1.1 ALIGN

```
#define ALIGN(
            x,
            a ) (((x) + ((a) - 1)) & ∼((a) - 1))
```

Aligns a value x up to an alignment a.

Definition at line 41 of file crypto_init.c.

#### 7.236.1.2 TFM_CRYPTO_IOVEC_ALIGNMENT

```
#define TFM_CRYPTO_IOVEC_ALIGNMENT (4u)
```

Maximum alignment required by any iovec parameters to the TF-M Crypto partition.
Definition at line 47 of file crypto_init.c.

### 7.236.2 Function Documentation

#### 7.236.2.1 tfm_crypto_api_dispatcher()

```
psa_status_t tfm_crypto_api_dispatcher (
            psa_invec in_vec[],
            size_t in_len,
            psa_outvec out_vec[],
            size_t out_len )
```

This function acts as interface from the framework dispatching calls to the set of functions that implement the PSA Crypto APIs. It is based on the Uniform Signatures prototype.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| in | *in_len* | Length of the valid entries in in_vec |
| out | *out_vec* | Array of outvec parameters |
| in | *out_len* | Length of the valid entries in out_vec |

**Returns**

Return values as described in psa_status_t

Definition at line 340 of file crypto_init.c.

Here is the call graph for this function:

#### 7.236.2.2 tfm_crypto_get_caller_id()

```
psa_status_t tfm_crypto_get_caller_id (
            int32_t * id )
```

Returns the ID of the caller.

**Parameters**

| | | |
|---|---|---|
| out | *id* | Pointer to hold the ID of the caller |

**Returns**

Return values as described in psa_status_t

Definition at line 152 of file crypto_init.c.
Here is the caller graph for this function:

#### 7.236.2.3 tfm_crypto_init()

```
psa_status_t tfm_crypto_init (
            void  )
```

Initialise the service.

**Returns**

Return values as described in psa_status_t

Definition at line 310 of file crypto_init.c.

#### 7.236.2.4 tfm_crypto_sfn()

```
psa_status_t tfm_crypto_sfn (
            const psa_msg_t * msg )
```

Definition at line 329 of file crypto_init.c.

## 7.237 secure_fw/partitions/crypto/crypto_key_derivation.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <string.h>
#include "config_tfm.h"
#include "static_checks.h"
#include "tfm_sp_log.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_key.h"
#include "tfm_crypto_defs.h"
#include "crypto_library.h"
```

Include dependency graph for crypto_key_derivation.c:

### Functions

- psa_status_t tfm_crypto_key_derivation_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Key derivation module.*

## 7.238 secure_fw/partitions/crypto/crypto_key_management.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <string.h>
```

```
#include "config_tfm.h"
#include "static_checks.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_key.h"
#include "tfm_crypto_defs.h"
#include "crypto_library.h"
```

Include dependency graph for crypto_key_management.c:

## Functions

- psa_status_t tfm_crypto_key_management_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s ∗encoded_key)

  *This function acts as interface for the Key management module.*

## 7.239 secure_fw/partitions/crypto/crypto_library.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <string.h>
#include "tfm_sp_log.h"
#include "config_tfm.h"
#include "psa/crypto.h"
#include "psa/error.h"
#include "crypto_library.h"
#include "tfm_plat_crypto_keys.h"
#include "mbedtls/memory_buffer_alloc.h"
#include "mbedtls/platform.h"
#include "mbedtls/version.h"
#include "config_engine_buf.h"
```

Include dependency graph for crypto_library.c:

## Functions

- tfm_crypto_library_key_id_t tfm_crypto_library_key_id_init (int32_t owner, psa_key_id_t key_id)

  *Function used to initialise an object of tfm_crypto_library_key_id_t to a (owner, key_id) pair.*

- char ∗ tfm_crypto_library_get_info (void)

*This function is used to retrieve a string describing the library used in the backend to provide information to the crypto service and the user.*

- psa_status_t tfm_crypto_core_library_init (void)

  *This function is used to perform the necessary steps to initialise the underlying library that provides the implementation of the PSA Crypto core to the TF-M Crypto service.*
- void tfm_crypto_library_get_library_key_id_set_owner (int32_t owner, psa_key_attributes_t *attr)

  *Allows to set the owner of a library key embedded into the key attributes structure.*
- psa_status_t mbedtls_psa_platform_get_builtin_key (mbedtls_svc_key_id_t key_id, psa_key_lifetime_↩t *lifetime, psa_drv_slot_number_t *slot_number)

  *This function is required by mbed TLS to enable support for platform builtin keys in the PSA Crypto core layer implemented by mbed TLS. This function is not standardized by the API hence this layer directly provides the symbol required by the library.*

## 7.240 secure_fw/partitions/crypto/crypto_library.h File Reference

This file contains some abstractions required to interface the TF-M Crypto service to an underlying cryptographic library that implements the PSA Crypto API. The TF-M Crypto service uses this library to provide a PSA Crypto core layer implementation and a software or hardware based implementation of crypto algorithms.

`#include "psa/crypto.h"`

Include dependency graph for crypto_library.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define CRYPTO_LIBRARY_GET_KEY_ID(encoded_key_library) MBEDTLS_SVC_KEY_ID_GET_KEY_↩ID(encoded_key_library)

  *This macro extracts the key ID from the library encoded key passed as parameter.*
- #define CRYPTO_LIBRARY_GET_OWNER(encoded_key_library) MBEDTLS_SVC_KEY_ID_GET_OWN↩ER_ID(encoded_key_library)

  *This macro extracts the owner from the library encoded key passed as parameter.*

### Typedefs

- typedef mbedtls_svc_key_id_t tfm_crypto_library_key_id_t

  *The following typedef must be defined to the type associated to the key_id in the underlying library.*

## Functions

- tfm_crypto_library_key_id_t tfm_crypto_library_key_id_init (int32_t owner, psa_key_id_t key_id)

  *Function used to initialise an object of tfm_crypto_library_key_id_t to a (owner, key_id) pair.*
- char * tfm_crypto_library_get_info (void)

  *This function is used to retrieve a string describing the library used in the backend to provide information to the crypto service and the user.*
- void tfm_crypto_library_get_library_key_id_set_owner (int32_t owner, psa_key_attributes_t *attr)

  *Allows to set the owner of a library key embedded into the key attributes structure.*
- psa_status_t tfm_crypto_core_library_init (void)

  *This function is used to perform the necessary steps to initialise the underlying library that provides the implementation of the PSA Crypto core to the TF-M Crypto service.*

### 7.240.1 Detailed Description

This file contains some abstractions required to interface the TF-M Crypto service to an underlying cryptographic library that implements the PSA Crypto API. The TF-M Crypto service uses this library to provide a PSA Crypto core layer implementation and a software or hardware based implementation of crypto algorithms.

### 7.240.2 Macro Definition Documentation

#### 7.240.2.1 CRYPTO_LIBRARY_GET_KEY_ID

```
#define CRYPTO_LIBRARY_GET_KEY_ID(
            encoded_key_library ) MBEDTLS_SVC_KEY_ID_GET_KEY_ID(encoded_key_library)
```

This macro extracts the key ID from the library encoded key passed as parameter.
Definition at line 31 of file crypto_library.h.

#### 7.240.2.2 CRYPTO_LIBRARY_GET_OWNER

```
#define CRYPTO_LIBRARY_GET_OWNER(
            encoded_key_library ) MBEDTLS_SVC_KEY_ID_GET_OWNER_ID(encoded_key_library)
```

This macro extracts the owner from the library encoded key passed as parameter.
Definition at line 37 of file crypto_library.h.

### 7.240.3 Typedef Documentation

#### 7.240.3.1 tfm_crypto_library_key_id_t

```
typedef mbedtls_svc_key_id_t tfm_crypto_library_key_id_t
```

The following typedef must be defined to the type associated to the key_id in the underlying library.
Definition at line 43 of file crypto_library.h.

## 7.241 secure_fw/partitions/crypto/crypto_mac.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "config_tfm.h"
#include "static_checks.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_key.h"
```

```
#include "tfm_crypto_defs.h"
#include "crypto_library.h"
```

Include dependency graph for crypto_mac.c:

## Functions

- psa_status_t tfm_crypto_mac_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the MAC module.*

## 7.242 secure_fw/partitions/crypto/crypto_rng.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "config_tfm.h"
#include "tfm_mbedcrypto_include.h"
#include "tfm_crypto_api.h"
#include "tfm_crypto_defs.h"
```

Include dependency graph for crypto_rng.c:

## Functions

- psa_status_t tfm_crypto_random_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ])

  *This function acts as interface for the Random module.*

## 7.243 secure_fw/partitions/crypto/crypto_spe.h File Reference

When Mbed Crypto is built with the MBEDTLS_PSA_CRYPTO_SPM option enabled, this header is included by all .c files in Mbed Crypto that use PSA Crypto function names. This avoids duplication of symbols between TF-M and Mbed Crypto.

This graph shows which files directly or indirectly include this file:

## Macros

- #define PSA_FUNCTION_NAME(x) mbedcrypto__ ## x
- #define psa_crypto_init PSA_FUNCTION_NAME(psa_crypto_init)
- #define psa_key_derivation_get_capacity PSA_FUNCTION_NAME(psa_key_derivation_get_capacity)
- #define psa_key_derivation_set_capacity PSA_FUNCTION_NAME(psa_key_derivation_set_capacity)
- #define psa_key_derivation_input_bytes PSA_FUNCTION_NAME(psa_key_derivation_input_bytes)
- #define psa_key_derivation_input_integer PSA_FUNCTION_NAME(psa_key_derivation_input_integer)
- #define psa_key_derivation_output_bytes PSA_FUNCTION_NAME(psa_key_derivation_output_bytes)
- #define psa_key_derivation_input_key PSA_FUNCTION_NAME(psa_key_derivation_input_key)
- #define psa_key_derivation_output_key PSA_FUNCTION_NAME(psa_key_derivation_output_key)
- #define psa_key_derivation_setup PSA_FUNCTION_NAME(psa_key_derivation_setup)
- #define psa_key_derivation_abort PSA_FUNCTION_NAME(psa_key_derivation_abort)
- #define psa_key_derivation_key_agreement PSA_FUNCTION_NAME(psa_key_derivation_key_agreement)
- #define psa_raw_key_agreement PSA_FUNCTION_NAME(psa_raw_key_agreement)
- #define psa_generate_random PSA_FUNCTION_NAME(psa_generate_random)
- #define psa_aead_encrypt PSA_FUNCTION_NAME(psa_aead_encrypt)
- #define psa_aead_decrypt PSA_FUNCTION_NAME(psa_aead_decrypt)
- #define psa_aead_encrypt_setup PSA_FUNCTION_NAME(psa_aead_encrypt_setup)
- #define psa_aead_decrypt_setup PSA_FUNCTION_NAME(psa_aead_decrypt_setup)
- #define psa_aead_generate_nonce PSA_FUNCTION_NAME(psa_aead_generate_nonce)
- #define psa_aead_set_nonce PSA_FUNCTION_NAME(psa_aead_set_nonce)
- #define psa_aead_set_lengths PSA_FUNCTION_NAME(psa_aead_set_lengths)
- #define psa_aead_update_ad PSA_FUNCTION_NAME(psa_aead_update_ad)
- #define psa_aead_update PSA_FUNCTION_NAME(psa_aead_update)
- #define psa_aead_finish PSA_FUNCTION_NAME(psa_aead_finish)
- #define psa_aead_verify PSA_FUNCTION_NAME(psa_aead_verify)
- #define psa_aead_abort PSA_FUNCTION_NAME(psa_aead_abort)
- #define psa_open_key PSA_FUNCTION_NAME(psa_open_key)
- #define psa_close_key PSA_FUNCTION_NAME(psa_close_key)
- #define psa_import_key PSA_FUNCTION_NAME(psa_import_key)
- #define psa_destroy_key PSA_FUNCTION_NAME(psa_destroy_key)
- #define psa_get_key_attributes PSA_FUNCTION_NAME(psa_get_key_attributes)
- #define psa_reset_key_attributes PSA_FUNCTION_NAME(psa_reset_key_attributes)
- #define psa_export_key PSA_FUNCTION_NAME(psa_export_key)
- #define psa_export_public_key PSA_FUNCTION_NAME(psa_export_public_key)
- #define psa_purge_key PSA_FUNCTION_NAME(psa_purge_key)
- #define psa_copy_key PSA_FUNCTION_NAME(psa_copy_key)
- #define psa_cipher_operation_init PSA_FUNCTION_NAME(psa_cipher_operation_init)
- #define psa_cipher_generate_iv PSA_FUNCTION_NAME(psa_cipher_generate_iv)
- #define psa_cipher_set_iv PSA_FUNCTION_NAME(psa_cipher_set_iv)
- #define psa_cipher_encrypt_setup PSA_FUNCTION_NAME(psa_cipher_encrypt_setup)
- #define psa_cipher_decrypt_setup PSA_FUNCTION_NAME(psa_cipher_decrypt_setup)
- #define psa_cipher_update PSA_FUNCTION_NAME(psa_cipher_update)
- #define psa_cipher_encrypt PSA_FUNCTION_NAME(psa_cipher_encrypt)
- #define psa_cipher_decrypt PSA_FUNCTION_NAME(psa_cipher_decrypt)
- #define psa_cipher_finish PSA_FUNCTION_NAME(psa_cipher_finish)

- #define psa_cipher_abort PSA_FUNCTION_NAME(psa_cipher_abort)
- #define psa_hash_operation_init PSA_FUNCTION_NAME(psa_hash_operation_init)
- #define psa_hash_setup PSA_FUNCTION_NAME(psa_hash_setup)
- #define psa_hash_update PSA_FUNCTION_NAME(psa_hash_update)
- #define psa_hash_finish PSA_FUNCTION_NAME(psa_hash_finish)
- #define psa_hash_verify PSA_FUNCTION_NAME(psa_hash_verify)
- #define psa_hash_abort PSA_FUNCTION_NAME(psa_hash_abort)
- #define psa_hash_clone PSA_FUNCTION_NAME(psa_hash_clone)
- #define psa_hash_compute PSA_FUNCTION_NAME(psa_hash_compute)
- #define psa_hash_compare PSA_FUNCTION_NAME(psa_hash_compare)
- #define psa_mac_operation_init PSA_FUNCTION_NAME(psa_mac_operation_init)
- #define psa_mac_sign_setup PSA_FUNCTION_NAME(psa_mac_sign_setup)
- #define psa_mac_verify_setup PSA_FUNCTION_NAME(psa_mac_verify_setup)
- #define psa_mac_update PSA_FUNCTION_NAME(psa_mac_update)
- #define psa_mac_sign_finish PSA_FUNCTION_NAME(psa_mac_sign_finish)
- #define psa_mac_verify_finish PSA_FUNCTION_NAME(psa_mac_verify_finish)
- #define psa_mac_compute PSA_FUNCTION_NAME(psa_mac_compute)
- #define psa_mac_verify PSA_FUNCTION_NAME(psa_mac_verify)
- #define psa_mac_abort PSA_FUNCTION_NAME(psa_mac_abort)
- #define psa_sign_message PSA_FUNCTION_NAME(psa_sign_message)
- #define psa_verify_message PSA_FUNCTION_NAME(psa_verify_message)
- #define psa_sign_hash PSA_FUNCTION_NAME(psa_sign_hash)
- #define psa_verify_hash PSA_FUNCTION_NAME(psa_verify_hash)
- #define psa_asymmetric_encrypt PSA_FUNCTION_NAME(psa_asymmetric_encrypt)
- #define psa_asymmetric_decrypt PSA_FUNCTION_NAME(psa_asymmetric_decrypt)
- #define psa_generate_key PSA_FUNCTION_NAME(psa_generate_key)

## 7.243.1 Detailed Description

When Mbed Crypto is built with the MBEDTLS_PSA_CRYPTO_SPM option enabled, this header is included by all .c files in Mbed Crypto that use PSA Crypto function names. This avoids duplication of symbols between TF-M and Mbed Crypto.

**Note**

This file should be included before including any PSA Crypto headers from Mbed Crypto.

## 7.243.2 Macro Definition Documentation

### 7.243.2.1 psa_aead_abort

```
#define psa_aead_abort PSA_FUNCTION_NAME(psa_aead_abort)
```

Definition at line 73 of file crypto_spe.h.

### 7.243.2.2 psa_aead_decrypt

```
#define psa_aead_decrypt PSA_FUNCTION_NAME(psa_aead_decrypt)
```

Definition at line 53 of file crypto_spe.h.

### 7.243.2.3 psa_aead_decrypt_setup

```
#define psa_aead_decrypt_setup PSA_FUNCTION_NAME(psa_aead_decrypt_setup)
```

Definition at line 57 of file crypto_spe.h.

#### 7.243.2.4 psa_aead_encrypt

```
#define psa_aead_encrypt PSA_FUNCTION_NAME(psa_aead_encrypt)
```

Definition at line 51 of file crypto_spe.h.

#### 7.243.2.5 psa_aead_encrypt_setup

```
#define psa_aead_encrypt_setup PSA_FUNCTION_NAME(psa_aead_encrypt_setup)
```

Definition at line 55 of file crypto_spe.h.

#### 7.243.2.6 psa_aead_finish

```
#define psa_aead_finish PSA_FUNCTION_NAME(psa_aead_finish)
```

Definition at line 69 of file crypto_spe.h.

#### 7.243.2.7 psa_aead_generate_nonce

```
#define psa_aead_generate_nonce PSA_FUNCTION_NAME(psa_aead_generate_nonce)
```

Definition at line 59 of file crypto_spe.h.

#### 7.243.2.8 psa_aead_set_lengths

```
#define psa_aead_set_lengths PSA_FUNCTION_NAME(psa_aead_set_lengths)
```

Definition at line 63 of file crypto_spe.h.

#### 7.243.2.9 psa_aead_set_nonce

```
#define psa_aead_set_nonce PSA_FUNCTION_NAME(psa_aead_set_nonce)
```

Definition at line 61 of file crypto_spe.h.

#### 7.243.2.10 psa_aead_update

```
#define psa_aead_update PSA_FUNCTION_NAME(psa_aead_update)
```

Definition at line 67 of file crypto_spe.h.

#### 7.243.2.11 psa_aead_update_ad

```
#define psa_aead_update_ad PSA_FUNCTION_NAME(psa_aead_update_ad)
```

Definition at line 65 of file crypto_spe.h.

#### 7.243.2.12 psa_aead_verify

```
#define psa_aead_verify PSA_FUNCTION_NAME(psa_aead_verify)
```

Definition at line 71 of file crypto_spe.h.

#### 7.243.2.13 psa_asymmetric_decrypt

```
#define psa_asymmetric_decrypt PSA_FUNCTION_NAME(psa_asymmetric_decrypt)
```

Definition at line 161 of file crypto_spe.h.

#### 7.243.2.14 psa_asymmetric_encrypt

```
#define psa_asymmetric_encrypt PSA_FUNCTION_NAME(psa_asymmetric_encrypt)
```

Definition at line 159 of file crypto_spe.h.

#### 7.243.2.15 psa_cipher_abort

```
#define psa_cipher_abort PSA_FUNCTION_NAME(psa_cipher_abort)
```

Definition at line 113 of file crypto_spe.h.

#### 7.243.2.16 psa_cipher_decrypt

```
#define psa_cipher_decrypt PSA_FUNCTION_NAME(psa_cipher_decrypt)
```

Definition at line 109 of file crypto_spe.h.

#### 7.243.2.17 psa_cipher_decrypt_setup

```
#define psa_cipher_decrypt_setup PSA_FUNCTION_NAME(psa_cipher_decrypt_setup)
```

Definition at line 103 of file crypto_spe.h.

#### 7.243.2.18 psa_cipher_encrypt

```
#define psa_cipher_encrypt PSA_FUNCTION_NAME(psa_cipher_encrypt)
```

Definition at line 107 of file crypto_spe.h.

#### 7.243.2.19 psa_cipher_encrypt_setup

```
#define psa_cipher_encrypt_setup PSA_FUNCTION_NAME(psa_cipher_encrypt_setup)
```

Definition at line 101 of file crypto_spe.h.

#### 7.243.2.20 psa_cipher_finish

```
#define psa_cipher_finish PSA_FUNCTION_NAME(psa_cipher_finish)
```

Definition at line 111 of file crypto_spe.h.

#### 7.243.2.21 psa_cipher_generate_iv

```
#define psa_cipher_generate_iv PSA_FUNCTION_NAME(psa_cipher_generate_iv)
```

Definition at line 97 of file crypto_spe.h.

#### 7.243.2.22 psa_cipher_operation_init

```
#define psa_cipher_operation_init PSA_FUNCTION_NAME(psa_cipher_operation_init)
```

Definition at line 95 of file crypto_spe.h.

#### 7.243.2.23 psa_cipher_set_iv

```
#define psa_cipher_set_iv PSA_FUNCTION_NAME(psa_cipher_set_iv)
```

Definition at line 99 of file crypto_spe.h.

#### 7.243.2.24 psa_cipher_update

```
#define psa_cipher_update PSA_FUNCTION_NAME(psa_cipher_update)
```

Definition at line 105 of file crypto_spe.h.

#### 7.243.2.25 psa_close_key

```
#define psa_close_key PSA_FUNCTION_NAME(psa_close_key)
```

Definition at line 77 of file crypto_spe.h.

#### 7.243.2.26 psa_copy_key

```
#define psa_copy_key PSA_FUNCTION_NAME(psa_copy_key)
```

Definition at line 93 of file crypto_spe.h.

#### 7.243.2.27 psa_crypto_init

```
#define psa_crypto_init PSA_FUNCTION_NAME(psa_crypto_init)
```

Definition at line 25 of file crypto_spe.h.

#### 7.243.2.28 psa_destroy_key

```
#define psa_destroy_key PSA_FUNCTION_NAME(psa_destroy_key)
```

Definition at line 81 of file crypto_spe.h.

#### 7.243.2.29 psa_export_key

```
#define psa_export_key PSA_FUNCTION_NAME(psa_export_key)
```

Definition at line 87 of file crypto_spe.h.

#### 7.243.2.30 psa_export_public_key

```
#define psa_export_public_key PSA_FUNCTION_NAME(psa_export_public_key)
```

Definition at line 89 of file crypto_spe.h.

#### 7.243.2.31 PSA_FUNCTION_NAME

```
#define PSA_FUNCTION_NAME(
              x ) mbedcrypto__ ## x
```

Definition at line 23 of file crypto_spe.h.

#### 7.243.2.32 psa_generate_key

```
#define psa_generate_key PSA_FUNCTION_NAME(psa_generate_key)
```

Definition at line 163 of file crypto_spe.h.

#### 7.243.2.33 psa_generate_random

```
#define psa_generate_random PSA_FUNCTION_NAME(psa_generate_random)
```

Definition at line 49 of file crypto_spe.h.

#### 7.243.2.34 psa_get_key_attributes

```
#define psa_get_key_attributes PSA_FUNCTION_NAME(psa_get_key_attributes)
```

Definition at line 83 of file crypto_spe.h.

#### 7.243.2.35 psa_hash_abort

```
#define psa_hash_abort PSA_FUNCTION_NAME(psa_hash_abort)
```

Definition at line 125 of file crypto_spe.h.

#### 7.243.2.36 psa_hash_clone

```
#define psa_hash_clone PSA_FUNCTION_NAME(psa_hash_clone)
```

Definition at line 127 of file crypto_spe.h.

#### 7.243.2.37 psa_hash_compare

```
#define psa_hash_compare PSA_FUNCTION_NAME(psa_hash_compare)
```

Definition at line 131 of file crypto_spe.h.

#### 7.243.2.38 psa_hash_compute

```
#define psa_hash_compute PSA_FUNCTION_NAME(psa_hash_compute)
```

Definition at line 129 of file crypto_spe.h.

#### 7.243.2.39 psa_hash_finish

```
#define psa_hash_finish PSA_FUNCTION_NAME(psa_hash_finish)
```

Definition at line 121 of file crypto_spe.h.

#### 7.243.2.40 psa_hash_operation_init

```
#define psa_hash_operation_init PSA_FUNCTION_NAME(psa_hash_operation_init)
```

Definition at line 115 of file crypto_spe.h.

#### 7.243.2.41 psa_hash_setup

```
#define psa_hash_setup PSA_FUNCTION_NAME(psa_hash_setup)
```

Definition at line 117 of file crypto_spe.h.

#### 7.243.2.42 psa_hash_update

```
#define psa_hash_update PSA_FUNCTION_NAME(psa_hash_update)
```

Definition at line 119 of file crypto_spe.h.

#### 7.243.2.43 psa_hash_verify

```
#define psa_hash_verify PSA_FUNCTION_NAME(psa_hash_verify)
```

Definition at line 123 of file crypto_spe.h.

#### 7.243.2.44 psa_import_key

```
#define psa_import_key PSA_FUNCTION_NAME(psa_import_key)
```

Definition at line 79 of file crypto_spe.h.

#### 7.243.2.45 psa_key_derivation_abort

```
#define psa_key_derivation_abort PSA_FUNCTION_NAME(psa_key_derivation_abort)
```

Definition at line 43 of file crypto_spe.h.

#### 7.243.2.46 psa_key_derivation_get_capacity

```
#define psa_key_derivation_get_capacity PSA_FUNCTION_NAME(psa_key_derivation_get_capacity)
```

Definition at line 27 of file crypto_spe.h.

#### 7.243.2.47 psa_key_derivation_input_bytes

```
#define psa_key_derivation_input_bytes PSA_FUNCTION_NAME(psa_key_derivation_input_bytes)
```

Definition at line 31 of file crypto_spe.h.

#### 7.243.2.48 psa_key_derivation_input_integer

```
#define psa_key_derivation_input_integer PSA_FUNCTION_NAME(psa_key_derivation_input_integer)
```

Definition at line 33 of file crypto_spe.h.

#### 7.243.2.49 psa_key_derivation_input_key

```
#define psa_key_derivation_input_key PSA_FUNCTION_NAME(psa_key_derivation_input_key)
```

Definition at line 37 of file crypto_spe.h.

#### 7.243.2.50 psa_key_derivation_key_agreement

```
#define psa_key_derivation_key_agreement PSA_FUNCTION_NAME(psa_key_derivation_key_agreement)
```

Definition at line 45 of file crypto_spe.h.

#### 7.243.2.51 psa_key_derivation_output_bytes

```
#define psa_key_derivation_output_bytes PSA_FUNCTION_NAME(psa_key_derivation_output_bytes)
```

Definition at line 35 of file crypto_spe.h.

#### 7.243.2.52 psa_key_derivation_output_key

```
#define psa_key_derivation_output_key PSA_FUNCTION_NAME(psa_key_derivation_output_key)
```

Definition at line 39 of file crypto_spe.h.

#### 7.243.2.53 psa_key_derivation_set_capacity

```
#define psa_key_derivation_set_capacity PSA_FUNCTION_NAME(psa_key_derivation_set_capacity)
```

Definition at line 29 of file crypto_spe.h.

#### 7.243.2.54 psa_key_derivation_setup

```
#define psa_key_derivation_setup PSA_FUNCTION_NAME(psa_key_derivation_setup)
```

Definition at line 41 of file crypto_spe.h.

#### 7.243.2.55 psa_mac_abort

```
#define psa_mac_abort PSA_FUNCTION_NAME(psa_mac_abort)
```

Definition at line 149 of file crypto_spe.h.

#### 7.243.2.56 psa_mac_compute

```
#define psa_mac_compute PSA_FUNCTION_NAME(psa_mac_compute)
```

Definition at line 145 of file crypto_spe.h.

#### 7.243.2.57 psa_mac_operation_init

```
#define psa_mac_operation_init PSA_FUNCTION_NAME(psa_mac_operation_init)
```

Definition at line 133 of file crypto_spe.h.

#### 7.243.2.58 psa_mac_sign_finish

```
#define psa_mac_sign_finish PSA_FUNCTION_NAME(psa_mac_sign_finish)
```

Definition at line 141 of file crypto_spe.h.

#### 7.243.2.59 psa_mac_sign_setup

```
#define psa_mac_sign_setup PSA_FUNCTION_NAME(psa_mac_sign_setup)
```

Definition at line 135 of file crypto_spe.h.

#### 7.243.2.60 psa_mac_update

```
#define psa_mac_update PSA_FUNCTION_NAME(psa_mac_update)
```

Definition at line 139 of file crypto_spe.h.

#### 7.243.2.61 psa_mac_verify

```
#define psa_mac_verify PSA_FUNCTION_NAME(psa_mac_verify)
```

Definition at line 147 of file crypto_spe.h.

#### 7.243.2.62 psa_mac_verify_finish

```
#define psa_mac_verify_finish PSA_FUNCTION_NAME(psa_mac_verify_finish)
```

Definition at line 143 of file crypto_spe.h.

#### 7.243.2.63 psa_mac_verify_setup

```
#define psa_mac_verify_setup PSA_FUNCTION_NAME(psa_mac_verify_setup)
```

Definition at line 137 of file crypto_spe.h.

#### 7.243.2.64 psa_open_key

```
#define psa_open_key PSA_FUNCTION_NAME(psa_open_key)
```

Definition at line 75 of file crypto_spe.h.

#### 7.243.2.65 psa_purge_key

```
#define psa_purge_key PSA_FUNCTION_NAME(psa_purge_key)
```

Definition at line 91 of file crypto_spe.h.

#### 7.243.2.66 psa_raw_key_agreement

```
#define psa_raw_key_agreement PSA_FUNCTION_NAME(psa_raw_key_agreement)
```

Definition at line 47 of file crypto_spe.h.

#### 7.243.2.67 psa_reset_key_attributes

```
#define psa_reset_key_attributes PSA_FUNCTION_NAME(psa_reset_key_attributes)
```

Definition at line 85 of file crypto_spe.h.

#### 7.243.2.68 psa_sign_hash

```
#define psa_sign_hash PSA_FUNCTION_NAME(psa_sign_hash)
```

Definition at line 155 of file crypto_spe.h.

#### 7.243.2.69 psa_sign_message

```
#define psa_sign_message PSA_FUNCTION_NAME(psa_sign_message)
```

Definition at line 151 of file crypto_spe.h.

#### 7.243.2.70 psa_verify_hash

```
#define psa_verify_hash PSA_FUNCTION_NAME(psa_verify_hash)
```

Definition at line 157 of file crypto_spe.h.

#### 7.243.2.71 psa_verify_message

```
#define psa_verify_message PSA_FUNCTION_NAME(psa_verify_message)
```

Definition at line 153 of file crypto_spe.h.

## 7.244 secure_fw/partitions/crypto/dir_crypto.dox File Reference

## 7.245 secure_fw/partitions/crypto/psa_driver_api/tfm_builtin_key_↩ loader.c File Reference

```
#include <string.h>
#include "tfm_builtin_key_loader.h"
#include "tfm_mbedcrypto_include.h"
#include "psa_manifest/pid.h"
#include "tfm_plat_crypto_keys.h"
```

```
#include "crypto_library.h"
```

Include dependency graph for tfm_builtin_key_loader.c:

## Data Structures

- struct tfm_builtin_key_t

  *A structure which describes a builtin key slot.*

## Macros

- #define TFM_BUILTIN_MAX_KEY_LEN (48)
- #define TFM_BUILTIN_MAX_KEYS (TFM_BUILTIN_KEY_SLOT_MAX)
- #define NUMBER_OF_ELEMENTS_OF(x) sizeof(x)/sizeof(∗x)

## Functions

- psa_status_t tfm_builtin_key_loader_init (void)

  *This is the initialisation function for the tfm_builtin_key_laoder driver, to be called from the PSA core initialisation subsystem. It discovers the keys available in the underlying hardware platform and loads them in memory visible to the PSA Crypto subsystem to be used to the normal APIs.*
- psa_status_t tfm_builtin_key_loader_get_key_buffer_size (tfm_crypto_library_key_id_t key_id, size_t ∗len)

  *Returns the length of a key from the builtin driver.*
- psa_status_t tfm_builtin_key_loader_get_builtin_key (psa_drv_slot_number_t slot_number, psa_key_↩attributes_t ∗attributes, uint8_t ∗key_buffer, size_t key_buffer_size, size_t ∗key_buffer_length)

  *Returns the attributes and key material of a key from the builtin driver to be used by the PSA Crypto core.*

## 7.245.1 Macro Definition Documentation

### 7.245.1.1 NUMBER_OF_ELEMENTS_OF

```
#define NUMBER_OF_ELEMENTS_OF(
            x ) sizeof(x)/sizeof(*x)
```

Definition at line 22 of file tfm_builtin_key_loader.c.

### 7.245.1.2 TFM_BUILTIN_MAX_KEY_LEN

```
#define TFM_BUILTIN_MAX_KEY_LEN (48)
```

Definition at line 15 of file tfm_builtin_key_loader.c.

### 7.245.1.3 TFM_BUILTIN_MAX_KEYS

```
#define TFM_BUILTIN_MAX_KEYS (TFM_BUILTIN_KEY_SLOT_MAX)
```

Definition at line 19 of file tfm_builtin_key_loader.c.

## 7.246 secure_fw/partitions/crypto/psa_driver_api/tfm_builtin_key_loader.h File Reference

```
#include <psa/crypto.h>
#include "platform_builtin_key_loader_ids.h"
```

Include dependency graph for tfm_builtin_key_loader.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define PSA_CRYPTO_DRIVER_TFM_BUILTIN_KEY_LOADER

  *Enables the tfm_builtin_key_loader driver in the PSA Crypto core subsystem.*
- #define TFM_BUILTIN_KEY_LOADER_KEY_LOCATION ((psa_key_location_t)0x800001)

  *The PSA driver location for TF-M builtin keys. Arbitrary within the ranges documented at* `https://armmbed.↩github.io/mbed-crypto/html/api/keys/lifetimes.html#c.psa_key_location_t`.
- #define TFM_BUILTIN_KEY_LOADER_LIFETIME

  *This macro defines the lifetime associated to TF-M builtin keys as persistent and as an ad-hoc location associated to the TFM_BUILTIN_KEY_LOADER driver. To be handled by the tfm_builtin_ker_loader driver, the lifetime of the platform keys must be set equal to this particular lifetime value.*

### Functions

- psa_status_t tfm_builtin_key_loader_init (void)

  *This is the initialisation function for the tfm_builtin_key_laoder driver, to be called from the PSA core initialisation subsystem. It discovers the keys available in the underlying hardware platform and loads them in memory visible to the PSA Crypto subsystem to be used to the normal APIs.*
- psa_status_t tfm_builtin_key_loader_get_key_buffer_size (mbedtls_svc_key_id_t key_id, size_t *len)

*Returns the length of a key from the builtin driver.*

- psa_status_t tfm_builtin_key_loader_get_builtin_key (psa_drv_slot_number_t slot_number, psa_key_↩attributes_t *attributes, uint8_t *key_buffer, size_t key_buffer_size, size_t *key_buffer_length)

  *Returns the attributes and key material of a key from the builtin driver to be used by the PSA Crypto core.*

### 7.246.1 Macro Definition Documentation

#### 7.246.1.1 PSA_CRYPTO_DRIVER_TFM_BUILTIN_KEY_LOADER

```
#define PSA_CRYPTO_DRIVER_TFM_BUILTIN_KEY_LOADER
```

Enables the tfm_builtin_key_loader driver in the PSA Crypto core subsystem.
Definition at line 37 of file tfm_builtin_key_loader.h.

#### 7.246.1.2 TFM_BUILTIN_KEY_LOADER_KEY_LOCATION

```
#define TFM_BUILTIN_KEY_LOADER_KEY_LOCATION ((psa_key_location_t)0x800001)
```

The PSA driver location for TF-M builtin keys. Arbitrary within the ranges documented at https↩://armmbed.github.io/mbed-crypto/html/api/keys/lifetimes.html#c.psa_key_↩location_t.
Definition at line 45 of file tfm_builtin_key_loader.h.

#### 7.246.1.3 TFM_BUILTIN_KEY_LOADER_LIFETIME

```
#define TFM_BUILTIN_KEY_LOADER_LIFETIME
```

**Value:**

```
(PSA_KEY_LIFETIME_FROM_PERSISTENCE_AND_LOCATION( \
PSA_KEY_LIFETIME_PERSISTENT,
TFM_BUILTIN_KEY_LOADER_KEY_LOCATION))
```

This macro defines the lifetime associated to TF-M builtin keys as persistent and as an ad-hoc location associated to the TFM_BUILTIN_KEY_LOADER driver. To be handled by the tfm_builtin_ker_loader driver, the lifetime of the platform keys must be set equal to this particular lifetime value.
Definition at line 53 of file tfm_builtin_key_loader.h.

## 7.247 secure_fw/partitions/crypto/tfm_crypto_api.h File Reference

```
#include <limits.h>
#include <stdint.h>
#include "tfm_crypto_defs.h"
#include "tfm_crypto_key.h"
#include "psa/client.h"
```

Include dependency graph for tfm_crypto_api.h:

This graph shows which files directly or indirectly include this file:

## Enumerations

- enum tfm_crypto_operation_type {
  TFM_CRYPTO_OPERATION_NONE = 0, TFM_CRYPTO_CIPHER_OPERATION = 1, TFM_CRYPTO_MAC_OPERATION = 2, TFM_CRYPTO_HASH_OPERATION = 3,
  TFM_CRYPTO_KEY_DERIVATION_OPERATION = 4, TFM_CRYPTO_AEAD_OPERATION = 5,
  TFM_CRYPTO_OPERATION_TYPE_MAX = INT_MAX }

  *List of possible operation types supported by the TFM based implementation. This type is needed by the operation allocation, lookup and release functions.*

## Functions

- psa_status_t tfm_crypto_init (void)

  *Initialise the service.*
- psa_status_t tfm_crypto_init_alloc (void)

  *Initialise the Alloc module.*
- psa_status_t tfm_crypto_get_caller_id (int32_t *id)

  *Returns the ID of the caller.*
- psa_status_t tfm_crypto_operation_alloc (enum tfm_crypto_operation_type type, uint32_t *handle, void **ctx)

  *Allocate an operation context in the backend.*
- psa_status_t tfm_crypto_operation_release (uint32_t *handle)

  *Release an operation context in the backend.*
- psa_status_t tfm_crypto_operation_lookup (enum tfm_crypto_operation_type type, uint32_t handle, void **ctx)

  *Look up an operation context in the backend for the corresponding frontend operation.*
- psa_status_t tfm_crypto_api_dispatcher (psa_invec in_vec[], size_t in_len, psa_outvec out_vec[], size_↩t out_len)

  *This function acts as interface from the framework dispatching calls to the set of functions that implement the PSA Crypto APIs. It is based on the Uniform Signatures prototype.*
- psa_status_t tfm_crypto_key_management_interface (psa_invec in_vec[], psa_outvec out_vec[], struct tfm_crypto_key_id_s *encoded_key)

*This function acts as interface for the Key management module.*

- psa_status_t tfm_crypto_mac_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the MAC module.*
- psa_status_t tfm_crypto_cipher_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Cipher module.*
- psa_status_t tfm_crypto_aead_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the AEAD module.*
- psa_status_t tfm_crypto_asymmetric_sign_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Asymmetric signing module.*
- psa_status_t tfm_crypto_asymmetric_encrypt_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Asymmetric encryption module.*
- psa_status_t tfm_crypto_key_derivation_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ], struct tfm_crypto_key_id_s *encoded_key)

  *This function acts as interface for the Key derivation module.*
- psa_status_t tfm_crypto_random_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ])

  *This function acts as interface for the Random module.*
- psa_status_t tfm_crypto_hash_interface (psa_invec in_vec[ ], psa_outvec out_vec[ ])

  *This function acts as interface for the Hash module.*

## 7.247.1 Enumeration Type Documentation

### 7.247.1.1 tfm_crypto_operation_type

`enum tfm_crypto_operation_type`

List of possible operation types supported by the TFM based implementation. This type is needed by the operation allocation, lookup and release functions.

**Enumerator**

| | |
|---|---|
| TFM_CRYPTO_OPERATION_NONE | |
| TFM_CRYPTO_CIPHER_OPERATION | |
| TFM_CRYPTO_MAC_OPERATION | |
| TFM_CRYPTO_HASH_OPERATION | |
| TFM_CRYPTO_KEY_DERIVATION_OPERATION | |
| TFM_CRYPTO_AEAD_OPERATION | |
| TFM_CRYPTO_OPERATION_TYPE_MAX | |

Definition at line 27 of file tfm_crypto_api.h.

## 7.247.2 Function Documentation

### 7.247.2.1 tfm_crypto_api_dispatcher()

```
psa_status_t tfm_crypto_api_dispatcher (
            psa_invec in_vec[],
            size_t in_len,
```

```
psa_outvec out_vec[],
size_t out_len )
```

This function acts as interface from the framework dispatching calls to the set of functions that implement the PSA Crypto APIs. It is based on the Uniform Signatures prototype.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Array of invec parameters |
| in | *in_len* | Length of the valid entries in in_vec |
| out | *out_vec* | Array of outvec parameters |
| in | *out_len* | Length of the valid entries in out_vec |

**Returns**

Return values as described in psa_status_t

Definition at line 340 of file crypto_init.c.
Here is the call graph for this function:

### 7.247.2.2 tfm_crypto_get_caller_id()

```
psa_status_t tfm_crypto_get_caller_id (
int32_t * id )
```

Returns the ID of the caller.

**Parameters**

| | | |
|---|---|---|
| out | *id* | Pointer to hold the ID of the caller |

**Returns**

Return values as described in psa_status_t

Definition at line 152 of file crypto_init.c.
Here is the caller graph for this function:

#### 7.247.2.3 tfm_crypto_init()

```
psa_status_t tfm_crypto_init (
            void  )
```

Initialise the service.

**Returns**

Return values as described in psa_status_t

Definition at line 310 of file crypto_init.c.

## 7.248 secure_fw/partitions/crypto/tfm_crypto_key.h File Reference

```
#include <stdint.h>
#include "psa/crypto.h"
```

Include dependency graph for tfm_crypto_key.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct tfm_crypto_key_id_s

  *The type which describes a key identifier to the Crypto service. The key identifiers must clearly provide a dedicated indication of the entity owner which owns the key.*

### Macros

- #define TFM_CRYPTO_KEY_ID_S_INIT {0, 0}

### 7.248.1 Macro Definition Documentation

#### 7.248.1.1 TFM_CRYPTO_KEY_ID_S_INIT

```
#define TFM_CRYPTO_KEY_ID_S_INIT {0, 0}
```

Definition at line 32 of file tfm_crypto_key.h.

## 7.249 secure_fw/partitions/crypto/tfm_mbedcrypto_alt.c File Reference

```
#include "tfm_mbedcrypto_include.h"
```

Include dependency graph for tfm_mbedcrypto_alt.c:

## 7.250 secure_fw/partitions/crypto/tfm_mbedcrypto_include.h File Reference

```
#include "crypto_spe.h"
#include "psa/crypto.h"
```

Include dependency graph for tfm_mbedcrypto_include.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define PSA_CRYPTO_SECURE 1

### 7.250.1 Macro Definition Documentation

#### 7.250.1.1 PSA_CRYPTO_SECURE

```
#define PSA_CRYPTO_SECURE 1
```

Definition at line 12 of file tfm_mbedcrypto_include.h.

## 7.251 secure_fw/partitions/dir_services.dox File Reference

## 7.252 secure_fw/partitions/firmware_update/bootloader/mcuboot/tfm_↩ mcuboot_fwu.c File Reference

```
#include <string.h>
#include "psa/crypto.h"
#include "psa/error.h"
#include "tfm_sp_log.h"
#include "bootutil_priv.h"
#include "bootutil/bootutil.h"
#include "bootutil/image.h"
#include "flash_map_backend/flash_map_backend.h"
#include "sysflash/sysflash.h"
#include "tfm_bootloader_fwu_abstraction.h"
#include "tfm_boot_status.h"
```

```
#include "service_api.h"
```

Include dependency graph for tfm_mcuboot_fwu.c:

## Data Structures

- struct fwu_image_info_data_s
- struct tfm_fwu_mcuboot_ctx_s

## Macros

- #define MAX_IMAGE_INFO_LENGTH

## Typedefs

- typedef struct fwu_image_info_data_s fwu_image_info_data_t
- typedef struct tfm_fwu_mcuboot_ctx_s tfm_fwu_mcuboot_ctx_t

## Functions

- psa_status_t fwu_bootloader_init (void)
- psa_status_t fwu_bootloader_staging_area_init (psa_fwu_component_t component, const void *manifest, size_t manifest_size)

  *Initialize the staging area of the component.*
- psa_status_t fwu_bootloader_load_image (psa_fwu_component_t component, size_t block_offset, const void *block, size_t block_size)

  *Load the image into the target component.*
- psa_status_t fwu_bootloader_install_image (const psa_fwu_component_t *candidates, uint8_t number)

  *Starts the installation of an image.*
- psa_status_t fwu_bootloader_mark_image_accepted (const psa_fwu_component_t *trials, uint8_t number)

  *Mark the TRIAL(running) image in component as confirmed.*
- psa_status_t fwu_bootloader_reject_staged_image (psa_fwu_component_t component)

  *Uninstall the staged image in the component.*
- psa_status_t fwu_bootloader_reject_trial_image (psa_fwu_component_t component)

  *Reject the trial image in the component.*
- psa_status_t fwu_bootloader_abort (psa_fwu_component_t component)
- psa_status_t fwu_bootloader_get_image_info (psa_fwu_component_t component, bool query_state, bool query_impl_info, psa_fwu_component_info_t *info)

  *Get the image information.*
- psa_status_t fwu_bootloader_clean_component (psa_fwu_component_t component)

  *The component is in FAILED or UPDATED state. Clean the staging area of the component.*

### 7.252.1 Macro Definition Documentation

#### 7.252.1.1 MAX_IMAGE_INFO_LENGTH

```
#define MAX_IMAGE_INFO_LENGTH
```

**Value:**

```
(MCUBOOT_IMAGE_NUMBER * \
(sizeof(struct image_version) + \
 SHARED_DATA_ENTRY_HEADER_SIZE))
```

Definition at line 24 of file tfm_mcuboot_fwu.c.

### 7.252.2 Typedef Documentation

#### 7.252.2.1 fwu_image_info_data_t

```
typedef struct fwu_image_info_data_s fwu_image_info_data_t
```

#### 7.252.2.2 tfm_fwu_mcuboot_ctx_t

```
typedef struct tfm_fwu_mcuboot_ctx_s tfm_fwu_mcuboot_ctx_t
```

### 7.252.3 Function Documentation

#### 7.252.3.1 fwu_bootloader_abort()

```
psa_status_t fwu_bootloader_abort (
            psa_fwu_component_t component )
```

Definition at line 487 of file tfm_mcuboot_fwu.c.

#### 7.252.3.2 fwu_bootloader_clean_component()

```
psa_status_t fwu_bootloader_clean_component (
            psa_fwu_component_t component )
```

The component is in FAILED or UPDATED state. Clean the staging area of the component.

**Returns**

PSA_SUCCESS On success PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR↩_STORAGE_FAILURE

Definition at line 659 of file tfm_mcuboot_fwu.c.

#### 7.252.3.3 fwu_bootloader_get_image_info()

```
psa_status_t fwu_bootloader_get_image_info (
            psa_fwu_component_t component,
            bool query_state,
            bool query_impl_info,
            psa_fwu_component_info_t * info )
```

Get the image information.
Get the image information of the given component in staging area or the running area.

**Parameters**

| | | |
|---|---|---|
| in | *component* | The identifier of the target component in bootloader. |
| in | *query_state* | Query 'state' field. |

**Parameters**

| | | |
|---|---|---|
| in | *query_impl_info* | Query 'impl' field. |
| out | *info* | Buffer containing return the component information. |

**Returns**

PSA_SUCCESS On success PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR↩_GENERIC_ERROR A fatal error occurred

Definition at line 586 of file tfm_mcuboot_fwu.c.

#### 7.252.3.4 fwu_bootloader_init()

```
psa_status_t fwu_bootloader_init (
            void  )
```

Bootloader related initialization for firmware update, such as reading some necessary shared data from the memory if needed.

**Returns**

PSA_SUCCESS On success PSA_ERROR_GENERIC_ERROR On failure

Definition at line 95 of file tfm_mcuboot_fwu.c.
Here is the caller graph for this function:

#### 7.252.3.5 fwu_bootloader_install_image()

```
psa_status_t fwu_bootloader_install_image (
            const psa_fwu_component_t * candidates,
            uint8_t number )
```

Starts the installation of an image.
The components are in CANDIDATE state. Check the authenticity and integrity of the staged image in the components. If a reboot is required to complete the check, then mark this image as a candidate so that the next time bootloader runs it will take this image as a candidate one to bootup. Return the error code PSA_SUCCESS_RE↩BOOT.

**Parameters**

| | | |
|---|---|---|
| in | *candidates* | A list of components in CANDIDATE state. |
| in | *number* | Number of components in CANDIDATE state. |

**Returns**

PSA_SUCCESS On success PSA_SUCCESS_REBOOT A system reboot is needed to finish installation TFM_SUCCESS_RESTART A restart of the updated component is required to complete the update PSA_ERROR_DEPENDENCY_NEEDED Another image needs to be installed to finish installation PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR_INVALID_SIGNATURE The signature is incorrect PSA_ERROR_GENERIC_ERROR A fatal error occurred TFM_ERROR_ROLLBACK_DETECTED The image is too old to be installed. If the image metadata contains a timestamp and it has expired, then this error is also returned.

Definition at line 201 of file tfm_mcuboot_fwu.c.

#### 7.252.3.6 fwu_bootloader_load_image()

```
psa_status_t fwu_bootloader_load_image (
            psa_fwu_component_t component,
            size_t image_offset,
            const void * block,
            size_t block_size )
```

Load the image into the target component.
The component is in WRITING state. Write the image data into the target component.

**Parameters**

| | | |
|---|---|---|
| in | *component* | The identifier of the target component in bootloader. |
| in | *image_offset* | The offset of the image being passed into block, in bytes |
| in | *block* | A buffer containing a block of image data. This might be a complete image or a subset. |
| in | *block_size* | Size of block. |

**Returns**

PSA_SUCCESS On success PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR_INSUFFICIENT_MEMORY There is no enough memory to process the update PSA_ERROR_INSUFFICIENT_STORAGE There is no enough storage to process the update PSA_ERROR_GENERIC_ERROR A fatal error occurred

Definition at line 138 of file tfm_mcuboot_fwu.c.

#### 7.252.3.7 fwu_bootloader_mark_image_accepted()

```
psa_status_t fwu_bootloader_mark_image_accepted (
            const psa_fwu_component_t * trials,
            uint8_t number )
```

Mark the TRIAL(running) image in component as confirmed.
Call this API to mark the running images as permanent/accepted to avoid revert when next time bootup. Usually, this API is called after the running images have been verified as valid.

**Parameters**

| | | |
|---|---|---|
| in | *candidates* | A list of components in TRIAL state. |
| in | *number* | Number of components in TRIAL state. |

**Returns**

PSA_SUCCESS On success PSA_ERROR_INSUFFICIENT_MEMORY PSA_ERROR_INSUFFICIENT_S↩ TORAGE PSA_ERROR_COMMUNICATION_FAILURE PSA_ERROR_STORAGE_FAILURE

Definition at line 389 of file tfm_mcuboot_fwu.c.

#### 7.252.3.8 fwu_bootloader_reject_staged_image()

```
psa_status_t fwu_bootloader_reject_staged_image (
            psa_fwu_component_t component )
```

Uninstall the staged image in the component.
The component is in STAGED state. Uninstall the staged image in the component so that this image will not be treated as a candidate next time bootup.

**Returns**

PSA_SUCCESS On success PSA_ERROR_INSUFFICIENT_MEMORY PSA_ERROR_INSUFFICIENT_S↩ TORAGE PSA_ERROR_COMMUNICATION_FAILURE PSA_ERROR_STORAGE_FAILURE

Definition at line 461 of file tfm_mcuboot_fwu.c.

#### 7.252.3.9 fwu_bootloader_reject_trial_image()

```
psa_status_t fwu_bootloader_reject_trial_image (
            psa_fwu_component_t component )
```

Reject the trial image in the component.
The component is in TRIAL state. Mark the running image in the component as rejected.

**Returns**

PSA_SUCCESS On success PSA_ERROR_INSUFFICIENT_MEMORY PSA_ERROR_INSUFFICIENT_S↩ TORAGE PSA_ERROR_COMMUNICATION_FAILURE PSA_ERROR_STORAGE_FAILURE

Definition at line 477 of file tfm_mcuboot_fwu.c.

#### 7.252.3.10 fwu_bootloader_staging_area_init()

```
psa_status_t fwu_bootloader_staging_area_init (
            psa_fwu_component_t component,
            const void * manifest,
            size_t manifest_size )
```

Initialize the staging area of the component.
The component is in READY state. Prepare the staging area of the component for image download. For example, initialize the staging area, open the flash area, and so on. The image will be written into the staging area later.

**Parameters**

| | | |
|---|---|---|
| in | *component* | The identifier of the target component in bootloader. |
| in | *manifest* | A pointer to a buffer containing a detached manifest for the update. If the manifest is bundled with the firmware image, manifest must be NULL. |
| in | *manifest_size* | Size of the manifest buffer in bytes. |

**Returns**

PSA_SUCCESS On success PSA_ERROR_INVALID_SIGNATURE A signature or integrity check on the manifest has failed. PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR_INSUFF↩ ICIENT_MEMORY PSA_ERROR_INSUFFICIENT_STORAGE PSA_ERROR_COMMUNICATION_FAILURE PSA_ERROR_STORAGE_FAILURE

Definition at line 108 of file tfm_mcuboot_fwu.c.

## 7.253 secure_fw/partitions/firmware_update/bootloader/tfm_↩ bootloader_fwu_abstraction.h File Reference

```
#include "stdbool.h"
#include "psa/update.h"
```

Include dependency graph for tfm_bootloader_fwu_abstraction.h:

This graph shows which files directly or indirectly include this file:

### Functions

- psa_status_t fwu_bootloader_init (void)
- psa_status_t fwu_bootloader_staging_area_init (psa_fwu_component_t component, const void *manifest, size_t manifest_size)

  *Initialize the staging area of the component.*
- psa_status_t fwu_bootloader_load_image (psa_fwu_component_t component, size_t image_offset, const void *block, size_t block_size)

  *Load the image into the target component.*

- psa_status_t fwu_bootloader_install_image (const psa_fwu_component_t *candidates, uint8_t number)

  *Starts the installation of an image.*
- psa_status_t fwu_bootloader_mark_image_accepted (const psa_fwu_component_t *trials, uint8_t number)

  *Mark the TRIAL(running) image in component as confirmed.*
- psa_status_t fwu_bootloader_reject_staged_image (psa_fwu_component_t component)

  *Uninstall the staged image in the component.*
- psa_status_t fwu_bootloader_reject_trial_image (psa_fwu_component_t component)

  *Reject the trial image in the component.*
- psa_status_t fwu_bootloader_clean_component (psa_fwu_component_t component)

  *The component is in FAILED or UPDATED state. Clean the staging area of the component.*
- psa_status_t fwu_bootloader_get_image_info (psa_fwu_component_t component, bool query_state, bool query_impl_info, psa_fwu_component_info_t *info)

  *Get the image information.*

## 7.253.1 Function Documentation

### 7.253.1.1 fwu_bootloader_clean_component()

```
psa_status_t fwu_bootloader_clean_component (
            psa_fwu_component_t component )
```

The component is in FAILED or UPDATED state. Clean the staging area of the component.

**Returns**

PSA_SUCCESS On success PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR↩_STORAGE_FAILURE

Definition at line 659 of file tfm_mcuboot_fwu.c.

### 7.253.1.2 fwu_bootloader_get_image_info()

```
psa_status_t fwu_bootloader_get_image_info (
            psa_fwu_component_t component,
            bool query_state,
            bool query_impl_info,
            psa_fwu_component_info_t * info )
```

Get the image information.
Get the image information of the given component in staging area or the running area.

**Parameters**

| | | |
|---|---|---|
| in | *component* | The identifier of the target component in bootloader. |
| in | *query_state* | Query 'state' field. |
| in | *query_impl_info* | Query 'impl' field. |
| out | *info* | Buffer containing return the component information. |

**Returns**

PSA_SUCCESS On success PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR↩_GENERIC_ERROR A fatal error occurred

Definition at line 586 of file tfm_mcuboot_fwu.c.

#### 7.253.1.3 fwu_bootloader_init()

```
psa_status_t fwu_bootloader_init (
            void  )
```

Bootloader related initialization for firmware update, such as reading some necessary shared data from the memory if needed.

**Returns**

PSA_SUCCESS On success PSA_ERROR_GENERIC_ERROR On failure

Definition at line 95 of file tfm_mcuboot_fwu.c.
Here is the caller graph for this function:

#### 7.253.1.4 fwu_bootloader_install_image()

```
psa_status_t fwu_bootloader_install_image (
            const psa_fwu_component_t * candidates,
            uint8_t number )
```

Starts the installation of an image.
The components are in CANDIDATE state. Check the authenticity and integrity of the staged image in the components. If a reboot is required to complete the check, then mark this image as a candidate so that the next time bootloader runs it will take this image as a candidate one to bootup. Return the error code PSA_SUCCESS_RE↩ BOOT.

**Parameters**

| | | |
|---|---|---|
| in | *candidates* | A list of components in CANDIDATE state. |
| in | *number* | Number of components in CANDIDATE state. |

**Returns**

PSA_SUCCESS On success PSA_SUCCESS_REBOOT A system reboot is needed to finish installation T↩ FM_SUCCESS_RESTART A restart of the updated component is required to complete the update PSA↩ _ERROR_DEPENDENCY_NEEDED Another image needs to be installed to finish installation PSA_ERR↩ OR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR_INVALID_SIGNATURE The signature is incorrect PSA_ERROR_GENERIC_ERROR A fatal error occurred TFM_ERROR_ROLLBACK_DETECTED The image is too old to be installed. If the image metadata contains a timestamp and it has expired, then this error is also returned.

Definition at line 201 of file tfm_mcuboot_fwu.c.

#### 7.253.1.5 fwu_bootloader_load_image()

```
psa_status_t fwu_bootloader_load_image (
            psa_fwu_component_t component,
            size_t image_offset,
```

```
const void * block,
size_t block_size )
```

Load the image into the target component.
The component is in WRITING state. Write the image data into the target component.

**Parameters**

| | | |
|---|---|---|
| in | *component* | The identifier of the target component in bootloader. |
| in | *image_offset* | The offset of the image being passed into block, in bytes |
| in | *block* | A buffer containing a block of image data. This might be a complete image or a subset. |
| in | *block_size* | Size of block. |

**Returns**

PSA_SUCCESS On success PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR↩_INSUFFICIENT_MEMORY There is no enough memory to process the update PSA_ERROR_INSUFFI↩CIENT_STORAGE There is no enough storage to process the update PSA_ERROR_GENERIC_ERROR A fatal error occurred

Definition at line 138 of file tfm_mcuboot_fwu.c.

#### 7.253.1.6 fwu_bootloader_mark_image_accepted()

```
psa_status_t fwu_bootloader_mark_image_accepted (
            const psa_fwu_component_t * trials,
            uint8_t number )
```

Mark the TRIAL(running) image in component as confirmed.
Call this API to mark the running images as permanent/accepted to avoid revert when next time bootup. Usually, this API is called after the running images have been verified as valid.

**Parameters**

| | | |
|---|---|---|
| in | *candidates* | A list of components in TRIAL state. |
| in | *number* | Number of components in TRIAL state. |

**Returns**

PSA_SUCCESS On success PSA_ERROR_INSUFFICIENT_MEMORY PSA_ERROR_INSUFFICIENT_S↩TORAGE PSA_ERROR_COMMUNICATION_FAILURE PSA_ERROR_STORAGE_FAILURE

Definition at line 389 of file tfm_mcuboot_fwu.c.

#### 7.253.1.7 fwu_bootloader_reject_staged_image()

```
psa_status_t fwu_bootloader_reject_staged_image (
            psa_fwu_component_t component )
```

Uninstall the staged image in the component.
The component is in STAGED state. Uninstall the staged image in the component so that this image will not be treated as a candidate next time bootup.

**Returns**

PSA_SUCCESS On success PSA_ERROR_INSUFFICIENT_MEMORY PSA_ERROR_INSUFFICIENT_S↩TORAGE PSA_ERROR_COMMUNICATION_FAILURE PSA_ERROR_STORAGE_FAILURE

Definition at line 461 of file tfm_mcuboot_fwu.c.

#### 7.253.1.8 fwu_bootloader_reject_trial_image()

```
psa_status_t fwu_bootloader_reject_trial_image (
            psa_fwu_component_t component )
```

Reject the trial image in the component.
The component is in TRIAL state. Mark the running image in the component as rejected.

**Returns**

PSA_SUCCESS On success PSA_ERROR_INSUFFICIENT_MEMORY PSA_ERROR_INSUFFICIENT_S↩ TORAGE PSA_ERROR_COMMUNICATION_FAILURE PSA_ERROR_STORAGE_FAILURE

Definition at line 477 of file tfm_mcuboot_fwu.c.

#### 7.253.1.9 fwu_bootloader_staging_area_init()

```
psa_status_t fwu_bootloader_staging_area_init (
            psa_fwu_component_t component,
            const void * manifest,
            size_t manifest_size )
```

Initialize the staging area of the component.
The component is in READY state. Prepare the staging area of the component for image download. For example, initialize the staging area, open the flash area, and so on. The image will be written into the staging area later.

**Parameters**

| | | |
|---|---|---|
| in | *component* | The identifier of the target component in bootloader. |
| in | *manifest* | A pointer to a buffer containing a detached manifest for the update. If the manifest is bundled with the firmware image, manifest must be NULL. |
| in | *manifest_size* | Size of the manifest buffer in bytes. |

**Returns**

PSA_SUCCESS On success PSA_ERROR_INVALID_SIGNATURE A signature or integrity check on the manifest has failed. PSA_ERROR_INVALID_ARGUMENT Invalid input parameter PSA_ERROR_INSUFF↩ ICIENT_MEMORY PSA_ERROR_INSUFFICIENT_STORAGE PSA_ERROR_COMMUNICATION_FAILURE PSA_ERROR_STORAGE_FAILURE

Definition at line 108 of file tfm_mcuboot_fwu.c.

## 7.254 secure_fw/partitions/firmware_update/tfm_fwu_req_mngr.c File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include "config_tfm.h"
#include "tfm_platform_api.h"
#include "tfm_bootloader_fwu_abstraction.h"
#include "psa/update.h"
#include "service_api.h"
#include "psa/service.h"
#include "psa_manifest/tfm_firmware_update.h"
#include "compiler_ext_defs.h"
```

Include dependency graph for tfm_fwu_req_mngr.c:

## Data Structures

- struct tfm_fwu_ctx_s

## Macros

- #define COMPONENTS_ITER(x) for ((x) = 0; (x) < (FWU_COMPONENT_NUMBER); (x)++)

## Typedefs

- typedef struct tfm_fwu_ctx_s tfm_fwu_ctx_t

## Functions

- psa_status_t tfm_fwu_reject (const psa_msg_t *msg)
- psa_status_t tfm_firmware_update_service_sfn (const psa_msg_t *msg)
- psa_status_t tfm_fwu_entry (void)

## 7.254.1 Macro Definition Documentation

### 7.254.1.1 COMPONENTS_ITER

```
#define COMPONENTS_ITER(
            x ) for ((x) = 0; (x) < (FWU_COMPONENT_NUMBER); (x)++)
```

Definition at line 20 of file tfm_fwu_req_mngr.c.

## 7.254.2 Typedef Documentation

### 7.254.2.1 tfm_fwu_ctx_t

```
typedef struct tfm_fwu_ctx_s tfm_fwu_ctx_t
```

## 7.254.3 Function Documentation

### 7.254.3.1 tfm_firmware_update_service_sfn()

```
psa_status_t tfm_firmware_update_service_sfn (
            const psa_msg_t * msg )
```

Definition at line 506 of file tfm_fwu_req_mngr.c.

#### 7.254.3.2 tfm_fwu_entry()

```
psa_status_t tfm_fwu_entry (
            void  )
```

Definition at line 534 of file tfm_fwu_req_mngr.c.
Here is the call graph for this function:

#### 7.254.3.3 tfm_fwu_reject()

```
psa_status_t tfm_fwu_reject (
            const psa_msg_t * msg )
```

Definition at line 355 of file tfm_fwu_req_mngr.c.
Here is the call graph for this function:

## 7.255 secure_fw/partitions/idle_partition/idle_partition.c File Reference

```
#include "tfm_hal_device_header.h"
#include "fih.h"
#include "psa/service.h"
```

Include dependency graph for idle_partition.c:

## Functions

- void tfm_idle_thread (void)

### 7.255.1 Function Documentation

#### 7.255.1.1 tfm_idle_thread()

```
void tfm_idle_thread (
            void  )
```

Definition at line 15 of file idle_partition.c.
Here is the call graph for this function:

## 7.256 secure_fw/partitions/idle_partition/load_info_idle_sp.c File Reference

```
#include <stdint.h>
#include <stddef.h>
```

```
#include "spm.h"
#include "load/partition_defs.h"
#include "load/service_defs.h"
#include "load/asset_defs.h"
#include "region_defs.h"
#include "tfm_s_linker_alignments.h"
```

Include dependency graph for load_info_idle_sp.c:

## Data Structures

- struct partition_tfm_sp_idle_load_info_t

## Macros

- #define IDLE_SP_STACK_SIZE ROUND_UP_TO_MULTIPLE(0x100, TFM_LINKER_IDLE_PARTITION_↩ STACK_ALIGNMENT)

## Functions

- void tfm_idle_thread (void)

## Variables

- uint8_t idle_sp_stack [ROUND_UP_TO_MULTIPLE(0x100, TFM_LINKER_IDLE_PARTITION_STACK_A↩ LIGNMENT)]
- const struct partition_tfm_sp_idle_load_info_t tfm_sp_idle_load

## 7.256.1 Macro Definition Documentation

### 7.256.1.1 IDLE_SP_STACK_SIZE

```
#define IDLE_SP_STACK_SIZE ROUND_UP_TO_MULTIPLE(0x100, TFM_LINKER_IDLE_PARTITION_STACK_ALIGN↩
MENT)
```

Definition at line 25 of file load_info_idle_sp.c.

## 7.256.2 Function Documentation

### 7.256.2.1 tfm_idle_thread()

```
void tfm_idle_thread (
            void  )
```

Definition at line 15 of file idle_partition.c.

Here is the call graph for this function:

### 7.256.3 Variable Documentation

#### 7.256.3.1 idle_sp_stack

```
uint8_t idle_sp_stack[ROUND_UP_TO_MULTIPLE(0x100, TFM_LINKER_IDLE_PARTITION_STACK_ALIGNMENT)]
```

Definition at line 42 of file load_info_idle_sp.c.

#### 7.256.3.2 tfm_sp_idle_load

```
const struct partition_tfm_sp_idle_load_info_t tfm_sp_idle_load
```

**Initial value:**

```
= {
    .load_info = {
        .psa_ff_ver                 = 0x0101 | PARTITION_INFO_MAGIC,
        .pid                        = TFM_SP_IDLE,
        .flags                      = PARTITION_PRI_LOWEST | PARTITION_MODEL_IPC
                                      | PARTITION_MODEL_PSA_ROT,
        .entry                      = ENTRY_TO_POSITION(tfm_idle_thread),
        .stack_size                 =  ROUND_UP_TO_MULTIPLE(0x100,
      TFM_LINKER_IDLE_PARTITION_STACK_ALIGNMENT) ,
        .heap_size                  = 0,
        .ndeps                      = 0,
        .nservices                  = 0,
        .nassets                    = 0,
    },
    .stack_addr                     = (uintptr_t)idle_sp_stack,
    .heap_addr                      = 0,
}
```

Definition at line 51 of file load_info_idle_sp.c.

## 7.257 secure_fw/partitions/initial_attestation/attest.h File Reference

```
#include "psa/initial_attestation.h"
#include "psa/client.h"
#include "tfm_boot_status.h"
```

Include dependency graph for attest.h:

This graph shows which files directly or indirectly include this file:

## Enumerations

- enum psa_attest_err_t { PSA_ATTEST_ERR_SUCCESS = 0, PSA_ATTEST_ERR_INIT_FAILED, PSA_ATTEST_ERR_BUFFER_OVERFLOW, PSA_ATTEST_ERR_CLAIM_UNAVAILABLE, PSA_ATTEST_ERR_INVALID_INPUT, PSA_ATTEST_ERR_GENERAL, PSA_ATTEST_ERR_FORCE_INT_SIZE = INT_MAX }

  *Initial attestation service error types.*

## Functions

- enum psa_attest_err_t attest_get_boot_data (uint8_t major_type, struct tfm_boot_data *boot_data, uint32_t len)

  *Copy the boot data (coming from boot loader) from shared memory area to service memory area.*
- enum psa_attest_err_t attest_get_caller_client_id (int32_t *caller_id)

  *Get the ID of the caller thread.*
- psa_status_t attest_init (void)

  *Initialise the initial attestation service during the TF-M boot up process.*
- psa_status_t initial_attest_get_token (const void *challenge_buf, size_t challenge_size, void *token_buf, size_t token_buf_size, size_t *token_size)

  *Get initial attestation token.*
- psa_status_t initial_attest_get_token_size (size_t challenge_size, size_t *token_size)

*Get the size of the initial attestation token.*

## 7.257.1 Enumeration Type Documentation

### 7.257.1.1 psa_attest_err_t

`enum psa_attest_err_t`

Initial attestation service error types.

**Enumerator**

| | |
|---|---|
| PSA_ATTEST_ERR_SUCCESS | Action was performed successfully |
| PSA_ATTEST_ERR_INIT_FAILED | Boot status data is unavailable or malformed |
| PSA_ATTEST_ERR_BUFFER_OVERFLOW | Buffer is too small to store required data |
| PSA_ATTEST_ERR_CLAIM_UNAVAILABLE | Some of the mandatory claims are unavailable |
| PSA_ATTEST_ERR_INVALID_INPUT | Some parameter or combination of parameters are recognised as invalid:<br>• challenge size is not allowed<br>• challenge object is unavailable<br>• token buffer is unavailable |
| PSA_ATTEST_ERR_GENERAL | Unexpected error happened during operation |
| PSA_ATTEST_ERR_FORCE_INT_SIZE | Following entry is only to ensure the error code of integer size |

Definition at line 25 of file attest.h.

## 7.257.2 Function Documentation

### 7.257.2.1 attest_get_boot_data()

```
enum psa_attest_err_t attest_get_boot_data (
            uint8_t major_type,
            struct tfm_boot_data * boot_data,
            uint32_t len )
```

Copy the boot data (coming from boot loader) from shared memory area to service memory area.

**Parameters**

| | | |
|---|---|---|
| in | *major_type* | Major type of TLV entries to copy |
| out | *ptr* | Pointer to the buffer to store the boot data \parma[in] len Size of the buffer to store the boot data |

**Returns**

Returns error code as specified in psa_attest_err_t

Definition at line 26 of file tfm_attest.c.
Here is the caller graph for this function:

### 7.257.2.2 attest_get_caller_client_id()

```
enum psa_attest_err_t attest_get_caller_client_id (
            int32_t * caller_id )
```

Get the ID of the caller thread.

**Parameters**

| | | |
|---|---|---|
| out | *caller↩ _id* | Pointer where to store caller ID |

**Returns**

Returns error code as specified in psa_attest_err_t

Definition at line 17 of file tfm_attest.c.
Here is the caller graph for this function:

### 7.257.2.3 attest_init()

```
psa_status_t attest_init (
            void )
```

Initialise the initial attestation service during the TF-M boot up process.

**Returns**

Returns PSA_SUCCESS if init has been completed, otherwise error as specified in psa_status_t

Definition at line 65 of file attest_core.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.257.2.4 initial_attest_get_token()

```
psa_status_t initial_attest_get_token (
            const void * challenge_buf,
            size_t challenge_size,
            void * token_buf,
            size_t token_buf_size,
            size_t * token_size )
```

Get initial attestation token.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Pointer to in_vec array, which contains input data to attestation service |
| in | *num_invec* | Number of elements in in_vec array |
| in,out | *out_vec* | Pointer out_vec array, which contains output data to attestation service |
| in | *num_outvec* | Number of elements in out_vec array |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 716 of file attest_core.c.

#### 7.257.2.5 initial_attest_get_token_size()

```
psa_status_t initial_attest_get_token_size (
            size_t challenge_size,
            size_t * token_size )
```

Get the size of the initial attestation token.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Pointer to in_vec array, which contains input data to attestation service |
| in | *num_invec* | Number of elements in in_vec array |
| out | *out_vec* | Pointer to out_vec array, which contains pointer where to store the output data |
| in | *num_outvec* | Number of elements in out_vec array |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 752 of file attest_core.c.

## 7.258 secure_fw/partitions/initial_attestation/attest_asymmetric_key.c File Reference

```
#include "attest_key.h"
#include <stdint.h>
#include <stddef.h>
#include "config_tfm.h"
#include "tfm_plat_defs.h"
#include "tfm_plat_device_id.h"
#include "t_cose_standard_constants.h"
#include "q_useful_buf.h"
#include "qcbor/qcbor.h"
#include "tfm_crypto_defs.h"
```

Include dependency graph for attest_asymmetric_key.c:

### Macros

- #define ATTEST_ECC_PUBLIC_KEY_SIZE PSA_KEY_EXPORT_ECC_PUBLIC_KEY_MAX_SIZE(ATTE↩ ST_KEY_BITS)
- #define ECC_P256_COORD_SIZE PSA_BITS_TO_BYTES(256) /∗ 256 bits -> 32 bytes ∗/

### Functions

- enum psa_attest_err_t attest_get_instance_id (struct q_useful_buf_c ∗id_buf)

  *Get the buffer of Instance ID data.*

### 7.258.1 Macro Definition Documentation

#### 7.258.1.1 ATTEST_ECC_PUBLIC_KEY_SIZE

```
#define ATTEST_ECC_PUBLIC_KEY_SIZE PSA_KEY_EXPORT_ECC_PUBLIC_KEY_MAX_SIZE(ATTEST_KEY_BITS)
```

Definition at line 20 of file attest_asymmetric_key.c.

#### 7.258.1.2 ECC_P256_COORD_SIZE

```
#define ECC_P256_COORD_SIZE PSA_BITS_TO_BYTES(256) /* 256 bits -> 32 bytes */
```

The size of X and Y coordinate in 2 parameter style EC public key. Format is as defined in COSE (RFC 8152) and SEC 1: Elliptic Curve Cryptography.
This size is well-known and documented in public standards.
Definition at line 31 of file attest_asymmetric_key.c.

### 7.258.2 Function Documentation

#### 7.258.2.1 attest_get_instance_id()

```
enum psa_attest_err_t attest_get_instance_id (
            struct q_useful_buf_c * id_buf )
```

Get the buffer of Instance ID data.

**Parameters**

| | | |
|---|---|---|
| out | *id_buf* | Address and length of Instance ID buffer |

**Return values**

| | |
|---|---|
| *PSA_ATTEST_ERR_SUCCESS* | Instance ID was successfully returned. |
| *PSA_ATTEST_ERR_CLAIM_UNAVAILABLE* | Instance ID is unavailable |
| *PSA_ATTEST_ERR_GENERAL* | Instance ID could not be returned. |

Definition at line 116 of file attest_asymmetric_key.c.

## 7.259 secure_fw/partitions/initial_attestation/attest_boot_data.c File Reference

```
#include <string.h>
#include <stddef.h>
#include <stdbool.h>
#include "attest_boot_data.h"
#include "tfm_boot_status.h"
#include "tfm_attest_iat_defs.h"
#include "q_useful_buf.h"
#include "static_checks.h"
```

Include dependency graph for attest_boot_data.c:

## Data Structures

- struct attest_boot_data

  *Contains the received boot status information from bootloader.*

## Macros

- #define MAX_BOOT_STATUS 512

## Functions

- enum psa_attest_err_t attest_encode_sw_components_array (QCBOREncodeContext ∗encode_ctx, const int32_t ∗map_label, uint32_t ∗cnt)

  *Function to encode all the software components.*
- enum psa_attest_err_t attest_boot_data_init (void)

  *Gets the IAS TLV entries (boot data coming from boot loader) from shared memory area to service memory area.*

## 7.259.1 Macro Definition Documentation

### 7.259.1.1 MAX_BOOT_STATUS

```
#define MAX_BOOT_STATUS 512
```

Definition at line 23 of file attest_boot_data.c.

## 7.259.2 Function Documentation

### 7.259.2.1 attest_boot_data_init()

```
enum psa_attest_err_t attest_boot_data_init (
            void  )
```

Gets the IAS TLV entries (boot data coming from boot loader) from shared memory area to service memory area.

**Returns**

Returns error code as specified in psa_attest_err_t

Definition at line 344 of file attest_boot_data.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.259.2.2 attest_encode_sw_components_array()

```
enum psa_attest_err_t attest_encode_sw_components_array (
            QCBOREncodeContext * encode_ctx,
            const int32_t * map_label,
            uint32_t * cnt )
```

Function to encode all the software components.
The function creates a CBOR array in which 1 item is a SW component. The encoded list of these software components makes up the SW components claim. This encoded array can be stand-alone or part of a map, depending on the `map_label` parameter. The CBOR array is only created if at least 1 SW component has been found.

**Parameters**

| | | |
|---|---|---|
| out | *encode_ctx* | Pointer to the encoding context |
| in | *map_label* | Label/key of the map item to be added to the context. If NULL, the SW components are added as a stand-alone array |
| out | *cnt* | Number of SW component in the encoded array |

**Returns**

Returns error code as specified in psa_attest_err_t

Definition at line 211 of file attest_boot_data.c.

## 7.260 secure_fw/partitions/initial_attestation/attest_boot_data.h File Reference

```
#include <stdint.h>
```

```
#include "attest.h"
#include "psa/initial_attestation.h"
#include "qcbor/qcbor.h"
```

Include dependency graph for attest_boot_data.h:

This graph shows which files directly or indirectly include this file:

## Functions

- enum psa_attest_err_t attest_encode_sw_components_array (QCBOREncodeContext ∗encode_ctx, const int32_t ∗map_label, uint32_t ∗cnt)

  *Function to encode all the software components.*

- enum psa_attest_err_t attest_boot_data_init (void)

  *Gets the IAS TLV entries (boot data coming from boot loader) from shared memory area to service memory area.*

## 7.260.1 Function Documentation

### 7.260.1.1 attest_boot_data_init()

```
enum psa_attest_err_t attest_boot_data_init (
            void  )
```

Gets the IAS TLV entries (boot data coming from boot loader) from shared memory area to service memory area.

**Returns**

Returns error code as specified in psa_attest_err_t

Definition at line 344 of file attest_boot_data.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.260.1.2 attest_encode_sw_components_array()

```
enum psa_attest_err_t attest_encode_sw_components_array (
            QCBOREncodeContext * encode_ctx,
            const int32_t * map_label,
            uint32_t * cnt )
```

Function to encode all the software components.
The function creates a CBOR array in which 1 item is a SW component. The encoded list of these software components makes up the SW components claim. This encoded array can be stand-alone or part of a map, depending on the `map_label` parameter. The CBOR array is only created if at least 1 SW component has been found.

**Parameters**

| | | |
|---|---|---|
| out | *encode_ctx* | Pointer to the encoding context |
| in | *map_label* | Label/key of the map item to be added to the context. If NULL, the SW components are added as a stand-alone array |
| out | *cnt* | Number of SW component in the encoded array |

**Returns**

Returns error code as specified in psa_attest_err_t

Definition at line 211 of file attest_boot_data.c.

## 7.261 secure_fw/partitions/initial_attestation/attest_core.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <stddef.h>
#include "psa/client.h"
#include "attest.h"
#include "attest_boot_data.h"
#include "attest_key.h"
#include "attest_token.h"
#include "config_tfm.h"
#include "tfm_plat_defs.h"
#include "tfm_plat_device_id.h"
#include "tfm_plat_boot_seed.h"
#include "tfm_attest_hal.h"
#include "tfm_attest_iat_defs.h"
#include "t_cose_common.h"
#include "tfm_crypto_defs.h"
#include "tfm_sp_log.h"
```

Include dependency graph for attest_core.c:

### Macros

- #define ARRAY_LENGTH(array) (sizeof(array) / sizeof(∗(array)))

### Functions

- psa_status_t attest_init (void)

  *Initialise the initial attestation service during the TF-M boot up process.*

- psa_status_t initial_attest_get_token (const void ∗challenge_buf, size_t challenge_size, void ∗token_buf, size_t token_buf_size, size_t ∗token_size)

  *Get initial attestation token.*

- psa_status_t initial_attest_get_token_size (const size_t challenge_size, size_t ∗token_size)

  *Get the size of the initial attestation token.*

### 7.261.1 Macro Definition Documentation

#### 7.261.1.1 ARRAY_LENGTH

```
#define ARRAY_LENGTH(
            array ) (sizeof(array) / sizeof(*(array)))
```

Definition at line 28 of file attest_core.c.

### 7.261.2 Function Documentation

#### 7.261.2.1 attest_init()

```
psa_status_t attest_init (
            void  )
```

Initialise the initial attestation service during the TF-M boot up process.

**Returns**

Returns PSA_SUCCESS if init has been completed, otherwise error as specified in psa_status_t

Definition at line 65 of file attest_core.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.261.2.2 initial_attest_get_token()

```
psa_status_t initial_attest_get_token (
            const void * challenge_buf,
            size_t challenge_size,
            void * token_buf,
            size_t token_buf_size,
            size_t * token_size )
```

Get initial attestation token.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Pointer to in_vec array, which contains input data to attestation service |
| in | *num_invec* | Number of elements in in_vec array |
| in,out | *out_vec* | Pointer out_vec array, which contains output data to attestation service |
| in | *num_outvec* | Number of elements in out_vec array |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 716 of file attest_core.c.

#### 7.261.2.3 initial_attest_get_token_size()

```
psa_status_t initial_attest_get_token_size (
            size_t challenge_size,
            size_t * token_size )
```

Get the size of the initial attestation token.

**Parameters**

| | | |
|---|---|---|
| in | *in_vec* | Pointer to in_vec array, which contains input data to attestation service |
| in | *num_invec* | Number of elements in in_vec array |
| out | *out_vec* | Pointer to out_vec array, which contains pointer where to store the output data |
| in | *num_outvec* | Number of elements in out_vec array |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 752 of file attest_core.c.

## 7.262 secure_fw/partitions/initial_attestation/attest_key.h File Reference

```
#include "attest.h"
#include "config_tfm.h"
#include "psa/initial_attestation.h"
#include "psa/crypto.h"
#include "q_useful_buf.h"
```

Include dependency graph for attest_key.h:

This graph shows which files directly or indirectly include this file:

## Functions

- enum psa_attest_err_t attest_get_instance_id (struct q_useful_buf_c *id_buf)

  *Get the buffer of Instance ID data.*

### 7.262.1 Function Documentation

#### 7.262.1.1 attest_get_instance_id()

```
enum psa_attest_err_t attest_get_instance_id (
            struct q_useful_buf_c * id_buf )
```

Get the buffer of Instance ID data.

**Parameters**

| | | |
|---|---|---|
| out | *id_buf* | Address and length of Instance ID buffer |

**Return values**

| | |
|---|---|
| *PSA_ATTEST_ERR_SUCCESS* | Instance ID was successfully returned. |
| *PSA_ATTEST_ERR_CLAIM_UNAVAILABLE* | Instance ID is unavailable |
| *PSA_ATTEST_ERR_GENERAL* | Instance ID could not be returned. |

Definition at line 116 of file attest_asymmetric_key.c.

## 7.263 secure_fw/partitions/initial_attestation/attest_symmetric_key.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <string.h>
#include "attest_key.h"
#include "config_tfm.h"
#include "tfm_plat_defs.h"
#include "psa/crypto.h"
#include "tfm_crypto_defs.h"
```

Include dependency graph for attest_symmetric_key.c:

## Macros

- #define SYMMETRIC_IAK_MAX_SIZE PSA_MAC_MAX_SIZE
- #define INSTANCE_ID_HASH_ALG PSA_ALG_SHA_256
- #define KID_BUF_LEN 32

## Functions

- enum psa_attest_err_t attest_get_instance_id (struct q_useful_buf_c ∗id_buf)

  *Get the buffer of Instance ID data.*

## 7.263.1 Macro Definition Documentation

### 7.263.1.1 INSTANCE_ID_HASH_ALG

```
#define INSTANCE_ID_HASH_ALG PSA_ALG_SHA_256
```

Definition at line 23 of file attest_symmetric_key.c.

### 7.263.1.2 KID_BUF_LEN

```
#define KID_BUF_LEN 32
```

Definition at line 26 of file attest_symmetric_key.c.

### 7.263.1.3 SYMMETRIC_IAK_MAX_SIZE

```
#define SYMMETRIC_IAK_MAX_SIZE PSA_MAC_MAX_SIZE
```

Definition at line 20 of file attest_symmetric_key.c.

## 7.263.2 Function Documentation

### 7.263.2.1 attest_get_instance_id()

```
enum psa_attest_err_t attest_get_instance_id (
            struct q_useful_buf_c * id_buf )
```

Get the buffer of Instance ID data.

**Parameters**

| | | |
|---|---|---|
| out | *id_buf* | Address and length of Instance ID buffer |

**Return values**

| | |
|---|---|
| *PSA_ATTEST_ERR_SUCCESS* | Instance ID was successfully returned. |
| *PSA_ATTEST_ERR_CLAIM_UNAVAILABLE* | Instance ID is unavailable |
| *PSA_ATTEST_ERR_GENERAL* | Instance ID could not be returned. |

Definition at line 129 of file attest_symmetric_key.c.

## 7.264 secure_fw/partitions/initial_attestation/attest_token.h File Reference

Attestation Token Creation Interface.

```
#include <stdint.h>
#include "qcbor/qcbor.h"
#include "t_cose_sign1_sign.h"
```

Include dependency graph for attest_token.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct attest_token_encode_ctx

## Macros

- #define TOKEN_OPT_OMIT_CLAIMS 0x40000000
- #define TOKEN_OPT_SHORT_CIRCUIT_SIGN 0x80000000

## Enumerations

- enum attest_token_err_t {
  ATTEST_TOKEN_ERR_SUCCESS = 0, ATTEST_TOKEN_ERR_TOO_SMALL, ATTEST_TOKEN_ERR_CBOR_FORMATTIN
  ATTEST_TOKEN_ERR_GENERAL,
  ATTEST_TOKEN_ERR_HASH_UNAVAILABLE, ATTEST_TOKEN_ERR_CBOR_NOT_WELL_FORMED,
  ATTEST_TOKEN_ERR_CBOR_STRUCTURE, ATTEST_TOKEN_ERR_CBOR_TYPE,
  ATTEST_TOKEN_ERR_INTEGER_VALUE, ATTEST_TOKEN_ERR_COSE_FORMAT, ATTEST_TOKEN_ERR_COSE_VALI
  ATTEST_TOKEN_ERR_UNSUPPORTED_SIG_ALG,
  ATTEST_TOKEN_ERR_INSUFFICIENT_MEMORY, ATTEST_TOKEN_ERR_TAMPERING_DETECTED,
  ATTEST_TOKEN_ERR_SIGNING_KEY, ATTEST_TOKEN_ERR_VERIFICATION_KEY,
  ATTEST_TOKEN_ERR_NO_VALID_TOKEN, ATTEST_TOKEN_ERR_NOT_FOUND, ATTEST_TOKEN_ERR_SW_COMPON
  }

## Functions

- enum attest_token_err_t attest_token_encode_start (struct attest_token_encode_ctx *me, uint32_t opt_↩ flags, int32_t key_select, int32_t cose_alg_id, const struct q_useful_buf *out_buffer)

  *Initialize a token creation context.*
- QCBOREncodeContext * attest_token_encode_borrow_cbor_cntxt (struct attest_token_encode_ctx *me)

  *Get a copy of the CBOR encoding context.*
- void attest_token_encode_add_integer (struct attest_token_encode_ctx *me, int32_t label, int64_t value)

  *Add a 64-bit signed integer claim.*
- void attest_token_encode_add_bstr (struct attest_token_encode_ctx *me, int32_t label, const struct q_↩ useful_buf_c *value)

  *Add a binary string claim.*

- void attest_token_encode_add_tstr (struct attest_token_encode_ctx *me, int32_t label, const struct q_useful_buf_c *value)

  *Add a text string claim.*
- void attest_token_encode_add_cbor (struct attest_token_encode_ctx *me, int32_t label, const struct q_useful_buf_c *encoded)

  *Add some already-encoded CBOR to payload.*
- enum attest_token_err_t attest_token_encode_finish (struct attest_token_encode_ctx *me, struct q_useful_buf_c *completed_token)

  *Finish the token, complete the signing and get the result.*

## 7.264.1 Detailed Description

Attestation Token Creation Interface.
The context and functions here are the way to create an attestation token. The steps are roughly:

1. Create and initialize an attest_token_encode_ctx indicating the options, key and such using attest_token_encode_start().
2. Use various add methods to fill in the payload with claims. The encoding context can also be borrowed for more rich payloads.
3. Call attest_token_encode_finish() to create the signature and finish formatting the COSE signed output.

## 7.264.2 Macro Definition Documentation

### 7.264.2.1 TOKEN_OPT_OMIT_CLAIMS

```
#define TOKEN_OPT_OMIT_CLAIMS 0x40000000
```

Request that the claims internally generated not be added to the token. This is a test mode that results in a static token that never changes. Only the nonce is included. The nonce is under the callers control unlike the other claims.
Definition at line 103 of file attest_token.h.

### 7.264.2.2 TOKEN_OPT_SHORT_CIRCUIT_SIGN

```
#define TOKEN_OPT_SHORT_CIRCUIT_SIGN 0x80000000
```

A special test mode where a proper signature is not produced. In its place there is a concatenation of hashes of the payload to be the same size as the signature. This works and can be used to verify all of the SW stack except the public signature part. The token has no security value in this mode because anyone can replicate it.
Definition at line 112 of file attest_token.h.

## 7.264.3 Enumeration Type Documentation

### 7.264.3.1 attest_token_err_t

```
enum attest_token_err_t
```

Error codes returned from attestation token creation.

**Enumerator**

| | |
|---|---|
| ATTEST_TOKEN_ERR_SUCCESS | Success |
| ATTEST_TOKEN_ERR_TOO_SMALL | The buffer passed in to receive the output is too small. |
| ATTEST_TOKEN_ERR_CBOR_FORMATTING | Something went wrong formatting the CBOR, most likely the payload has maps or arrays that are not closed. |

**Enumerator**

| | |
|---|---|
| ATTEST_TOKEN_ERR_GENERAL | A general, unspecific error when creating or decoding the token. |
| ATTEST_TOKEN_ERR_HASH_UNAVAILABLE | A hash function that is needed to make the token is not available. |
| ATTEST_TOKEN_ERR_CBOR_NOT_WELL_FOR↩MED | CBOR Syntax not well-formed – a CBOR syntax error. |
| ATTEST_TOKEN_ERR_CBOR_STRUCTURE | Bad CBOR structure, for example not a map when was is required. |
| ATTEST_TOKEN_ERR_CBOR_TYPE | Bad CBOR type, for example an not a text string, when a text string is required. |
| ATTEST_TOKEN_ERR_INTEGER_VALUE | Integer too large, for example an `int32_t` is required, but value only fits in `int64_t` |
| ATTEST_TOKEN_ERR_COSE_FORMAT | Something is wrong with the COSE message structure, missing headers or such. |
| ATTEST_TOKEN_ERR_COSE_VALIDATION | COSE signature or authentication tag is invalid, data is corrupted. |
| ATTEST_TOKEN_ERR_UNSUPPORTED_SIG_ALG | The signing algorithm is not supported. |
| ATTEST_TOKEN_ERR_INSUFFICIENT_MEMORY | Out of memory. |
| ATTEST_TOKEN_ERR_TAMPERING_DETECTED | Tampering detected in cryptographic function. |
| ATTEST_TOKEN_ERR_SIGNING_KEY | Signing key is not found or of wrong type. |
| ATTEST_TOKEN_ERR_VERIFICATION_KEY | Verification key is not found or of wrong type. |
| ATTEST_TOKEN_ERR_NO_VALID_TOKEN | No token was given or validated |
| ATTEST_TOKEN_ERR_NOT_FOUND | Data item with label wasn't found. |
| ATTEST_TOKEN_ERR_SW_COMPONENTS_MI↩SSING | SW Components absence not correctly indicated. |

Definition at line 48 of file attest_token.h.

## 7.264.4 Function Documentation

### 7.264.4.1 attest_token_encode_add_bstr()

```
void attest_token_encode_add_bstr (
            struct attest_token_encode_ctx * me,
            int32_t label,
            const struct q_useful_buf_c * value )
```

Add a binary string claim.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |
| in | *label* | Integer label for claim. |
| in | *value* | The binary claim data. |

Definition at line 346 of file attest_token_encode.c.

### 7.264.4.2 attest_token_encode_add_cbor()

```
void attest_token_encode_add_cbor (
            struct attest_token_encode_ctx * me,
```

```
            int32_t label,
            const struct q_useful_buf_c * encoded )
```

Add some already-encoded CBOR to payload.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |
| in | *label* | Integer label for claim. |
| in | *encoded* | The already-encoded CBOR. |

Encoded CBOR must be a full map or full array or a non-aggregate type. It cannot be a partial map or array. It can be nested maps and arrays, but they must all be complete.
Definition at line 370 of file attest_token_encode.c.

#### 7.264.4.3 attest_token_encode_add_integer()

```
void attest_token_encode_add_integer (
            struct attest_token_encode_ctx * me,
            int32_t label,
            int64_t value )
```

Add a 64-bit signed integer claim.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |
| in | *label* | Integer label for claim. |
| in | *value* | The integer claim data. |

Definition at line 335 of file attest_token_encode.c.

#### 7.264.4.4 attest_token_encode_add_tstr()

```
void attest_token_encode_add_tstr (
            struct attest_token_encode_ctx * me,
            int32_t label,
            const struct q_useful_buf_c * value )
```

Add a text string claim.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |
| in | *label* | Integer label for claim. |
| in | *value* | The text claim data. |

Definition at line 359 of file attest_token_encode.c.

#### 7.264.4.5 attest_token_encode_borrow_cbor_cntxt()

```
QCBOREncodeContext* attest_token_encode_borrow_cbor_cntxt (
            struct attest_token_encode_ctx * me )
```

Get a copy of the CBOR encoding context.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |

**Returns**

The CBOR encoding context

Allows the caller to encode CBOR right into the output buffer using any of the `QCBOREncode_AddXXXX()` methods. Anything added here will be part of the payload that gets hashed. This can be used to make complex CBOR structures. All open arrays and maps must be close before calling any other `attest_token_encode` methods. `QCBOREncode_Finish()` should not be closed on this context.
Definition at line 326 of file attest_token_encode.c.

#### 7.264.4.6 attest_token_encode_finish()

```
enum attest_token_err_t attest_token_encode_finish (
            struct attest_token_encode_ctx * me,
            struct q_useful_buf_c * completed_token )
```

Finish the token, complete the signing and get the result.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token Creation Context. |
| out | *completed_token* | Pointer and length to completed token. |

**Returns**

one of the attest_token_err_t errors.

This completes the token after the payload has been added. When this is called the signing algorithm is run and the final formatting of the token is completed.
Definition at line 284 of file attest_token_encode.c.

#### 7.264.4.7 attest_token_encode_start()

```
enum attest_token_err_t attest_token_encode_start (
            struct attest_token_encode_ctx * me,
            uint32_t opt_flags,
            int32_t key_select,
            int32_t cose_alg_id,
            const struct q_useful_buf * out_buffer )
```

Initialize a token creation context.

**Parameters**

| | | |
|---|---|---|
| in | *me* | The token creation context to be initialized. |
| in | *opt_flags* | Flags to select different custom options, for example TOKEN_OPT_OMIT_CLAIMS. |
| in | *key_select* | Selects which attestation key to sign with. |
| in | *cose_alg↩ _id* | The algorithm to sign with. The IDs are defined in `COSE (RFC 8152)` or in the `IANA COSE Registry`. |
| out | *out_buffer* | The output buffer to write the encoded token into. |

**Returns**

one of the attest_token_err_t errors.

The size of the buffer in `out_buffer->len` determines the size of the token that can be created. It must be able to hold the final encoded and signed token. The data encoding overhead is just that of CBOR. The signing overhead depends on the signing key size. It is about 150 bytes for 256-bit ECDSA.
If `out_buffer->ptr` is `NULL` and `out_buffer_ptr->len` is large like `UINT32_MAX` no token will be created but the length of the token that would be created will be in `completed_token` as returned by

attest_token_encode_finish(). None of the cryptographic functions run during this, but the sizes of what they would output is taken into account.
Definition at line 227 of file attest_token_encode.c.

## 7.265 secure_fw/partitions/initial_attestation/attest_token_encode.c File Reference

Attestation token creation implementation.

```
#include "attest_token.h"
#include "config_tfm.h"
#include "qcbor/qcbor.h"
#include "t_cose_sign1_sign.h"
#include "t_cose_common.h"
#include "q_useful_buf.h"
#include "psa/crypto.h"
#include "attest_key.h"
#include "tfm_crypto_defs.h"
```

Include dependency graph for attest_token_encode.c:

### Functions

- enum attest_token_err_t attest_token_encode_start (struct attest_token_encode_ctx ∗me, uint32_t opt_↩ flags, int32_t key_select, int32_t cose_alg_id, const struct q_useful_buf ∗out_buf)

  *Initialize a token creation context.*
- enum attest_token_err_t attest_token_encode_finish (struct attest_token_encode_ctx ∗me, struct q_useful↩ _buf_c ∗completed_token)

  *Finish the token, complete the signing and get the result.*
- QCBOREncodeContext ∗ attest_token_encode_borrow_cbor_cntxt (struct attest_token_encode_ctx ∗me)

  *Get a copy of the CBOR encoding context.*
- void attest_token_encode_add_integer (struct attest_token_encode_ctx ∗me, int32_t label, int64_t Value)

  *Add a 64-bit signed integer claim.*
- void attest_token_encode_add_bstr (struct attest_token_encode_ctx ∗me, int32_t label, const struct q_↩ useful_buf_c ∗bstr)

  *Add a binary string claim.*
- void attest_token_encode_add_tstr (struct attest_token_encode_ctx ∗me, int32_t label, const struct q_↩ useful_buf_c ∗tstr)

  *Add a text string claim.*
- void attest_token_encode_add_cbor (struct attest_token_encode_ctx ∗me, int32_t label, const struct q_↩ useful_buf_c ∗encoded)

  *Add some already-encoded CBOR to payload.*

## 7.265.1 Detailed Description

Attestation token creation implementation.

## 7.265.2 Function Documentation

### 7.265.2.1 attest_token_encode_add_bstr()

```
void attest_token_encode_add_bstr (
            struct attest_token_encode_ctx * me,
            int32_t label,
            const struct q_useful_buf_c * value )
```

Add a binary string claim.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |
| in | *label* | Integer label for claim. |
| in | *value* | The binary claim data. |

Definition at line 346 of file attest_token_encode.c.

### 7.265.2.2 attest_token_encode_add_cbor()

```
void attest_token_encode_add_cbor (
            struct attest_token_encode_ctx * me,
            int32_t label,
            const struct q_useful_buf_c * encoded )
```

Add some already-encoded CBOR to payload.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |
| in | *label* | Integer label for claim. |
| in | *encoded* | The already-encoded CBOR. |

Encoded CBOR must be a full map or full array or a non-aggregate type. It cannot be a partial map or array. It can be nested maps and arrays, but they must all be complete.
Definition at line 370 of file attest_token_encode.c.

### 7.265.2.3 attest_token_encode_add_integer()

```
void attest_token_encode_add_integer (
            struct attest_token_encode_ctx * me,
            int32_t label,
            int64_t value )
```

Add a 64-bit signed integer claim.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |
| in | *label* | Integer label for claim. |
| in | *value* | The integer claim data. |

Definition at line 335 of file attest_token_encode.c.

#### 7.265.2.4 attest_token_encode_add_tstr()

```
void attest_token_encode_add_tstr (
            struct attest_token_encode_ctx * me,
            int32_t label,
            const struct q_useful_buf_c * value )
```

Add a text string claim.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |
| in | *label* | Integer label for claim. |
| in | *value* | The text claim data. |

Definition at line 359 of file attest_token_encode.c.

#### 7.265.2.5 attest_token_encode_borrow_cbor_cntxt()

```
QCBOREncodeContext* attest_token_encode_borrow_cbor_cntxt (
            struct attest_token_encode_ctx * me )
```

Get a copy of the CBOR encoding context.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token creation context. |

**Returns**

The CBOR encoding context

Allows the caller to encode CBOR right into the output buffer using any of the `QCBOREncode_AddXXXX()` methods. Anything added here will be part of the payload that gets hashed. This can be used to make complex CBOR structures. All open arrays and maps must be close before calling any other `attest_token_encode` methods. `QCBOREncode_Finish()` should not be closed on this context.
Definition at line 326 of file attest_token_encode.c.

#### 7.265.2.6 attest_token_encode_finish()

```
enum attest_token_err_t attest_token_encode_finish (
            struct attest_token_encode_ctx * me,
            struct q_useful_buf_c * completed_token )
```

Finish the token, complete the signing and get the result.

**Parameters**

| | | |
|---|---|---|
| in | *me* | Token Creation Context. |
| out | *completed_token* | Pointer and length to completed token. |

**Returns**

one of the attest_token_err_t errors.

This completes the token after the payload has been added. When this is called the signing algorithm is run and the final formatting of the token is completed.

Definition at line 284 of file attest_token_encode.c.

#### 7.265.2.7 attest_token_encode_start()

```
enum attest_token_err_t attest_token_encode_start (
            struct attest_token_encode_ctx * me,
            uint32_t opt_flags,
            int32_t key_select,
            int32_t cose_alg_id,
            const struct q_useful_buf * out_buffer )
```

Initialize a token creation context.

**Parameters**

| | | |
|---|---|---|
| in | *me* | The token creation context to be initialized. |
| in | *opt_flags* | Flags to select different custom options, for example TOKEN_OPT_OMIT_CLAIMS. |
| in | *key_select* | Selects which attestation key to sign with. |
| in | *cose_alg↩ _id* | The algorithm to sign with. The IDs are defined in COSE (RFC 8152) or in the IANA COSE Registry. |
| out | *out_buffer* | The output buffer to write the encoded token into. |

**Returns**

one of the attest_token_err_t errors.

The size of the buffer in `out_buffer->len` determines the size of the token that can be created. It must be able to hold the final encoded and signed token. The data encoding overhead is just that of CBOR. The signing overhead depends on the signing key size. It is about 150 bytes for 256-bit ECDSA.
If `out_buffer->ptr` is `NULL` and `out_buffer_ptr->len` is large like `UINT32_MAX` no token will be created but the length of the token that would be created will be in `completed_token` as returned by attest_token_encode_finish(). None of the cryptographic functions run during this, but the sizes of what they would output is taken into account.
Definition at line 227 of file attest_token_encode.c.

## 7.266 secure_fw/partitions/initial_attestation/dir_initial_attestation.dox File Reference

## 7.267 secure_fw/partitions/initial_attestation/tfm_attest.c File Reference

```
#include "service_api.h"
#include "attest.h"
#include "psa/error.h"
#include "psa/initial_attestation.h"
#include "tfm_boot_status.h"
```

Include dependency graph for tfm_attest.c:

## Functions

- enum psa_attest_err_t attest_get_caller_client_id (int32_t ∗caller_id)

  *Get the ID of the caller thread.*
- enum psa_attest_err_t attest_get_boot_data (uint8_t major_type, struct tfm_boot_data ∗boot_data, uint32_t len)

  *Copy the boot data (coming from boot loader) from shared memory area to service memory area.*

## Variables

- int32_t g_attest_caller_id

## 7.267.1 Function Documentation

### 7.267.1.1 attest_get_boot_data()

```
enum psa_attest_err_t attest_get_boot_data (
            uint8_t major_type,
            struct tfm_boot_data * boot_data,
            uint32_t len )
```

Copy the boot data (coming from boot loader) from shared memory area to service memory area.

**Parameters**

| | | |
|---|---|---|
| in | *major_type* | Major type of TLV entries to copy |
| out | *ptr* | Pointer to the buffer to store the boot data \parma[in] len Size of the buffer to store the boot data |

**Returns**

Returns error code as specified in psa_attest_err_t

Definition at line 26 of file tfm_attest.c.
Here is the caller graph for this function:

#### 7.267.1.2 attest_get_caller_client_id()

```
enum psa_attest_err_t attest_get_caller_client_id (
            int32_t * caller_id )
```

Get the ID of the caller thread.

**Parameters**

| | | |
|---|---|---|
| out | *caller↩ _id* | Pointer where to store caller ID |

**Returns**

Returns error code as specified in psa_attest_err_t

Definition at line 17 of file tfm_attest.c.
Here is the caller graph for this function:

### 7.267.2 Variable Documentation

#### 7.267.2.1 g_attest_caller_id

```
int32_t g_attest_caller_id
```

Definition at line 32 of file tfm_attest_req_mngr.c.

## 7.268 secure_fw/partitions/initial_attestation/tfm_attest_req_mngr.c File Reference

```
#include <string.h>
#include "psa/error.h"
```

```
#include "psa/client.h"
#include "psa/initial_attestation.h"
#include "psa/crypto.h"
#include "attest.h"
#include "array.h"
#include "psa/framework_feature.h"
#include "psa/service.h"
#include "psa_manifest/tfm_initial_attestation.h"
#include "tfm_attest_defs.h"
```

Include dependency graph for tfm_attest_req_mngr.c:

## Macros

- #define ECC_P256_PUBLIC_KEY_SIZE PSA_KEY_EXPORT_ECC_PUBLIC_KEY_MAX_SIZE(256)

## Typedefs

- typedef psa_status_t(∗ attest_func_t) (const psa_msg_t ∗msg)

## Functions

- psa_status_t tfm_attestation_service_sfn (const psa_msg_t ∗msg)
- psa_status_t attest_partition_init (void)

## Variables

- int32_t g_attest_caller_id

## 7.268.1 Macro Definition Documentation

### 7.268.1.1 ECC_P256_PUBLIC_KEY_SIZE

```
#define ECC_P256_PUBLIC_KEY_SIZE PSA_KEY_EXPORT_ECC_PUBLIC_KEY_MAX_SIZE(256)
```

Definition at line 28 of file tfm_attest_req_mngr.c.

## 7.268.2 Typedef Documentation

### 7.268.2.1 attest_func_t

```
typedef psa_status_t(* attest_func_t) (const psa_msg_t *msg)
```

Definition at line 30 of file tfm_attest_req_mngr.c.

### 7.268.3 Function Documentation

#### 7.268.3.1 attest_partition_init()

```
psa_status_t attest_partition_init (
            void  )
```

Definition at line 159 of file tfm_attest_req_mngr.c.
Here is the call graph for this function:

#### 7.268.3.2 tfm_attestation_service_sfn()

```
psa_status_t tfm_attestation_service_sfn (
            const psa_msg_t * msg )
```

Definition at line 147 of file tfm_attest_req_mngr.c.

### 7.268.4 Variable Documentation

#### 7.268.4.1 g_attest_caller_id

```
int32_t g_attest_caller_id
```

Definition at line 32 of file tfm_attest_req_mngr.c.

## 7.269 secure_fw/partitions/internal_trusted_storage/flash/its_flash.c File Reference

```
#include "its_flash.h"
#include "config_tfm.h"
```

Include dependency graph for its_flash.c:

## 7.270 secure_fw/partitions/internal_trusted_storage/flash/its_flash.h File Reference

```
#include "config_tfm.h"
#include "its_flash_hal.h"
#include "its_utils.h"
#include "tfm_hal_its.h"
#include "its_flash_nor.h"
```

Include dependency graph for its_flash.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define ITS_FLASH_DEV &TFM_HAL_ITS_FLASH_DRIVER
- #define ITS_FLASH_ALIGNMENT TFM_HAL_ITS_PROGRAM_UNIT
- #define ITS_FLASH_OPS its_flash_ops_nor
- #define PS_FLASH_ALIGNMENT 1
- #define ITS_FLASH_MAX_ALIGNMENT

  *Provides a compile-time constant for the maximum program unit required by any flash device that can be accessed through this interface.*

## 7.270.1 Macro Definition Documentation

### 7.270.1.1 ITS_FLASH_ALIGNMENT

```
#define ITS_FLASH_ALIGNMENT TFM_HAL_ITS_PROGRAM_UNIT
```

Definition at line 53 of file its_flash.h.

### 7.270.1.2 ITS_FLASH_DEV

```
#define ITS_FLASH_DEV &TFM_HAL_ITS_FLASH_DRIVER
```

Definition at line 52 of file its_flash.h.

### 7.270.1.3 ITS_FLASH_MAX_ALIGNMENT

```
#define ITS_FLASH_MAX_ALIGNMENT
```

**Value:**

```
ITS_UTILS_MAX(ITS_FLASH_ALIGNMENT, \
PS_FLASH_ALIGNMENT)
```

Provides a compile-time constant for the maximum program unit required by any flash device that can be accessed through this interface.
Definition at line 100 of file its_flash.h.

### 7.270.1.4 ITS_FLASH_OPS

```
#define ITS_FLASH_OPS its_flash_ops_nor
```

Definition at line 54 of file its_flash.h.

### 7.270.1.5 PS_FLASH_ALIGNMENT

```
#define PS_FLASH_ALIGNMENT 1
```

Definition at line 93 of file its_flash.h.

## 7.271 secure_fw/partitions/internal_trusted_storage/flash/its_flash_hal.h File Reference

Internal Trusted Storage file system driver abstraction APIs. The purpose of this abstraction is to provide interface between ITS file system and implementation of storage back-end.

```
#include <stdint.h>
#include <stddef.h>
#include <psa/error.h>
```

Include dependency graph for its_flash_hal.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct its_flash_config_t

  *Structure containing ITS flash driver instance configuration used by ITS flash FS layer.*

- struct its_flash_ops_t

  *Structure containing the ITS flash driver operations.*

### Macros

- #define ITS_BLOCK_INVALID_ID 0xFFFFFFFFU

### 7.271.1 Detailed Description

Internal Trusted Storage file system driver abstraction APIs. The purpose of this abstraction is to provide interface between ITS file system and implementation of storage back-end.

#### Encryption of ITS storage

TF-M can use this layer to implement additional encryption layer for Internal Trusted Storage by implementing of ITS FS driver that is storage independent and sits between ITS file system and storage located on a physical device.

Write sequence with encryption of ITS:

1. TFM ITS FS calls storage write operation (its_flash_ops_t::write) that is handled by encryption driver.
2. Encryption driver uses PSA Crypto API to encrypt chunk and pass encrypted data to underlying storage driver by calling its_flash_ops_t::write.
3. Underlying storage driver (NOR/NAND/RAM or platform specific) store encrypted data on the physical device.

Read sequence with encryption of ITS:

1. TFM ITS FS calls storage read operation (its_flash_ops_t::read) that is handled by encryption driver.
2. Encryption driver uses underlying storage driver to read encrypted chunk by calling its_flash_ops_t::read.
3. Underlying storage driver (NOR/NAND/RAM or platform specific) read encrypted data from the physical device.
4. Encryption driver uses PSA Crypto API to decrypt chunk of data into result buffer of its_flash_ops_t::read request.

So, from ITS FS point of view there is no need to change anything. All magic could be implemented during initialization of ITS service.

.
TODO: ITS file system driver currently depends on sector size, see its_flash_info_t.sector_size. This dependency make no sense for ITS file system or Internal Storage service. Sector size is used only by Internal Storage service to calculate block size during initialization. It make sense to remove its_flash_info_t and its_flash_ops_t.get_info.

### 7.271.2 Macro Definition Documentation

#### 7.271.2.1 ITS_BLOCK_INVALID_ID

```
#define ITS_BLOCK_INVALID_ID 0xFFFFFFFFU
```

Definition at line 61 of file its_flash_hal.h.

## 7.272 secure_fw/partitions/internal_trusted_storage/flash/its_flash_nand.c File Reference

```
#include <string.h>
#include "its_flash_nand.h"
#include "static_checks.h"
```

Include dependency graph for its_flash_nand.c:

## Variables

- const struct its_flash_ops_t its_flash_ops_nand

### 7.272.1 Variable Documentation

#### 7.272.1.1 its_flash_ops_nand

```
const struct its_flash_ops_t its_flash_ops_nand
```

**Initial value:**

```
= {
    .init = its_flash_nand_init,
    .read = its_flash_nand_read,
    .write = its_flash_nand_write,
    .flush = its_flash_nand_flush,
    .erase = its_flash_nand_erase,
}
```

Definition at line 260 of file its_flash_nand.c.

## 7.273 secure_fw/partitions/internal_trusted_storage/flash/its_flash_nand.h File Reference

Implementations of the flash interface functions for a NAND flash device. See its_flash_ops_t for full documentation of functions.

```
#include <stddef.h>
#include <stdint.h>
#include "driver/Driver_Flash.h"
```

```
#include "its_flash_hal.h"
```

Include dependency graph for its_flash_nand.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct its_flash_nand_dev_t

## Variables

- const struct its_flash_ops_t its_flash_ops_nand

### 7.273.1 Detailed Description

Implementations of the flash interface functions for a NAND flash device. See its_flash_ops_t for full documentation of functions.

### 7.273.2 Variable Documentation

#### 7.273.2.1 its_flash_ops_nand

```
const struct its_flash_ops_t its_flash_ops_nand
```

Definition at line 260 of file its_flash_nand.c.

## 7.274 secure_fw/partitions/internal_trusted_storage/flash/its_flash_↩ nor.c File Reference

```
#include <string.h>
#include "static_checks.h"
#include "its_flash_nor.h"
```

Include dependency graph for its_flash_nor.c:

### Variables

- const struct its_flash_ops_t its_flash_ops_nor

  *ITS NOR flash driver.*

### 7.274.1 Variable Documentation

#### 7.274.1.1 its_flash_ops_nor

```
const struct its_flash_ops_t its_flash_ops_nor
```

**Initial value:**

```
= {
    .init = its_flash_nor_init,
    .read = its_flash_nor_read,
    .write = its_flash_nor_write,
    .flush = its_flash_nor_flush,
    .erase = its_flash_nor_erase,
}
```

ITS NOR flash driver.

its_flash_config_t::context should point to ARM_DRIVER_FLASH.

Definition at line 219 of file its_flash_nor.c.

## 7.275 secure_fw/partitions/internal_trusted_storage/flash/its_flash_nor.h File Reference

Implementations of the flash interface functions for a NOR flash device. See its_flash_ops_t for full documentation of functions.

```
#include "driver/Driver_Flash.h"
#include "its_flash_hal.h"
```

Include dependency graph for its_flash_nor.h:

This graph shows which files directly or indirectly include this file:

### Variables

- const struct its_flash_ops_t its_flash_ops_nor

  *ITS NOR flash driver.*

### 7.275.1 Detailed Description

Implementations of the flash interface functions for a NOR flash device. See its_flash_ops_t for full documentation of functions.

### 7.275.2 Variable Documentation

#### 7.275.2.1 its_flash_ops_nor

```
const struct its_flash_ops_t its_flash_ops_nor
```

ITS NOR flash driver.
its_flash_config_t::context should point to ARM_DRIVER_FLASH.
Definition at line 219 of file its_flash_nor.c.

## 7.276 secure_fw/partitions/internal_trusted_storage/flash/its_flash_↩ ram.c File Reference

```
#include <string.h>
#include "its_flash_ram.h"
#include "static_checks.h"
```

Include dependency graph for its_flash_ram.c:

### Variables

- const struct its_flash_ops_t its_flash_fs_ops_ram

## 7.276.1 Variable Documentation

#### 7.276.1.1 its_flash_fs_ops_ram

```
const struct its_flash_ops_t its_flash_fs_ops_ram
```

**Initial value:**

```
= {
    .init = its_flash_ram_init,
    .read = its_flash_ram_read,
    .write = its_flash_ram_write,
    .flush = its_flash_ram_flush,
    .erase = its_flash_ram_erase,
}
```

Definition at line 110 of file its_flash_ram.c.

# 7.277 secure_fw/partitions/internal_trusted_storage/flash/its_flash_↩ ram.h File Reference

Implementations of the flash interface functions for an emulated flash device using RAM. See its_flash_ops_t for full documentation of functions.

```
#include <stdint.h>
#include "its_flash_hal.h"
```

Include dependency graph for its_flash_ram.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct its_flash_ram_dev_t

  *ITS RAM driver context.*

## Variables

- const struct its_flash_ops_t its_flash_fs_ops_ram

### 7.277.1 Detailed Description

Implementations of the flash interface functions for an emulated flash device using RAM. See its_flash_ops_t for full documentation of functions.

### 7.277.2 Variable Documentation

#### 7.277.2.1 its_flash_fs_ops_ram

```
const struct its_flash_ops_t its_flash_fs_ops_ram
```

Definition at line 110 of file its_flash_ram.c.

## 7.278 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_↩ fs.c File Reference

```
#include <stdbool.h>
#include <string.h>
#include "config_tfm.h"
#include "its_flash_fs.h"
#include "its_flash_fs_dblock.h"
#include "its_utils.h"
```

Include dependency graph for its_flash_fs.c:

## Macros

- #define ITS_FLASH_FS_INTERNAL_FLAGS_MASK (UINT32_MAX - ((1U << 24) - 1))
- #define ITS_FLASH_FS_FLAG_DELETE (1U << 24)

## Functions

- psa_status_t its_flash_fs_init_ctx (its_flash_fs_ctx_t *fs_ctx, const struct its_flash_fs_config_t *fs_cfg)

  *Initialises the filesystem context. Must be called successfully before any other filesystem API is called.*
- psa_status_t its_flash_fs_prepare (its_flash_fs_ctx_t *fs_ctx)

  *Prepares the filesystem to accept operations on the files.*
- psa_status_t its_flash_fs_wipe_all (struct its_flash_fs_ctx_t *fs_ctx)

  *Wipes all files from the filesystem.*
- psa_status_t its_flash_fs_file_get_info (struct its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, struct its_flash_fs_file_info_t *info)

  *Gets the file information referenced by the file ID.*
- psa_status_t its_flash_fs_file_write (struct its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, struct its_flash_fs_file_info_t *finfo, size_t data_size, size_t offset, const uint8_t *data)

  *Writes data to a file.*
- psa_status_t its_flash_fs_file_delete (struct its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid)

  *Deletes file referenced by the file ID.*
- psa_status_t its_flash_fs_file_read (struct its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, size_t size, size_t offset, uint8_t *data)

  *Reads data from an existing file.*

## 7.278.1 Macro Definition Documentation

### 7.278.1.1 ITS_FLASH_FS_FLAG_DELETE

```
#define ITS_FLASH_FS_FLAG_DELETE (1U << 24)
```

Definition at line 22 of file its_flash_fs.c.

### 7.278.1.2 ITS_FLASH_FS_INTERNAL_FLAGS_MASK

```
#define ITS_FLASH_FS_INTERNAL_FLAGS_MASK (UINT32_MAX - ((1U << 24) - 1))
```

Definition at line 20 of file its_flash_fs.c.

## 7.278.2 Function Documentation

### 7.278.2.1 its_flash_fs_file_delete()

```
psa_status_t its_flash_fs_file_delete (
            its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid )
```

Deletes file referenced by the file ID.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 559 of file its_flash_fs.c.
Here is the call graph for this function:

### 7.278.2.2 its_flash_fs_file_get_info()

```
psa_status_t its_flash_fs_file_get_info (
            its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            struct its_flash_fs_file_info_t * info )
```

Gets the file information referenced by the file ID.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |
| out | *info* | Pointer to the file information structure its_flash_fs_file_info_t |

**Returns**

Returns error code specified in psa_status_t

Definition at line 203 of file its_flash_fs.c.
Here is the call graph for this function:

### 7.278.2.3 its_flash_fs_file_read()

```
psa_status_t its_flash_fs_file_read (
            its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            size_t size,
            size_t offset,
            uint8_t * data )
```

Reads data from an existing file.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |
| in | *size* | Size to be read |
| in | *offset* | Offset in the file |
| out | *data* | Pointer to buffer to store the data |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 574 of file its_flash_fs.c.
Here is the call graph for this function:

### 7.278.2.4 its_flash_fs_file_write()

```
psa_status_t its_flash_fs_file_write (
            its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            struct its_flash_fs_file_info_t * finfo,
            size_t data_size,
            size_t offset,
            const uint8_t * data )
```

Writes data to a file.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |
| in | *finfo* | Pointer to its_flash_fs_file_info_t |
| in | *data_size* | Size of the incoming write data. |
| in | *offset* | Offset in the file to write. Must be less than or equal to the current file size. |
| in | *data* | Pointer to buffer containing data to be written |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 228 of file its_flash_fs.c.
Here is the call graph for this function:

### 7.278.2.5 its_flash_fs_init_ctx()

```
psa_status_t its_flash_fs_init_ctx (
            its_flash_fs_ctx_t * fs_ctx,
            const struct its_flash_fs_config_t * fs_cfg )
```

Initialises the filesystem context. Must be called successfully before any other filesystem API is called.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context to initialise. Must have been allocated by the caller. |
| in,out | *fs_cfg* | Filesystem configuration to associate with the context. |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 142 of file its_flash_fs.c.
Here is the caller graph for this function:

### 7.278.2.6 its_flash_fs_prepare()

```
psa_status_t its_flash_fs_prepare (
            its_flash_fs_ctx_t * fs_ctx )
```

Prepares the filesystem to accept operations on the files.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context to prepare. Must have been initialised by its_flash_fs_init_ctx(). |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 172 of file its_flash_fs.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.278.2.7 its_flash_fs_wipe_all()

```
psa_status_t its_flash_fs_wipe_all (
            its_flash_fs_ctx_t * fs_ctx )
```

Wipes all files from the filesystem.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context to wipe. Must be prepared again before further use. |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 197 of file its_flash_fs.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.279 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_↩ fs.h File Reference

Internal Trusted Storage service filesystem abstraction APIs. The purpose of this abstraction is to have the ability to plug-in other filesystems or filesystem proxies (supplicant).

```
#include <stddef.h>
#include <stdint.h>
#include "its_flash_fs_mblock.h"
#include "psa/error.h"
#include "../flash/its_flash_hal.h"
```

Include dependency graph for its_flash_fs.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct its_flash_fs_config_t

  *Structure containing the flash filesystem configuration parameters.*
- struct its_flash_fs_file_info_t

  *Structure containing file information.*

## Macros

- #define ITS_FLASH_FS_USER_FLAGS_MASK ((1UL << 16) - 1)
- #define ITS_FLASH_FS_WRITE_FLAGS_MASK ((1UL << 24) - (1UL << 16))
- #define ITS_FLASH_FS_FLAG_CREATE (1UL << 16)
- #define ITS_FLASH_FS_FLAG_TRUNCATE (1UL << 17)

## Typedefs

- typedef struct its_flash_fs_ctx_t its_flash_fs_ctx_t

  *ITS flash filesystem context type, used to maintain state across FS operations.*

## Functions

- psa_status_t its_flash_fs_init_ctx (its_flash_fs_ctx_t *fs_ctx, const struct its_flash_fs_config_t *fs_cfg)

  *Initialises the filesystem context. Must be called successfully before any other filesystem API is called.*
- psa_status_t its_flash_fs_prepare (its_flash_fs_ctx_t *fs_ctx)

  *Prepares the filesystem to accept operations on the files.*
- psa_status_t its_flash_fs_wipe_all (its_flash_fs_ctx_t *fs_ctx)

  *Wipes all files from the filesystem.*
- psa_status_t its_flash_fs_file_get_info (its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, struct its_flash_fs_file_info_t *info)

  *Gets the file information referenced by the file ID.*
- psa_status_t its_flash_fs_file_write (its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, struct its_flash_fs_file_info_t *finfo, size_t data_size, size_t offset, const uint8_t *data)

  *Writes data to a file.*
- psa_status_t its_flash_fs_file_read (its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, size_t size, size_t offset, uint8_t *data)

  *Reads data from an existing file.*
- psa_status_t its_flash_fs_file_delete (its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid)

  *Deletes file referenced by the file ID.*

### 7.279.1 Detailed Description

Internal Trusted Storage service filesystem abstraction APIs. The purpose of this abstraction is to have the ability to plug-in other filesystems or filesystem proxies (supplicant).

### 7.279.2 Macro Definition Documentation

#### 7.279.2.1 ITS_FLASH_FS_FLAG_CREATE

```
#define ITS_FLASH_FS_FLAG_CREATE (1UL << 16)
```

Definition at line 40 of file its_flash_fs.h.

#### 7.279.2.2 ITS_FLASH_FS_FLAG_TRUNCATE

```
#define ITS_FLASH_FS_FLAG_TRUNCATE (1UL << 17)
```

Definition at line 42 of file its_flash_fs.h.

#### 7.279.2.3 ITS_FLASH_FS_USER_FLAGS_MASK

```
#define ITS_FLASH_FS_USER_FLAGS_MASK ((1UL << 16) - 1)
```

Definition at line 35 of file its_flash_fs.h.

#### 7.279.2.4 ITS_FLASH_FS_WRITE_FLAGS_MASK

```
#define ITS_FLASH_FS_WRITE_FLAGS_MASK ((1UL << 24) - (1UL << 16))
```

Definition at line 38 of file its_flash_fs.h.

### 7.279.3 Typedef Documentation

#### 7.279.3.1 its_flash_fs_ctx_t

```
typedef struct its_flash_fs_ctx_t its_flash_fs_ctx_t
```

ITS flash filesystem context type, used to maintain state across FS operations.
The user should allocate a variable of this type, initialised to zero, before calling its_flash_fs_prepare, and then pass it to each subsequent FS operation. The contents are internal to the filesystem.
Definition at line 65 of file its_flash_fs.h.

### 7.279.4 Function Documentation

#### 7.279.4.1 its_flash_fs_file_delete()

```
psa_status_t its_flash_fs_file_delete (
            its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid )
```

Deletes file referenced by the file ID.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 559 of file its_flash_fs.c.
Here is the call graph for this function:

#### 7.279.4.2 its_flash_fs_file_get_info()

```
psa_status_t its_flash_fs_file_get_info (
            its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            struct its_flash_fs_file_info_t * info )
```

Gets the file information referenced by the file ID.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |
| out | *info* | Pointer to the file information structure its_flash_fs_file_info_t |

**Returns**

Returns error code specified in psa_status_t

Definition at line 203 of file its_flash_fs.c.
Here is the call graph for this function:

### 7.279.4.3 its_flash_fs_file_read()

```
psa_status_t its_flash_fs_file_read (
            its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            size_t size,
            size_t offset,
            uint8_t * data )
```

Reads data from an existing file.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |
| in | *size* | Size to be read |
| in | *offset* | Offset in the file |
| out | *data* | Pointer to buffer to store the data |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 574 of file its_flash_fs.c.
Here is the call graph for this function:

#### 7.279.4.4 its_flash_fs_file_write()

```
psa_status_t its_flash_fs_file_write (
            its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            struct its_flash_fs_file_info_t * finfo,
            size_t data_size,
            size_t offset,
            const uint8_t * data )
```

Writes data to a file.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |
| in | *finfo* | Pointer to its_flash_fs_file_info_t |
| in | *data_size* | Size of the incoming write data. |
| in | *offset* | Offset in the file to write. Must be less than or equal to the current file size. |
| in | *data* | Pointer to buffer containing data to be written |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 228 of file its_flash_fs.c.
Here is the call graph for this function:

#### 7.279.4.5 its_flash_fs_init_ctx()

```
psa_status_t its_flash_fs_init_ctx (
            its_flash_fs_ctx_t * fs_ctx,
            const struct its_flash_fs_config_t * fs_cfg )
```

Initialises the filesystem context. Must be called successfully before any other filesystem API is called.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context to initialise. Must have been allocated by the caller. |
| in,out | *fs_cfg* | Filesystem configuration to associate with the context. |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 142 of file its_flash_fs.c.
Here is the caller graph for this function:

### 7.279.4.6 its_flash_fs_prepare()

```
psa_status_t its_flash_fs_prepare (
            its_flash_fs_ctx_t * fs_ctx )
```

Prepares the filesystem to accept operations on the files.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context to prepare. Must have been initialised by its_flash_fs_init_ctx(). |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 172 of file its_flash_fs.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.279.4.7 its_flash_fs_wipe_all()

```
psa_status_t its_flash_fs_wipe_all (
            its_flash_fs_ctx_t * fs_ctx )
```

Wipes all files from the filesystem.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context to wipe. Must be prepared again before further use. |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 197 of file its_flash_fs.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.280 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_↩ fs_dblock.c File Reference

```
#include "its_flash_fs_dblock.h"
#include "its_flash_fs.h"
```

Include dependency graph for its_flash_fs_dblock.c:

## Functions

- psa_status_t its_flash_fs_dblock_compact_block (struct its_flash_fs_ctx_t ∗fs_ctx, uint32_t lblock, size_t free_size, size_t src_offset, size_t dst_offset, size_t size)

  *Compacts block data for the given logical block.*
- psa_status_t its_flash_fs_dblock_read_file (struct its_flash_fs_ctx_t ∗fs_ctx, const struct its_file_meta_t ∗file_meta, size_t offset, size_t size, uint8_t ∗buf)

  *Reads the file content.*
- psa_status_t its_flash_fs_dblock_write_file (struct its_flash_fs_ctx_t ∗fs_ctx, const struct its_block_meta_t ∗block_meta, const struct its_file_meta_t ∗file_meta, size_t offset, size_t size, const uint8_t ∗data)

  *Writes scratch data block content with requested data and the rest of the data from the given logical block.*

## 7.280.1 Function Documentation

### 7.280.1.1 its_flash_fs_dblock_compact_block()

```
psa_status_t its_flash_fs_dblock_compact_block (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock,
            size_t free_size,
            size_t src_offset,
            size_t dst_offset,
            size_t size )
```

Compacts block data for the given logical block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical data block to compact |

**Parameters**

| | | |
|---|---|---|
| in | *free_size* | Available data size to compact |
| in | *src_offset* | Offset in the current data block which points to the data position to reallocate |
| in | *dst_offset* | Offset in the scratch block which points to the data position to store the data to be reallocated |
| in | *size* | Number of bytes to be reallocated |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 36 of file its_flash_fs_dblock.c.
Here is the call graph for this function:

### 7.280.1.2 its_flash_fs_dblock_read_file()

```
psa_status_t its_flash_fs_dblock_read_file (
            struct its_flash_fs_ctx_t * fs_ctx,
            const struct its_file_meta_t * file_meta,
            size_t offset,
            size_t size,
            uint8_t * buf )
```

Reads the file content.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *file_meta* | File metadata |
| in | *offset* | Offset in the file |
| in | *size* | Size to be read |
| out | *buf* | Buffer pointer to store the data |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 119 of file its_flash_fs_dblock.c.
Here is the caller graph for this function:

### 7.280.1.3 its_flash_fs_dblock_write_file()

```
psa_status_t its_flash_fs_dblock_write_file (
            struct its_flash_fs_ctx_t * fs_ctx,
            const struct its_block_meta_t * block_meta,
            const struct its_file_meta_t * file_meta,
            size_t offset,
            size_t size,
            const uint8_t * data )
```

Writes scratch data block content with requested data and the rest of the data from the given logical block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *block_meta* | Block metadata |
| in | *file_meta* | File metadata |
| in | *offset* | Offset in the scratch data block where to start the copy of the incoming data |
| in | *size* | Size of the incoming data |
| in | *data* | Pointer to data buffer to copy in the scratch data block |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 139 of file its_flash_fs_dblock.c.
Here is the call graph for this function:

## 7.281 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_↩ fs_dblock.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "psa/error.h"
#include "its_flash_fs_mblock.h"
```

Include dependency graph for its_flash_fs_dblock.h:

This graph shows which files directly or indirectly include this file:

## Functions

- psa_status_t its_flash_fs_dblock_compact_block (struct its_flash_fs_ctx_t ∗fs_ctx, uint32_t lblock, size_t free_size, size_t src_offset, size_t dst_offset, size_t size)

  *Compacts block data for the given logical block.*
- psa_status_t its_flash_fs_dblock_read_file (struct its_flash_fs_ctx_t ∗fs_ctx, const struct its_file_meta_t ∗file_meta, size_t offset, size_t size, uint8_t ∗buf)

  *Reads the file content.*
- psa_status_t its_flash_fs_dblock_write_file (struct its_flash_fs_ctx_t ∗fs_ctx, const struct its_block_meta_t ∗block_meta, const struct its_file_meta_t ∗file_meta, size_t offset, size_t size, const uint8_t ∗data)

  *Writes scratch data block content with requested data and the rest of the data from the given logical block.*

## 7.281.1 Function Documentation

### 7.281.1.1 its_flash_fs_dblock_compact_block()

```
psa_status_t its_flash_fs_dblock_compact_block (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock,
            size_t free_size,
            size_t src_offset,
            size_t dst_offset,
            size_t size )
```

Compacts block data for the given logical block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical data block to compact |
| in | *free_size* | Available data size to compact |
| in | *src_offset* | Offset in the current data block which points to the data position to reallocate |
| in | *dst_offset* | Offset in the scratch block which points to the data position to store the data to be reallocated |
| in | *size* | Number of bytes to be reallocated |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 36 of file its_flash_fs_dblock.c.
Here is the call graph for this function:

### 7.281.1.2 its_flash_fs_dblock_read_file()

```
psa_status_t its_flash_fs_dblock_read_file (
            struct its_flash_fs_ctx_t * fs_ctx,
            const struct its_file_meta_t * file_meta,
            size_t offset,
            size_t size,
            uint8_t * buf )
```

Reads the file content.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *file_meta* | File metadata |
| in | *offset* | Offset in the file |
| in | *size* | Size to be read |
| out | *buf* | Buffer pointer to store the data |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 119 of file its_flash_fs_dblock.c.
Here is the caller graph for this function:

### 7.281.1.3 its_flash_fs_dblock_write_file()

```
psa_status_t its_flash_fs_dblock_write_file (
            struct its_flash_fs_ctx_t * fs_ctx,
            const struct its_block_meta_t * block_meta,
            const struct its_file_meta_t * file_meta,
            size_t offset,
            size_t size,
            const uint8_t * data )
```

Writes scratch data block content with requested data and the rest of the data from the given logical block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *block_meta* | Block metadata |
| in | *file_meta* | File metadata |
| in | *offset* | Offset in the scratch data block where to start the copy of the incoming data |
| in | *size* | Size of the incoming data |
| in | *data* | Pointer to data buffer to copy in the scratch data block |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 139 of file its_flash_fs_dblock.c.
Here is the call graph for this function:

## 7.282 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_↩ fs_mblock.c File Reference

```
#include <string.h>
#include "config_tfm.h"
#include "its_flash_fs_mblock.h"
#include "psa/storage_common.h"
#include "static_checks.h"
```

Include dependency graph for its_flash_fs_mblock.c:

### Macros

- #define ITS_MAX_BLOCK_DATA_COPY 256
- #define ITS_METADATA_BLOCK0 0
- #define ITS_METADATA_BLOCK1 1
- #define ITS_OTHER_META_BLOCK(metablock)

  *Macro to get the the swap metadata block.*
- #define ITS_BLOCK_META_HEADER_SIZE sizeof(struct its_metadata_block_header_t)

- #define ITS_BLOCK_METADATA_SIZE sizeof(struct its_block_meta_t)
- #define ITS_FILE_METADATA_SIZE sizeof(struct its_file_meta_t)

## Functions

- psa_status_t its_flash_fs_mblock_cp_file_meta (struct its_flash_fs_ctx_t *fs_ctx, uint32_t idx_start, uint32_t idx_end)

  *Copies the file metadata entries between two indexes from the active metadata block to the scratch metadata block.*
- uint32_t its_flash_fs_mblock_cur_data_scratch_id (struct its_flash_fs_ctx_t *fs_ctx, uint32_t lblock)

  *Gets current scratch datablock physical ID.*
- psa_status_t its_flash_fs_mblock_get_file_idx_meta (struct its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, uint32_t *idx, struct its_file_meta_t *file_meta)

  *Gets file metadata entry index and file metadata.*
- psa_status_t its_flash_fs_mblock_get_file_idx_flag (struct its_flash_fs_ctx_t *fs_ctx, uint32_t flags, uint32_t *idx)

  *Gets file metadata entry index of the first file with one of the provided flags set.*
- psa_status_t its_flash_fs_mblock_init (struct its_flash_fs_ctx_t *fs_ctx)

  *Initializes metadata block with the valid/active metablock.*
- psa_status_t its_flash_fs_mblock_meta_update_finalize (struct its_flash_fs_ctx_t *fs_ctx)

  *Finalizes an update operation. Last step when a create/write/delete is performed.*
- psa_status_t its_flash_fs_mblock_migrate_lb0_data_to_scratch (struct its_flash_fs_ctx_t *fs_ctx)

  *Writes the files data area of logical block 0 into the scratch block.*
- psa_status_t its_flash_fs_mblock_read_file_meta (struct its_flash_fs_ctx_t *fs_ctx, uint32_t idx, struct its_file_meta_t *file_meta)

  *Reads specified file metadata.*
- psa_status_t its_flash_fs_mblock_read_block_metadata (struct its_flash_fs_ctx_t *fs_ctx, uint32_t lblock, struct its_block_meta_t *block_meta)

  *Reads specified logical block metadata.*
- psa_status_t its_flash_fs_mblock_read_block_metadata_comp (struct its_flash_fs_ctx_t *fs_ctx, uint32_t lblock, struct its_block_meta_t *block_meta)

  *Reads specified logical block metadata based on the backward compatible FS.*
- psa_status_t its_flash_fs_mblock_reserve_file (struct its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, bool use_spare, size_t size, uint32_t flags, uint32_t *idx, struct its_file_meta_t *file_meta, struct its_block_meta_t *block_meta)

  *Reserves space for a file.*
- psa_status_t its_flash_fs_mblock_reset_metablock (struct its_flash_fs_ctx_t *fs_ctx)

  *Resets metablock by cleaning and initializing the metadatablock.*
- void its_flash_fs_mblock_set_data_scratch (struct its_flash_fs_ctx_t *fs_ctx, uint32_t phy_id, uint32_t lblock)

  *Sets current data scratch block.*
- psa_status_t its_flash_fs_mblock_update_scratch_block_meta (struct its_flash_fs_ctx_t *fs_ctx, uint32_t lblock, struct its_block_meta_t *block_meta)

  *Puts logical block's metadata in scratch metadata block.*
- psa_status_t its_flash_fs_mblock_update_scratch_file_meta (struct its_flash_fs_ctx_t *fs_ctx, uint32_t idx, const struct its_file_meta_t *file_meta)

  *Writes a file metadata entry into scratch metadata block.*
- psa_status_t its_flash_fs_block_to_block_move (struct its_flash_fs_ctx_t *fs_ctx, uint32_t dst_block, size_t dst_offset, uint32_t src_block, size_t src_offset, size_t size)

  *Moves data from source block ID to destination block ID.*

### 7.282.1 Macro Definition Documentation

#### 7.282.1.1 ITS_BLOCK_META_HEADER_SIZE

```
#define ITS_BLOCK_META_HEADER_SIZE sizeof(struct its_metadata_block_header_t)
```

Definition at line 37 of file its_flash_fs_mblock.c.

#### 7.282.1.2 ITS_BLOCK_METADATA_SIZE

```
#define ITS_BLOCK_METADATA_SIZE sizeof(struct its_block_meta_t)
```

Definition at line 38 of file its_flash_fs_mblock.c.

#### 7.282.1.3 ITS_FILE_METADATA_SIZE

```
#define ITS_FILE_METADATA_SIZE sizeof(struct its_file_meta_t)
```

Definition at line 39 of file its_flash_fs_mblock.c.

#### 7.282.1.4 ITS_MAX_BLOCK_DATA_COPY

```
#define ITS_MAX_BLOCK_DATA_COPY 256
```

Definition at line 18 of file its_flash_fs_mblock.c.

#### 7.282.1.5 ITS_METADATA_BLOCK0

```
#define ITS_METADATA_BLOCK0 0
```

Definition at line 25 of file its_flash_fs_mblock.c.

#### 7.282.1.6 ITS_METADATA_BLOCK1

```
#define ITS_METADATA_BLOCK1 1
```

Definition at line 26 of file its_flash_fs_mblock.c.

#### 7.282.1.7 ITS_OTHER_META_BLOCK

```
#define ITS_OTHER_META_BLOCK(
            metablock )
```

**Value:**

```
(((metablock) == ITS_METADATA_BLOCK0) ?  \
(ITS_METADATA_BLOCK1) :  (ITS_METADATA_BLOCK0))
```

Macro to get the the swap metadata block.
Definition at line 33 of file its_flash_fs_mblock.c.

### 7.282.2 Function Documentation

#### 7.282.2.1 its_flash_fs_block_to_block_move()

```
psa_status_t its_flash_fs_block_to_block_move (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t dst_block,
            size_t dst_offset,
            uint32_t src_block,
            size_t src_offset,
            size_t size )
```

Moves data from source block ID to destination block ID.

**Parameters**

| | | |
|---|---|---|
| in | *fs_ctx* | Filesystem context |
| in | *dst_block* | Destination block ID |
| in | *dst_offset* | Destination offset position from the init of the destination block |
| in | *src_block* | Source block ID |
| in | *src_offset* | Source offset position from the init of the source block |
| in | *size* | Number of bytes to moves |

**Note**

This function assumes all input values are valid. That is, the address range, based on blockid, offset and size, is a valid range in flash. It also assumes that the destination block is already erased and ready to be written.

**Returns**

Returns PSA_SUCCESS if the function is executed correctly. Otherwise, it returns PSA_ERROR_STORA↩GE_FAILURE.

Definition at line 1379 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

### 7.282.2.2 its_flash_fs_mblock_cp_file_meta()

```
psa_status_t its_flash_fs_mblock_cp_file_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t idx_start,
            uint32_t idx_end )
```

Copies the file metadata entries between two indexes from the active metadata block to the scratch metadata block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *idx_start* | File metadata entry index to start copy, inclusive |
| in | *idx_end* | File metadata entry index to end copy, exclusive |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 976 of file its_flash_fs_mblock.c.

### 7.282.2.3 its_flash_fs_mblock_cur_data_scratch_id()

```
uint32_t its_flash_fs_mblock_cur_data_scratch_id (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock )
```

Gets current scratch datablock physical ID.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical block number |

**Returns**

current scratch data block

Definition at line 992 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

### 7.282.2.4 its_flash_fs_mblock_get_file_idx_flag()

```
psa_status_t its_flash_fs_mblock_get_file_idx_flag (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t flags,
            uint32_t * idx )
```

Gets file metadata entry index of the first file with one of the provided flags set.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *flags* | Flags to search for |
| out | *idx* | Index of the file metadata in the file system |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1033 of file its_flash_fs_mblock.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.282.2.5 its_flash_fs_mblock_get_file_idx_meta()

```
psa_status_t its_flash_fs_mblock_get_file_idx_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            uint32_t * idx,
            struct its_file_meta_t * file_meta )
```

Gets file metadata entry index and file metadata.

**Note**

A NULL [file_meta] indicates ignoring file meta.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | ID of the file |
| out | *idx* | Index of the file metadata in the file system |
| out | *file_meta* | Pointer to file meta structure |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1004 of file its_flash_fs_mblock.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.282.2.6 its_flash_fs_mblock_init()

```
psa_status_t its_flash_fs_mblock_init (
            struct its_flash_fs_ctx_t * fs_ctx )
```

Initializes metadata block with the valid/active metablock.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |

**Returns**

Returns value as specified in psa_status_t

Definition at line 1058 of file its_flash_fs_mblock.c.

Here is the caller graph for this function:

#### 7.282.2.7 its_flash_fs_mblock_meta_update_finalize()

```
psa_status_t its_flash_fs_mblock_meta_update_finalize (
            struct its_flash_fs_ctx_t * fs_ctx )
```

Finalizes an update operation. Last step when a create/write/delete is performed.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |

**Returns**

Returns offset value in metadata block

Definition at line 1084 of file its_flash_fs_mblock.c.

#### 7.282.2.8 its_flash_fs_mblock_migrate_lb0_data_to_scratch()

```
psa_status_t its_flash_fs_mblock_migrate_lb0_data_to_scratch (
            struct its_flash_fs_ctx_t * fs_ctx )
```

Writes the files data area of logical block 0 into the scratch block.

**Note**

The files data in the logical block 0 is stored in same physical block where the metadata is stored. A change in the metadata requires a swap of physical blocks. So, the files data stored in the current medadata block needs to be copied in the scratch block, unless the data of the file processed is located in the logical block 0.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1108 of file its_flash_fs_mblock.c.
Here is the call graph for this function:

### 7.282.2.9 its_flash_fs_mblock_read_block_metadata()

```
psa_status_t its_flash_fs_mblock_read_block_metadata (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock,
            struct its_block_meta_t * block_meta )
```

Reads specified logical block metadata.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical block number |
| out | *block_meta* | Pointer to block meta structure |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1153 of file its_flash_fs_mblock.c.

Here is the caller graph for this function:

#### 7.282.2.10 its_flash_fs_mblock_read_block_metadata_comp()

```
psa_status_t its_flash_fs_mblock_read_block_metadata_comp (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock,
            struct its_block_meta_t * block_meta )
```

Reads specified logical block metadata based on the backward compatible FS.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical block number |
| out | *block_meta* | Pointer to block meta structure |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1175 of file its_flash_fs_mblock.c.

#### 7.282.2.11 its_flash_fs_mblock_read_file_meta()

```
psa_status_t its_flash_fs_mblock_read_file_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t idx,
            struct its_file_meta_t * file_meta )
```

Reads specified file metadata.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *idx* | File metadata entry index |
| out | *file_meta* | Pointer to file meta structure |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1131 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

### 7.282.2.12 its_flash_fs_mblock_reserve_file()

```
psa_status_t its_flash_fs_mblock_reserve_file (
            struct its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            bool use_spare,
            size_t size,
            uint32_t flags,
            uint32_t * file_meta_idx,
            struct its_file_meta_t * file_meta,
            struct its_block_meta_t * block_meta )
```

Reserves space for a file.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |
| in | *use_spare* | If true then the spare file will be used, otherwise at least one file will be left free |
| in | *size* | Size of the file for which space is reserve |
| in | *flags* | Flags set when the file was created |
| out | *file_meta_idx* | File metadata entry index |
| out | *file_meta* | File metadata entry |
| out | *block_meta* | Block metadata entry |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1199 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

### 7.282.2.13 its_flash_fs_mblock_reset_metablock()

```
psa_status_t its_flash_fs_mblock_reset_metablock (
            struct its_flash_fs_ctx_t * fs_ctx )
```

Resets metablock by cleaning and initializing the metadatablock.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |

**Returns**

Returns value as specified in psa_status_t

Definition at line 1223 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

### 7.282.2.14 its_flash_fs_mblock_set_data_scratch()

```
void its_flash_fs_mblock_set_data_scratch (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t phy_id,
            uint32_t lblock )
```

Sets current data scratch block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *phy↩ _id* | Physical ID of scratch data block |
| in | *lblock* | Logical block number |

Definition at line 1334 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

#### 7.282.2.15 its_flash_fs_mblock_update_scratch_block_meta()

```
psa_status_t its_flash_fs_mblock_update_scratch_block_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock,
            struct its_block_meta_t * block_meta )
```

Puts logical block's metadata in scratch metadata block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical block number |
| in | *block_meta* | Pointer to block's metadata |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1342 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

#### 7.282.2.16 its_flash_fs_mblock_update_scratch_file_meta()

```
psa_status_t its_flash_fs_mblock_update_scratch_file_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t idx,
            const struct its_file_meta_t * file_meta )
```

Writes a file metadata entry into scratch metadata block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *idx* | File's index in the metadata table |
| in | *file_meta* | Metadata pointer |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1365 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

## 7.283 secure_fw/partitions/internal_trusted_storage/flash_fs/its_flash_↩ fs_mblock.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include "flash/its_flash_hal.h"
#include "its_flash_fs.h"
#include "its_utils.h"
#include "psa/error.h"
```

Include dependency graph for its_flash_fs_mblock.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct its_metadata_block_header_t

  *Structure to store the metadata block header.*
- struct its_metadata_block_header_comp_t

  *The struture of metadata block header in ITS_BACKWARD_SUPPORTED_VERSION.*
- struct its_block_meta_t

  *Structure to store information about each physical flash memory block.*
- struct its_file_meta_t

  *Structure to store file metadata.*
- struct its_flash_fs_ctx_t

  *Structure to store the ITS flash file system context.*

## Macros

- #define ITS_SUPPORTED_VERSION 0x02

  *Defines the supported version.*
- #define ITS_BACKWARD_SUPPORTED_VERSION 0x01

  *Defines the backward supported version.*
- #define ITS_METADATA_INVALID_INDEX 0xFFFF

  *Defines the invalid index value when the metadata table is full.*
- #define ITS_LOGICAL_DBLOCK0 0

  *Defines logical data block 0 ID.*
- #define _T1
- #define _T1_COMP
- #define _T2
- #define _T3

## Functions

- psa_status_t its_flash_fs_mblock_init (struct its_flash_fs_ctx_t ∗fs_ctx)

  *Initializes metadata block with the valid/active metablock.*
- psa_status_t its_flash_fs_mblock_cp_file_meta (struct its_flash_fs_ctx_t ∗fs_ctx, uint32_t idx_start, uint32↩_t idx_end)

  *Copies the file metadata entries between two indexes from the active metadata block to the scratch metadata block.*
- uint32_t its_flash_fs_mblock_cur_data_scratch_id (struct its_flash_fs_ctx_t ∗fs_ctx, uint32_t lblock)

  *Gets current scratch datablock physical ID.*
- psa_status_t its_flash_fs_mblock_get_file_idx_meta (struct its_flash_fs_ctx_t ∗fs_ctx, const uint8_t ∗fid, uint32_t ∗idx, struct its_file_meta_t ∗file_meta)

  *Gets file metadata entry index and file metadata.*

- psa_status_t its_flash_fs_mblock_get_file_idx_flag (struct its_flash_fs_ctx_t *fs_ctx, uint32_t flags, uint32_t *idx)

  *Gets file metadata entry index of the first file with one of the provided flags set.*
- psa_status_t its_flash_fs_mblock_meta_update_finalize (struct its_flash_fs_ctx_t *fs_ctx)

  *Finalizes an update operation. Last step when a create/write/delete is performed.*
- psa_status_t its_flash_fs_mblock_migrate_lb0_data_to_scratch (struct its_flash_fs_ctx_t *fs_ctx)

  *Writes the files data area of logical block 0 into the scratch block.*
- psa_status_t its_flash_fs_mblock_read_file_meta (struct its_flash_fs_ctx_t *fs_ctx, uint32_t idx, struct its_file_meta_t *file_meta)

  *Reads specified file metadata.*
- psa_status_t its_flash_fs_mblock_read_block_metadata (struct its_flash_fs_ctx_t *fs_ctx, uint32_t lblock, struct its_block_meta_t *block_meta)

  *Reads specified logical block metadata.*
- psa_status_t its_flash_fs_mblock_read_block_metadata_comp (struct its_flash_fs_ctx_t *fs_ctx, uint32_↩t lblock, struct its_block_meta_t *block_meta)

  *Reads specified logical block metadata based on the backward compatible FS.*
- psa_status_t its_flash_fs_mblock_reserve_file (struct its_flash_fs_ctx_t *fs_ctx, const uint8_t *fid, bool use_spare, size_t size, uint32_t flags, uint32_t *file_meta_idx, struct its_file_meta_t *file_meta, struct its_block_meta_t *block_meta)

  *Reserves space for a file.*
- psa_status_t its_flash_fs_mblock_reset_metablock (struct its_flash_fs_ctx_t *fs_ctx)

  *Resets metablock by cleaning and initializing the metadatablock.*
- void its_flash_fs_mblock_set_data_scratch (struct its_flash_fs_ctx_t *fs_ctx, uint32_t phy_id, uint32_t lblock)

  *Sets current data scratch block.*
- psa_status_t its_flash_fs_mblock_update_scratch_block_meta (struct its_flash_fs_ctx_t *fs_ctx, uint32_↩t lblock, struct its_block_meta_t *block_meta)

  *Puts logical block's metadata in scratch metadata block.*
- psa_status_t its_flash_fs_mblock_update_scratch_file_meta (struct its_flash_fs_ctx_t *fs_ctx, uint32_t idx, const struct its_file_meta_t *file_meta)

  *Writes a file metadata entry into scratch metadata block.*
- psa_status_t its_flash_fs_block_to_block_move (struct its_flash_fs_ctx_t *fs_ctx, uint32_t dst_block, size_t dst_offset, uint32_t src_block, size_t src_offset, size_t size)

  *Moves data from source block ID to destination block ID.*

## 7.283.1 Macro Definition Documentation

### 7.283.1.1 _T1

```
#define _T1
```

**Value:**

```
    uint32_t scratch_dblock;      \
    uint8_t fs_version;           \
    uint8_t metadata_xor;         \
    uint8_t active_swap_count;
```

Number of times the metadata blocks have \ been swapped \

Definition at line 67 of file its_flash_fs_mblock.h.

### 7.283.1.2 _T1_COMP

```
#define _T1_COMP
```

**Value:**

```
    uint32_t scratch_dblock;      \
    uint8_t fs_version;           \
    uint8_t active_swap_count;
```

Number of times the metadata blocks have \ been swapped \
Definition at line 99 of file its_flash_fs_mblock.h.

#### 7.283.1.3 _T2

```
#define _T2
```

**Value:**

```
    uint32_t phy_id;      \
    size_t data_start;    \
    size_t free_size;
```

Number of bytes free at end of block (set during \ block compaction for gap reuse) \
Definition at line 123 of file its_flash_fs_mblock.h.

#### 7.283.1.4 _T3

```
#define _T3
```

**Value:**

```
    uint32_t lblock;              /* Logical datablock where file is stored */ \
    size_t data_idx;              /* Offset in the logical data block */ \
    size_t cur_size;              /* Size in storage system for this fragment */ \
    size_t max_size;              /* Maximum size of this file */ \
    uint32_t flags;               /* Flags set when the file was created */ \
    uint8_t id[ITS_FILE_ID_SIZE]  /* ID of this file */
```

Definition at line 158 of file its_flash_fs_mblock.h.

#### 7.283.1.5 ITS_BACKWARD_SUPPORTED_VERSION

```
#define ITS_BACKWARD_SUPPORTED_VERSION 0x01
```

Defines the backward supported version.
Definition at line 38 of file its_flash_fs_mblock.h.

#### 7.283.1.6 ITS_LOGICAL_DBLOCK0

```
#define ITS_LOGICAL_DBLOCK0 0
```

Defines logical data block 0 ID.
Definition at line 52 of file its_flash_fs_mblock.h.

#### 7.283.1.7 ITS_METADATA_INVALID_INDEX

```
#define ITS_METADATA_INVALID_INDEX 0xFFFF
```

Defines the invalid index value when the metadata table is full.
Definition at line 45 of file its_flash_fs_mblock.h.

#### 7.283.1.8 ITS_SUPPORTED_VERSION

```
#define ITS_SUPPORTED_VERSION 0x02
```

Defines the supported version.
Definition at line 31 of file its_flash_fs_mblock.h.

### 7.283.2 Function Documentation

#### 7.283.2.1 its_flash_fs_block_to_block_move()

```
psa_status_t its_flash_fs_block_to_block_move (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t dst_block,
            size_t dst_offset,
            uint32_t src_block,
            size_t src_offset,
            size_t size )
```

Moves data from source block ID to destination block ID.

**Parameters**

| | | |
|---|---|---|
| in | *fs_ctx* | Filesystem context |
| in | *dst_block* | Destination block ID |
| in | *dst_offset* | Destination offset position from the init of the destination block |
| in | *src_block* | Source block ID |
| in | *src_offset* | Source offset position from the init of the source block |
| in | *size* | Number of bytes to moves |

**Note**

This function assumes all input values are valid. That is, the address range, based on blockid, offset and size, is a valid range in flash. It also assumes that the destination block is already erased and ready to be written.

**Returns**

Returns PSA_SUCCESS if the function is executed correctly. Otherwise, it returns PSA_ERROR_STORA↩GE_FAILURE.

Definition at line 1379 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

#### 7.283.2.2 its_flash_fs_mblock_cp_file_meta()

```
psa_status_t its_flash_fs_mblock_cp_file_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
```

```
uint32_t idx_start,
uint32_t idx_end )
```

Copies the file metadata entries between two indexes from the active metadata block to the scratch metadata block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *idx_start* | File metadata entry index to start copy, inclusive |
| in | *idx_end* | File metadata entry index to end copy, exclusive |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 976 of file its_flash_fs_mblock.c.

#### 7.283.2.3 its_flash_fs_mblock_cur_data_scratch_id()

```
uint32_t its_flash_fs_mblock_cur_data_scratch_id (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock )
```

Gets current scratch datablock physical ID.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical block number |

**Returns**

current scratch data block

Definition at line 992 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

#### 7.283.2.4 its_flash_fs_mblock_get_file_idx_flag()

```
psa_status_t its_flash_fs_mblock_get_file_idx_flag (
            struct its_flash_fs_ctx_t * fs_ctx,
```

```
uint32_t flags,
uint32_t * idx )
```

Gets file metadata entry index of the first file with one of the provided flags set.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *flags* | Flags to search for |
| out | *idx* | Index of the file metadata in the file system |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1033 of file its_flash_fs_mblock.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.283.2.5 its_flash_fs_mblock_get_file_idx_meta()

```
psa_status_t its_flash_fs_mblock_get_file_idx_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            uint32_t * idx,
            struct its_file_meta_t * file_meta )
```

Gets file metadata entry index and file metadata.

**Note**

A NULL [file_meta] indicates ignoring file meta.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | ID of the file |
| out | *idx* | Index of the file metadata in the file system |
| out | *file_meta* | Pointer to file meta structure |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1004 of file its_flash_fs_mblock.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.283.2.6 its_flash_fs_mblock_init()

```
psa_status_t its_flash_fs_mblock_init (
            struct its_flash_fs_ctx_t * fs_ctx )
```

Initializes metadata block with the valid/active metablock.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |

**Returns**

Returns value as specified in psa_status_t

Definition at line 1058 of file its_flash_fs_mblock.c.

Here is the caller graph for this function:

#### 7.283.2.7 its_flash_fs_mblock_meta_update_finalize()

```
psa_status_t its_flash_fs_mblock_meta_update_finalize (
            struct its_flash_fs_ctx_t * fs_ctx )
```

Finalizes an update operation. Last step when a create/write/delete is performed.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |

**Returns**

Returns offset value in metadata block

Definition at line 1084 of file its_flash_fs_mblock.c.

#### 7.283.2.8 its_flash_fs_mblock_migrate_lb0_data_to_scratch()

```
psa_status_t its_flash_fs_mblock_migrate_lb0_data_to_scratch (
            struct its_flash_fs_ctx_t * fs_ctx )
```

Writes the files data area of logical block 0 into the scratch block.

**Note**

The files data in the logical block 0 is stored in same physical block where the metadata is stored. A change in the metadata requires a swap of physical blocks. So, the files data stored in the current medadata block needs to be copied in the scratch block, unless the data of the file processed is located in the logical block 0.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1108 of file its_flash_fs_mblock.c.
Here is the call graph for this function:

### 7.283.2.9 its_flash_fs_mblock_read_block_metadata()

```
psa_status_t its_flash_fs_mblock_read_block_metadata (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock,
            struct its_block_meta_t * block_meta )
```

Reads specified logical block metadata.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical block number |
| out | *block_meta* | Pointer to block meta structure |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1153 of file its_flash_fs_mblock.c.

Here is the caller graph for this function:

#### 7.283.2.10 its_flash_fs_mblock_read_block_metadata_comp()

```
psa_status_t its_flash_fs_mblock_read_block_metadata_comp (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock,
            struct its_block_meta_t * block_meta )
```

Reads specified logical block metadata based on the backward compatible FS.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical block number |
| out | *block_meta* | Pointer to block meta structure |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1175 of file its_flash_fs_mblock.c.

#### 7.283.2.11 its_flash_fs_mblock_read_file_meta()

```
psa_status_t its_flash_fs_mblock_read_file_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t idx,
            struct its_file_meta_t * file_meta )
```

Reads specified file metadata.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *idx* | File metadata entry index |
| out | *file_meta* | Pointer to file meta structure |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1131 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

### 7.283.2.12 its_flash_fs_mblock_reserve_file()

```
psa_status_t its_flash_fs_mblock_reserve_file (
            struct its_flash_fs_ctx_t * fs_ctx,
            const uint8_t * fid,
            bool use_spare,
            size_t size,
            uint32_t flags,
            uint32_t * file_meta_idx,
            struct its_file_meta_t * file_meta,
            struct its_block_meta_t * block_meta )
```

Reserves space for a file.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *fid* | File ID |
| in | *use_spare* | If true then the spare file will be used, otherwise at least one file will be left free |
| in | *size* | Size of the file for which space is reserve |
| in | *flags* | Flags set when the file was created |
| out | *file_meta_idx* | File metadata entry index |
| out | *file_meta* | File metadata entry |
| out | *block_meta* | Block metadata entry |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1199 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

### 7.283.2.13 its_flash_fs_mblock_reset_metablock()

```
psa_status_t its_flash_fs_mblock_reset_metablock (
            struct its_flash_fs_ctx_t * fs_ctx )
```

Resets metablock by cleaning and initializing the metadatablock.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |

**Returns**

Returns value as specified in psa_status_t

Definition at line 1223 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

### 7.283.2.14 its_flash_fs_mblock_set_data_scratch()

```
void its_flash_fs_mblock_set_data_scratch (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t phy_id,
            uint32_t lblock )
```

Sets current data scratch block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *phy↩ _id* | Physical ID of scratch data block |
| in | *lblock* | Logical block number |

Definition at line 1334 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

#### 7.283.2.15 its_flash_fs_mblock_update_scratch_block_meta()

```
psa_status_t its_flash_fs_mblock_update_scratch_block_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t lblock,
            struct its_block_meta_t * block_meta )
```

Puts logical block's metadata in scratch metadata block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *lblock* | Logical block number |
| in | *block_meta* | Pointer to block's metadata |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1342 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

#### 7.283.2.16 its_flash_fs_mblock_update_scratch_file_meta()

```
psa_status_t its_flash_fs_mblock_update_scratch_file_meta (
            struct its_flash_fs_ctx_t * fs_ctx,
            uint32_t idx,
            const struct its_file_meta_t * file_meta )
```

Writes a file metadata entry into scratch metadata block.

**Parameters**

| | | |
|---|---|---|
| in,out | *fs_ctx* | Filesystem context |
| in | *idx* | File's index in the metadata table |
| in | *file_meta* | Metadata pointer |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1365 of file its_flash_fs_mblock.c.
Here is the caller graph for this function:

## 7.284 secure_fw/partitions/internal_trusted_storage/its_crypto_interface.c File Reference

```
#include <string.h>
#include "flash_fs/its_flash_fs.h"
#include "flash/its_flash.h"
#include "its_utils.h"
#include "psa_manifest/pid.h"
#include "tfm_hal_its_encryption.h"
#include "tfm_hal_ps.h"
#include "tfm_internal_trusted_storage.h"
#include "tfm_its_defs.h"
#include "tfm_sp_log.h"
```

Include dependency graph for its_crypto_interface.c:

### Functions

- psa_status_t tfm_its_crypt_file (struct its_flash_fs_file_info_t ∗finfo, uint8_t ∗fid, const size_t fid_size, const uint8_t ∗input, const size_t input_size, uint8_t ∗output, const size_t output_size, const bool is_encrypt)

  *Perform encryption/decryption of the buffer using the tfm_hal_its APIs.*

### 7.284.1 Function Documentation

#### 7.284.1.1 tfm_its_crypt_file()

```
psa_status_t tfm_its_crypt_file (
            struct its_flash_fs_file_info_t * finfo,
            uint8_t * fid,
            const size_t fid_size,
            const uint8_t * input,
            const size_t input_size,
            uint8_t * output,
            const size_t output_size,
            const bool is_encrypt )
```

Perform encryption/decryption of the buffer using the tfm_hal_its APIs.

**Parameters**

| | | |
|---|---|---|
| in | *finfo* | Pointer to its_flash_fs_file_info_t |
| in | *fid* | File identifier |
| in | *fid_size* | File identifier size in bytes |
| in | *input* | Input buffer |
| in | *input_size* | Input size in bytes |
| out | *output* | Output buffer |
| in | *output_size* | Output size in bytes |
| in | *is_encrypt* | Set the operation type (encryption/decryption) |

**Returns**

PSA_SUCCESS on successful operation or a valid PSA error code

Definition at line 86 of file its_crypto_interface.c.

## 7.285 secure_fw/partitions/internal_trusted_storage/its_crypto_interface.h File Reference

```
#include "flash_fs/its_flash_fs.h"
#include "flash/its_flash.h"
#include "its_utils.h"
#include "psa_manifest/pid.h"
#include "tfm_hal_its.h"
#include "tfm_hal_ps.h"
#include "tfm_internal_trusted_storage.h"
#include "tfm_its_defs.h"
#include "tfm_sp_log.h"
```

Include dependency graph for its_crypto_interface.h:

## Functions

- psa_status_t tfm_its_crypt_file (struct its_flash_fs_file_info_t ∗finfo, uint8_t ∗fid, const size_t fid_size, const uint8_t ∗input, const size_t input_size, uint8_t ∗output, const size_t output_size, const bool is_encrypt)

  *Perform encryption/decryption of the buffer using the tfm_hal_its APIs.*

## 7.285.1 Function Documentation

### 7.285.1.1 tfm_its_crypt_file()

```
psa_status_t tfm_its_crypt_file (
            struct its_flash_fs_file_info_t * finfo,
            uint8_t * fid,
            const size_t fid_size,
            const uint8_t * input,
            const size_t input_size,
            uint8_t * output,
            const size_t output_size,
            const bool is_encrypt )
```

Perform encryption/decryption of the buffer using the tfm_hal_its APIs.

**Parameters**

| | | |
|---|---|---|
| in | *finfo* | Pointer to its_flash_fs_file_info_t |
| in | *fid* | File identifier |
| in | *fid_size* | File identifier size in bytes |
| in | *input* | Input buffer |
| in | *input_size* | Input size in bytes |
| out | *output* | Output buffer |
| in | *output_size* | Output size in bytes |
| in | *is_encrypt* | Set the operation type (encryption/decryption) |

**Returns**

PSA_SUCCESS on successful operation or a valid PSA error code

Definition at line 86 of file its_crypto_interface.c.

## 7.286 secure_fw/partitions/internal_trusted_storage/its_utils.c File Reference

```
#include "its_utils.h"
```

Include dependency graph for its_utils.c:

### Functions

- psa_status_t its_utils_check_contained_in (size_t superset_size, size_t subset_offset, size_t subset_size)

  *Checks if a subset region is fully contained within a superset region.*
- psa_status_t its_utils_validate_fid (const uint8_t *fid)

  *Validates file ID.*

### 7.286.1 Function Documentation

#### 7.286.1.1 its_utils_check_contained_in()

```
psa_status_t its_utils_check_contained_in (
            size_t superset_size,
            size_t subset_offset,
            size_t subset_size )
```

Checks if a subset region is fully contained within a superset region.

**Parameters**

| | | |
|---|---|---|
| in | *superset_size* | Size of superset region |
| in | *subset_offset* | Offset of start of subset region from start of superset region |
| in | *subset_size* | Size of subset region |

**Returns**

Returns error code as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The subset is contained within the superset |
| *PSA_ERROR_INVALID_ARGUMENT* | Otherwise |

Definition at line 10 of file its_utils.c.
Here is the caller graph for this function:

#### 7.286.1.2 its_utils_validate_fid()

```
psa_status_t its_utils_validate_fid (
            const uint8_t * fid )
```

Validates file ID.

**Parameters**

| | | |
|---|---|---|
| in | *fid* | File ID |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 30 of file its_utils.c.

## 7.287 secure_fw/partitions/internal_trusted_storage/its_utils.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "psa/error.h"
```

Include dependency graph for its_utils.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define ITS_FILE_ID_SIZE 12
- #define ITS_DATA_SIZE_FIELD_SIZE 4
- #define ITS_FLAG_SIZE 4
- #define ITS_DEFAULT_EMPTY_BUFF_VAL 0
- #define ITS_UTILS_BOUND_CHECK(err_msg, data_size, data_buf_size) typedef char err_msg[(data_size <= data_buf_size)∗2 - 1]

  *Macro to check, at compilation time, if data fits in data buffer.*
- #define ITS_UTILS_MIN(x, y) (((x) < (y)) ? (x) : (y))

  *Evaluates to the minimum of the two parameters.*
- #define ITS_UTILS_MAX(x, y) (((x) > (y)) ? (x) : (y))

  *Evaluates to the maximum of the two parameters.*
- #define ITS_UTILS_ALIGN(x, a) (((x) + ((a) - 1)) & ∼((a) - 1))

  *Aligns a value up to the provided alignment.*
- #define ITS_UTILS_IS_ALIGNED(x, a) (((x) & ((a) - 1)) == 0)

  *Checks that a value is aligned to the provided alignment.*

## Functions

- psa_status_t its_utils_check_contained_in (size_t superset_size, size_t subset_offset, size_t subset_size)

  *Checks if a subset region is fully contained within a superset region.*
- psa_status_t its_utils_validate_fid (const uint8_t *fid)

  *Validates file ID.*

## 7.287.1 Macro Definition Documentation

### 7.287.1.1 ITS_DATA_SIZE_FIELD_SIZE

```
#define ITS_DATA_SIZE_FIELD_SIZE 4
```

Definition at line 21 of file its_utils.h.

### 7.287.1.2 ITS_DEFAULT_EMPTY_BUFF_VAL

```
#define ITS_DEFAULT_EMPTY_BUFF_VAL 0
```

Definition at line 23 of file its_utils.h.

### 7.287.1.3 ITS_FILE_ID_SIZE

```
#define ITS_FILE_ID_SIZE 12
```

Definition at line 20 of file its_utils.h.

### 7.287.1.4 ITS_FLAG_SIZE

```
#define ITS_FLAG_SIZE 4
```

Definition at line 22 of file its_utils.h.

### 7.287.1.5 ITS_UTILS_ALIGN

```
#define ITS_UTILS_ALIGN(
            x,
            a ) (((x) + ((a) - 1)) & ~((a) - 1))
```

Aligns a value up to the provided alignment.

**Parameters**

| | | |
|---|---|---|
| in | *x* | Value to be aligned |
| in | *a* | Alignment (must be a power of two) |

**Returns**

The least value not less than `x` that is aligned to `a`.

Definition at line 58 of file its_utils.h.

### 7.287.1.6 ITS_UTILS_BOUND_CHECK

```
#define ITS_UTILS_BOUND_CHECK(
            err_msg,
```

```
data_size,
data_buf_size ) typedef char err_msg[(data_size <= data_buf_size)*2 - 1]
```

Macro to check, at compilation time, if data fits in data buffer.

**Parameters**

| | | |
|---|---|---|
| in | *err_msg* | Error message which will be displayed in first instance if the error is triggered |
| in | *data_size* | Data size to check if it fits |
| in | *data_buf_size* | Size of the data buffer |

**Returns**

Triggers a compilation error if data_size is bigger than data_buf_size. The compilation error should be "... error: 'err_msg' declared as an array with a negative size"

Definition at line 37 of file its_utils.h.

#### 7.287.1.7 ITS_UTILS_IS_ALIGNED

```
#define ITS_UTILS_IS_ALIGNED(
            x,
            a ) (((x) & ((a) - 1)) == 0)
```

Checks that a value is aligned to the provided alignment.

**Parameters**

| | | |
|---|---|---|
| in | *x* | Value to check for alignment |
| in | *a* | Alignment (must be a power of two) |

**Returns**

1 if `x` is aligned to `a`, 0 otherwise.

Definition at line 68 of file its_utils.h.

#### 7.287.1.8 ITS_UTILS_MAX

```
#define ITS_UTILS_MAX(
            x,
            y ) (((x) > (y)) ?  (x) :  (y))
```

Evaluates to the maximum of the two parameters.
Definition at line 48 of file its_utils.h.

#### 7.287.1.9 ITS_UTILS_MIN

```
#define ITS_UTILS_MIN(
            x,
            y ) (((x) < (y)) ?  (x) :  (y))
```

Evaluates to the minimum of the two parameters.
Definition at line 43 of file its_utils.h.

### 7.287.2 Function Documentation

#### 7.287.2.1 its_utils_check_contained_in()

```
psa_status_t its_utils_check_contained_in (
            size_t superset_size,
            size_t subset_offset,
            size_t subset_size )
```

Checks if a subset region is fully contained within a superset region.

**Parameters**

| | | |
|---|---|---|
| in | *superset_size* | Size of superset region |
| in | *subset_offset* | Offset of start of subset region from start of superset region |
| in | *subset_size* | Size of subset region |

**Returns**

Returns error code as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The subset is contained within the superset |
| *PSA_ERROR_INVALID_ARGUMENT* | Otherwise |

Definition at line 10 of file its_utils.c.
Here is the caller graph for this function:

#### 7.287.2.2 its_utils_validate_fid()

```
psa_status_t its_utils_validate_fid (
            const uint8_t * fid )
```

Validates file ID.

**Parameters**

| | | |
|---|---|---|
| in | *fid* | File ID |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 30 of file its_utils.c.

## 7.288 secure_fw/partitions/internal_trusted_storage/tfm_internal_↩ trusted_storage.c File Reference

```
#include <string.h>
#include "psa/framework_feature.h"
#include "cmsis_compiler.h"
#include "config_tfm.h"
#include "tfm_internal_trusted_storage.h"
#include "tfm_its_req_mngr.h"
#include "tfm_hal_its.h"
#include "flash/its_flash.h"
#include "flash_fs/its_flash_fs.h"
#include "psa_manifest/pid.h"
#include "tfm_its_defs.h"
#include "its_utils.h"
#include "tfm_sp_log.h"
```

Include dependency graph for tfm_internal_trusted_storage.c:

### Functions

- psa_status_t tfm_its_init (void)

  *Initializes the internal trusted storage system.*
- psa_status_t tfm_its_set (int32_t client_id, psa_storage_uid_t uid, size_t data_length, psa_storage_create_flags_t create_flags)

  *Create a new, or modify an existing, uid/value pair.*
- psa_status_t tfm_its_get (int32_t client_id, psa_storage_uid_t uid, size_t data_offset, size_t data_size, size↩ _t *p_data_length)

  *Retrieve data associated with a provided UID.*
- psa_status_t tfm_its_get_info (int32_t client_id, psa_storage_uid_t uid, struct psa_storage_info_t *p_info)

  *Retrieve the metadata about the provided uid.*
- psa_status_t tfm_its_remove (int32_t client_id, psa_storage_uid_t uid)

  *Remove the provided uid and its associated data from the storage.*

### Variables

- uint8_t * p_psa_src_data
- uint8_t * p_psa_dest_data

## 7.288.1 Function Documentation

### 7.288.1.1 tfm_its_get()

```
psa_status_t tfm_its_get (
            int32_t client_id,
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_size,
            size_t * p_data_length )
```

Retrieve data associated with a provided UID.

Retrieves up to `data_size` bytes of the data associated with `uid`, starting at `data_offset` bytes from the beginning of the data. Upon successful completion, the data will be placed in the `p_data` buffer, which must be at least `data_size` bytes in size. The length of the data returned will be in `p_data_length`. If `data_size` is 0, the contents of `p_data_length` will be set to zero.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | The uid value |
| in | *data_offset* | The starting offset of the data requested |
| in | *data_size* | The amount of data requested |
| out | *p_data_length* | On success, this will contain size of the data placed in `p_data`. |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided `uid` value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided arguments (`p_data`, `p_data_length`) is invalid, for example is `NULL` or references memory the caller cannot access. In addition, this can also happen if `data_offset` is larger than the size of the data associated with `uid`. |

Definition at line 573 of file tfm_internal_trusted_storage.c.

Here is the caller graph for this function:

#### 7.288.1.2 tfm_its_get_info()

```
psa_status_t tfm_its_get_info (
            int32_t client_id,
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Retrieve the metadata about the provided uid.
Retrieves the metadata stored for a given `uid` as a `psa_storage_info_t` structure.

**Parameters**

| | | |
|---|---|---|
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| in | *uid* | The `uid` value |
| out | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_info`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 637 of file tfm_internal_trusted_storage.c.

Here is the caller graph for this function:

#### 7.288.1.3 tfm_its_init()

```
psa_status_t tfm_its_init (
            void  )
```

Initializes the internal trusted storage system.

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the storage system initialization has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 280 of file tfm_internal_trusted_storage.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.288.1.4 tfm_its_remove()

```
psa_status_t tfm_its_remove (
            int32_t client_id,
            psa_storage_uid_t uid )
```

Remove the provided uid and its associated data from the storage.
Deletes the data from internal storage.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | The `uid` value |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, wrong flags and so on) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |

Definition at line 656 of file tfm_internal_trusted_storage.c.

Here is the caller graph for this function:

#### 7.288.1.5 tfm_its_set()

```
psa_status_t tfm_its_set (
            int32_t client_id,
            psa_storage_uid_t uid,
            size_t data_length,
            psa_storage_create_flags_t create_flags )
```

Create a new, or modify an existing, uid/value pair.
Stores data in the internal storage.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | The identifier for the data |
| in | *data_length* | The size in bytes of the data in `p_data` |
| in | *create_flags* | The flags that the data will be stored with |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because one or more of the flags provided in `create_flags` is not supported or is not valid |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because there was insufficient space on the storage medium |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers (`p_data`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 422 of file tfm_internal_trusted_storage.c.

Here is the caller graph for this function:

### 7.288.2 Variable Documentation

#### 7.288.2.1 p_psa_dest_data

`uint8_t* p_psa_dest_data`
Definition at line 15 of file ps_filesystem_interface.c.

#### 7.288.2.2 p_psa_src_data

`uint8_t* p_psa_src_data`
Definition at line 14 of file ps_filesystem_interface.c.

## 7.289 secure_fw/partitions/internal_trusted_storage/tfm_internal_trusted_storage.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include "psa/error.h"
#include "psa/storage_common.h"
#include "flash_fs/its_flash_fs.h"
#include "its_utils.h"
```

Include dependency graph for tfm_internal_trusted_storage.h:

This graph shows which files directly or indirectly include this file:

## Functions

- psa_status_t tfm_its_init (void)

  *Initializes the internal trusted storage system.*
- psa_status_t tfm_its_set (int32_t client_id, psa_storage_uid_t uid, size_t data_length, psa_storage_create_flags_t create_flags)

  *Create a new, or modify an existing, uid/value pair.*
- psa_status_t tfm_its_get (int32_t client_id, psa_storage_uid_t uid, size_t data_offset, size_t data_size, size↩_t *p_data_length)

  *Retrieve data associated with a provided UID.*
- psa_status_t tfm_its_get_info (int32_t client_id, psa_storage_uid_t uid, struct psa_storage_info_t *p_info)

  *Retrieve the metadata about the provided uid.*
- psa_status_t tfm_its_remove (int32_t client_id, psa_storage_uid_t uid)

  *Remove the provided uid and its associated data from the storage.*

## 7.289.1 Function Documentation

### 7.289.1.1 tfm_its_get()

```
psa_status_t tfm_its_get (
            int32_t client_id,
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_size,
            size_t * p_data_length )
```

Retrieve data associated with a provided UID.

Retrieves up to `data_size` bytes of the data associated with `uid`, starting at `data_offset` bytes from the beginning of the data. Upon successful completion, the data will be placed in the `p_data` buffer, which must be at least `data_size` bytes in size. The length of the data returned will be in `p_data_length`. If `data_size` is 0, the contents of `p_data_length` will be set to zero.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | The uid value |
| in | *data_offset* | The starting offset of the data requested |
| in | *data_size* | The amount of data requested |
| out | *p_data_length* | On success, this will contain size of the data placed in `p_data`. |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided `uid` value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided arguments (`p_data`, `p_data_length`) is invalid, for example is `NULL` or references memory the caller cannot access. In addition, this can also happen if `data_offset` is larger than the size of the data associated with `uid`. |

Definition at line 573 of file tfm_internal_trusted_storage.c.
Here is the caller graph for this function:

### 7.289.1.2 tfm_its_get_info()

```
psa_status_t tfm_its_get_info (
            int32_t client_id,
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Retrieve the metadata about the provided uid.
Retrieves the metadata stored for a given `uid` as a `psa_storage_info_t` structure.

**Parameters**

| | | |
|---|---|---|
| `in` | *client↩ _id* | Identifier of the asset's owner (client) |
| `in` | *uid* | The `uid` value |
| `out` | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |

**Return values**

| | |
|---|---|
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_info`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 637 of file tfm_internal_trusted_storage.c.
Here is the caller graph for this function:

#### 7.289.1.3 tfm_its_init()

```
psa_status_t tfm_its_init (
            void  )
```

Initializes the internal trusted storage system.

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the storage system initialization has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 280 of file tfm_internal_trusted_storage.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.289.1.4 tfm_its_remove()

```
psa_status_t tfm_its_remove (
            int32_t client_id,
            psa_storage_uid_t uid )
```

Remove the provided uid and its associated data from the storage.
Deletes the data from internal storage.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | The `uid` value |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, wrong flags and so on) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |

Definition at line 656 of file tfm_internal_trusted_storage.c.

Here is the caller graph for this function:

#### 7.289.1.5 tfm_its_set()

```
psa_status_t tfm_its_set (
            int32_t client_id,
            psa_storage_uid_t uid,
            size_t data_length,
            psa_storage_create_flags_t create_flags )
```

Create a new, or modify an existing, uid/value pair.
Stores data in the internal storage.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | The identifier for the data |
| in | *data_length* | The size in bytes of the data in `p_data` |
| in | *create_flags* | The flags that the data will be stored with |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because one or more of the flags provided in `create_flags` is not supported or is not valid |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because there was insufficient space on the storage medium |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers (`p_data`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 422 of file tfm_internal_trusted_storage.c.

Here is the caller graph for this function:

## 7.290 secure_fw/partitions/internal_trusted_storage/tfm_its_req_mngr.c File Reference

```
#include <stdint.h>
#include <string.h>
#include <stdbool.h>
#include "cmsis_compiler.h"
#include "config_tfm.h"
#include "psa/storage_common.h"
#include "tfm_internal_trusted_storage.h"
#include "psa/framework_feature.h"
#include "psa/service.h"
#include "psa_manifest/tfm_internal_trusted_storage.h"
#include "tfm_its_defs.h"
```

Include dependency graph for tfm_its_req_mngr.c:

### Functions

- psa_status_t tfm_its_entry (void)
- psa_status_t tfm_internal_trusted_storage_service_sfn (const psa_msg_t *msg)
- size_t its_req_mngr_read (uint8_t *buf, size_t num_bytes)
- void its_req_mngr_write (const uint8_t *buf, size_t num_bytes)

### 7.290.1 Function Documentation

#### 7.290.1.1 its_req_mngr_read()

```
size_t its_req_mngr_read (
            uint8_t * buf,
            size_t num_bytes )
```

Definition at line 178 of file tfm_its_req_mngr.c.

Here is the call graph for this function:

#### 7.290.1.2 its_req_mngr_write()

```
void its_req_mngr_write (
            const uint8_t * buf,
            size_t num_bytes )
```

Definition at line 183 of file tfm_its_req_mngr.c.
Here is the call graph for this function:

#### 7.290.1.3 tfm_internal_trusted_storage_service_sfn()

```
psa_status_t tfm_internal_trusted_storage_service_sfn (
            const psa_msg_t * msg )
```

Definition at line 155 of file tfm_its_req_mngr.c.

#### 7.290.1.4 tfm_its_entry()

```
psa_status_t tfm_its_entry (
            void  )
```

Definition at line 150 of file tfm_its_req_mngr.c.
Here is the call graph for this function:

## 7.291 secure_fw/partitions/internal_trusted_storage/tfm_its_req_mngr.h File Reference

```
#include <stdint.h>
#include "config_tfm.h"
#include "psa/framework_feature.h"
```

Include dependency graph for tfm_its_req_mngr.h:

This graph shows which files directly or indirectly include this file:

### Functions

- size_t its_req_mngr_read (uint8_t *buf, size_t num_bytes)
- void its_req_mngr_write (const uint8_t *buf, size_t num_bytes)

### 7.291.1 Function Documentation

#### 7.291.1.1 its_req_mngr_read()

```
size_t its_req_mngr_read (
            uint8_t * buf,
            size_t num_bytes )
```

Definition at line 178 of file tfm_its_req_mngr.c.
Here is the call graph for this function:

#### 7.291.1.2 its_req_mngr_write()

```
void its_req_mngr_write (
            const uint8_t * buf,
            size_t num_bytes )
```

Definition at line 183 of file tfm_its_req_mngr.c.
Here is the call graph for this function:

## 7.292 secure_fw/partitions/lib/runtime/crt_memcmp.c File Reference

```
#include <stddef.h>
#include <stdint.h>
```

Include dependency graph for crt_memcmp.c:

## Functions

- int memcmp (const void *s1, const void *s2, size_t n)

### 7.292.1 Function Documentation

#### 7.292.1.1 memcmp()

```
int memcmp (
            const void * s1,
            const void * s2,
            size_t n )
```

Definition at line 11 of file crt_memcmp.c.
Here is the caller graph for this function:

## 7.293 secure_fw/partitions/lib/runtime/crt_memmove.c File Reference

```
#include <string.h>
#include "crt_impl_private.h"
#include "static_checks.h"
```

Include dependency graph for crt_memmove.c:

### Macros

- #define memcpy_f memcpy

### Functions

- void * memmove (void *dest, const void *src, size_t n)

### 7.293.1 Macro Definition Documentation

#### 7.293.1.1 memcpy_f

```
#define memcpy_f memcpy
```

Definition at line 45 of file crt_memmove.c.

### 7.293.2 Function Documentation

#### 7.293.2.1 memmove()

```
void* memmove (
            void * dest,
            const void * src,
            size_t n )
```

Definition at line 52 of file crt_memmove.c.

## 7.294 secure_fw/partitions/lib/runtime/crt_strnlen.c File Reference

```
#include <stddef.h>
```

Include dependency graph for crt_strnlen.c:

### Functions

- size_t tfm_strnlen (const char *s, size_t maxlen)

  *Return the length of a given string, up to a maximum of maxlen bytes.*

### 7.294.1 Function Documentation

#### 7.294.1.1 tfm_strnlen()

```
size_t tfm_strnlen (
            const char * s,
            size_t maxlen )
```

Return the length of a given string, up to a maximum of maxlen bytes.

**Parameters**

| | | |
|---|---|---|
| in | *s* | Points to the string to be examined. |
| in | *maxlen* | The maximum number of characters to examine. |

Definition at line 10 of file crt_strnlen.c.

## 7.295 secure_fw/partitions/lib/runtime/include/service_api.h File Reference

```
#include <stdint.h>
#include "tfm_boot_status.h"
#include "psa/error.h"
```

Include dependency graph for service_api.h:

This graph shows which files directly or indirectly include this file:

## Functions

- psa_status_t tfm_core_get_boot_data (uint8_t major_type, struct tfm_boot_data ∗boot_data, uint32_t len)

  *Retrieve secure partition related data from shared memory area, which stores shared data between bootloader and runtime firmware.*

### 7.295.1 Function Documentation

#### 7.295.1.1 tfm_core_get_boot_data()

```
psa_status_t tfm_core_get_boot_data (
            uint8_t major_type,
            struct tfm_boot_data * boot_data,
            uint32_t len )
```

Retrieve secure partition related data from shared memory area, which stores shared data between bootloader and runtime firmware.

**Parameters**

| | | |
|---|---|---|
| in | *major_type* | Major type. |
| out | *boot_data* | Pointer to boot data. |
| in | *len* | The length of the boot data. |

Definition at line 15 of file service_api.c.

## 7.296 secure_fw/partitions/lib/runtime/include/tfm_sp_log.h File Reference

This graph shows which files directly or indirectly include this file:

## Macros

- #define TFM_PARTITION_LOG_LEVEL_DEBUG 3 /∗ All log APIs output ∗/
- #define TFM_PARTITION_LOG_LEVEL_INFO
- #define TFM_PARTITION_LOG_LEVEL_ERROR
- #define TFM_PARTITION_LOG_LEVEL_SILENCE 0 /∗ All log APIs are suppressed ∗/
- #define LOG_DBGFMT(...)
- #define LOG_INFFMT(...)
- #define LOG_ERRFMT(...)

## Functions

- int printf (const char ∗fmt,...)

### 7.296.1 Macro Definition Documentation

#### 7.296.1.1 LOG_DBGFMT

```
#define LOG_DBGFMT(
                ...  )
```

Definition at line 35 of file tfm_sp_log.h.

#### 7.296.1.2 LOG_ERRFMT

```
#define LOG_ERRFMT(
                ...  )
```

Definition at line 47 of file tfm_sp_log.h.

#### 7.296.1.3 LOG_INFFMT

```
#define LOG_INFFMT(
                ...  )
```

Definition at line 41 of file tfm_sp_log.h.

#### 7.296.1.4 TFM_PARTITION_LOG_LEVEL_DEBUG

```
#define TFM_PARTITION_LOG_LEVEL_DEBUG 3 /* All log APIs output */
```

Definition at line 18 of file tfm_sp_log.h.

#### 7.296.1.5 TFM_PARTITION_LOG_LEVEL_ERROR

```
#define TFM_PARTITION_LOG_LEVEL_ERROR
```

**Value:**

```
                                        1  /*
* Only LOG_ERRFMT APIs output.
*/
```

Definition at line 20 of file tfm_sp_log.h.

#### 7.296.1.6 TFM_PARTITION_LOG_LEVEL_INFO

```
#define TFM_PARTITION_LOG_LEVEL_INFO
```

**Value:**

```
                                        2  /*
* All log APIs output except
* LOG_DBGFMT
*/
```

Definition at line 19 of file tfm_sp_log.h.

#### 7.296.1.7 TFM_PARTITION_LOG_LEVEL_SILENCE

```
#define TFM_PARTITION_LOG_LEVEL_SILENCE 0 /* All log APIs are suppressed */
```

Definition at line 21 of file tfm_sp_log.h.

### 7.296.2 Function Documentation

#### 7.296.2.1 printf()

```
int printf (
            const char * fmt,
             ...  )
```

Definition at line 165 of file tfm_sp_log_raw.c.
Here is the call graph for this function:

## 7.297 secure_fw/partitions/lib/runtime/include/tfm_strnlen.h File Reference

### Functions

- size_t tfm_strnlen (const char *s, size_t maxlen)

  *Return the length of a given string, up to a maximum of maxlen bytes.*

### 7.297.1 Function Documentation

#### 7.297.1.1 tfm_strnlen()

```
size_t tfm_strnlen (
            const char * s,
            size_t maxlen )
```

Return the length of a given string, up to a maximum of maxlen bytes.

**Parameters**

| | | |
|---|---|---|
| in | *s* | Points to the string to be examined. |
| in | *maxlen* | The maximum number of characters to examine. |

Definition at line 10 of file crt_strnlen.c.

## 7.298 secure_fw/partitions/lib/runtime/psa_api_ipc.c File Reference

```
#include <stdint.h>
#include "psa/client.h"
#include "config_impl.h"
#include "tfm_psa_call_pack.h"
#include "sprt_partition_metadata_indicator.h"
#include "runtime_defs.h"
```

Include dependency graph for psa_api_ipc.c:

## Functions

- uint32_t psa_framework_version (void)

  *Retrieve the version of the PSA Framework API that is implemented.*
- uint32_t psa_version (uint32_t sid)

  *Retrieve the version of an RoT Service or indicate that it is not present on this system.*
- psa_status_t tfm_psa_call_pack (psa_handle_t handle, uint32_t ctrl_param, const psa_invec ∗in_vec, psa_outvec ∗out_vec)
- psa_signal_t psa_wait (psa_signal_t signal_mask, uint32_t timeout)

  *Return the Secure Partition interrupt signals that have been asserted from a subset of signals provided by the caller.*
- psa_status_t psa_get (psa_signal_t signal, psa_msg_t ∗msg)

  *Retrieve the message which corresponds to a given RoT Service signal and remove the message from the RoT Service queue.*
- size_t psa_read (psa_handle_t msg_handle, uint32_t invec_idx, void ∗buffer, size_t num_bytes)

  *Read a message parameter or part of a message parameter from a client input vector.*
- size_t psa_skip (psa_handle_t msg_handle, uint32_t invec_idx, size_t num_bytes)

  *Skip over part of a client input vector.*
- void psa_write (psa_handle_t msg_handle, uint32_t outvec_idx, const void ∗buffer, size_t num_bytes)

  *Write a message response to a client output vector.*
- void psa_reply (psa_handle_t msg_handle, psa_status_t retval)

  *Complete handling of a specific message and unblock the client.*
- void psa_panic (void)

  *Terminate execution within the calling Secure Partition and will not return.*
- uint32_t psa_rot_lifecycle_state (void)

## 7.298.1 Function Documentation

### 7.298.1.1 psa_framework_version()

```
uint32_t psa_framework_version (
            void  )
```

Retrieve the version of the PSA Framework API that is implemented.

**Returns**

version The version of the PSA Framework implementation that is providing the runtime services to the caller. The major and minor version are encoded as follows:

- version[15:8] – major version number.
- version[7:0] – minor version number.

Definition at line 18 of file psa_api_ipc.c.

### 7.298.1.2 psa_get()

```
psa_status_t psa_get (
            psa_signal_t signal,
            psa_msg_t * msg )
```

Retrieve the message which corresponds to a given RoT Service signal and remove the message from the RoT Service queue.

**Parameters**

| | | |
|---|---|---|
| in | *signal* | The signal value for an asserted RoT Service. |
| out | *msg* | Pointer to psa_msg_t object for receiving the message. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success, ∗msg will contain the delivered message. |
| *PSA_ERROR_DOES_NOT_EXIST* | Message could not be delivered. |
| *PROGRAMMER ERROR* | The call is invalid because one or more of the following are true:<br>• signal has more than a single bit set.<br>• signal does not correspond to an RoT Service.<br>• The RoT Service signal is not currently asserted.<br>• The msg pointer provided is not a valid memory reference. |

Definition at line 42 of file psa_api_ipc.c.
Here is the caller graph for this function:

### 7.298.1.3 psa_panic()

```
void psa_panic (
            void )
```

Terminate execution within the calling Secure Partition and will not return.

**Return values**

| | |
|---|---|
| *Does not return* | |

Definition at line 69 of file psa_api_ipc.c.

#### 7.298.1.4 psa_read()

```
size_t psa_read (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            void * buffer,
            size_t num_bytes )
```

Read a message parameter or part of a message parameter from a client input vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of the input vector to read from. Must be less than PSA_MAX_IOVEC. |
| out | *buffer* | Buffer in the Secure Partition to copy the requested data to. |
| in | *num_bytes* | Maximum number of bytes to be read from the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes copied. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a PSA_IPC_CALL message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• the memory reference for buffer is invalid or not writable. |

Definition at line 47 of file psa_api_ipc.c.

#### 7.298.1.5 psa_reply()

```
void psa_reply (
            psa_handle_t msg_handle,
            psa_status_t status )
```

Complete handling of a specific message and unblock the client.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *status* | Message result value to be reported to the client. |

**Return values**

| | |
|---|---|
| *void* | Success. |

**Return values**

| | |
|---|---|
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• An invalid status code is specified for the type of message. |

Definition at line 64 of file psa_api_ipc.c.
Here is the caller graph for this function:

### 7.298.1.6 psa_rot_lifecycle_state()

```
uint32_t psa_rot_lifecycle_state (
            void  )
```

Definition at line 74 of file psa_api_ipc.c.

### 7.298.1.7 psa_skip()

```
size_t psa_skip (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            size_t num_bytes )
```

Skip over part of a client input vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of input vector to skip from. Must be less than PSA_MAX_IOVEC. |
| in | *num_bytes* | Maximum number of bytes to skip in the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes skipped. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC. |

Definition at line 53 of file psa_api_ipc.c.

#### 7.298.1.8 psa_version()

```
uint32_t psa_version (
            uint32_t sid )
```

Retrieve the version of an RoT Service or indicate that it is not present on this system.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of the RoT Service to query. |

**Return values**

| | |
|---|---|
| *PSA_VERSION_NONE* | The RoT Service is not implemented, or the caller is not permitted to access the service. |
| > | 0 The version of the implemented RoT Service. |

Definition at line 23 of file psa_api_ipc.c.

#### 7.298.1.9 psa_wait()

```
psa_signal_t psa_wait (
            psa_signal_t signal_mask,
            uint32_t timeout )
```

Return the Secure Partition interrupt signals that have been asserted from a subset of signals provided by the caller.

**Parameters**

| | | |
|---|---|---|
| in | *signal_mask* | A set of signals to query. Signals that are not in this set will be ignored. |
| in | *timeout* | Specify either blocking PSA_BLOCK or polling PSA_POLL operation. |

**Return values**

| | |
|---|---|
| *>0* | At least one signal is asserted. |
| *0* | No signals are asserted. This is only seen when a polling timeout is used. |

Definition at line 37 of file psa_api_ipc.c.
Here is the caller graph for this function:

#### 7.298.1.10 psa_write()

```
void psa_write (
            psa_handle_t msg_handle,
            uint32_t outvec_idx,
            const void * buffer,
            size_t num_bytes )
```

Write a message response to a client output vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| out | *outvec_idx* | Index of output vector in message to write to. Must be less than PSA_MAX_IOVEC. |
| in | *buffer* | Buffer with the data to write. |
| in | *num_bytes* | Number of bytes to write to the client output vector. |

**Return values**

| | |
|---|---|
| *void* | Success |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• outvec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• The memory reference for buffer is invalid.<br>• The call attempts to write data past the end of the client output vector. |

Definition at line 58 of file psa_api_ipc.c.

#### 7.298.1.11 tfm_psa_call_pack()

```
psa_status_t tfm_psa_call_pack (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * in_vec,
            psa_outvec * out_vec )
```

Definition at line 28 of file psa_api_ipc.c.

## 7.299 secure_fw/partitions/lib/runtime/service_api.c File Reference

```
#include "cmsis_compiler.h"
#include "service_api.h"
#include "psa/service.h"
#include "svc_num.h"
#include "utilities.h"
```

Include dependency graph for service_api.c:

## Functions

- psa_status_t tfm_core_get_boot_data (uint8_t major_type, struct tfm_boot_data ∗boot_status, uint32_t len)

  *Retrieve secure partition related data from shared memory area, which stores shared data between bootloader and runtime firmware.*
- void tfm_flih_func_return (psa_flih_result_t result)

## 7.299.1 Function Documentation

### 7.299.1.1 tfm_core_get_boot_data()

```
psa_status_t tfm_core_get_boot_data (
            uint8_t major_type,
            struct tfm_boot_data * boot_data,
            uint32_t len )
```

Retrieve secure partition related data from shared memory area, which stores shared data between bootloader and runtime firmware.

**Parameters**

| | | |
|---|---|---|
| in | *major_type* | Major type. |
| out | *boot_data* | Pointer to boot data. |
| in | *len* | The length of the boot data. |

Definition at line 15 of file service_api.c.

### 7.299.1.2 tfm_flih_func_return()

```
void tfm_flih_func_return (
            psa_flih_result_t result )
```

Definition at line 28 of file service_api.c.
Here is the caller graph for this function:

## 7.300 secure_fw/partitions/lib/runtime/sfn_common_thread.c File Reference

```
#include <stdint.h>
#include "runtime_defs.h"
#include "sprt_partition_metadata_indicator.h"
#include "tfm_sp_log.h"
#include "psa/error.h"
#include "psa/service.h"
```

Include dependency graph for sfn_common_thread.c:

### Functions

- void common_sfn_thread (void *param)

### 7.300.1 Function Documentation

#### 7.300.1.1 common_sfn_thread()

```
void common_sfn_thread (
            void * param )
```

Definition at line 20 of file sfn_common_thread.c.

Here is the call graph for this function:

## 7.301 secure_fw/partitions/lib/runtime/sprt_partition_metadata_indicator.c File Reference

```
#include <stdint.h>
```

Include dependency graph for sprt_partition_metadata_indicator.c:

### Variables

- const struct runtime_metadata_t * p_partition_metadata

### 7.301.1 Variable Documentation

#### 7.301.1.1 p_partition_metadata

```
const struct runtime_metadata_t* p_partition_metadata
```

Definition at line 15 of file sprt_partition_metadata_indicator.c.

## 7.302 secure_fw/partitions/lib/runtime/sprt_partition_metadata_indicator.h File Reference

```
#include <stdint.h>
```

Include dependency graph for sprt_partition_metadata_indicator.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define PART_METADATA() p_partition_metadata

## Variables

- const struct runtime_metadata_t * p_partition_metadata

### 7.302.1 Macro Definition Documentation

#### 7.302.1.1 PART_METADATA

```
#define PART_METADATA( ) p_partition_metadata
```

Definition at line 17 of file sprt_partition_metadata_indicator.h.

### 7.302.2 Variable Documentation

#### 7.302.2.1 p_partition_metadata

```
const struct runtime_metadata_t* p_partition_metadata
```

Definition at line 15 of file sprt_partition_metadata_indicator.c.

## 7.303 secure_fw/partitions/lib/runtime/tfm_sp_log_raw.c File Reference

```
#include <stdarg.h>
#include <stddef.h>
#include <stdint.h>
#include "tfm_hal_defs.h"
#include "tfm_hal_sp_logdev.h"
```

Include dependency graph for tfm_sp_log_raw.c:

### Data Structures

- struct formatted_buffer_t

### Macros

- #define PRINT_BUFF_SIZE 32
- #define NUM_BUFF_SIZE 12

### Functions

- int vprintf (const char *fmt, va_list ap)
- int printf (const char *fmt,...)

### 7.303.1 Macro Definition Documentation

#### 7.303.1.1 NUM_BUFF_SIZE

```
#define NUM_BUFF_SIZE 12
```

Definition at line 15 of file tfm_sp_log_raw.c.

#### 7.303.1.2 PRINT_BUFF_SIZE

```
#define PRINT_BUFF_SIZE 32
```

Definition at line 14 of file tfm_sp_log_raw.c.

### 7.303.2 Function Documentation

#### 7.303.2.1 printf()

```
int printf (
            const char * fmt,
             ...  )
```

Definition at line 165 of file tfm_sp_log_raw.c.
Here is the call graph for this function:

#### 7.303.2.2 vprintf()

```
int vprintf (
            const char * fmt,
            va_list ap )
```

Definition at line 98 of file tfm_sp_log_raw.c.
Here is the caller graph for this function:

## 7.304 secure_fw/partitions/ns_agent_mailbox/ns_agent_mailbox.c File Reference

```
#include "async.h"
#include "config_tfm.h"
#include "ns_agent_mailbox_defs.h"
#include "psa/service.h"
#include "psa_manifest/ns_agent_mailbox.h"
#include "tfm_hal_mailbox.h"
#include "tfm_hal_multi_core.h"
#include "tfm_hal_platform.h"
#include "tfm_multi_core.h"
#include "tfm_rpc.h"
#include "tfm_sp_log.h"
```

Include dependency graph for ns_agent_mailbox.c:

## Enumerations

- enum mailbox_state { START, NS_CORE_RUNNING, LINK_ESTABLISHED }

## Functions

- void ns_agent_mailbox_entry (void)

### 7.304.1 Enumeration Type Documentation

#### 7.304.1.1 mailbox_state

```
enum mailbox_state
```

**Enumerator**

| | |
|---|---|
| START | |
| NS_CORE_RUNNING | |
| LINK_ESTABLISHED | |

Definition at line 23 of file ns_agent_mailbox.c.

### 7.304.2 Function Documentation

#### 7.304.2.1 ns_agent_mailbox_entry()

```
void ns_agent_mailbox_entry (
            void  )
```

Definition at line 94 of file ns_agent_mailbox.c.

## 7.305 secure_fw/partitions/ns_agent_mailbox/tfm_multi_core_client_id.c File Reference

```
#include <stddef.h>
#include "internal_status_code.h"
#include "tfm_plat_defs.h"
#include "tfm_multi_core.h"
```

Include dependency graph for tfm_multi_core_client_id.c:

## Data Structures

- struct client_id_region_t

## Macros

- #define MAX_MAILBOX_NUMBER 2

## Functions

- int32_t tfm_multi_core_register_client_id_range (void *owner, int32_t client_id_base, int32_t client_id_limit)

  *Register a non-secure client ID range.*
- int32_t tfm_multi_core_hal_client_id_translate (void *owner, int32_t client_id_in, int32_t *client_id_out)

  *Translate a non-secure client ID range.*

## 7.305.1 Macro Definition Documentation

### 7.305.1.1 MAX_MAILBOX_NUMBER

```
#define MAX_MAILBOX_NUMBER 2
```

Definition at line 13 of file tfm_multi_core_client_id.c.

## 7.305.2 Function Documentation

### 7.305.2.1 tfm_multi_core_hal_client_id_translate()

```
int32_t tfm_multi_core_hal_client_id_translate (
            void * owner,
            int32_t client_id_in,
            int32_t * client_id_out )
```

Translate a non-secure client ID range.

**Parameters**

| | | |
|---|---|---|
| in | *owner* | Identifier of the non-secure client. |
| in | *client_id_in* | The input client ID. |
| out | *client_id_out* | The translated client ID. Undefined if SPM_ERROR_GENERIC is returned by the function |

**Returns**

SPM_SUCCESS if the translation is successful, SPM_ERROR_GENERIC otherwise.

Definition at line 42 of file tfm_multi_core_client_id.c.

#### 7.305.2.2 tfm_multi_core_register_client_id_range()

```
int32_t tfm_multi_core_register_client_id_range (
            void * owner,
            int32_t client_id_base,
            int32_t client_id_limit )
```

Register a non-secure client ID range.

**Parameters**

| | | |
|---|---|---|
| in | *owner* | Identifier of the non-secure client. |
| in | *client_id_base* | The minimum client ID for this client. |
| in | *client_id_limit* | The maximum client ID for this client. |

**Returns**

SPM_SUCCESS if the registration is successful, SPM_ERROR_GENERIC if owner is null, or no free slots left.

Definition at line 23 of file tfm_multi_core_client_id.c.

## 7.306 secure_fw/partitions/ns_agent_mailbox/tfm_spe_mailbox.c File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include "cmsis.h"
#include "cmsis_compiler.h"
#include "async.h"
#include "config_impl.h"
#include "internal_status_code.h"
#include "psa/error.h"
#include "utilities.h"
#include "tfm_arch.h"
#include "thread.h"
#include "tfm_psa_call_pack.h"
#include "tfm_spe_mailbox.h"
#include "tfm_rpc.h"
#include "tfm_hal_multi_core.h"
#include "tfm_multi_core.h"
#include "ffm/mailbox_agent_api.h"
#include "static_checks.h"
```

Include dependency graph for tfm_spe_mailbox.c:

## Data Structures

- struct vectors

## Macros

- #define MAILBOX_CLEAN_CACHE(addr, size) { __DSB(); }
- #define MAILBOX_INVALIDATE_CACHE(addr, size) do {} while (0)

## Functions

- __STATIC_INLINE void set_spe_queue_empty_status (uint8_t idx)
- __STATIC_INLINE void clear_spe_queue_empty_status (uint8_t idx)
- __STATIC_INLINE bool get_spe_queue_empty_status (uint8_t idx)
- __STATIC_INLINE mailbox_queue_status_t get_nspe_queue_pend_status (const struct mailbox_status_t *ns_status)
- __STATIC_INLINE void set_nspe_queue_replied_status (struct mailbox_status_t *ns_status, mailbox_queue_status_t mask)
- __STATIC_INLINE void clear_nspe_queue_pend_status (struct mailbox_status_t *ns_status, mailbox_queue_status_t mask)
- __STATIC_INLINE int32_t get_spe_mailbox_msg_handle (uint8_t idx, mailbox_msg_handle_t *handle)
- __STATIC_INLINE int32_t get_spe_mailbox_msg_idx (mailbox_msg_handle_t handle, uint8_t *idx)
- __STATIC_INLINE struct mailbox_reply_t * get_nspe_reply_addr (uint8_t idx)
- __STATIC_INLINE int32_t check_mailbox_msg (const struct mailbox_msg_t *msg)
- int32_t tfm_mailbox_handle_msg (void)

  *Handle mailbox message(s) from NSPE.*
- int32_t tfm_mailbox_reply_msg (mailbox_msg_handle_t handle, int32_t reply)

  *Return PSA client call return result to NSPE.*
- int32_t tfm_inter_core_comm_init (void)

  *Initialization of the multi core communication.*
- int32_t tfm_inter_core_comm_enable (void)

  *Enable the multi core communication.*
- int32_t tfm_inter_core_comm_disable (void)

  *Disable the multi core communication.*
- int32_t tfm_inter_core_comm_deinit (void)

  *De-initialization of the multi core communication.*

## 7.306.1 Macro Definition Documentation

### 7.306.1.1 MAILBOX_CLEAN_CACHE

```
#define MAILBOX_CLEAN_CACHE(
            addr,
            size ) { __DSB(); }
```

Definition at line 35 of file tfm_spe_mailbox.c.

#### 7.306.1.2 MAILBOX_INVALIDATE_CACHE

```
#define MAILBOX_INVALIDATE_CACHE(
            addr,
            size ) do {} while (0)
```

Definition at line 36 of file tfm_spe_mailbox.c.

### 7.306.2 Function Documentation

#### 7.306.2.1 check_mailbox_msg()

```
__STATIC_INLINE int32_t check_mailbox_msg (
            const struct mailbox_msg_t * msg )
```

Definition at line 183 of file tfm_spe_mailbox.c.

#### 7.306.2.2 clear_nspe_queue_pend_status()

```
__STATIC_INLINE void clear_nspe_queue_pend_status (
            struct mailbox_status_t * ns_status,
            mailbox_queue_status_t mask )
```

Definition at line 97 of file tfm_spe_mailbox.c.

#### 7.306.2.3 clear_spe_queue_empty_status()

```
__STATIC_INLINE void clear_spe_queue_empty_status (
            uint8_t idx )
```

Definition at line 64 of file tfm_spe_mailbox.c.

#### 7.306.2.4 get_nspe_queue_pend_status()

```
__STATIC_INLINE mailbox_queue_status_t get_nspe_queue_pend_status (
            const struct mailbox_status_t * ns_status )
```

Definition at line 81 of file tfm_spe_mailbox.c.

#### 7.306.2.5 get_nspe_reply_addr()

```
__STATIC_INLINE struct mailbox_reply_t* get_nspe_reply_addr (
            uint8_t idx )
```

Definition at line 141 of file tfm_spe_mailbox.c.

Here is the call graph for this function:

#### 7.306.2.6 get_spe_mailbox_msg_handle()

```
__STATIC_INLINE int32_t get_spe_mailbox_msg_handle (
            uint8_t idx,
            mailbox_msg_handle_t * handle )
```

Definition at line 106 of file tfm_spe_mailbox.c.

#### 7.306.2.7 get_spe_mailbox_msg_idx()

```
__STATIC_INLINE int32_t get_spe_mailbox_msg_idx (
            mailbox_msg_handle_t handle,
            uint8_t * idx )
```

Definition at line 118 of file tfm_spe_mailbox.c.

#### 7.306.2.8 get_spe_queue_empty_status()

```
__STATIC_INLINE bool get_spe_queue_empty_status (
            uint8_t idx )
```

Definition at line 71 of file tfm_spe_mailbox.c.

#### 7.306.2.9 set_nspe_queue_replied_status()

```
__STATIC_INLINE void set_nspe_queue_replied_status (
            struct mailbox_status_t * ns_status,
            mailbox_queue_status_t mask )
```

Definition at line 88 of file tfm_spe_mailbox.c.

#### 7.306.2.10 set_spe_queue_empty_status()

```
__STATIC_INLINE void set_spe_queue_empty_status (
            uint8_t idx )
```

Definition at line 57 of file tfm_spe_mailbox.c.

#### 7.306.2.11 tfm_inter_core_comm_deinit()

```
int32_t tfm_inter_core_comm_deinit (
            void  )
```

De-initialization of the multi core communication.

This may be called to return the multi-core communication from the initialised, but not enabled state to the pre-initialised state.

**Return values**

| | |
|---|---|
| *0* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 620 of file tfm_spe_mailbox.c.

#### 7.306.2.12 tfm_inter_core_comm_disable()

```
int32_t tfm_inter_core_comm_disable (
            void  )
```

Disable the multi core communication.

This may be called to return the multi core communication to the initialised, but not enabled, state.

**Return values**

| | |
|---|---|
| *0* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 615 of file tfm_spe_mailbox.c.

#### 7.306.2.13 tfm_inter_core_comm_enable()

```
int32_t tfm_inter_core_comm_enable (
            void  )
```

Enable the multi core communication.
This is called after tfm_inter_core_init() and may be called again after tfm_inter_core_disable().

**Return values**

| | |
|---|---|
| *0* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 610 of file tfm_spe_mailbox.c.

#### 7.306.2.14 tfm_inter_core_comm_init()

```
int32_t tfm_inter_core_comm_init (
            void  )
```

Initialization of the multi core communication.
This is called once only, after the NS core has booted.

**Return values**

| | |
|---|---|
| *0* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 605 of file tfm_spe_mailbox.c.

#### 7.306.2.15 tfm_mailbox_handle_msg()

```
int32_t tfm_mailbox_handle_msg (
            void  )
```

Handle mailbox message(s) from NSPE.

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Successfully get PSA client call return result. |
| *Other* | return code Operation failed with an error code. |

Definition at line 348 of file tfm_spe_mailbox.c.

#### 7.306.2.16 tfm_mailbox_reply_msg()

```
int32_t tfm_mailbox_reply_msg (
```

```
mailbox_msg_handle_t handle,
int32_t reply )
```

Return PSA client call return result to NSPE.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | The handle to the mailbox message |
| in | *reply* | PSA client call return result to be written to NSPE. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 474 of file tfm_spe_mailbox.c.

## 7.307 secure_fw/partitions/ns_agent_mailbox/tfm_spe_mailbox.h File Reference

```
#include "tfm_mailbox.h"
#include "tfm_hal_mailbox.h"
```

Include dependency graph for tfm_spe_mailbox.h:

This graph shows which files directly or indirectly include this file:

## Functions

- int32_t tfm_mailbox_handle_msg (void)

  *Handle mailbox message(s) from NSPE.*
- int32_t tfm_mailbox_reply_msg (mailbox_msg_handle_t handle, int32_t reply)

  *Return PSA client call return result to NSPE.*

## 7.307.1 Function Documentation

### 7.307.1.1 tfm_mailbox_handle_msg()

```
int32_t tfm_mailbox_handle_msg (
            void  )
```

Handle mailbox message(s) from NSPE.

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Successfully get PSA client call return result. |
| *Other* | return code Operation failed with an error code. |

Definition at line 348 of file tfm_spe_mailbox.c.

### 7.307.1.2 tfm_mailbox_reply_msg()

```
int32_t tfm_mailbox_reply_msg (
            mailbox_msg_handle_t handle,
            int32_t reply )
```

Return PSA client call return result to NSPE.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | The handle to the mailbox message |
| in | *reply* | PSA client call return result to be written to NSPE. |

**Return values**

| | |
|---|---|
| *MAILBOX_SUCCESS* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 474 of file tfm_spe_mailbox.c.

## 7.308 secure_fw/partitions/ns_agent_tz/load_info_ns_agent_tz.c File Reference

```
#include <stdint.h>
#include <stddef.h>
#include "compiler_ext_defs.h"
#include "config_impl.h"
#include "spm.h"
#include "load/partition_defs.h"
#include "load/service_defs.h"
#include "load/asset_defs.h"
#include "region_defs.h"
#include "tfm_s_linker_alignments.h"
```

Include dependency graph for load_info_ns_agent_tz.c:

### Data Structures

- struct partition_tfm_sp_ns_agent_tz_load_info_t

### Macros

- #define TFM_SP_NS_AGENT_NDEPS (0)
- #define TFM_SP_NS_AGENT_NSERVS (0)

### Functions

- void ns_agent_tz_main (void)
- uint8_t ns_agent_tz_stack[TFM_NS_AGENT_TZ_STACK_SIZE_ALIGNED] __aligned (TFM_LINKER_NS↩_AGENT_TZ_STACK_ALIGNMENT)

### Variables

- const struct partition_tfm_sp_ns_agent_tz_load_info_t tfm_sp_ns_agent_tz_load

### 7.308.1 Macro Definition Documentation

#### 7.308.1.1 TFM_SP_NS_AGENT_NDEPS

```
#define TFM_SP_NS_AGENT_NDEPS (0)
```

Definition at line 27 of file load_info_ns_agent_tz.c.

#### 7.308.1.2 TFM_SP_NS_AGENT_NSERVS

```
#define TFM_SP_NS_AGENT_NSERVS (0)
```

Definition at line 28 of file load_info_ns_agent_tz.c.

### 7.308.2 Function Documentation

#### 7.308.2.1 __aligned()

```
uint8_t ns_agent_tz_stack [TFM_NS_AGENT_TZ_STACK_SIZE_ALIGNED] __aligned (
            TFM_LINKER_NS_AGENT_TZ_STACK_ALIGNMENT  )
```

#### 7.308.2.2 ns_agent_tz_main()

```
void ns_agent_tz_main (
            void  )
```

### 7.308.3 Variable Documentation

#### 7.308.3.1 tfm_sp_ns_agent_tz_load

```
const struct partition_tfm_sp_ns_agent_tz_load_info_t tfm_sp_ns_agent_tz_load
```

Definition at line 57 of file load_info_ns_agent_tz.c.

## 7.309 secure_fw/partitions/ns_agent_tz/ns_agent_tz.c File Reference

```
#include <stdint.h>
#include "compiler_ext_defs.h"
#include "security_defs.h"
#include "tfm_arch.h"
#include "tfm_hal_platform.h"
```

Include dependency graph for ns_agent_tz.c:

### Functions

- __naked void ns_agent_tz_main (uint32_t c_entry)

### 7.309.1 Function Documentation

#### 7.309.1.1 ns_agent_tz_main()

```
__naked void ns_agent_tz_main (
            uint32_t c_entry )
```

Definition at line 17 of file ns_agent_tz.c.

## 7.310 secure_fw/partitions/ns_agent_tz/ns_agent_tz_v80m.c File Reference

```
#include <stdint.h>
#include "compiler_ext_defs.h"
#include "security_defs.h"
#include "tfm_arch.h"
#include "tfm_hal_platform.h"
```

Include dependency graph for ns_agent_tz_v80m.c:

### Functions

- __naked void ns_agent_tz_main (uint32_t c_entry)

### 7.310.1 Function Documentation

#### 7.310.1.1 ns_agent_tz_main()

```
__naked void ns_agent_tz_main (
            uint32_t c_entry )
```

Definition at line 17 of file ns_agent_tz_v80m.c.

## 7.311 secure_fw/partitions/ns_agent_tz/psa_api_veneers.c File Reference

```
#include <stdint.h>
#include "config_impl.h"
```

```
#include "security_defs.h"
#include "tfm_arch.h"
#include "tfm_psa_call_pack.h"
#include "psa/client.h"
```

Include dependency graph for psa_api_veneers.c:

## Functions

- __tz_c_veneer uint32_t tfm_psa_framework_version_veneer (void)

  *Retrieve the version of the PSA Framework API that is implemented.*
- __tz_c_veneer uint32_t tfm_psa_version_veneer (uint32_t sid)

  *Return version of secure function provided by secure binary.*
- __tz_c_veneer psa_status_t tfm_psa_call_veneer (psa_handle_t handle, uint32_t ctrl_param, const psa_invec *in_vec, psa_outvec *out_vec)

  *Call a secure function referenced by a connection handle.*
- __tz_c_veneer psa_handle_t tfm_psa_connect_veneer (uint32_t sid, uint32_t version)

  *Connect to secure function.*
- __tz_c_veneer void tfm_psa_close_veneer (psa_handle_t handle)

  *Close connection to secure function referenced by a connection handle.*

## 7.311.1 Function Documentation

### 7.311.1.1 tfm_psa_call_veneer()

```
__tz_c_veneer psa_status_t tfm_psa_call_veneer (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * in_vec,
            psa_outvec * out_vec )
```

Call a secure function referenced by a connection handle.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Handle to connection. |
| in | *ctrl_param* | Parameters combined in uint32_t, includes request type, in_num and out_num. |
| in | *in_vec* | Array of input psa_invec structures. |
| in,out | *out_vec* | Array of output psa_outvec structures. |

**Returns**

Returns psa_status_t status code.

Definition at line 57 of file psa_api_veneers.c.

#### 7.311.1.2 tfm_psa_close_veneer()

```
__tz_c_veneer void tfm_psa_close_veneer (
            psa_handle_t handle )
```

Close connection to secure function referenced by a connection handle.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Handle to connection |

Definition at line 113 of file psa_api_veneers.c.

#### 7.311.1.3 tfm_psa_connect_veneer()

```
__tz_c_veneer psa_handle_t tfm_psa_connect_veneer (
            uint32_t sid,
            uint32_t version )
```

Connect to secure function.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of secure service. |
| in | *version* | Version of SF requested by client. |

**Returns**

Returns handle to connection.

Definition at line 104 of file psa_api_veneers.c.

#### 7.311.1.4 tfm_psa_framework_version_veneer()

```
__tz_c_veneer uint32_t tfm_psa_framework_version_veneer (
            void )
```

Retrieve the version of the PSA Framework API that is implemented.

**Returns**

The version of the PSA Framework.

Definition at line 29 of file psa_api_veneers.c.
Here is the call graph for this function:

### 7.311.1.5 tfm_psa_version_veneer()

```
__tz_c_veneer uint32_t tfm_psa_version_veneer (
            uint32_t sid )
```

Return version of secure function provided by secure binary.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of secure service. |

**Returns**

Version number of secure function.

Definition at line 43 of file psa_api_veneers.c.
Here is the call graph for this function:

## 7.312 secure_fw/partitions/ns_agent_tz/psa_api_veneers_v80m.c File Reference

```
#include <stdint.h>
#include "cmsis_compiler.h"
#include "compiler_ext_defs.h"
```

```
#include "config_impl.h"
#include "fih.h"
#include "security_defs.h"
#include "svc_num.h"
#include "tfm_psa_call_pack.h"
#include "utilities.h"
#include "psa/client.h"
#include "psa/service.h"
#include "tfm_arch.h"
```

Include dependency graph for psa_api_veneers_v80m.c:

## Functions

- void clear_caller_context (void)
- __tz_naked_veneer uint32_t tfm_psa_framework_version_veneer (void)

  *Retrieve the version of the PSA Framework API that is implemented.*
- __tz_naked_veneer uint32_t tfm_psa_version_veneer (uint32_t sid)

  *Return version of secure function provided by secure binary.*
- __tz_naked_veneer psa_status_t tfm_psa_call_veneer (psa_handle_t handle, uint32_t ctrl_param, const psa_invec *in_vec, psa_outvec *out_vec)

  *Call a secure function referenced by a connection handle.*
- __tz_naked_veneer psa_handle_t tfm_psa_connect_veneer (uint32_t sid, uint32_t version)

  *Connect to secure function.*
- __tz_naked_veneer void tfm_psa_close_veneer (psa_handle_t handle)

  *Close connection to secure function referenced by a connection handle.*

## Variables

- const __used int32_t ret_err = (int32_t)PSA_ERROR_NOT_SUPPORTED

## 7.312.1 Function Documentation

### 7.312.1.1 clear_caller_context()

```
void clear_caller_context (
            void  )
```

Definition at line 55 of file psa_api_veneers_v80m.c.

### 7.312.1.2 tfm_psa_call_veneer()

```
__tz_naked_veneer psa_status_t tfm_psa_call_veneer (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * in_vec,
            psa_outvec * out_vec )
```

Call a secure function referenced by a connection handle.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Handle to connection. |
| in | *ctrl_param* | Parameters combined in uint32_t, includes request type, in_num and out_num. |
| in | *in_vec* | Array of input psa_invec structures. |
| in,out | *out_vec* | Array of output psa_outvec structures. |

**Returns**

Returns psa_status_t status code.

Definition at line 159 of file psa_api_veneers_v80m.c.

#### 7.312.1.3 tfm_psa_close_veneer()

```
__tz_naked_veneer void tfm_psa_close_veneer (
            psa_handle_t handle )
```

Close connection to secure function referenced by a connection handle.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Handle to connection |

Definition at line 309 of file psa_api_veneers_v80m.c.

#### 7.312.1.4 tfm_psa_connect_veneer()

```
__tz_naked_veneer psa_handle_t tfm_psa_connect_veneer (
            uint32_t sid,
            uint32_t version )
```

Connect to secure function.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of secure service. |
| in | *version* | Version of SF requested by client. |

**Returns**

Returns handle to connection.

Definition at line 290 of file psa_api_veneers_v80m.c.
Here is the caller graph for this function:

#### 7.312.1.5 tfm_psa_framework_version_veneer()

```
__tz_naked_veneer uint32_t tfm_psa_framework_version_veneer (
            void  )
```

Retrieve the version of the PSA Framework API that is implemented.

**Returns**

The version of the PSA Framework.

Definition at line 88 of file psa_api_veneers_v80m.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.312.1.6 tfm_psa_version_veneer()

```
__tz_naked_veneer uint32_t tfm_psa_version_veneer (
            uint32_t sid )
```

Return version of secure function provided by secure binary.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of secure service. |

**Returns**

Version number of secure function.

Definition at line 123 of file psa_api_veneers_v80m.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.312.2 Variable Documentation

#### 7.312.2.1 ret_err

```
const __used int32_t ret_err = (int32_t)PSA_ERROR_NOT_SUPPORTED
```

Definition at line 284 of file psa_api_veneers_v80m.c.

## 7.313 secure_fw/partitions/platform/dir_platform.dox File Reference

## 7.314 secure_fw/partitions/platform/platform_sp.c File Reference

```
#include "config_tfm.h"
#include "platform_sp.h"
#include "tfm_platform_system.h"
#include "load/partition_defs.h"
#include "psa_manifest/pid.h"
#include "tfm_plat_nv_counters.h"
#include "psa/client.h"
#include "psa/service.h"
#include "region_defs.h"
#include "psa_manifest/tfm_platform.h"
#include "static_checks.h"
```

Include dependency graph for platform_sp.c:

## Macros

- #define NV_COUNTER_ID_SIZE sizeof(enum tfm_nv_counter_t)
- #define NV_COUNTER_SIZE 4

## Typedefs

- typedef enum tfm_platform_err_t(∗ plat_func_t) (const psa_msg_t ∗msg)

## Functions

- enum tfm_platform_err_t platform_sp_system_reset (void)

  *Resets the system.*
- psa_status_t tfm_platform_service_sfn (const psa_msg_t ∗msg)
- psa_status_t platform_sp_init (void)

  *Initializes the secure partition.*

### 7.314.1 Macro Definition Documentation

#### 7.314.1.1 NV_COUNTER_ID_SIZE

```
#define NV_COUNTER_ID_SIZE sizeof(enum tfm_nv_counter_t)
```

Definition at line 28 of file platform_sp.c.

#### 7.314.1.2 NV_COUNTER_SIZE

```
#define NV_COUNTER_SIZE 4
```

Definition at line 29 of file platform_sp.c.

### 7.314.2 Typedef Documentation

#### 7.314.2.1 plat_func_t

```
typedef enum tfm_platform_err_t(* plat_func_t) (const psa_msg_t *msg)
```

Definition at line 32 of file platform_sp.c.

### 7.314.3 Function Documentation

#### 7.314.3.1 platform_sp_init()

```
psa_status_t platform_sp_init (
            void  )
```

Initializes the secure partition.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 270 of file platform_sp.c.
Here is the call graph for this function:

#### 7.314.3.2 platform_sp_system_reset()

```
enum tfm_platform_err_t platform_sp_system_reset (
            void  )
```

Resets the system.

**Returns**

Returns values as specified by the tfm_platform_err_t

Definition at line 34 of file platform_sp.c.
Here is the call graph for this function:

#### 7.314.3.3 tfm_platform_service_sfn()

```
psa_status_t tfm_platform_service_sfn (
            const psa_msg_t * msg )
```

Definition at line 248 of file platform_sp.c.

## 7.315 secure_fw/partitions/platform/platform_sp.h File Reference

```
#include "tfm_platform_api.h"
```

Include dependency graph for platform_sp.h:

This graph shows which files directly or indirectly include this file:

## Functions

- psa_status_t platform_sp_init (void)

  *Initializes the secure partition.*

- enum tfm_platform_err_t platform_sp_system_reset (void)

  *Resets the system.*

### 7.315.1 Function Documentation

#### 7.315.1.1 platform_sp_init()

```
psa_status_t platform_sp_init (
            void  )
```

Initializes the secure partition.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 270 of file platform_sp.c.
Here is the call graph for this function:

#### 7.315.1.2 platform_sp_system_reset()

```
enum tfm_platform_err_t platform_sp_system_reset (
            void  )
```

Resets the system.

**Returns**

Returns values as specified by the tfm_platform_err_t

Definition at line 34 of file platform_sp.c.
Here is the call graph for this function:

## 7.316 secure_fw/partitions/protected_storage/config_ps_check.h File Reference

```
#include "config_tfm.h"
```

Include dependency graph for config_ps_check.h:

This graph shows which files directly or indirectly include this file:

## 7.317 secure_fw/partitions/protected_storage/crypto/ps_crypto_interface.c File Reference

```
#include "ps_crypto_interface.h"
#include <stdbool.h>
#include <string.h>
#include "config_tfm.h"
#include "tfm_crypto_defs.h"
#include "psa/crypto.h"
```

Include dependency graph for ps_crypto_interface.c:

## Macros

- #define LABEL_LEN (sizeof(int32_t) + sizeof(psa_storage_uid_t))
- #define ps_crypto_setkey tfm_platform_ps_set_key

## Typedefs

- typedef char PS_ERROR_NOT_AEAD_ALG[(PSA_ALG_IS_AEAD(PS_CRYPTO_ALG)) ? 1 :-1]

## Functions

- psa_status_t ps_crypto_init (void)

  *Initializes the crypto engine.*
- uint32_t ps_crypto_to_blocks (size_t in_len)

  *Convert lengths to block count.*
- void ps_crypto_set_iv (const union ps_crypto_t ∗crypto)

  *Provides current IV value to crypto layer.*
- psa_status_t ps_crypto_get_iv (union ps_crypto_t ∗crypto)

  *Gets a new IV value into the crypto union.*
- psa_status_t ps_crypto_encrypt_and_tag (union ps_crypto_t ∗crypto, const uint8_t ∗add, size_t add_len, const uint8_t ∗in, size_t in_len, uint8_t ∗out, size_t out_size, size_t ∗out_len)

  *Encrypts and tags the given plaintext data.*
- psa_status_t ps_crypto_auth_and_decrypt (const union ps_crypto_t ∗crypto, const uint8_t ∗add, size_t add_len, uint8_t ∗in, size_t in_len, uint8_t ∗out, size_t out_size, size_t ∗out_len)

  *Decrypts and authenticates the given encrypted data.*
- psa_status_t ps_crypto_generate_auth_tag (union ps_crypto_t ∗crypto, const uint8_t ∗add, uint32_t add_len)

  *Generates authentication tag for given data.*
- psa_status_t ps_crypto_authenticate (const union ps_crypto_t ∗crypto, const uint8_t ∗add, uint32_t add_len)

  *Authenticate given data against the tag.*

## 7.317.1 Macro Definition Documentation

### 7.317.1.1 LABEL_LEN

```
#define LABEL_LEN (sizeof(int32_t) + sizeof(psa_storage_uid_t))
```

Definition at line 21 of file ps_crypto_interface.c.

#### 7.317.1.2 ps_crypto_setkey

```
#define ps_crypto_setkey tfm_platform_ps_set_key
```

Definition at line 116 of file ps_crypto_interface.c.

### 7.317.2 Typedef Documentation

#### 7.317.2.1 PS_ERROR_NOT_AEAD_ALG

```
typedef char PS_ERROR_NOT_AEAD_ALG[(PSA_ALG_IS_AEAD(PS_CRYPTO_ALG)) ? 1 :-1]
```

Definition at line 33 of file ps_crypto_interface.c.

### 7.317.3 Function Documentation

#### 7.317.3.1 ps_crypto_auth_and_decrypt()

```
psa_status_t ps_crypto_auth_and_decrypt (
            const union ps_crypto_t * crypto,
            const uint8_t * add,
            size_t add_len,
            uint8_t * in,
            size_t in_len,
            uint8_t * out,
            size_t out_size,
            size_t * out_len )
```

Decrypts and authenticates the given encrypted data.

**Parameters**

| | | |
|---|---|---|
| in | *crypto* | Pointer to the crypto union |
| in | *add* | Pointer to the associated data |
| in | *add_len* | Length of the associated data |
| in | *in* | Pointer to the input data |
| in | *in_len* | Length of the input data |
| out | *out* | Pointer to the output buffer for decrypted data |
| in | *out_size* | Size of the output buffer |
| out | *out_len* | On success, the length of the output data |

**Returns**

Returns values as described in psa_status_t

Definition at line 246 of file ps_crypto_interface.c.

#### 7.317.3.2 ps_crypto_authenticate()

```
psa_status_t ps_crypto_authenticate (
            const union ps_crypto_t * crypto,
            const uint8_t * add,
            uint32_t add_len )
```

Authenticate given data against the tag.

**Parameters**

| | | |
|---|---|---|
| in | *crypto* | Pointer to the crypto union |
| in | *add* | Pointer to the data to authenticate |
| in | *add_len* | Length of the data to authenticate |

**Returns**

Returns values as described in psa_status_t

Definition at line 324 of file ps_crypto_interface.c.

#### 7.317.3.3 ps_crypto_encrypt_and_tag()

```
psa_status_t ps_crypto_encrypt_and_tag (
            union ps_crypto_t * crypto,
            const uint8_t * add,
            size_t add_len,
            const uint8_t * in,
            size_t in_len,
            uint8_t * out,
            size_t out_size,
            size_t * out_len )
```

Encrypts and tags the given plaintext data.

**Parameters**

| | | |
|---|---|---|
| in,out | *crypto* | Pointer to the crypto union |
| in | *add* | Pointer to the associated data |
| in | *add_len* | Length of the associated data |
| in | *in* | Pointer to the input data |
| in | *in_len* | Length of the input data |
| out | *out* | Pointer to the output buffer for encrypted data |
| in | *out_size* | Size of the output buffer |
| out | *out_len* | On success, the length of the output data |

**Returns**

Returns values as described in psa_status_t

Definition at line 203 of file ps_crypto_interface.c.

#### 7.317.3.4 ps_crypto_generate_auth_tag()

```
psa_status_t ps_crypto_generate_auth_tag (
            union ps_crypto_t * crypto,
            const uint8_t * add,
            uint32_t add_len )
```

Generates authentication tag for given data.

**Parameters**

| | | |
|---|---|---|
| in,out | *crypto* | Pointer to the crypto union |
| in | *add* | Pointer to the data to authenticate |
| in | *add_len* | Length of the data to authenticate |

**Returns**

Returns values as described in psa_status_t

Definition at line 289 of file ps_crypto_interface.c.

#### 7.317.3.5 ps_crypto_get_iv()

```
psa_status_t ps_crypto_get_iv (
            union ps_crypto_t * crypto )
```

Gets a new IV value into the crypto union.

**Parameters**

| | | |
|---|---|---|
| out | *crypto* | Pointer to the crypto union |

**Returns**

Returns values as described in psa_status_t

Definition at line 149 of file ps_crypto_interface.c.
Here is the call graph for this function:

#### 7.317.3.6 ps_crypto_init()

```
psa_status_t ps_crypto_init (
            void )
```

Initializes the crypto engine.

**Returns**

Returns values as described in psa_status_t

Definition at line 119 of file ps_crypto_interface.c.

#### 7.317.3.7 ps_crypto_set_iv()

```
void ps_crypto_set_iv (
            const union ps_crypto_t * crypto )
```

Provides current IV value to crypto layer.

**Parameters**

| | | |
|---|---|---|
| in | *crypto* | Pointer to the crypto union |

Definition at line 144 of file ps_crypto_interface.c.

Here is the call graph for this function:

#### 7.317.3.8 ps_crypto_to_blocks()

```
uint32_t ps_crypto_to_blocks (
            size_t in_len )
```

Convert lengths to block count.

**Parameters**

| | | |
|---|---|---|
| in | *in_len* | Length of the input data |

**Returns**

Returns number of blocks encrypted/decrypted

Definition at line 137 of file ps_crypto_interface.c.
Here is the caller graph for this function:

## 7.318 secure_fw/partitions/protected_storage/crypto/ps_crypto_interface.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "psa/protected_storage.h"
#include "tfm_plat_ps_crypto.h"
```

Include dependency graph for ps_crypto_interface.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- union ps_crypto_t

## Functions

- psa_status_t ps_crypto_init (void)

  *Initializes the crypto engine.*
- uint32_t ps_crypto_to_blocks (size_t in_len)

  *Convert lengths to block count.*
- psa_status_t ps_crypto_encrypt_and_tag (union ps_crypto_t ∗crypto, const uint8_t ∗add, size_t add_len, const uint8_t ∗in, size_t in_len, uint8_t ∗out, size_t out_size, size_t ∗out_len)

  *Encrypts and tags the given plaintext data.*
- psa_status_t ps_crypto_auth_and_decrypt (const union ps_crypto_t ∗crypto, const uint8_t ∗add, size_↩ t add_len, uint8_t ∗in, size_t in_len, uint8_t ∗out, size_t out_size, size_t ∗out_len)

  *Decrypts and authenticates the given encrypted data.*
- psa_status_t ps_crypto_generate_auth_tag (union ps_crypto_t ∗crypto, const uint8_t ∗add, uint32_t add_↩ len)

  *Generates authentication tag for given data.*
- psa_status_t ps_crypto_authenticate (const union ps_crypto_t ∗crypto, const uint8_t ∗add, uint32_t add_len)

  *Authenticate given data against the tag.*
- void ps_crypto_set_iv (const union ps_crypto_t ∗crypto)

*Provides current IV value to crypto layer.*

- psa_status_t ps_crypto_get_iv (union ps_crypto_t *crypto)

  *Gets a new IV value into the crypto union.*

## 7.318.1 Function Documentation

### 7.318.1.1 ps_crypto_auth_and_decrypt()

```
psa_status_t ps_crypto_auth_and_decrypt (
            const union ps_crypto_t * crypto,
            const uint8_t * add,
            size_t add_len,
            uint8_t * in,
            size_t in_len,
            uint8_t * out,
            size_t out_size,
            size_t * out_len )
```

Decrypts and authenticates the given encrypted data.

**Parameters**

| | | |
|---|---|---|
| in | *crypto* | Pointer to the crypto union |
| in | *add* | Pointer to the associated data |
| in | *add_len* | Length of the associated data |
| in | *in* | Pointer to the input data |
| in | *in_len* | Length of the input data |
| out | *out* | Pointer to the output buffer for decrypted data |
| in | *out_size* | Size of the output buffer |
| out | *out_len* | On success, the length of the output data |

**Returns**

Returns values as described in psa_status_t

Definition at line 246 of file ps_crypto_interface.c.

### 7.318.1.2 ps_crypto_authenticate()

```
psa_status_t ps_crypto_authenticate (
            const union ps_crypto_t * crypto,
            const uint8_t * add,
            uint32_t add_len )
```

Authenticate given data against the tag.

**Parameters**

| | | |
|---|---|---|
| in | *crypto* | Pointer to the crypto union |
| in | *add* | Pointer to the data to authenticate |
| in | *add_len* | Length of the data to authenticate |

**Returns**

Returns values as described in psa_status_t

Definition at line 324 of file ps_crypto_interface.c.

#### 7.318.1.3 ps_crypto_encrypt_and_tag()

```
psa_status_t ps_crypto_encrypt_and_tag (
            union ps_crypto_t * crypto,
            const uint8_t * add,
            size_t add_len,
            const uint8_t * in,
            size_t in_len,
            uint8_t * out,
            size_t out_size,
            size_t * out_len )
```

Encrypts and tags the given plaintext data.

**Parameters**

| | | |
|---|---|---|
| in,out | *crypto* | Pointer to the crypto union |
| in | *add* | Pointer to the associated data |
| in | *add_len* | Length of the associated data |
| in | *in* | Pointer to the input data |
| in | *in_len* | Length of the input data |
| out | *out* | Pointer to the output buffer for encrypted data |
| in | *out_size* | Size of the output buffer |
| out | *out_len* | On success, the length of the output data |

**Returns**

Returns values as described in psa_status_t

Definition at line 203 of file ps_crypto_interface.c.

#### 7.318.1.4 ps_crypto_generate_auth_tag()

```
psa_status_t ps_crypto_generate_auth_tag (
            union ps_crypto_t * crypto,
            const uint8_t * add,
            uint32_t add_len )
```

Generates authentication tag for given data.

**Parameters**

| | | |
|---|---|---|
| in,out | *crypto* | Pointer to the crypto union |
| in | *add* | Pointer to the data to authenticate |
| in | *add_len* | Length of the data to authenticate |

**Returns**

Returns values as described in psa_status_t

Definition at line 289 of file ps_crypto_interface.c.

#### 7.318.1.5 ps_crypto_get_iv()

```
psa_status_t ps_crypto_get_iv (
            union ps_crypto_t * crypto )
```

Gets a new IV value into the crypto union.

**Parameters**

| | | |
|---|---|---|
| out | *crypto* | Pointer to the crypto union |

**Returns**

Returns values as described in psa_status_t

Definition at line 149 of file ps_crypto_interface.c.
Here is the call graph for this function:

#### 7.318.1.6 ps_crypto_init()

```
psa_status_t ps_crypto_init (
            void )
```

Initializes the crypto engine.

**Returns**

Returns values as described in psa_status_t

Definition at line 119 of file ps_crypto_interface.c.

#### 7.318.1.7 ps_crypto_set_iv()

```
void ps_crypto_set_iv (
            const union ps_crypto_t * crypto )
```

Provides current IV value to crypto layer.

**Parameters**

| | | |
|---|---|---|
| in | *crypto* | Pointer to the crypto union |

Definition at line 144 of file ps_crypto_interface.c.

Here is the call graph for this function:

#### 7.318.1.8 ps_crypto_to_blocks()

```
uint32_t ps_crypto_to_blocks (
            size_t in_len )
```

Convert lengths to block count.

**Parameters**

| | | |
|---|---|---|
| in | *in_len* | Length of the input data |

**Returns**

Returns number of blocks encrypted/decrypted

Definition at line 137 of file ps_crypto_interface.c.
Here is the caller graph for this function:

## 7.319 secure_fw/partitions/protected_storage/dir_protected_storage.dox File Reference

## 7.320 secure_fw/partitions/protected_storage/nv_counters/ps_nv_counters.c File Reference

```
#include "ps_nv_counters.h"
#include "tfm_platform_api.h"
```

Include dependency graph for ps_nv_counters.c:

## Functions

- psa_status_t ps_read_nv_counter (enum tfm_nv_counter_t counter_id, uint32_t *val)

  *Reads the given non-volatile (NV) counter.*
- psa_status_t ps_increment_nv_counter (enum tfm_nv_counter_t counter_id)

  *Increments the given non-volatile (NV) counter.*

## 7.320.1 Function Documentation

### 7.320.1.1 ps_increment_nv_counter()

```
psa_status_t ps_increment_nv_counter (
            enum tfm_nv_counter_t counter_id )
```

Increments the given non-volatile (NV) counter.

**Parameters**

| | | |
|---|---|---|
| in | *counter↩ _id* | NV counter ID. |

**Returns**

If the counter is incremented correctly, it returns PSA_SUCCESS. Otherwise, PSA_ERROR_GENERIC_E↩ RROR.

Definition at line 25 of file ps_nv_counters.c.
Here is the call graph for this function:

#### 7.320.1.2 ps_read_nv_counter()

```
psa_status_t ps_read_nv_counter (
            enum tfm_nv_counter_t counter_id,
            uint32_t * val )
```

Reads the given non-volatile (NV) counter.

**Parameters**

| | | |
|---|---|---|
| in | *counter↩ _id* | NV counter ID. |
| out | *val* | Pointer to store the current NV counter value. |

**Returns**

PSA_SUCCESS if the value is read correctly, otherwise PSA_ERROR_GENERIC_ERROR

Definition at line 11 of file ps_nv_counters.c.
Here is the call graph for this function:

## 7.321 secure_fw/partitions/protected_storage/nv_counters/ps_nv_↩ counters.h File Reference

```
#include <stdint.h>
#include "psa/protected_storage.h"
#include "tfm_plat_nv_counters.h"
```

Include dependency graph for ps_nv_counters.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define TFM_PS_NV_COUNTER_1 PLAT_NV_COUNTER_PS_0
- #define TFM_PS_NV_COUNTER_2 PLAT_NV_COUNTER_PS_1
- #define TFM_PS_NV_COUNTER_3 PLAT_NV_COUNTER_PS_2
- #define PS_NV_COUNTER_SIZE 4 /∗ In bytes ∗/

## Functions

- psa_status_t ps_read_nv_counter (enum tfm_nv_counter_t counter_id, uint32_t ∗val)

  *Reads the given non-volatile (NV) counter.*
- psa_status_t ps_increment_nv_counter (enum tfm_nv_counter_t counter_id)

*Increments the given non-volatile (NV) counter.*

## 7.321.1 Macro Definition Documentation

### 7.321.1.1 PS_NV_COUNTER_SIZE

```
#define PS_NV_COUNTER_SIZE 4 /* In bytes */
```

Definition at line 27 of file ps_nv_counters.h.

### 7.321.1.2 TFM_PS_NV_COUNTER_1

```
#define TFM_PS_NV_COUNTER_1 PLAT_NV_COUNTER_PS_0
```

Definition at line 23 of file ps_nv_counters.h.

### 7.321.1.3 TFM_PS_NV_COUNTER_2

```
#define TFM_PS_NV_COUNTER_2 PLAT_NV_COUNTER_PS_1
```

Definition at line 24 of file ps_nv_counters.h.

### 7.321.1.4 TFM_PS_NV_COUNTER_3

```
#define TFM_PS_NV_COUNTER_3 PLAT_NV_COUNTER_PS_2
```

Definition at line 25 of file ps_nv_counters.h.

## 7.321.2 Function Documentation

### 7.321.2.1 ps_increment_nv_counter()

```
psa_status_t ps_increment_nv_counter (
            enum tfm_nv_counter_t counter_id )
```

Increments the given non-volatile (NV) counter.

**Parameters**

| | | |
|---|---|---|
| in | *counter↩ _id* | NV counter ID. |

**Returns**

If the counter is incremented correctly, it returns PSA_SUCCESS. Otherwise, PSA_ERROR_GENERIC_E↩ RROR.

Definition at line 25 of file ps_nv_counters.c.
Here is the call graph for this function:

#### 7.321.2.2 ps_read_nv_counter()

```
psa_status_t ps_read_nv_counter (
            enum tfm_nv_counter_t counter_id,
            uint32_t * val )
```

Reads the given non-volatile (NV) counter.

**Parameters**

| | | |
|---|---|---|
| in | *counter_id* | NV counter ID. |
| out | *val* | Pointer to store the current NV counter value. |

**Returns**

PSA_SUCCESS if the value is read correctly, otherwise PSA_ERROR_GENERIC_ERROR

Definition at line 11 of file ps_nv_counters.c.
Here is the call graph for this function:

## 7.322 secure_fw/partitions/protected_storage/ps_encrypted_object.c File Reference

```
#include "ps_encrypted_object.h"
#include <stddef.h>
#include <string.h>
#include "cmsis_compiler.h"
#include "crypto/ps_crypto_interface.h"
#include "psa/internal_trusted_storage.h"
#include "ps_object_defs.h"
#include "ps_utils.h"
```

Include dependency graph for ps_encrypted_object.c:

## Macros

- #define STORED_HEADER_DATA_SIZE
- #define PS_ENCRYPT_SIZE(plaintext_size) ((plaintext_size) + offsetof(struct ps_object_t, data) - offsetof(struct ps_object_t, header.info))
- #define PS_OBJECT_START_POSITION 0
- #define PS_MAX_ENCRYPTED_OBJ_SIZE PS_ENCRYPT_SIZE(PS_MAX_OBJECT_DATA_SIZE)
- #define PS_CRYPTO_BUF_LEN (PS_MAX_ENCRYPTED_OBJ_SIZE + PS_TAG_LEN_BYTES)

## Functions

- psa_status_t ps_encrypted_object_read (uint32_t fid, struct ps_object_t *obj, uint32_t *p_blocks)

  *Reads object referenced by the object File ID.*
- uint32_t ps_encrypted_object_blocks (uint32_t size)

  *Determines the number of encryption blocks that will be used to write an object of the specified size.*
- psa_status_t ps_encrypted_object_write (uint32_t fid, struct ps_object_t *obj)

  *Creates and writes a new encrypted object based on the given ps_object_t structure data.*

## Variables

- __PACKED_STRUCT auth_data_t

## 7.322.1 Macro Definition Documentation

### 7.322.1.1 PS_CRYPTO_BUF_LEN

```
#define PS_CRYPTO_BUF_LEN (PS_MAX_ENCRYPTED_OBJ_SIZE + PS_TAG_LEN_BYTES)
```

Definition at line 38 of file ps_encrypted_object.c.

### 7.322.1.2 PS_ENCRYPT_SIZE

```
#define PS_ENCRYPT_SIZE(
            plaintext_size ) ((plaintext_size) + offsetof(struct ps_object_t, data) - offsetof(struct
ps_object_t, header.info))
```

Definition at line 26 of file ps_encrypted_object.c.

### 7.322.1.3 PS_MAX_ENCRYPTED_OBJ_SIZE

```
#define PS_MAX_ENCRYPTED_OBJ_SIZE PS_ENCRYPT_SIZE(PS_MAX_OBJECT_DATA_SIZE)
```

Definition at line 33 of file ps_encrypted_object.c.

### 7.322.1.4 PS_OBJECT_START_POSITION

```
#define PS_OBJECT_START_POSITION 0
```

Definition at line 29 of file ps_encrypted_object.c.

### 7.322.1.5 STORED_HEADER_DATA_SIZE

```
#define STORED_HEADER_DATA_SIZE
```

**Value:**

```
(offsetof(struct ps_object_t, header.info) \
- offsetof(struct ps_object_t, header.crypto.ref.iv))
```

Definition at line 22 of file ps_encrypted_object.c.

## 7.322.2 Function Documentation

### 7.322.2.1 ps_encrypted_object_blocks()

```
uint32_t ps_encrypted_object_blocks (
            uint32_t size )
```

Determines the number of encryption blocks that will be used to write an object of the specified size.

**Parameters**

| | | |
|---|---|---|
| in | *size* | Size in bytes of the object to be written. |

**Returns**

Returns number of blocks

Definition at line 186 of file ps_encrypted_object.c.
Here is the call graph for this function:

### 7.322.2.2 ps_encrypted_object_read()

```
psa_status_t ps_encrypted_object_read (
            uint32_t fid,
            struct ps_object_t * obj,
            uint32_t * p_blocks )
```

Reads object referenced by the object File ID.

**Parameters**

| | | |
|---|---|---|
| in | *fid* | File ID |
| out | *obj* | Pointer to the object structure to fill in |
| out | *p_blocks* | Pointer to a counter of decryption blocks used. |

**Returns**

Returns error code specified in psa_status_t

Definition at line 150 of file ps_encrypted_object.c.
Here is the call graph for this function:

#### 7.322.2.3 ps_encrypted_object_write()

```
psa_status_t ps_encrypted_object_write (
            uint32_t fid,
            struct ps_object_t * obj )
```

Creates and writes a new encrypted object based on the given ps_object_t structure data.

**Parameters**

| | | |
|---|---|---|
| in | *fid* | File ID |
| in,out | *obj* | Pointer to the object structure to write. |

Note: The function will use obj to store the encrypted data before write it into the flash to reduce the memory requirements and the number of internal copies. So, this object will contain the encrypted object stored in the flash.

**Returns**

Returns error code specified in psa_status_t

Definition at line 193 of file ps_encrypted_object.c.

### 7.322.3 Variable Documentation

#### 7.322.3.1 auth_data_t

```
__PACKED_STRUCT auth_data_t
```

**Initial value:**

```
{
    uint32_t fid
```

Definition at line 40 of file ps_encrypted_object.c.

## 7.323 secure_fw/partitions/protected_storage/ps_encrypted_object.h File Reference

```
#include <stdint.h>
#include "ps_object_defs.h"
```

```
#include "psa/protected_storage.h"
```

Include dependency graph for ps_encrypted_object.h:

This graph shows which files directly or indirectly include this file:

## Functions

- psa_status_t ps_encrypted_object_read (uint32_t fid, struct ps_object_t ∗obj, uint32_t ∗p_blocks)

  *Reads object referenced by the object File ID.*
- psa_status_t ps_encrypted_object_write (uint32_t fid, struct ps_object_t ∗obj)

  *Creates and writes a new encrypted object based on the given ps_object_t structure data.*
- uint32_t ps_encrypted_object_blocks (uint32_t size)

  *Determines the number of encryption blocks that will be used to write an object of the specified size.*

### 7.323.1 Function Documentation

#### 7.323.1.1 ps_encrypted_object_blocks()

```
uint32_t ps_encrypted_object_blocks (
            uint32_t size )
```

Determines the number of encryption blocks that will be used to write an object of the specified size.

**Parameters**

| | | |
|---|---|---|
| in | *size* | Size in bytes of the object to be written. |

**Returns**

Returns number of blocks

Definition at line 186 of file ps_encrypted_object.c.
Here is the call graph for this function:

#### 7.323.1.2 ps_encrypted_object_read()

```
psa_status_t ps_encrypted_object_read (
            uint32_t fid,
            struct ps_object_t * obj,
            uint32_t * p_blocks )
```

Reads object referenced by the object File ID.

**Parameters**

| | | |
|---|---|---|
| in | *fid* | File ID |
| out | *obj* | Pointer to the object structure to fill in |
| out | *p_blocks* | Pointer to a counter of decryption blocks used. |

**Returns**

Returns error code specified in psa_status_t

Definition at line 150 of file ps_encrypted_object.c.
Here is the call graph for this function:

#### 7.323.1.3 ps_encrypted_object_write()

```
psa_status_t ps_encrypted_object_write (
            uint32_t fid,
            struct ps_object_t * obj )
```

Creates and writes a new encrypted object based on the given ps_object_t structure data.

**Parameters**

| | | |
|---|---|---|
| in | *fid* | File ID |
| in,out | *obj* | Pointer to the object structure to write. |

Note: The function will use obj to store the encrypted data before write it into the flash to reduce the memory requirements and the number of internal copies. So, this object will contain the encrypted object stored in the flash.

**Returns**

Returns error code specified in psa_status_t

Definition at line 193 of file ps_encrypted_object.c.

## 7.324 secure_fw/partitions/protected_storage/ps_filesystem_interface.c File Reference

```
#include "psa/client.h"
#include "psa_manifest/sid.h"
#include "tfm_its_defs.h"
#include "psa_manifest/pid.h"
#include "tfm_internal_trusted_storage.h"
```

Include dependency graph for ps_filesystem_interface.c:

## Functions

- psa_status_t psa_its_set (psa_storage_uid_t uid, size_t data_length, void ∗p_data, psa_storage_create_flags_t create_flags)
- psa_status_t psa_its_get (psa_storage_uid_t uid, size_t data_offset, size_t data_size, void ∗p_data, size_t ∗p_data_length)

  *Retrieve data associated with a provided UID.*
- psa_status_t psa_its_get_info (psa_storage_uid_t uid, struct psa_storage_info_t ∗p_info)

  *Retrieve the metadata about the provided uid.*
- psa_status_t psa_its_remove (psa_storage_uid_t uid)

  *Remove the provided uid and its associated data from the storage.*

## Variables

- uint8_t ∗ p_psa_src_data
- uint8_t ∗ p_psa_dest_data

## 7.324.1 Function Documentation

### 7.324.1.1 psa_its_get()

```
psa_status_t psa_its_get (
            psa_storage_uid_t uid,
            size_t data_offset,
            size_t data_size,
            void * p_data,
            size_t * p_data_length )
```

Retrieve data associated with a provided UID.

Retrieves up to `data_size` bytes of the data associated with `uid`, starting at `data_offset` bytes from the beginning of the data. Upon successful completion, the data will be placed in the `p_data` buffer, which must be at least `data_size` bytes in size. The length of the data returned will be in `p_data_length`. If `data_size` is 0, the contents of `p_data_length` will be set to zero.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The uid value |
| in | *data_offset* | The starting offset of the data requested |

**Parameters**

| | | |
|---|---|---|
| in | *data_size* | The amount of data requested |
| out | *p_data* | On success, the buffer where the data will be placed |
| out | *p_data_length* | On success, this will contain size of the data placed in `p_data` |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided `uid` value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided arguments (`p_data`, `p_data_length`) is invalid, for example is `NULL` or references memory the caller cannot access. In addition, this can also happen if `data_offset` is larger than the size of the data associated with `uid` |

Definition at line 27 of file ps_filesystem_interface.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.324.1.2 psa_its_get_info()

```
psa_status_t psa_its_get_info (
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Retrieve the metadata about the provided uid.
Retrieves the metadata stored for a given `uid` as a `psa_storage_info_t` structure.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |
| out | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one of the provided pointers(`p_info`) is invalid, for example is `NULL` or references memory the caller cannot access |

Definition at line 42 of file ps_filesystem_interface.c.
Here is the call graph for this function:

#### 7.324.1.3 psa_its_remove()

```
psa_status_t psa_its_remove (
            psa_storage_uid_t uid )
```

Remove the provided uid and its associated data from the storage.
Deletes the data from internal storage.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | The `uid` value |

**Returns**

A status indicating the success/failure of the operation

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, wrong flags and so on) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (Fatal error) |

Definition at line 48 of file ps_filesystem_interface.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.324.1.4 psa_its_set()

```
psa_status_t psa_its_set (
            psa_storage_uid_t uid,
            size_t data_length,
            void * p_data,
            psa_storage_create_flags_t create_flags )
```

Definition at line 17 of file ps_filesystem_interface.c.

Here is the call graph for this function:

### 7.324.2 Variable Documentation

#### 7.324.2.1 p_psa_dest_data

`uint8_t* p_psa_dest_data`
Definition at line 15 of file ps_filesystem_interface.c.

#### 7.324.2.2 p_psa_src_data

`uint8_t* p_psa_src_data`
Definition at line 14 of file ps_filesystem_interface.c.

## 7.325 secure_fw/partitions/protected_storage/ps_object_defs.h File Reference

```
#include <stdint.h>
#include "config_tfm.h"
#include "psa/protected_storage.h"
```
Include dependency graph for ps_object_defs.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct ps_object_info_t

  *Object information.*
- struct ps_obj_header_t

  *Metadata attached as a header to object data before storage.*
- struct ps_object_t

  *The object to be written to the file system below. Made up of the object header and the object data.*

## Macros

- #define PS_MAX_OBJECT_DATA_SIZE PS_MAX_ASSET_SIZE
- #define PS_OBJECT_BUF_SIZE PS_MAX_OBJECT_DATA_SIZE
- #define PS_OBJECT_HEADER_SIZE sizeof(struct ps_obj_header_t)
- #define PS_MAX_OBJECT_SIZE sizeof(struct ps_object_t)
- #define PS_MAX_NUM_OBJECTS (PS_NUM_ASSETS + 3)

  *Specifies the maximum number of objects in the system, which is the number of defined assets, the object table and 2 temporary objects to store the temporary object table and temporary updated object.*

## 7.325.1 Macro Definition Documentation

### 7.325.1.1 PS_MAX_NUM_OBJECTS

```
#define PS_MAX_NUM_OBJECTS (PS_NUM_ASSETS + 3)
```

Specifies the maximum number of objects in the system, which is the number of defined assets, the object table and 2 temporary objects to store the temporary object table and temporary updated object.
Definition at line 80 of file ps_object_defs.h.

#### 7.325.1.2 PS_MAX_OBJECT_DATA_SIZE

```
#define PS_MAX_OBJECT_DATA_SIZE PS_MAX_ASSET_SIZE
```

Definition at line 49 of file ps_object_defs.h.

#### 7.325.1.3 PS_MAX_OBJECT_SIZE

```
#define PS_MAX_OBJECT_SIZE sizeof(struct ps_object_t)
```

Definition at line 71 of file ps_object_defs.h.

#### 7.325.1.4 PS_OBJECT_BUF_SIZE

```
#define PS_OBJECT_BUF_SIZE PS_MAX_OBJECT_DATA_SIZE
```

Definition at line 54 of file ps_object_defs.h.

#### 7.325.1.5 PS_OBJECT_HEADER_SIZE

```
#define PS_OBJECT_HEADER_SIZE sizeof(struct ps_obj_header_t)
```

Definition at line 70 of file ps_object_defs.h.

## 7.326 secure_fw/partitions/protected_storage/ps_object_system.c File Reference

```
#include "ps_object_system.h"
#include <stdbool.h>
#include <stddef.h>
#include <string.h>
#include "cmsis_compiler.h"
#include "psa/internal_trusted_storage.h"
#include "ps_object_defs.h"
#include "ps_object_table.h"
#include "ps_utils.h"
#include "tfm_ps_req_mngr.h"
#include "tfm_sp_log.h"
#include "utilities.h"
```

Include dependency graph for ps_object_system.c:

### Macros

- #define PS_OBJECT_SIZE(max_size) (PS_OBJECT_HEADER_SIZE + (max_size))
- #define PS_OBJECT_START_POSITION 0

## Enumerations

- enum read_type_t { READ_HEADER_ONLY = 0, READ_ALL_OBJECT }

## Functions

- __STATIC_INLINE void ps_init_empty_object (psa_storage_uid_t uid, int32_t client_id, psa_storage_create_flags_t create_flags, uint32_t size, struct ps_object_t *obj)

  *Initialize g_ps_object based on the input parameters and empty data.*
- psa_status_t ps_system_prepare (void)

  *Prepares the protected storage system for usage, populating internal structures. It identifies and validates the system metadata.*
- psa_status_t ps_object_read (psa_storage_uid_t uid, int32_t client_id, uint32_t offset, uint32_t size, size_t *p_data_length)

  *Gets the data of the object with the provided UID and client ID.*
- psa_status_t ps_object_create (psa_storage_uid_t uid, int32_t client_id, psa_storage_create_flags_t create_flags, uint32_t size)

  *Creates a new object with the provided UID and client ID.*
- psa_status_t ps_object_write (psa_storage_uid_t uid, int32_t client_id, uint32_t offset, uint32_t size)

  *Writes data into the object with the provided UID and client ID.*
- psa_status_t ps_object_get_info (psa_storage_uid_t uid, int32_t client_id, struct psa_storage_info_t *info)

  *Gets the asset information for the object with the provided UID and client ID.*
- psa_status_t ps_object_delete (psa_storage_uid_t uid, int32_t client_id)

  *Deletes the object with the provided UID and client ID.*
- psa_status_t ps_system_wipe_all (void)

  *Wipes the protected storage system and all object data.*

## 7.326.1 Macro Definition Documentation

### 7.326.1.1 PS_OBJECT_SIZE

```
#define PS_OBJECT_SIZE(
            max_size ) (PS_OBJECT_HEADER_SIZE + (max_size))
```

Definition at line 30 of file ps_object_system.c.

### 7.326.1.2 PS_OBJECT_START_POSITION

```
#define PS_OBJECT_START_POSITION 0
```

Definition at line 31 of file ps_object_system.c.

## 7.326.2 Enumeration Type Documentation

### 7.326.2.1 read_type_t

```
enum read_type_t
```

**Enumerator**

| | |
|---|---|
| READ_HEADER_ONLY | |
| READ_ALL_OBJECT | |

Definition at line 155 of file ps_object_system.c.

## 7.326.3 Function Documentation

### 7.326.3.1 ps_init_empty_object()

```
__STATIC_INLINE void ps_init_empty_object (
            psa_storage_uid_t uid,
            int32_t client_id,
            psa_storage_create_flags_t create_flags,
            uint32_t size,
            struct ps_object_t * obj )
```

Initialize g_ps_object based on the input parameters and empty data.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *create_flags* | Object create flags |
| in | *size* | Object size |
| out | *obj* | Object to initialize |

Definition at line 49 of file ps_object_system.c.
Here is the call graph for this function:

### 7.326.3.2 ps_object_create()

```
psa_status_t ps_object_create (
            psa_storage_uid_t uid,
            int32_t client_id,
            psa_storage_create_flags_t create_flags,
            uint32_t size )
```

Creates a new object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *create_flags* | Flags indicating the properties of the data |
| in | *size* | Size of the contents of `data` in bytes |

**Returns**

Returns error code specified in psa_status_t

Definition at line 378 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.326.3.3 ps_object_delete()

```
psa_status_t ps_object_delete (
            psa_storage_uid_t uid,
            int32_t client_id )
```

Deletes the object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client↩ _id* | Identifier of the asset's owner (client) |

**Returns**

Returns error code specified in psa_status_t

Definition at line 665 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.326.3.4 ps_object_get_info()

```
psa_status_t ps_object_get_info (
            psa_storage_uid_t uid,
            int32_t client_id,
            struct psa_storage_info_t * info )
```

Gets the asset information for the object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| out | *info* | Pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

Returns error code specified in psa_status_t

Definition at line 608 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.326.3.5 ps_object_read()

```
psa_status_t ps_object_read (
            psa_storage_uid_t uid,
            int32_t client_id,
            uint32_t offset,
            uint32_t size,
            size_t * p_data_length )
```

Gets the data of the object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *offset* | Offset in the object at which to begin the read |
| in | *size* | Size of the contents of `data` in bytes |
| out | *p_data_length* | On success, this will contain size of the data written to asset |

**Returns**

Returns error code specified in psa_status_t

Definition at line 305 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.326.3.6 ps_object_write()

```
psa_status_t ps_object_write (
            psa_storage_uid_t uid,
            int32_t client_id,
            uint32_t offset,
            uint32_t size )
```

Writes data into the object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| in | *offset* | Offset in the object at which to begin the write |
| in | *size* | Size of the contents of `data` in bytes |

**Returns**

Returns error code specified in psa_status_t

Definition at line 500 of file ps_object_system.c.
Here is the call graph for this function:

#### 7.326.3.7 ps_system_prepare()

```
psa_status_t ps_system_prepare (
            void  )
```

Prepares the protected storage system for usage, populating internal structures. It identifies and validates the system metadata.

**Returns**

Returns error code specified in psa_status_t

Definition at line 275 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.326.3.8 ps_system_wipe_all()

```
psa_status_t ps_system_wipe_all (
            void  )
```

Wipes the protected storage system and all object data.

**Returns**

Returns error code specified in psa_status_t

Definition at line 749 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.327 secure_fw/partitions/protected_storage/ps_object_system.h File Reference

```
#include <stdint.h>
#include "psa/protected_storage.h"
```

Include dependency graph for ps_object_system.h:

This graph shows which files directly or indirectly include this file:

## Functions

- psa_status_t ps_system_prepare (void)

  *Prepares the protected storage system for usage, populating internal structures. It identifies and validates the system metadata.*
- psa_status_t ps_object_create (psa_storage_uid_t uid, int32_t client_id, psa_storage_create_flags_t create_flags, uint32_t size)

  *Creates a new object with the provided UID and client ID.*
- psa_status_t ps_object_read (psa_storage_uid_t uid, int32_t client_id, uint32_t offset, uint32_t size, size_t *p_data_length)

  *Gets the data of the object with the provided UID and client ID.*
- psa_status_t ps_object_write (psa_storage_uid_t uid, int32_t client_id, uint32_t offset, uint32_t size)

  *Writes data into the object with the provided UID and client ID.*
- psa_status_t ps_object_delete (psa_storage_uid_t uid, int32_t client_id)

  *Deletes the object with the provided UID and client ID.*
- psa_status_t ps_object_get_info (psa_storage_uid_t uid, int32_t client_id, struct psa_storage_info_t *info)

  *Gets the asset information for the object with the provided UID and client ID.*
- psa_status_t ps_system_wipe_all (void)

  *Wipes the protected storage system and all object data.*

## 7.327.1 Function Documentation

### 7.327.1.1 ps_object_create()

```
psa_status_t ps_object_create (
            psa_storage_uid_t uid,
            int32_t client_id,
            psa_storage_create_flags_t create_flags,
            uint32_t size )
```

Creates a new object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *create_flags* | Flags indicating the properties of the data |
| in | *size* | Size of the contents of `data` in bytes |

**Returns**

Returns error code specified in psa_status_t

Definition at line 378 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.327.1.2 ps_object_delete()

```
psa_status_t ps_object_delete (
            psa_storage_uid_t uid,
            int32_t client_id )
```

Deletes the object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client↩ _id* | Identifier of the asset's owner (client) |

**Returns**

Returns error code specified in psa_status_t

Definition at line 665 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.327.1.3 ps_object_get_info()

```
psa_status_t ps_object_get_info (
            psa_storage_uid_t uid,
            int32_t client_id,
            struct psa_storage_info_t * info )
```

Gets the asset information for the object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| out | *info* | Pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

Returns error code specified in psa_status_t

Definition at line 608 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.327.1.4 ps_object_read()

```
psa_status_t ps_object_read (
            psa_storage_uid_t uid,
            int32_t client_id,
            uint32_t offset,
            uint32_t size,
            size_t * p_data_length )
```

Gets the data of the object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *offset* | Offset in the object at which to begin the read |
| in | *size* | Size of the contents of `data` in bytes |
| out | *p_data_length* | On success, this will contain size of the data written to asset |

**Returns**

Returns error code specified in psa_status_t

Definition at line 305 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.327.1.5 ps_object_write()

```
psa_status_t ps_object_write (
            psa_storage_uid_t uid,
            int32_t client_id,
            uint32_t offset,
            uint32_t size )
```

Writes data into the object with the provided UID and client ID.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Unique identifier for the data |
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| in | *offset* | Offset in the object at which to begin the write |
| in | *size* | Size of the contents of `data` in bytes |

**Returns**

Returns error code specified in psa_status_t

Definition at line 500 of file ps_object_system.c.
Here is the call graph for this function:

#### 7.327.1.6 ps_system_prepare()

```
psa_status_t ps_system_prepare (
            void  )
```

Prepares the protected storage system for usage, populating internal structures. It identifies and validates the system metadata.

**Returns**

Returns error code specified in psa_status_t

Definition at line 275 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.327.1.7 ps_system_wipe_all()

```
psa_status_t ps_system_wipe_all (
            void  )
```

Wipes the protected storage system and all object data.

**Returns**

Returns error code specified in psa_status_t

Definition at line 749 of file ps_object_system.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.328 secure_fw/partitions/protected_storage/ps_object_table.c File Reference

```
#include "ps_object_table.h"
#include <stddef.h>
#include <string.h>
#include "cmsis_compiler.h"
#include "config_tfm.h"
#include "crypto/ps_crypto_interface.h"
#include "psa_manifest/pid.h"
#include "nv_counters/ps_nv_counters.h"
#include "psa/internal_trusted_storage.h"
#include "ps_utils.h"
#include "tfm_ps_defs.h"
#include "utilities.h"
#include "static_checks.h"
```

Include dependency graph for ps_object_table.c:

### Data Structures

- struct ps_obj_table_entry_t

- struct ps_obj_table_t

  *Object table structure.*
- struct ps_obj_table_ctx_t

  *Object table context structure.*
- struct ps_obj_table_init_ctx_t

## Macros

- #define PS_FLASH_DEFAULT_VAL 0xFFU
- #define PS_OBJECT_SYSTEM_VERSION 0x01

  *Current object system version.*
- #define PS_OBJ_TABLE_ENTRIES (PS_NUM_ASSETS + 1)
- #define PS_OBJ_TABLE_IDX_0 0
- #define PS_OBJ_TABLE_IDX_1 1
- #define PS_NUM_OBJ_TABLES 2
- #define PS_TABLE_FS_ID(idx) (idx + 1)

  *File ID to be used in order to store the object table in the file system.*
- #define PS_OBJECT_FS_ID(idx)

  *File ID to be used in order to store an object in the file system.*
- #define PS_OBJECT_FS_ID_TO_IDX(fid)

  *Gets object index in the table based on the file ID.*
- #define PS_OBJ_TABLE_SIZE sizeof(struct ps_obj_table_t)
- #define PS_OBJECTS_TABLE_ENTRY_SIZE sizeof(struct ps_obj_table_entry_t)
- #define PS_NON_AUTH_OBJ_TABLE_SIZE sizeof(union ps_crypto_t)
- #define PS_OBJECT_TABLE_OBJECT_OFFSET 0
- #define PS_CRYPTO_ASSOCIATED_DATA(crypto)
- #define PS_CRYPTO_ASSOCIATED_DATA_LEN
- #define PS_INVALID_NVC_VALUE 0

## Typedefs

- typedef char OBJ_TABLE_NOT_FIT_IN_STATIC_OBJ_DATA_BUF[(sizeof(struct ps_obj_table_t)<=PS_↩ MAX_ASSET_SIZE) ∗2 - 1]

## Enumerations

- enum ps_obj_table_state { PS_OBJ_TABLE_VALID = 0, PS_OBJ_TABLE_INVALID, PS_OBJ_TABLE_NVC_1_VALID, PS_OBJ_TABLE_NVC_3_VALID }

## Functions

- __STATIC_INLINE void ps_object_table_fs_read_table (struct ps_obj_table_init_ctx_t ∗init_ctx)

  *Reads object table from persistent memory.*
- __STATIC_INLINE psa_status_t ps_object_table_fs_write_table (struct ps_obj_table_t ∗obj_table)

  *Writes object table in persistent memory.*
- __STATIC_INLINE void ps_object_table_validate_version (struct ps_obj_table_init_ctx_t ∗init_ctx)

  *Checks the validity of the table version.*
- __STATIC_INLINE psa_status_t ps_table_free_idx (uint32_t idx_num, uint32_t ∗idx)

  *Gets free index in the table.*
- psa_status_t ps_object_table_create (void)

  *Creates object table.*
- psa_status_t ps_object_table_init (uint8_t ∗obj_data)

  *Initializes object table.*
- psa_status_t ps_object_table_obj_exist (psa_storage_uid_t uid, int32_t client_id)

*Checks if there is an entry in the table for the provided UID and client ID pair.*

- psa_status_t ps_object_table_get_free_fid (uint32_t fid_num, uint32_t *p_fid)

  *Gets a not in use file ID.*

- uint32_t ps_object_table_current_gen (void)
- void ps_object_table_set_gen (uint32_t new_gen)
- psa_status_t ps_object_table_set_obj_tbl_info (psa_storage_uid_t uid, int32_t client_id, const struct ps_obj_table_info_t *obj_tbl_info)

  *Sets object table information in the object table and stores it persistently, for the provided UID and client ID pair.*

- psa_status_t ps_object_table_get_obj_tbl_info (psa_storage_uid_t uid, int32_t client_id, struct ps_obj_table_info_t *obj_tbl_info)

  *Gets object table information from the object table for the provided UID and client ID pair.*

- psa_status_t ps_object_table_delete_object (psa_storage_uid_t uid, int32_t client_id)

  *Deletes the table entry for the provided UID and client ID pair.*

- psa_status_t ps_object_table_delete_old_table (void)

  *Deletes old object table from the persistent area.*

## 7.328.1 Macro Definition Documentation

### 7.328.1.1 PS_CRYPTO_ASSOCIATED_DATA

```
#define PS_CRYPTO_ASSOCIATED_DATA(
            crypto )
```

**Value:**

```
((uint8_t *)crypto + \
PS_NON_AUTH_OBJ_TABLE_SIZE)
```

Definition at line 163 of file ps_object_table.c.

### 7.328.1.2 PS_CRYPTO_ASSOCIATED_DATA_LEN

```
#define PS_CRYPTO_ASSOCIATED_DATA_LEN
```

**Value:**

```
(PS_OBJ_TABLE_SIZE - \
PS_NON_AUTH_OBJ_TABLE_SIZE)
```

Definition at line 180 of file ps_object_table.c.

### 7.328.1.3 PS_FLASH_DEFAULT_VAL

```
#define PS_FLASH_DEFAULT_VAL 0xFFU
```

Definition at line 27 of file ps_object_table.c.

### 7.328.1.4 PS_INVALID_NVC_VALUE

```
#define PS_INVALID_NVC_VALUE 0
```

Definition at line 208 of file ps_object_table.c.

### 7.328.1.5 PS_NON_AUTH_OBJ_TABLE_SIZE

```
#define PS_NON_AUTH_OBJ_TABLE_SIZE sizeof(union ps_crypto_t)
```

Definition at line 157 of file ps_object_table.c.

#### 7.328.1.6 PS_NUM_OBJ_TABLES

```
#define PS_NUM_OBJ_TABLES 2
```

Definition at line 96 of file ps_object_table.c.

#### 7.328.1.7 PS_OBJ_TABLE_ENTRIES

```
#define PS_OBJ_TABLE_ENTRIES (PS_NUM_ASSETS + 1)
```

Definition at line 58 of file ps_object_table.c.

#### 7.328.1.8 PS_OBJ_TABLE_IDX_0

```
#define PS_OBJ_TABLE_IDX_0 0
```

Definition at line 92 of file ps_object_table.c.

#### 7.328.1.9 PS_OBJ_TABLE_IDX_1

```
#define PS_OBJ_TABLE_IDX_1 1
```

Definition at line 93 of file ps_object_table.c.

#### 7.328.1.10 PS_OBJ_TABLE_SIZE

```
#define PS_OBJ_TABLE_SIZE sizeof(struct ps_obj_table_t)
```

Definition at line 151 of file ps_object_table.c.

#### 7.328.1.11 PS_OBJECT_FS_ID

```
#define PS_OBJECT_FS_ID(
            idx )
```

**Value:**

```
((idx + 1) + \
PS_TABLE_FS_ID(PS_OBJ_TABLE_IDX_1))
```

File ID to be used in order to store an object in the file system.

**Parameters**

| | | |
|---|---|---|
| in | *idx* | Object table index to convert into a file ID. |

**Returns**

Returns file ID

Definition at line 121 of file ps_object_table.c.

#### 7.328.1.12 PS_OBJECT_FS_ID_TO_IDX

```
#define PS_OBJECT_FS_ID_TO_IDX(
            fid )
```

**Value:**

```
((fid - 1) - \
PS_TABLE_FS_ID(PS_OBJ_TABLE_IDX_1))
```

Gets object index in the table based on the file ID.

**Parameters**

| | | |
|---|---|---|
| in | *fid* | File ID of an object in the object table |

**Returns**

Returns object table index

Definition at line 133 of file ps_object_table.c.

#### 7.328.1.13 PS_OBJECT_SYSTEM_VERSION

```
#define PS_OBJECT_SYSTEM_VERSION 0x01
```

Current object system version.
Definition at line 34 of file ps_object_table.c.

#### 7.328.1.14 PS_OBJECT_TABLE_OBJECT_OFFSET

```
#define PS_OBJECT_TABLE_OBJECT_OFFSET 0
```

Definition at line 160 of file ps_object_table.c.

#### 7.328.1.15 PS_OBJECTS_TABLE_ENTRY_SIZE

```
#define PS_OBJECTS_TABLE_ENTRY_SIZE sizeof(struct ps_obj_table_entry_t)
```

Definition at line 154 of file ps_object_table.c.

#### 7.328.1.16 PS_TABLE_FS_ID

```
#define PS_TABLE_FS_ID(
            idx ) (idx + 1)
```

File ID to be used in order to store the object table in the file system.

**Parameters**

| | | |
|---|---|---|
| in | *idx* | Table index to convert into a file ID. |

**Returns**

Returns file ID

Definition at line 109 of file ps_object_table.c.

### 7.328.2 Typedef Documentation

#### 7.328.2.1 OBJ_TABLE_NOT_FIT_IN_STATIC_OBJ_DATA_BUF

```
typedef char OBJ_TABLE_NOT_FIT_IN_STATIC_OBJ_DATA_BUF[( sizeof(struct ps_obj_table_t)<= PS_M↩
AX_ASSET_SIZE) *2 - 1]
```

Definition at line 198 of file ps_object_table.c.

### 7.328.3 Enumeration Type Documentation

#### 7.328.3.1 ps_obj_table_state

```
enum ps_obj_table_state
```

**Enumerator**

| | |
|---|---|
| PS_OBJ_TABLE_VALID | Table content is valid |
| PS_OBJ_TABLE_INVALID | Table content is invalid |
| PS_OBJ_TABLE_NVC_1_VALID | Table content valid with NVC 1 value |
| PS_OBJ_TABLE_NVC_3_VALID | Table content valid with NVC 3 value |

Definition at line 200 of file ps_object_table.c.

## 7.328.4 Function Documentation

### 7.328.4.1 ps_object_table_create()

```
psa_status_t ps_object_table_create (
            void  )
```

Creates object table.

**Returns**

Returns error code as specified in psa_status_t

Definition at line 824 of file ps_object_table.c.
Here is the caller graph for this function:

### 7.328.4.2 ps_object_table_current_gen()

```
uint32_t ps_object_table_current_gen (
            void  )
```

Definition at line 955 of file ps_object_table.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.328.4.3 ps_object_table_delete_object()

```
psa_status_t ps_object_table_delete_object (
            psa_storage_uid_t uid,
            int32_t client_id )
```

Deletes the table entry for the provided UID and client ID pair.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Identifier for the data. |
| in | *client↩ _id* | Identifier of the asset's owner (client) |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1071 of file ps_object_table.c.

#### 7.328.4.4 ps_object_table_delete_old_table()

```
psa_status_t ps_object_table_delete_old_table (
            void  )
```

Deletes old object table from the persistent area.

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1107 of file ps_object_table.c.

#### 7.328.4.5 ps_object_table_fs_read_table()

```
__STATIC_INLINE void ps_object_table_fs_read_table (
            struct ps_obj_table_init_ctx_t * init_ctx )
```

Reads object table from persistent memory.

**Parameters**

| | | |
|---|---|---|
| out | *init_ctx* | Pointer to the init object table context |

Definition at line 238 of file ps_object_table.c.

Here is the call graph for this function:

#### 7.328.4.6 ps_object_table_fs_write_table()

```
__STATIC_INLINE psa_status_t ps_object_table_fs_write_table (
            struct ps_obj_table_t * obj_table )
```

Writes object table in persistent memory.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj_table* | Pointer to the object table to generate authentication |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 275 of file ps_object_table.c.

#### 7.328.4.7 ps_object_table_get_free_fid()

```
psa_status_t ps_object_table_get_free_fid (
            uint32_t fid_num,
            uint32_t * p_fid )
```

Gets a not in use file ID.

**Parameters**

| | | |
|---|---|---|
| in | *fid_num* | Amount of file IDs that the function will check are free before returning one. 0 is an invalid input and will error. Note that this function will only ever return 1 file ID. |
| out | *p_fid* | Pointer to the location to store the file ID |

**Returns**

Returns PSA_SUCCESS if the fid is valid and fid_num - 1 entries are still free in the table. Otherwise, it returns an error code as specified in psa_status_t

Definition at line 924 of file ps_object_table.c.
Here is the call graph for this function:

### 7.328.4.8 ps_object_table_get_obj_tbl_info()

```
psa_status_t ps_object_table_get_obj_tbl_info (
            psa_storage_uid_t uid,
            int32_t client_id,
            struct ps_obj_table_info_t * obj_tbl_info )
```

Gets object table information from the object table for the provided UID and client ID pair.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Identifier for the data. |
| in | *client_id* | Identifier of the asset's owner (client) |
| out | *obj_tbl_info* | Pointer to the location to store object table information |

**Returns**

Returns PSA_SUCCESS if the object exists. Otherwise, it returns PSA_ERROR_DOES_NOT_EXIST.

Definition at line 1043 of file ps_object_table.c.
Here is the caller graph for this function:

#### 7.328.4.9 ps_object_table_init()

```
psa_status_t ps_object_table_init (
            uint8_t * obj_data )
```

Initializes object table.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj_data* | Pointer to the static object data allocated in other to reuse that memory to allocated a temporary object table. |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 852 of file ps_object_table.c.
Here is the caller graph for this function:

#### 7.328.4.10 ps_object_table_obj_exist()

```
psa_status_t ps_object_table_obj_exist (
            psa_storage_uid_t uid,
            int32_t client_id )
```

Checks if there is an entry in the table for the provided UID and client ID pair.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Identifier for the data |
| in | *client↩ _id* | Identifier of the asset's owner (client) |

**Returns**

Returns error code as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | If there is a table entry for the object |
| *PSA_ERROR_DOES_NOT_EXIST* | If no table entry exists for the object |

Definition at line 916 of file ps_object_table.c.

#### 7.328.4.11 ps_object_table_set_gen()

```
void ps_object_table_set_gen (
```

```
uint32_t new_gen )
```

Definition at line 967 of file ps_object_table.c.
Here is the call graph for this function:

#### 7.328.4.12 ps_object_table_set_obj_tbl_info()

```
psa_status_t ps_object_table_set_obj_tbl_info (
            psa_storage_uid_t uid,
            int32_t client_id,
            const struct ps_obj_table_info_t * obj_tbl_info )
```

Sets object table information in the object table and stores it persistently, for the provided UID and client ID pair.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Identifier for the data. |
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *obj_tbl_info* | Pointer to the location to store object table information ps_obj_table_info_t |

**Note**

A call to this function results in writing the table to the file system.

**Returns**

Returns error code as specified in psa_status_t

Definition at line 984 of file ps_object_table.c.

#### 7.328.4.13 ps_object_table_validate_version()

```
__STATIC_INLINE void ps_object_table_validate_version (
            struct ps_obj_table_init_ctx_t * init_ctx )
```

Checks the validity of the table version.

**Parameters**

| | | |
|---|---|---|
| in,out | *init_ctx* | Pointer to the init object table context |

Definition at line 612 of file ps_object_table.c.

#### 7.328.4.14 ps_table_free_idx()

```
__STATIC_INLINE psa_status_t ps_table_free_idx (
            uint32_t idx_num,
            uint32_t * idx )
```

Gets free index in the table.

**Parameters**

| | | |
|---|---|---|
| in | *idx_num* | The number of indices required to be free before one can be allocated. Primarily used to prevent index exhaustion.Note that this function will only ever return 1 index. |
| out | *idx* | Pointer to store the free index |

**Note**

The table is dimensioned to fit PS_NUM_ASSETS + 1

**Returns**

Returns PSA_SUCCESS and a table index if idx_num free indices are available. Otherwise, it returns PSA↩ _ERROR_INSUFFICIENT_STORAGE.

Definition at line 785 of file ps_object_table.c.
Here is the caller graph for this function:

## 7.329 secure_fw/partitions/protected_storage/ps_object_table.h File Reference

```
#include <stdint.h>
#include "psa/protected_storage.h"
```

Include dependency graph for ps_object_table.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct ps_obj_table_info_t

  *Object table information structure.*

## Functions

- psa_status_t ps_object_table_create (void)

  *Creates object table.*
- psa_status_t ps_object_table_init (uint8_t *obj_data)

  *Initializes object table.*

- psa_status_t ps_object_table_obj_exist (psa_storage_uid_t uid, int32_t client_id)

  *Checks if there is an entry in the table for the provided UID and client ID pair.*
- psa_status_t ps_object_table_get_free_fid (uint32_t fid_num, uint32_t *p_fid)

  *Gets a not in use file ID.*
- psa_status_t ps_object_table_set_obj_tbl_info (psa_storage_uid_t uid, int32_t client_id, const struct ps_obj_table_info_t *obj_tbl_info)

  *Sets object table information in the object table and stores it persistently, for the provided UID and client ID pair.*
- psa_status_t ps_object_table_get_obj_tbl_info (psa_storage_uid_t uid, int32_t client_id, struct ps_obj_table_info_t *obj_tbl_info)

  *Gets object table information from the object table for the provided UID and client ID pair.*
- psa_status_t ps_object_table_delete_object (psa_storage_uid_t uid, int32_t client_id)

  *Deletes the table entry for the provided UID and client ID pair.*
- psa_status_t ps_object_table_delete_old_table (void)

  *Deletes old object table from the persistent area.*

## 7.329.1 Function Documentation

### 7.329.1.1 ps_object_table_create()

```
psa_status_t ps_object_table_create (
            void  )
```

Creates object table.

**Returns**

Returns error code as specified in psa_status_t

Definition at line 824 of file ps_object_table.c.
Here is the caller graph for this function:

### 7.329.1.2 ps_object_table_delete_object()

```
psa_status_t ps_object_table_delete_object (
            psa_storage_uid_t uid,
            int32_t client_id )
```

Deletes the table entry for the provided UID and client ID pair.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Identifier for the data. |
| in | *client_id* | Identifier of the asset's owner (client) |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1071 of file ps_object_table.c.

#### 7.329.1.3 ps_object_table_delete_old_table()

```
psa_status_t ps_object_table_delete_old_table (
            void  )
```

Deletes old object table from the persistent area.

**Returns**

Returns error code as specified in psa_status_t

Definition at line 1107 of file ps_object_table.c.

#### 7.329.1.4 ps_object_table_get_free_fid()

```
psa_status_t ps_object_table_get_free_fid (
            uint32_t fid_num,
            uint32_t * p_fid )
```

Gets a not in use file ID.

**Parameters**

| | | |
|---|---|---|
| in | *fid_num* | Amount of file IDs that the function will check are free before returning one. 0 is an invalid input and will error. Note that this function will only ever return 1 file ID. |
| out | *p_fid* | Pointer to the location to store the file ID |

**Returns**

Returns PSA_SUCCESS if the fid is valid and fid_num - 1 entries are still free in the table. Otherwise, it returns an error code as specified in psa_status_t

Definition at line 924 of file ps_object_table.c.
Here is the call graph for this function:

#### 7.329.1.5 ps_object_table_get_obj_tbl_info()

```
psa_status_t ps_object_table_get_obj_tbl_info (
            psa_storage_uid_t uid,
            int32_t client_id,
            struct ps_obj_table_info_t * obj_tbl_info )
```

Gets object table information from the object table for the provided UID and client ID pair.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Identifier for the data. |
| in | *client_id* | Identifier of the asset's owner (client) |
| out | *obj_tbl_info* | Pointer to the location to store object table information |

**Returns**

Returns PSA_SUCCESS if the object exists. Otherwise, it returns PSA_ERROR_DOES_NOT_EXIST.

Definition at line 1043 of file ps_object_table.c.
Here is the caller graph for this function:

### 7.329.1.6 ps_object_table_init()

```
psa_status_t ps_object_table_init (
            uint8_t * obj_data )
```

Initializes object table.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj_data* | Pointer to the static object data allocated in other to reuse that memory to allocated a temporary object table. |

**Returns**

Returns error code as specified in psa_status_t

Definition at line 852 of file ps_object_table.c.
Here is the caller graph for this function:

#### 7.329.1.7 ps_object_table_obj_exist()

```
psa_status_t ps_object_table_obj_exist (
            psa_storage_uid_t uid,
            int32_t client_id )
```

Checks if there is an entry in the table for the provided UID and client ID pair.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Identifier for the data |
| in | *client_id* | Identifier of the asset's owner (client) |

**Returns**

Returns error code as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | If there is a table entry for the object |
| *PSA_ERROR_DOES_NOT_EXIST* | If no table entry exists for the object |

Definition at line 916 of file ps_object_table.c.

#### 7.329.1.8 ps_object_table_set_obj_tbl_info()

```
psa_status_t ps_object_table_set_obj_tbl_info (
            psa_storage_uid_t uid,
            int32_t client_id,
            const struct ps_obj_table_info_t * obj_tbl_info )
```

Sets object table information in the object table and stores it persistently, for the provided UID and client ID pair.

**Parameters**

| | | |
|---|---|---|
| in | *uid* | Identifier for the data. |
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *obj_tbl_info* | Pointer to the location to store object table information ps_obj_table_info_t |

**Note**

A call to this function results in writing the table to the file system.

**Returns**

Returns error code as specified in psa_status_t

Definition at line 984 of file ps_object_table.c.

## 7.330 secure_fw/partitions/protected_storage/ps_utils.c File Reference

```
#include "ps_utils.h"
```

Include dependency graph for ps_utils.c:

## Functions

- psa_status_t ps_utils_check_contained_in (uint32_t superset_size, uint32_t subset_offset, uint32_t subset↩ _size)

  *Checks if a subset region is fully contained within a superset region.*

## 7.330.1 Function Documentation

### 7.330.1.1 ps_utils_check_contained_in()

```
psa_status_t ps_utils_check_contained_in (
            uint32_t superset_size,
            uint32_t subset_offset,
            uint32_t subset_size )
```

Checks if a subset region is fully contained within a superset region.

**Parameters**

| | | |
|---|---|---|
| in | *superset_size* | Size of superset region |
| in | *subset_offset* | Offset of start of subset region from start of superset region |
| in | *subset_size* | Size of subset region |

**Returns**

Returns error code as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The subset is contained within the superset |
| *PSA_ERROR_INVALID_ARGUMENT* | The subset offset is greater than the size of the superset or when the subset offset is valid, but the subset offset + size is greater than the size of the superset |

Definition at line 10 of file ps_utils.c.

## 7.331 secure_fw/partitions/protected_storage/ps_utils.h File Reference

```
#include <stdint.h>
#include "psa/error.h"
#include "psa/protected_storage.h"
```

Include dependency graph for ps_utils.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define PS_INVALID_FID 0
- #define PS_DEFAULT_EMPTY_BUFF_VAL 0

- #define PS_UTILS_BOUND_CHECK(err_msg, data_size, data_buf_size) typedef char err_msg[(data_size <= data_buf_size)*2 - 1]

  *Macro to check, at compilation time, if data fits in data buffer.*
- #define PS_UTILS_MIN(x, y) (((x) < (y)) ? (x) : (y))

  *Evaluates to the minimum of the two parameters.*

## Functions

- psa_status_t ps_utils_check_contained_in (uint32_t superset_size, uint32_t subset_offset, uint32_t subset_size)

  *Checks if a subset region is fully contained within a superset region.*

## 7.331.1 Macro Definition Documentation

### 7.331.1.1 PS_DEFAULT_EMPTY_BUFF_VAL

```
#define PS_DEFAULT_EMPTY_BUFF_VAL 0
```

Definition at line 21 of file ps_utils.h.

### 7.331.1.2 PS_INVALID_FID

```
#define PS_INVALID_FID 0
```

Definition at line 20 of file ps_utils.h.

### 7.331.1.3 PS_UTILS_BOUND_CHECK

```
#define PS_UTILS_BOUND_CHECK(
            err_msg,
            data_size,
            data_buf_size ) typedef char err_msg[(data_size <= data_buf_size)*2 - 1]
```

Macro to check, at compilation time, if data fits in data buffer.

**Parameters**

| | | |
|---|---|---|
| in | *err_msg* | Error message which will be displayed in first instance if the error is triggered |
| in | *data_size* | Data size to check if it fits |
| in | *data_buf_size* | Size of the data buffer |

**Returns**

Triggers a compilation error if data_size is bigger than data_buf_size. The compilation error should be "... error: 'err_msg' declared as an array with a negative size"

Definition at line 35 of file ps_utils.h.

### 7.331.1.4 PS_UTILS_MIN

```
#define PS_UTILS_MIN(
            x,
            y ) (((x) < (y)) ? (x) : (y))
```

Evaluates to the minimum of the two parameters.
Definition at line 41 of file ps_utils.h.

### 7.331.2 Function Documentation

#### 7.331.2.1 ps_utils_check_contained_in()

```
psa_status_t ps_utils_check_contained_in (
            uint32_t superset_size,
            uint32_t subset_offset,
            uint32_t subset_size )
```

Checks if a subset region is fully contained within a superset region.

**Parameters**

| | | |
|---|---|---|
| in | *superset_size* | Size of superset region |
| in | *subset_offset* | Offset of start of subset region from start of superset region |
| in | *subset_size* | Size of subset region |

**Returns**

Returns error code as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The subset is contained within the superset |
| *PSA_ERROR_INVALID_ARGUMENT* | The subset offset is greater than the size of the superset or when the subset offset is valid, but the subset offset + size is greater than the size of the superset |

Definition at line 10 of file ps_utils.c.

## 7.332 secure_fw/partitions/protected_storage/tfm_protected_storage.c File Reference

```
#include "config_tfm.h"
#include "config_ps_check.h"
#include "tfm_protected_storage.h"
#include "ps_object_system.h"
#include "tfm_ps_defs.h"
#include "tfm_internal_trusted_storage.h"
#include "tfm_sp_log.h"
#include "psa/crypto.h"
```

Include dependency graph for tfm_protected_storage.c:

## Functions

- psa_status_t tfm_ps_init (void)

  *Initializes the protected storage system.*
- psa_status_t tfm_ps_set (int32_t client_id, psa_storage_uid_t uid, uint32_t data_length, psa_storage_create_flags_t create_flags)

  *Creates a new or modifies an existing asset.*
- psa_status_t tfm_ps_get (int32_t client_id, psa_storage_uid_t uid, uint32_t data_offset, uint32_t data_size, size_t *p_data_length)

  *Gets the asset data for the provided uid.*
- psa_status_t tfm_ps_get_info (int32_t client_id, psa_storage_uid_t uid, struct psa_storage_info_t *p_info)

  *Gets the metadata for the provided uid.*
- psa_status_t tfm_ps_remove (int32_t client_id, psa_storage_uid_t uid)

  *Removes the provided uid and its associated data from storage.*
- uint32_t tfm_ps_get_support (void)

  *Gets a bitmask with flags set for all of the optional features supported by the implementation.*

## 7.332.1 Function Documentation

### 7.332.1.1 tfm_ps_get()

```
psa_status_t tfm_ps_get (
            int32_t client_id,
            psa_storage_uid_t uid,
            uint32_t data_offset,
            uint32_t data_size,
            size_t * p_data_length )
```

Gets the asset data for the provided uid.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | Unique identifier for the data |
| in | *data_offset* | The offset within the data associated with the `uid` to start retrieving data |
| in | *data_size* | The amount of data to read (and the minimum allocated size of the `p_data` buffer) |
| out | *p_data_length* | The pointer to the size of the data retrieved upon success. |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, etc.) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the data associated with the UID was corrupt |
| *PSA_ERROR_INVALID_SIGNATURE* | The operation failed because the data associated with the UID failed authentication |

Definition at line 90 of file tfm_protected_storage.c.
Here is the call graph for this function:

### 7.332.1.2 tfm_ps_get_info()

```
psa_status_t tfm_ps_get_info (
            int32_t client_id,
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Gets the metadata for the provided uid.

**Parameters**

| | | |
|---|---|---|
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| in | *uid* | Unique identifier for the data |
| out | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |

**Return values**

| | |
|---|---|
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, etc.) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the data associated with the UID was corrupt |
| *PSA_ERROR_INVALID_SIGNATURE* | The operation failed because the data associated with the UID failed authentication |

Definition at line 106 of file tfm_protected_storage.c.
Here is the call graph for this function:

#### 7.332.1.3 tfm_ps_get_support()

```
uint32_t tfm_ps_get_support (
            void  )
```

Gets a bitmask with flags set for all of the optional features supported by the implementation.

**Returns**

Bitmask value which contains all the bits set for all the optional features supported by the implementation

Definition at line 141 of file tfm_protected_storage.c.

#### 7.332.1.4 tfm_ps_init()

```
psa_status_t tfm_ps_init (
            void  )
```

Initializes the protected storage system.

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the storage system initialization has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 23 of file tfm_protected_storage.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.332.1.5 tfm_ps_remove()

```
psa_status_t tfm_ps_remove (
            int32_t client_id,
            psa_storage_uid_t uid )
```

Removes the provided uid and its associated data from storage.

**Parameters**

| | | |
|---|---|---|
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| in | *uid* | Unique identifier for the data to be removed |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, etc.) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (fatal error) |

**Return values**

| | |
|---|---|
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 118 of file tfm_protected_storage.c.
Here is the call graph for this function:

#### 7.332.1.6 tfm_ps_set()

```
psa_status_t tfm_ps_set (
            int32_t client_id,
            psa_storage_uid_t uid,
            uint32_t data_length,
            psa_storage_create_flags_t create_flags )
```

Creates a new or modifies an existing asset.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | Unique identifier for the data |
| in | *data_length* | The size in bytes of the data in `p_data` |
| in | *create_flags* | The flags indicating the properties of the data |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was already created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, etc.) |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because one or more of the flags provided in `create_flags` is not supported or is not valid |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because there was insufficient space on the storage medium |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure. |

Definition at line 69 of file tfm_protected_storage.c.

Here is the call graph for this function:

## 7.333 secure_fw/partitions/protected_storage/tfm_protected_storage.h File Reference

```
#include <stdint.h>
#include "psa/protected_storage.h"
```

Include dependency graph for tfm_protected_storage.h:

This graph shows which files directly or indirectly include this file:

## Functions

- psa_status_t tfm_ps_init (void)

  *Initializes the protected storage system.*
- psa_status_t tfm_ps_set (int32_t client_id, psa_storage_uid_t uid, uint32_t data_length, psa_storage_create_flags_t create_flags)

  *Creates a new or modifies an existing asset.*
- psa_status_t tfm_ps_get (int32_t client_id, psa_storage_uid_t uid, uint32_t data_offset, uint32_t data_size, size_t *p_data_length)

  *Gets the asset data for the provided uid.*
- psa_status_t tfm_ps_get_info (int32_t client_id, psa_storage_uid_t uid, struct psa_storage_info_t *p_info)

  *Gets the metadata for the provided uid.*
- psa_status_t tfm_ps_remove (int32_t client_id, psa_storage_uid_t uid)

  *Removes the provided uid and its associated data from storage.*
- uint32_t tfm_ps_get_support (void)

  *Gets a bitmask with flags set for all of the optional features supported by the implementation.*

## 7.333.1 Function Documentation

### 7.333.1.1 tfm_ps_get()

```
psa_status_t tfm_ps_get (
            int32_t client_id,
            psa_storage_uid_t uid,
            uint32_t data_offset,
            uint32_t data_size,
            size_t * p_data_length )
```

Gets the asset data for the provided uid.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | Unique identifier for the data |
| in | *data_offset* | The offset within the data associated with the `uid` to start retrieving data |

**Parameters**

| | | |
|---|---|---|
| in | *data_size* | The amount of data to read (and the minimum allocated size of the `p_data` buffer) |
| out | *p_data_length* | The pointer to the size of the data retrieved upon success. |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, etc.) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the data associated with the UID was corrupt |
| *PSA_ERROR_INVALID_SIGNATURE* | The operation failed because the data associated with the UID failed authentication |

Definition at line 90 of file tfm_protected_storage.c.
Here is the call graph for this function:

### 7.333.1.2 tfm_ps_get_info()

```
psa_status_t tfm_ps_get_info (
            int32_t client_id,
            psa_storage_uid_t uid,
            struct psa_storage_info_t * p_info )
```

Gets the metadata for the provided uid.

**Parameters**

| | | |
|---|---|---|
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| in | *uid* | Unique identifier for the data |
| out | *p_info* | A pointer to the `psa_storage_info_t` struct that will be populated with the metadata |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, etc.) |
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |
| *PSA_ERROR_DATA_CORRUPT* | The operation failed because the data associated with the UID was corrupt |
| *PSA_ERROR_INVALID_SIGNATURE* | The operation failed because the data associated with the UID failed authentication |

Definition at line 106 of file tfm_protected_storage.c.
Here is the call graph for this function:

#### 7.333.1.3 tfm_ps_get_support()

```
uint32_t tfm_ps_get_support (
            void  )
```

Gets a bitmask with flags set for all of the optional features supported by the implementation.

**Returns**

Bitmask value which contains all the bits set for all the optional features supported by the implementation

Definition at line 141 of file tfm_protected_storage.c.

#### 7.333.1.4 tfm_ps_init()

```
psa_status_t tfm_ps_init (
            void  )
```

Initializes the protected storage system.

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |

**Return values**

| | |
|---|---|
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the storage system initialization has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 23 of file tfm_protected_storage.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.333.1.5 tfm_ps_remove()

```
psa_status_t tfm_ps_remove (
            int32_t client_id,
            psa_storage_uid_t uid )
```

Removes the provided uid and its associated data from storage.

**Parameters**

| | | |
|---|---|---|
| in | *client↩ _id* | Identifier of the asset's owner (client) |
| in | *uid* | Unique identifier for the data to be removed |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, etc.) |

**Return values**

| | |
|---|---|
| *PSA_ERROR_DOES_NOT_EXIST* | The operation failed because the provided uid value was not found in the storage |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure |

Definition at line 118 of file tfm_protected_storage.c.
Here is the call graph for this function:

### 7.333.1.6 tfm_ps_set()

```
psa_status_t tfm_ps_set (
            int32_t client_id,
            psa_storage_uid_t uid,
            uint32_t data_length,
            psa_storage_create_flags_t create_flags )
```

Creates a new or modifies an existing asset.

**Parameters**

| | | |
|---|---|---|
| in | *client_id* | Identifier of the asset's owner (client) |
| in | *uid* | Unique identifier for the data |
| in | *data_length* | The size in bytes of the data in `p_data` |
| in | *create_flags* | The flags indicating the properties of the data |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | The operation completed successfully |
| *PSA_ERROR_NOT_PERMITTED* | The operation failed because the provided uid value was already created with PSA_STORAGE_FLAG_WRITE_ONCE |
| *PSA_ERROR_INVALID_ARGUMENT* | The operation failed because one or more of the given arguments were invalid (null pointer, etc.) |
| *PSA_ERROR_NOT_SUPPORTED* | The operation failed because one or more of the flags provided in `create_flags` is not supported or is not valid |
| *PSA_ERROR_INSUFFICIENT_STORAGE* | The operation failed because there was insufficient space on the storage medium |

**Return values**

| | |
|---|---|
| *PSA_ERROR_STORAGE_FAILURE* | The operation failed because the physical storage has failed (fatal error) |
| *PSA_ERROR_GENERIC_ERROR* | The operation failed because of an unspecified internal failure. |

Definition at line 69 of file tfm_protected_storage.c.
Here is the call graph for this function:

## 7.334 secure_fw/partitions/protected_storage/tfm_ps_req_mngr.c File Reference

```
#include "tfm_ps_req_mngr.h"
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include "psa/protected_storage.h"
#include "tfm_protected_storage.h"
#include "psa/service.h"
#include "psa_manifest/tfm_protected_storage.h"
#include "tfm_ps_defs.h"
```

Include dependency graph for tfm_ps_req_mngr.c:

### Functions

- psa_status_t tfm_protected_storage_service_sfn (const psa_msg_t ∗msg)
- psa_status_t tfm_ps_entry (void)
- psa_status_t ps_req_mngr_read_asset_data (uint8_t ∗out_data, uint32_t size)

  *Writes the asset data of a client iovec onto an output buffer.*
- void ps_req_mngr_write_asset_data (const uint8_t ∗in_data, uint32_t size)

  *Takes an input buffer containing asset data and writes its contents to the client iovec.*

### 7.334.1 Function Documentation

#### 7.334.1.1 ps_req_mngr_read_asset_data()

```
psa_status_t ps_req_mngr_read_asset_data (
            uint8_t * out_data,
            uint32_t size )
```

Writes the asset data of a client iovec onto an output buffer.

**Parameters**

| | | |
|---|---|---|
| out | *out_data* | Pointer to the buffer data will be written to. |
| in | *size* | The amount of data to write. |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

Definition at line 164 of file tfm_ps_req_mngr.c.
Here is the call graph for this function:

#### 7.334.1.2 ps_req_mngr_write_asset_data()

```
void ps_req_mngr_write_asset_data (
            const uint8_t * in_data,
            uint32_t size )
```

Takes an input buffer containing asset data and writes its contents to the client iovec.

**Parameters**

| | | |
|---|---|---|
| in | *in_data* | Pointer to the buffer data will read from. |
| in | *size* | The amount of data to read. |

Definition at line 175 of file tfm_ps_req_mngr.c.
Here is the call graph for this function:

#### 7.334.1.3 tfm_protected_storage_service_sfn()

```
psa_status_t tfm_protected_storage_service_sfn (
            const psa_msg_t * msg )
```

Definition at line 139 of file tfm_ps_req_mngr.c.

#### 7.334.1.4 tfm_ps_entry()

```
psa_status_t tfm_ps_entry (
            void  )
```

Definition at line 159 of file tfm_ps_req_mngr.c.
Here is the call graph for this function:

## 7.335 secure_fw/partitions/protected_storage/tfm_ps_req_mngr.h File Reference

```
#include <stddef.h>
#include "psa/client.h"
#include "psa/protected_storage.h"
```

Include dependency graph for tfm_ps_req_mngr.h:

This graph shows which files directly or indirectly include this file:

## Functions

- void ps_req_mngr_write_asset_data (const uint8_t ∗in_data, uint32_t size)

  *Takes an input buffer containing asset data and writes its contents to the client iovec.*
- psa_status_t ps_req_mngr_read_asset_data (uint8_t ∗out_data, uint32_t size)

  *Writes the asset data of a client iovec onto an output buffer.*

## 7.335.1 Function Documentation

### 7.335.1.1 ps_req_mngr_read_asset_data()

```
psa_status_t ps_req_mngr_read_asset_data (
            uint8_t * out_data,
            uint32_t size )
```

Writes the asset data of a client iovec onto an output buffer.

**Parameters**

| | | |
|---|---|---|
| out | *out_data* | Pointer to the buffer data will be written to. |
| in | *size* | The amount of data to write. |

**Returns**

A status indicating the success/failure of the operation as specified in psa_status_t

Definition at line 164 of file tfm_ps_req_mngr.c.
Here is the call graph for this function:

#### 7.335.1.2 ps_req_mngr_write_asset_data()

```
void ps_req_mngr_write_asset_data (
            const uint8_t * in_data,
            uint32_t size )
```

Takes an input buffer containing asset data and writes its contents to the client iovec.

**Parameters**

| | | |
|---|---|---|
| in | *in_data* | Pointer to the buffer data will read from. |
| in | *size* | The amount of data to read. |

Definition at line 175 of file tfm_ps_req_mngr.c.
Here is the call graph for this function:

## 7.336 secure_fw/shared/crt_memcpy.c File Reference

```
#include "crt_impl_private.h"
#include "static_checks.h"
```

Include dependency graph for crt_memcpy.c:

### Functions

- void ∗ memcpy (void ∗dest, const void ∗src, size_t n)

### 7.336.1 Function Documentation

#### 7.336.1.1 memcpy()

```
void* memcpy (
            void * dest,
            const void * src,
            size_t n )
```

Definition at line 13 of file crt_memcpy.c.
Here is the caller graph for this function:

## 7.337 secure_fw/shared/crt_memset.c File Reference

```
#include "crt_impl_private.h"
```

Include dependency graph for crt_memset.c:

## Functions

- void * memset (void *s, int c, size_t n)

## 7.337.1 Function Documentation

### 7.337.1.1 memset()

```
void* memset (
            void * s,
            int c,
            size_t n )
```

Definition at line 10 of file crt_memset.c.
Here is the caller graph for this function:

## 7.338 secure_fw/spm/core/arch/tfm_arch.c File Reference

```
#include "compiler_ext_defs.h"
#include "security_defs.h"
#include "tfm_arch.h"
#include "tfm_core_trustzone.h"
#include "utilities.h"
#include "config_impl.h"
#include "static_checks.h"
```

Include dependency graph for tfm_arch.c:

### Functions

- __naked void tfm_arch_free_msp_and_exc_ret (uint32_t msp_base, uint32_t exc_return)
- void tfm_arch_init_context (struct context_ctrl_t *p_ctx_ctrl, uintptr_t pfn, void *param, uintptr_t pfnlr)
- uint32_t tfm_arch_refresh_hardware_context (const struct context_ctrl_t *p_ctx_ctrl)

### 7.338.1 Function Documentation

#### 7.338.1.1 tfm_arch_free_msp_and_exc_ret()

```
__naked void tfm_arch_free_msp_and_exc_ret (
            uint32_t msp_base,
            uint32_t exc_return )
```

Definition at line 22 of file tfm_arch.c.

#### 7.338.1.2 tfm_arch_init_context()

```
void tfm_arch_init_context (
            struct context_ctrl_t * p_ctx_ctrl,
            uintptr_t pfn,
            void * param,
            uintptr_t pfnlr )
```

Definition at line 138 of file tfm_arch.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.338.1.3 tfm_arch_refresh_hardware_context()

```
uint32_t tfm_arch_refresh_hardware_context (
            const struct context_ctrl_t * p_ctx_ctrl )
```

Definition at line 173 of file tfm_arch.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.339 secure_fw/spm/core/arch/tfm_arch_v6m_v7m.c File Reference

```
#include <inttypes.h>
#include "compiler_ext_defs.h"
#include "security_defs.h"
#include "utilities.h"
#include "spm.h"
#include "tfm_arch.h"
#include "tfm_hal_device_header.h"
#include "tfm_svcalls.h"
#include "svc_num.h"
```

Include dependency graph for tfm_arch_v6m_v7m.c:

## Functions

- void SVC_Handler (void)
- void tfm_arch_set_secure_exception_priorities (void)
- void tfm_arch_config_extensions (void)

## Variables

- uint32_t psp_limit
- uint32_t scheduler_lock = SCHEDULER_UNLOCKED

## 7.339.1 Function Documentation

### 7.339.1.1 SVC_Handler()

```
void SVC_Handler (
            void  )
```

Definition at line 138 of file tfm_arch_v6m_v7m.c.

### 7.339.1.2 tfm_arch_config_extensions()

```
void tfm_arch_config_extensions (
            void  )
```

Definition at line 246 of file tfm_arch_v6m_v7m.c.

### 7.339.1.3 tfm_arch_set_secure_exception_priorities()

```
void tfm_arch_set_secure_exception_priorities (
            void  )
```

Definition at line 207 of file tfm_arch_v6m_v7m.c.

## 7.339.2 Variable Documentation

### 7.339.2.1 psp_limit

```
uint32_t psp_limit
```

Definition at line 23 of file tfm_arch_v6m_v7m.c.

#### 7.339.2.2 scheduler_lock

```
uint32_t scheduler_lock = SCHEDULER_UNLOCKED
```

Definition at line 26 of file tfm_arch_v6m_v7m.c.

## 7.340 secure_fw/spm/core/arch/tfm_arch_v6m_v7m.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "cmsis_compiler.h"
#include "utilities.h"
```

Include dependency graph for tfm_arch_v6m_v7m.h:

### Macros

- #define EXC_RETURN_SPSEL (1UL << 2)
- #define EXC_RETURN_MODE (1UL << 3)
- #define SCB_ICSR_ADDR (0xE000ED04)
- #define SCB_ICSR_PENDSVSET_BIT (0x10000000)

### Functions

- __STATIC_INLINE bool is_return_secure_stack (uint32_t lr)

  *Check whether Secure or Non-secure stack is used to restore stack frame on exception return.*
- __STATIC_INLINE bool is_default_stacking_rules_apply (uint32_t lr)

  *Check whether the default stacking rules apply, or whether the Additional state context, also known as callee registers, are already on the stack.*
- __STATIC_INLINE uint32_t tfm_arch_get_psplim (void)

  *Get PSP Limit.*
- __STATIC_INLINE void tfm_arch_set_psplim (uint32_t psplim)

  *Set PSP limit value.*
- __STATIC_INLINE void tfm_arch_set_msplim (uint32_t msplim)

  *Set MSP limit value.*
- __STATIC_INLINE uintptr_t arch_seal_thread_stack (uintptr_t stk)

  *Seal the thread stack.*
- __STATIC_INLINE void tfm_arch_check_msp_sealing (void)

  *Check MSP sealing.*
- __STATIC_INLINE void arch_update_process_sp (uint32_t bottom, uint32_t toplimit)

## Variables

- uint32_t psp_limit

### 7.340.1 Macro Definition Documentation

#### 7.340.1.1 EXC_RETURN_MODE

```
#define EXC_RETURN_MODE (1UL << 3)
```

Definition at line 32 of file tfm_arch_v6m_v7m.h.

#### 7.340.1.2 EXC_RETURN_SPSEL

```
#define EXC_RETURN_SPSEL (1UL << 2)
```

Definition at line 30 of file tfm_arch_v6m_v7m.h.

#### 7.340.1.3 SCB_ICSR_ADDR

```
#define SCB_ICSR_ADDR (0xE000ED04)
```

Definition at line 34 of file tfm_arch_v6m_v7m.h.

#### 7.340.1.4 SCB_ICSR_PENDSVSET_BIT

```
#define SCB_ICSR_PENDSVSET_BIT (0x10000000)
```

Definition at line 35 of file tfm_arch_v6m_v7m.h.

### 7.340.2 Function Documentation

#### 7.340.2.1 arch_seal_thread_stack()

```
__STATIC_INLINE uintptr_t arch_seal_thread_stack (
            uintptr_t stk )
```

Seal the thread stack.

**Parameters**

| | | |
|---|---|---|
| in | *stk* | Thread stack address. |

**Return values**

| | |
|---|---|
| *stack* | Updated thread stack address. |

Definition at line 142 of file tfm_arch_v6m_v7m.h.

Here is the caller graph for this function:

#### 7.340.2.2 arch_update_process_sp()

```
__STATIC_INLINE void arch_update_process_sp (
            uint32_t bottom,
            uint32_t toplimit )
```

Definition at line 160 of file tfm_arch_v6m_v7m.h.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.340.2.3 is_default_stacking_rules_apply()

```
__STATIC_INLINE bool is_default_stacking_rules_apply (
            uint32_t lr )
```

Check whether the default stacking rules apply, or whether the Additional state context, also known as callee registers, are already on the stack.

**Parameters**

| | | |
|---|---|---|
| in | *lr* | LR register containing the EXC_RETURN value. |

**Return values**

| | |
|---|---|
| *true* | Always use default stacking rules on v6m/v7m architectures. |

Definition at line 63 of file tfm_arch_v6m_v7m.h.
Here is the caller graph for this function:

#### 7.340.2.4 is_return_secure_stack()

```
__STATIC_INLINE bool is_return_secure_stack (
            uint32_t lr )
```

Check whether Secure or Non-secure stack is used to restore stack frame on exception return.

**Parameters**

| | | |
|---|---|---|
| in | *lr* | LR register containing the EXC_RETURN value. |

**Return values**

| | |
|---|---|
| *true* | Always return to Secure stack on secure core in multi-core topology. |

Definition at line 46 of file tfm_arch_v6m_v7m.h.
Here is the caller graph for this function:

#### 7.340.2.5 tfm_arch_check_msp_sealing()

```
__STATIC_INLINE void tfm_arch_check_msp_sealing (
            void )
```

Check MSP sealing.

Definition at line 151 of file tfm_arch_v6m_v7m.h.

#### 7.340.2.6 tfm_arch_get_psplim()

```
__STATIC_INLINE uint32_t tfm_arch_get_psplim (
            void  )
```

Get PSP Limit.

**Return values**

| | |
|---|---|
| *psp* | limit Limit of PSP stack. |

Definition at line 106 of file tfm_arch_v6m_v7m.h.

#### 7.340.2.7 tfm_arch_set_msplim()

```
__STATIC_INLINE void tfm_arch_set_msplim (
            uint32_t msplim )
```

Set MSP limit value.

**Parameters**

| | | |
|---|---|---|
| in | *msplim* | MSP limit value to be written. |

Definition at line 126 of file tfm_arch_v6m_v7m.h.
Here is the caller graph for this function:

#### 7.340.2.8 tfm_arch_set_psplim()

```
__STATIC_INLINE void tfm_arch_set_psplim (
            uint32_t psplim )
```

Set PSP limit value.

**Parameters**

| | | |
|---|---|---|
| in | *psplim* | PSP limit value to be written. |

Definition at line 116 of file tfm_arch_v6m_v7m.h.

### 7.340.3 Variable Documentation

#### 7.340.3.1 psp_limit

```
uint32_t psp_limit
```

Definition at line 23 of file tfm_arch_v6m_v7m.c.

## 7.341 secure_fw/spm/core/arch/tfm_arch_v8m_base.c File Reference

```
#include <inttypes.h>
#include "compiler_ext_defs.h"
#include "config_spm.h"
#include "security_defs.h"
#include "spm.h"
#include "svc_num.h"
#include "tfm_hal_device_header.h"
#include "tfm_arch.h"
#include "tfm_svcalls.h"
#include "utilities.h"
```

Include dependency graph for tfm_arch_v8m_base.c:

### Functions

- void SVC_Handler (void)
- void tfm_arch_set_secure_exception_priorities (void)
- void tfm_arch_config_extensions (void)

### Variables

- uint32_t scheduler_lock = SCHEDULER_UNLOCKED

### 7.341.1 Function Documentation

#### 7.341.1.1 SVC_Handler()

```
void SVC_Handler (
            void  )
```

Definition at line 172 of file tfm_arch_v8m_base.c.

#### 7.341.1.2 tfm_arch_config_extensions()

```
void tfm_arch_config_extensions (
            void  )
```

Definition at line 297 of file tfm_arch_v8m_base.c.

#### 7.341.1.3 tfm_arch_set_secure_exception_priorities()

```
void tfm_arch_set_secure_exception_priorities (
            void  )
```

Definition at line 252 of file tfm_arch_v8m_base.c.

### 7.341.2 Variable Documentation

#### 7.341.2.1 scheduler_lock

```
uint32_t scheduler_lock = SCHEDULER_UNLOCKED
```

Definition at line 27 of file tfm_arch_v8m_base.c.

## 7.342 secure_fw/spm/core/arch/tfm_arch_v8m_main.c File Reference

```
#include <inttypes.h>
#include "compiler_ext_defs.h"
#include "config_spm.h"
#include "fih.h"
#include "security_defs.h"
#include "region_defs.h"
#include "spm.h"
#include "svc_num.h"
#include "tfm_arch.h"
#include "tfm_hal_device_header.h"
#include "tfm_svcalls.h"
#include "utilities.h"
#include "core_ext.h"
#include "ffm/backend.h"
```

Include dependency graph for tfm_arch_v8m_main.c:

### Functions

- void SVC_Handler (void)
- void tfm_arch_set_secure_exception_priorities (void)
- void tfm_arch_config_extensions (void)

### Variables

- uint32_t scheduler_lock = SCHEDULER_UNLOCKED

### 7.342.1 Function Documentation

#### 7.342.1.1 SVC_Handler()

```
void SVC_Handler (
            void  )
```

Definition at line 179 of file tfm_arch_v8m_main.c.

#### 7.342.1.2 tfm_arch_config_extensions()

```
void tfm_arch_config_extensions (
            void  )
```

Definition at line 333 of file tfm_arch_v8m_main.c.
Here is the call graph for this function:

#### 7.342.1.3 tfm_arch_set_secure_exception_priorities()

```
void tfm_arch_set_secure_exception_priorities (
            void  )
```

Definition at line 270 of file tfm_arch_v8m_main.c.

### 7.342.2 Variable Documentation

#### 7.342.2.1 scheduler_lock

```
uint32_t scheduler_lock = SCHEDULER_UNLOCKED
```

Definition at line 35 of file tfm_arch_v8m_main.c.

## 7.343 secure_fw/spm/core/backend_ipc.c File Reference

```
#include <stdint.h>
#include "aapcs_local.h"
#include "async.h"
#include "config_spm.h"
#include "critical_section.h"
#include "compiler_ext_defs.h"
#include "ffm/psa_api.h"
#include "fih.h"
#include "runtime_defs.h"
#include "stack_watermark.h"
#include "spm.h"
#include "tfm_hal_isolation.h"
```

```
#include "tfm_hal_platform.h"
#include "tfm_nspm.h"
#include "ffm/backend.h"
#include "utilities.h"
#include "memory_symbols.h"
#include "load/partition_defs.h"
#include "load/service_defs.h"
#include "load/spm_load_api.h"
#include "psa/error.h"
#include "internal_status_code.h"
#include "sprt_partition_metadata_indicator.h"
#include "static_checks.h"
```

Include dependency graph for backend_ipc.c:

## Typedefs

- typedef thrd_fn_t(∗ comp_init_fn_t) (struct partition_t ∗, uint32_t, uint32_t ∗)

## Functions

- ARCH_CLAIM_CTXCTRL_INSTANCE (spm_thread_context, spm_thread_stack, sizeof(spm_thread_stack))
- psa_status_t backend_messaging (struct connection_t ∗p_connection)
- psa_status_t backend_replying (struct connection_t ∗handle, int32_t status)
- void common_sfn_thread (void ∗param)
- void backend_init_comp_assuredly (struct partition_t ∗p_pt, uint32_t service_setting)
- uint32_t backend_system_run (void)
- psa_signal_t backend_wait_signals (struct partition_t ∗p_pt, psa_signal_t signals)

  *Set the wait signal pattern in current partition.*
- psa_status_t backend_assert_signal (struct partition_t ∗p_pt, psa_signal_t signal)

  *Set the asserted signal pattern in current partition.*
- uint64_t backend_abi_entering_spm (void)
- uint32_t backend_abi_leaving_spm (uint32_t result)
- uint64_t ipc_schedule (uint32_t exc_return)

## Variables

- struct partition_head_t partition_listhead
- struct context_ctrl_t ∗ p_spm_thread_context = &spm_thread_context
- struct psa_api_tbl_t psa_api_thread_fn_call
- struct psa_api_tbl_t psa_api_svc
- comp_init_fn_t comp_init_fns [ ] = {partition_init, ns_agent_tz_init}

### 7.343.1 Typedef Documentation

#### 7.343.1.1 comp_init_fn_t

```
typedef thrd_fn_t(* comp_init_fn_t) (struct partition_t *, uint32_t, uint32_t *)
```

Definition at line 300 of file backend_ipc.c.

### 7.343.2 Function Documentation

#### 7.343.2.1 ARCH_CLAIM_CTXCTRL_INSTANCE()

```
ARCH_CLAIM_CTXCTRL_INSTANCE (
            spm_thread_context ,
            spm_thread_stack ,
            sizeof(spm_thread_stack)  )
```

#### 7.343.2.2 backend_abi_entering_spm()

```
uint64_t backend_abi_entering_spm (
            void  )
```

Definition at line 405 of file backend_ipc.c.
Here is the call graph for this function:

#### 7.343.2.3 backend_abi_leaving_spm()

```
uint32_t backend_abi_leaving_spm (
            uint32_t result )
```

Definition at line 436 of file backend_ipc.c.
Here is the call graph for this function:

#### 7.343.2.4 backend_assert_signal()

```
psa_status_t backend_assert_signal (
            struct partition_t * p_pt,
            psa_signal_t signal )
```

Set the asserted signal pattern in current partition.
Definition at line 385 of file backend_ipc.c.
Here is the caller graph for this function:

#### 7.343.2.5 backend_init_comp_assuredly()

```
void backend_init_comp_assuredly (
            struct partition_t * p_pt,
            uint32_t service_setting )
```

Definition at line 304 of file backend_ipc.c.

#### 7.343.2.6 backend_messaging()

```
psa_status_t backend_messaging (
            struct connection_t * p_connection )
```

Definition at line 174 of file backend_ipc.c.
Here is the caller graph for this function:

#### 7.343.2.7 backend_replying()

```
psa_status_t backend_replying (
            struct connection_t * handle,
            int32_t status )
```

Definition at line 212 of file backend_ipc.c.

#### 7.343.2.8 backend_system_run()

```
uint32_t backend_system_run (
            void  )
```

Definition at line 329 of file backend_ipc.c.

#### 7.343.2.9 backend_wait_signals()

```
psa_signal_t backend_wait_signals (
            struct partition_t * p_pt,
            psa_signal_t signals )
```

Set the wait signal pattern in current partition.
Definition at line 364 of file backend_ipc.c.
Here is the caller graph for this function:

#### 7.343.2.10 common_sfn_thread()

```
void common_sfn_thread (
            void * param )
```

Definition at line 20 of file sfn_common_thread.c.
Here is the call graph for this function:

#### 7.343.2.11 ipc_schedule()

```
uint64_t ipc_schedule (
            uint32_t exc_return )
```

Definition at line 454 of file backend_ipc.c.

Here is the call graph for this function:

## 7.343.3 Variable Documentation

### 7.343.3.1 comp_init_fns

```
comp_init_fn_t comp_init_fns[] = {partition_init, ns_agent_tz_init}
```

Definition at line 301 of file backend_ipc.c.

### 7.343.3.2 p_spm_thread_context

```
struct context_ctrl_t* p_spm_thread_context = &spm_thread_context
```

Definition at line 53 of file backend_ipc.c.

### 7.343.3.3 partition_listhead

```
struct partition_head_t partition_listhead
```

Definition at line 37 of file backend_ipc.c.

### 7.343.3.4 psa_api_svc

```
struct psa_api_tbl_t psa_api_svc
```

Definition at line 201 of file psa_interface_svc.c.

### 7.343.3.5 psa_api_thread_fn_call

```
struct psa_api_tbl_t psa_api_thread_fn_call
```

Definition at line 246 of file psa_interface_thread_fn_call.c.

## 7.344 secure_fw/spm/core/backend_sfn.c File Reference

```
#include <stdint.h>
#include "compiler_ext_defs.h"
#include "current.h"
#include "runtime_defs.h"
#include "tfm_hal_platform.h"
#include "tfm_nspm.h"
#include "ffm/backend.h"
#include "stack_watermark.h"
#include "load/partition_defs.h"
#include "load/service_defs.h"
#include "load/spm_load_api.h"
#include "psa/error.h"
#include "psa/service.h"
#include "spm.h"
#include "memory_symbols.h"
```

Include dependency graph for backend_sfn.c:

### Macros

- #define SFN_PARTITION_STATE_NOT_INITED 0
- #define SFN_PARTITION_STATE_INITED 1

### Functions

- psa_status_t backend_messaging (struct connection_t *p_connection)
- psa_status_t backend_replying (struct connection_t *handle, int32_t status)
- void backend_init_comp_assuredly (struct partition_t *p_pt, uint32_t service_set)
- uint32_t backend_system_run (void)
- psa_signal_t backend_wait_signals (struct partition_t *p_pt, psa_signal_t signals)

  *Set the wait signal pattern in current partition.*
- psa_status_t backend_assert_signal (struct partition_t *p_pt, psa_signal_t signal)

  *Set the asserted signal pattern in current partition.*

### Variables

- struct partition_head_t partition_listhead
- struct partition_t * p_current_partition

### 7.344.1 Macro Definition Documentation

#### 7.344.1.1 SFN_PARTITION_STATE_INITED

```
#define SFN_PARTITION_STATE_INITED 1
```

Definition at line 29 of file backend_sfn.c.

#### 7.344.1.2 SFN_PARTITION_STATE_NOT_INITED

```
#define SFN_PARTITION_STATE_NOT_INITED 0
```

Definition at line 28 of file backend_sfn.c.

### 7.344.2 Function Documentation

#### 7.344.2.1 backend_assert_signal()

```
psa_status_t backend_assert_signal (
            struct partition_t * p_pt,
            psa_signal_t signal )
```

Set the asserted signal pattern in current partition.
Definition at line 194 of file backend_sfn.c.
Here is the call graph for this function:

#### 7.344.2.2 backend_init_comp_assuredly()

```
void backend_init_comp_assuredly (
            struct partition_t * p_pt,
            uint32_t service_set )
```

Definition at line 126 of file backend_sfn.c.
Here is the call graph for this function:

#### 7.344.2.3 backend_messaging()

```
psa_status_t backend_messaging (
            struct connection_t * p_connection )
```

Definition at line 41 of file backend_sfn.c.
Here is the call graph for this function:

#### 7.344.2.4 backend_replying()

```
psa_status_t backend_replying (
            struct connection_t * handle,
            int32_t status )
```

Definition at line 75 of file backend_sfn.c.

#### 7.344.2.5 backend_system_run()

```
uint32_t backend_system_run (
            void  )
```

Definition at line 149 of file backend_sfn.c.
Here is the call graph for this function:

#### 7.344.2.6 backend_wait_signals()

```
psa_signal_t backend_wait_signals (
            struct partition_t * p_pt,
            psa_signal_t signals )
```

Set the wait signal pattern in current partition.
Definition at line 185 of file backend_sfn.c.
Here is the call graph for this function:

### 7.344.3 Variable Documentation

#### 7.344.3.1 p_current_partition

```
struct partition_t* p_current_partition
```
Definition at line 35 of file backend_sfn.c.

#### 7.344.3.2 partition_listhead

```
struct partition_head_t partition_listhead
```
Definition at line 32 of file backend_sfn.c.

## 7.345 secure_fw/spm/core/internal_status_code.h File Reference

```
#include "psa/error.h"
```
Include dependency graph for internal_status_code.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define SPM_SUCCESS PSA_SUCCESS
- #define SPM_ERROR_BAD_PARAMETERS ((psa_status_t)-249)
- #define SPM_ERROR_SHORT_BUFFER ((psa_status_t)-250)
- #define SPM_ERROR_VERSION ((psa_status_t)-251)
- #define SPM_ERROR_MEMORY_CHECK ((psa_status_t)-252)
- #define SPM_ERROR_GENERIC ((psa_status_t)-253)
- #define STATUS_NEED_SCHEDULE ((psa_status_t)-254)

### 7.345.1 Macro Definition Documentation

#### 7.345.1.1 SPM_ERROR_BAD_PARAMETERS

```
#define SPM_ERROR_BAD_PARAMETERS ((psa_status_t)-249)
```

Definition at line 15 of file internal_status_code.h.

#### 7.345.1.2 SPM_ERROR_GENERIC

```
#define SPM_ERROR_GENERIC ((psa_status_t)-253)
```

Definition at line 19 of file internal_status_code.h.

#### 7.345.1.3 SPM_ERROR_MEMORY_CHECK

```
#define SPM_ERROR_MEMORY_CHECK ((psa_status_t)-252)
```

Definition at line 18 of file internal_status_code.h.

#### 7.345.1.4 SPM_ERROR_SHORT_BUFFER

```
#define SPM_ERROR_SHORT_BUFFER ((psa_status_t)-250)
```

Definition at line 16 of file internal_status_code.h.

#### 7.345.1.5 SPM_ERROR_VERSION

```
#define SPM_ERROR_VERSION ((psa_status_t)-251)
```

Definition at line 17 of file internal_status_code.h.

#### 7.345.1.6 SPM_SUCCESS

```
#define SPM_SUCCESS PSA_SUCCESS
```

Definition at line 12 of file internal_status_code.h.

#### 7.345.1.7 STATUS_NEED_SCHEDULE

```
#define STATUS_NEED_SCHEDULE ((psa_status_t)-254)
```

Definition at line 21 of file internal_status_code.h.

## 7.346 secure_fw/spm/core/interrupt.c File Reference

```
#include "interrupt.h"
#include "bitops.h"
#include "current.h"
#include "fih.h"
#include "svc_num.h"
#include "tfm_arch.h"
#include "tfm_hal_interrupt.h"
#include "tfm_hal_isolation.h"
#include "tfm_svcalls.h"
#include "thread.h"
#include "utilities.h"
#include "load/spm_load_api.h"
#include "ffm/backend.h"
#include "internal_status_code.h"
#include "static_checks.h"
```

Include dependency graph for interrupt.c:

### Functions

- void tfm_flih_func_return (psa_flih_result_t result)
- uint32_t tfm_flih_prepare_depriv_flih (struct partition_t *p_owner_sp, uintptr_t flih_func)
- uint32_t tfm_flih_return_to_isr (psa_flih_result_t result, struct context_flih_ret_t *p_ctx_flih_ret)
- const struct irq_load_info_t * get_irq_info_for_signal (const struct partition_load_info_t *p_ldinf, psa_signal_t signal)

  *Return the IRQ load info context pointer associated with a signal.*
- void spm_handle_interrupt (void *p_pt, const struct irq_load_info_t *p_ildi)

  *Entry of Secure interrupt handler. Platforms can call this function to handle individual interrupts.*

### Variables

- uintptr_t spm_boundary

### 7.346.1 Function Documentation

#### 7.346.1.1 get_irq_info_for_signal()

```
const struct irq_load_info_t* get_irq_info_for_signal (
            const struct partition_load_info_t * p_ldinf,
            psa_signal_t signal )
```

Return the IRQ load info context pointer associated with a signal.

**Parameters**

| | | |
|---|---|---|
| in | *p_ldinf* | The load info of the partition in which we look for the signal. |
| in | *signal* | The signal to query for. |

**Return values**

| | |
|---|---|
| *NULL* | if one of more the following are true:<br>• the signal indicates more than one signal<br>• the signal does not belong to the partition. |
| *Any* | other value The load info pointer associated with the signal |

Definition at line 189 of file interrupt.c.
Here is the caller graph for this function:

#### 7.346.1.2 spm_handle_interrupt()

```
void spm_handle_interrupt (
            void * p_pt,
            const struct irq_load_info_t * p_ildi )
```

Entry of Secure interrupt handler. Platforms can call this function to handle individual interrupts.

**Parameters**

| | | |
|---|---|---|
| in | *p_pt* | The owner Partition of the interrupt to handle |
| in | *p_ildi* | The irq_load_info_t struct of the interrupt to handle |

Note: The input parameters are maintained by platforms and they must be init-ed in the interrupt init functions.
Definition at line 210 of file interrupt.c.

Here is the call graph for this function:

#### 7.346.1.3 tfm_flih_func_return()

```
void tfm_flih_func_return (
            psa_flih_result_t result )
```

Definition at line 28 of file service_api.c.
Here is the caller graph for this function:

#### 7.346.1.4 tfm_flih_prepare_depriv_flih()

```
uint32_t tfm_flih_prepare_depriv_flih (
            struct partition_t * p_owner_sp,
            uintptr_t flih_func )
```

Definition at line 43 of file interrupt.c.
Here is the call graph for this function:

#### 7.346.1.5 tfm_flih_return_to_isr()

```
uint32_t tfm_flih_return_to_isr (
            psa_flih_result_t result,
            struct context_flih_ret_t * p_ctx_flih_ret )
```

Definition at line 120 of file interrupt.c.
Here is the call graph for this function:

### 7.346.2 Variable Documentation

#### 7.346.2.1 spm_boundary

```
uintptr_t spm_boundary
```

Definition at line 29 of file main.c.

## 7.347 secure_fw/spm/core/interrupt.h File Reference

```
#include "spm.h"
#include "load/interrupt_defs.h"
#include "load/partition_defs.h"
#include "psa/service.h"
```

Include dependency graph for interrupt.h:

This graph shows which files directly or indirectly include this file:

## Functions

- const struct irq_load_info_t ∗ get_irq_info_for_signal (const struct partition_load_info_t ∗p_ldinf, psa_signal_t signal)

  *Return the IRQ load info context pointer associated with a signal.*
- void spm_handle_interrupt (void ∗p_pt, const struct irq_load_info_t ∗p_ildi)

  *Entry of Secure interrupt handler. Platforms can call this function to handle individual interrupts.*
- uint32_t tfm_flih_prepare_depriv_flih (struct partition_t ∗p_owner_sp, uintptr_t flih_func)
- uint32_t tfm_flih_return_to_isr (psa_flih_result_t result, struct context_flih_ret_t ∗p_ctx_flih_ret)

## 7.347.1 Function Documentation

### 7.347.1.1 get_irq_info_for_signal()

```
const struct irq_load_info_t* get_irq_info_for_signal (
            const struct partition_load_info_t * p_ldinf,
            psa_signal_t signal )
```

Return the IRQ load info context pointer associated with a signal.

**Parameters**

| | | |
|---|---|---|
| in | *p_ldinf* | The load info of the partition in which we look for the signal. |
| in | *signal* | The signal to query for. |

**Return values**

| | |
|---|---|
| *NULL* | if one of more the following are true:<br>• the signal indicates more than one signal<br>• the signal does not belong to the partition. |
| *Any* | other value The load info pointer associated with the signal |

Definition at line 189 of file interrupt.c.

Here is the caller graph for this function:

#### 7.347.1.2 spm_handle_interrupt()

```
void spm_handle_interrupt (
            void * p_pt,
            const struct irq_load_info_t * p_ildi )
```

Entry of Secure interrupt handler. Platforms can call this function to handle individual interrupts.

**Parameters**

| | | |
|---|---|---|
| in | *p_pt* | The owner Partition of the interrupt to handle |
| in | *p_ildi* | The irq_load_info_t struct of the interrupt to handle |

Note: The input parameters are maintained by platforms and they must be init-ed in the interrupt init functions.
Definition at line 210 of file interrupt.c.
Here is the call graph for this function:

#### 7.347.1.3 tfm_flih_prepare_depriv_flih()

```
uint32_t tfm_flih_prepare_depriv_flih (
            struct partition_t * p_owner_sp,
            uintptr_t flih_func )
```

Definition at line 43 of file interrupt.c.

Here is the call graph for this function:

#### 7.347.1.4 tfm_flih_return_to_isr()

```
uint32_t tfm_flih_return_to_isr (
            psa_flih_result_t result,
            struct context_flih_ret_t * p_ctx_flih_ret )
```

Definition at line 120 of file interrupt.c.
Here is the call graph for this function:

## 7.348 secure_fw/spm/core/mailbox_agent_api.c File Reference

```
#include "current.h"
#include "fih.h"
#include "internal_status_code.h"
#include "spm.h"
#include "tfm_hal_isolation.h"
#include "tfm_multi_core.h"
#include "ffm/mailbox_agent_api.h"
#include "ffm/backend.h"
#include "ffm/psa_api.h"
#include "psa/error.h"
```

Include dependency graph for mailbox_agent_api.c:

## Functions

- psa_status_t tfm_spm_agent_psa_call (psa_handle_t handle, uint32_t control, const struct client_params_t ∗params, const void ∗client_data_stateless)

### 7.348.1 Function Documentation

#### 7.348.1.1 tfm_spm_agent_psa_call()

```
psa_status_t tfm_spm_agent_psa_call (
            psa_handle_t handle,
            uint32_t control,
            const struct client_params_t * params,
            const void * client_data_stateless )
```

Definition at line 21 of file mailbox_agent_api.c.

## 7.349 secure_fw/spm/core/main.c File Reference

```
#include "build_config_check.h"
#include "internal_status_code.h"
#include "fih.h"
#include "tfm_boot_data.h"
#include "memory_symbols.h"
#include "spm.h"
#include "static_checks.h"
#include "tfm_hal_isolation.h"
#include "tfm_hal_platform.h"
#include "tfm_spm_log.h"
#include "tfm_version.h"
#include "tfm_plat_otp.h"
#include "tfm_plat_provisioning.h"
#include "ffm/backend.h"
```

Include dependency graph for main.c:

## Functions

- int main (void)

## Variables

- uintptr_t spm_boundary = (uintptr_t)NULL

### 7.349.1 Function Documentation

#### 7.349.1.1 main()

```
int main (
            void  )
```

Definition at line 99 of file main.c.
Here is the call graph for this function:

### 7.349.2 Variable Documentation

#### 7.349.2.1 spm_boundary

```
uintptr_t spm_boundary = (uintptr_t)NULL
```

Definition at line 29 of file main.c.

## 7.350 secure_fw/spm/core/memory_symbols.h File Reference

```
#include "region.h"
```

Include dependency graph for memory_symbols.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define SPM_BOOT_STACK_TOP (uint32_t)&REGION_NAME(Image$$, ARM_LIB_STACK, $$ZI$$Base)
- #define SPM_BOOT_STACK_BOTTOM (uint32_t)&REGION_NAME(Image$$, ARM_LIB_STACK, $$ZI$$↩ Limit)
- #define PART_INFOLIST_START (uintptr_t)&REGION_NAME(Image$$, TFM_SP_LOAD_LIST, $$RO$$↩ Base)
- #define PART_INFOLIST_END (uintptr_t)&REGION_NAME(Image$$, TFM_SP_LOAD_LIST, $$RO$$Limit)
- #define PART_INFORAM_START (uintptr_t)&REGION_NAME(Image$$, ER_PART_RT_POOL, $$ZI$$↩ Base)
- #define PART_INFORAM_END (uintptr_t)&REGION_NAME(Image$$, ER_PART_RT_POOL, $$ZI$$Limit)
- #define SERV_INFORAM_START (uintptr_t)&REGION_NAME(Image$$, ER_SERV_RT_POOL, $$ZI$$↩ Base)
- #define SERV_INFORAM_END (uintptr_t)&REGION_NAME(Image$$, ER_SERV_RT_POOL, $$ZI$$Limit)

## Functions

- REGION_DECLARE (Image$$, ARM_LIB_STACK, $ $ZI$ $Base)
- REGION_DECLARE (Image$$, TFM_SP_LOAD_LIST, $ $RO$ $Base)
- REGION_DECLARE (Image$$, ER_PART_RT_POOL, $ $ZI$ $Base)
- REGION_DECLARE (Image$$, ER_SERV_RT_POOL, $ $ZI$ $Base)

### 7.350.1 Macro Definition Documentation

#### 7.350.1.1 PART_INFOLIST_END

```
#define PART_INFOLIST_END (uintptr_t)&REGION_NAME(Image$$, TFM_SP_LOAD_LIST, $$RO$$Limit)
```

Definition at line 44 of file memory_symbols.h.

#### 7.350.1.2 PART_INFOLIST_START

```
#define PART_INFOLIST_START (uintptr_t)&REGION_NAME(Image$$, TFM_SP_LOAD_LIST, $$RO$$Base)
```

Definition at line 42 of file memory_symbols.h.

#### 7.350.1.3 PART_INFORAM_END

```
#define PART_INFORAM_END (uintptr_t)&REGION_NAME(Image$$, ER_PART_RT_POOL, $$ZI$$Limit)
```

Definition at line 48 of file memory_symbols.h.

#### 7.350.1.4 PART_INFORAM_START

```
#define PART_INFORAM_START (uintptr_t)&REGION_NAME(Image$$, ER_PART_RT_POOL, $$ZI$$Base)
```

Definition at line 46 of file memory_symbols.h.

#### 7.350.1.5 SERV_INFORAM_END

```
#define SERV_INFORAM_END (uintptr_t)&REGION_NAME(Image$$, ER_SERV_RT_POOL, $$ZI$$Limit)
```

Definition at line 52 of file memory_symbols.h.

#### 7.350.1.6 SERV_INFORAM_START

```
#define SERV_INFORAM_START (uintptr_t)&REGION_NAME(Image$$, ER_SERV_RT_POOL, $$ZI$$Base)
```

Definition at line 50 of file memory_symbols.h.

#### 7.350.1.7 SPM_BOOT_STACK_BOTTOM

```
#define SPM_BOOT_STACK_BOTTOM (uint32_t)&REGION_NAME(Image$$, ARM_LIB_STACK, $$ZI$$Limit)
```

Definition at line 27 of file memory_symbols.h.

#### 7.350.1.8 SPM_BOOT_STACK_TOP

```
#define SPM_BOOT_STACK_TOP (uint32_t)&REGION_NAME(Image$$, ARM_LIB_STACK, $$ZI$$Base)
```

Definition at line 25 of file memory_symbols.h.

### 7.350.2 Function Documentation

#### 7.350.2.1 REGION_DECLARE() [1/4]

```
REGION_DECLARE (
            Image$$ ,
            ARM_LIB_STACK ,
             $ $ZI$ )
```

#### 7.350.2.2 REGION_DECLARE() [2/4]

```
REGION_DECLARE (
            Image$$ ,
            ER_PART_RT_POOL ,
             $ $ZI$ )
```

#### 7.350.2.3 REGION_DECLARE() [3/4]

```
REGION_DECLARE (
            Image$$ ,
            ER_SERV_RT_POOL ,
             $ $ZI$ )
```

#### 7.350.2.4 REGION_DECLARE() [4/4]

```
REGION_DECLARE (
            Image$$ ,
            TFM_SP_LOAD_LIST ,
             $ $RO$ )
```

## 7.351 secure_fw/spm/core/ns_agent_mailbox_interface.c File Reference

```
#include "config_tfm.h"
#include "ns_agent_mailbox_defs.h"
#include "ns_agent_mailbox_interface.h"
#include "psa/client.h"
#include "psa_manifest/sid.h"
```

Include dependency graph for ns_agent_mailbox_interface.c:

### Functions

- psa_status_t ns_agent_boot_ns_core (void)

  *Boot the non-secure core.*
- psa_status_t ns_agent_mailbox_enable (void)

  *Enable the mailbox.*
- psa_status_t ns_agent_mailbox_disable (void)

  *Disable the mailbox.*
- psa_status_t ns_agent_ns_core_shutdown (void)

  *Inform ns_agent_mailbox that the NS core has shut down.*

### 7.351.1 Function Documentation

#### 7.351.1.1 ns_agent_boot_ns_core()

```
psa_status_t ns_agent_boot_ns_core (
            void  )
```

Boot the non-secure core.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 16 of file ns_agent_mailbox_interface.c.
Here is the call graph for this function:

#### 7.351.1.2 ns_agent_mailbox_disable()

```
psa_status_t ns_agent_mailbox_disable (
            void  )
```

Disable the mailbox.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 26 of file ns_agent_mailbox_interface.c.
Here is the call graph for this function:

#### 7.351.1.3 ns_agent_mailbox_enable()

```
psa_status_t ns_agent_mailbox_enable (
            void  )
```

Enable the mailbox.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 21 of file ns_agent_mailbox_interface.c.
Here is the call graph for this function:

#### 7.351.1.4 ns_agent_ns_core_shutdown()

```
psa_status_t ns_agent_ns_core_shutdown (
            void  )
```

Inform ns_agent_mailbox that the NS core has shut down.
Before re-enabling the mailbox, ns_agent_boot_ns_core() must be called.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 31 of file ns_agent_mailbox_interface.c.
Here is the call graph for this function:

## 7.352 secure_fw/spm/core/psa_api.c File Reference

```
#include <stdint.h>
#include "async.h"
#include "bitops.h"
#include "config_impl.h"
#include "config_spm.h"
#include "critical_section.h"
#include "internal_status_code.h"
#include "psa/lifecycle.h"
#include "psa/service.h"
#include "spm.h"
#include "tfm_arch.h"
#include "load/partition_defs.h"
#include "load/service_defs.h"
#include "load/interrupt_defs.h"
```

```
#include "utilities.h"
#include "ffm/backend.h"
#include "ffm/psa_api.h"
#include "tfm_hal_platform.h"
#include "tfm_plat_otp.h"
#include "tfm_psa_call_pack.h"
#include "tfm_spm_log.h"
#include "tfm_hal_isolation.h"
```

Include dependency graph for psa_api.c:

## Functions

- void spm_handle_programmer_errors (psa_status_t status)

  *This function handles the specific programmer error cases.*
- uint32_t tfm_spm_get_lifecycle_state (void)

  *This function get the current PSA RoT lifecycle state.*
- psa_status_t tfm_spm_partition_psa_reply (psa_handle_t msg_handle, psa_status_t status)

  *Function body of psa_reply.*
- psa_status_t tfm_spm_partition_psa_panic (void)

  *Function body of psa_panic.*

## 7.352.1 Function Documentation

### 7.352.1.1 spm_handle_programmer_errors()

```
void spm_handle_programmer_errors (
            psa_status_t status )
```

This function handles the specific programmer error cases.

**Parameters**

| | | |
|---|---|---|
| in | *status* | Standard error codes for the SPM. |

**Return values**

| | |
|---|---|
| *void* | Status will not cause SPM panic |
| *SPM panic* | Following programmer errors are triggered by SP:<br>• PSA_ERROR_PROGRAMMER_ERROR<br>• PSA_ERROR_CONNECTION_REFUSED<br>• PSA_ERROR_CONNECTION_BUSY |

Definition at line 34 of file psa_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.352.1.2 tfm_spm_get_lifecycle_state()

```
uint32_t tfm_spm_get_lifecycle_state (
            void  )
```

This function get the current PSA RoT lifecycle state.

**Returns**

state The current security lifecycle state of the PSA RoT. The PSA state and implementation state are encoded as follows:

- state[15:8] – PSA lifecycle state
- state[7:0] – IMPLEMENTATION DEFINED state

Definition at line 44 of file psa_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.352.1.3 tfm_spm_partition_psa_panic()

```
psa_status_t tfm_spm_partition_psa_panic (
            void  )
```

Function body of psa_panic.

**Return values**

| | |
|---|---|
| *Should not return* | |

Definition at line 358 of file psa_api.c.
Here is the caller graph for this function:

#### 7.352.1.4 tfm_spm_partition_psa_reply()

```
psa_status_t tfm_spm_partition_psa_reply (
            psa_handle_t msg_handle,
            psa_status_t status )
```

Function body of psa_reply.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *status* | Message result value to be reported to the client. |

**Return values**

| | |
|---|---|
| *Positive* | integer Success, the connection handle. |
| *PSA_SUCCESS* | Success |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• An invalid status code is specified for the type of message. |

Definition at line 201 of file psa_api.c.

Here is the caller graph for this function:

## 7.353 secure_fw/spm/core/psa_call_api.c File Reference

```
#include "config_impl.h"
#include "critical_section.h"
#include "ffm/backend.h"
#include "ffm/psa_api.h"
#include "tfm_hal_isolation.h"
#include "tfm_hal_multi_core.h"
#include "tfm_psa_call_pack.h"
#include "utilities.h"
#include "static_checks.h"
```

Include dependency graph for psa_call_api.c:

### Functions

- psa_status_t spm_associate_call_params (struct connection_t ∗p_connection, uint32_t ctrl_param, const psa_invec ∗inptr, psa_outvec ∗outptr)
- psa_status_t tfm_spm_client_psa_call (psa_handle_t handle, uint32_t ctrl_param, const psa_invec ∗inptr, psa_outvec ∗outptr)

  *handler for psa_call.*

### 7.353.1 Function Documentation

#### 7.353.1.1 spm_associate_call_params()

```
psa_status_t spm_associate_call_params (
            struct connection_t * p_connection,
            uint32_t ctrl_param,
            const psa_invec * inptr,
            psa_outvec * outptr )
```

Definition at line 21 of file psa_call_api.c.

Here is the call graph for this function:

#### 7.353.1.2 tfm_spm_client_psa_call()

```
psa_status_t tfm_spm_client_psa_call (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * inptr,
            psa_outvec * outptr )
```

handler for psa_call.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Service handle to the established connection, psa_handle_t |
| in | *ctrl_param* | Parameters combined in uint32_t, includes request type, in_num and out_num. |
| in | *inptr* | Array of input psa_invec structures. psa_invec |
| in | *outptr* | Array of output psa_outvec structures. psa_outvec |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |
| *Does not return* | The call is invalid, one or more of the following are true:<br>• An invalid handle was passed.<br>• The connection is already handling a request.<br>• An invalid memory reference was provided.<br>• in_num + out_num > PSA_MAX_IOVEC.<br>• The message is unrecognized by the RoT Service or incorrectly formatted. |

Definition at line 162 of file psa_call_api.c.

## 7.354 secure_fw/spm/core/psa_connection_api.c File Reference

```
#include "critical_section.h"
#include "ffm/backend.h"
#include "ffm/psa_api.h"
#include "load/service_defs.h"
#include "spm.h"
```

```
#include "utilities.h"
```

Include dependency graph for psa_connection_api.c:

## Functions

- psa_status_t tfm_spm_client_psa_connect (uint32_t sid, uint32_t version)
- psa_status_t spm_psa_connect_client_id_associated (struct connection_t **p_connection, uint32_t sid, uint32_t version, int32_t client_id)
- psa_status_t tfm_spm_client_psa_close (psa_handle_t handle)
- psa_status_t spm_psa_close_client_id_associated (psa_handle_t handle, int32_t client_id)
- psa_status_t tfm_spm_partition_psa_set_rhandle (psa_handle_t msg_handle, void *rhandle)

## 7.354.1 Function Documentation

### 7.354.1.1 spm_psa_close_client_id_associated()

```
psa_status_t spm_psa_close_client_id_associated (
            psa_handle_t handle,
            int32_t client_id )
```

Definition at line 100 of file psa_connection_api.c.

### 7.354.1.2 spm_psa_connect_client_id_associated()

```
psa_status_t spm_psa_connect_client_id_associated (
            struct connection_t ** p_connection,
            uint32_t sid,
            uint32_t version,
            int32_t client_id )
```

Definition at line 37 of file psa_connection_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.354.1.3 tfm_spm_client_psa_close()

```
psa_status_t tfm_spm_client_psa_close (
            psa_handle_t handle )
```

Definition at line 94 of file psa_connection_api.c.

#### 7.354.1.4 tfm_spm_client_psa_connect()

```
psa_status_t tfm_spm_client_psa_connect (
            uint32_t sid,
            uint32_t version )
```

Definition at line 20 of file psa_connection_api.c.
Here is the call graph for this function:

#### 7.354.1.5 tfm_spm_partition_psa_set_rhandle()

```
psa_status_t tfm_spm_partition_psa_set_rhandle (
            psa_handle_t msg_handle,
            void * rhandle )
```

Definition at line 129 of file psa_connection_api.c.

## 7.355 secure_fw/spm/core/psa_interface_sfn.c File Reference

```
#include <stdint.h>
#include "config_impl.h"
#include "current.h"
```

```
#include "tfm_psa_call_pack.h"
#include "ffm/backend.h"
#include "ffm/psa_api.h"
#include "psa/client.h"
```

Include dependency graph for psa_interface_sfn.c:

## Functions

- uint32_t psa_framework_version (void)

  *Retrieve the version of the PSA Framework API that is implemented.*

- uint32_t psa_version (uint32_t sid)

  *Retrieve the version of an RoT Service or indicate that it is not present on this system.*

- psa_status_t tfm_psa_call_pack (psa_handle_t handle, uint32_t ctrl_param, const psa_invec *in_vec, psa_outvec *out_vec)
- size_t psa_read (psa_handle_t msg_handle, uint32_t invec_idx, void *buffer, size_t num_bytes)

  *Read a message parameter or part of a message parameter from a client input vector.*

- size_t psa_skip (psa_handle_t msg_handle, uint32_t invec_idx, size_t num_bytes)

  *Skip over part of a client input vector.*

- void psa_write (psa_handle_t msg_handle, uint32_t outvec_idx, const void *buffer, size_t num_bytes)

  *Write a message response to a client output vector.*

- void psa_panic (void)

  *Terminate execution within the calling Secure Partition and will not return.*

## 7.355.1 Function Documentation

### 7.355.1.1 psa_framework_version()

```
uint32_t psa_framework_version (
            void  )
```

Retrieve the version of the PSA Framework API that is implemented.

**Returns**

version The version of the PSA Framework implementation that is providing the runtime services to the caller. The major and minor version are encoded as follows:

- version[15:8] – major version number.
- version[7:0] – minor version number.

Definition at line 17 of file psa_interface_sfn.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.355.1.2 psa_panic()

```
void psa_panic (
            void  )
```

Terminate execution within the calling Secure Partition and will not return.

**Return values**

| | |
|---|---|
| *Does not return* | |

Definition at line 99 of file psa_interface_sfn.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.355.1.3 psa_read()

```
size_t psa_read (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            void * buffer,
            size_t num_bytes )
```

Read a message parameter or part of a message parameter from a client input vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of the input vector to read from. Must be less than PSA_MAX_IOVEC. |
| out | *buffer* | Buffer in the Secure Partition to copy the requested data to. |
| in | *num_bytes* | Maximum number of bytes to be read from the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes copied. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a PSA_IPC_CALL message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• the memory reference for buffer is invalid or not writable. |

Definition at line 65 of file psa_interface_sfn.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.355.1.4 psa_skip()

```
size_t psa_skip (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            size_t num_bytes )
```

Skip over part of a client input vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of input vector to skip from. Must be less than PSA_MAX_IOVEC. |
| in | *num_bytes* | Maximum number of bytes to skip in the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes skipped. |
| *0* | There was no remaining data in this input vector. |

**Return values**

| | |
|---|---|
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC. |

Definition at line 77 of file psa_interface_sfn.c.
Here is the call graph for this function:

### 7.355.1.5 psa_version()

```
uint32_t psa_version (
            uint32_t sid )
```

Retrieve the version of an RoT Service or indicate that it is not present on this system.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | ID of the RoT Service to query. |

**Return values**

| | |
|---|---|
| *PSA_VERSION_NONE* | The RoT Service is not implemented, or the caller is not permitted to access the service. |
| > | 0 The version of the implemented RoT Service. |

Definition at line 27 of file psa_interface_sfn.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.355.1.6 psa_write()

```
void psa_write (
            psa_handle_t msg_handle,
            uint32_t outvec_idx,
            const void * buffer,
            size_t num_bytes )
```

Write a message response to a client output vector.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| out | *outvec_idx* | Index of output vector in message to write to. Must be less than PSA_MAX_IOVEC. |
| in | *buffer* | Buffer with the data to write. |
| in | *num_bytes* | Number of bytes to write to the client output vector. |

**Return values**

| | |
|---|---|
| *void* | Success |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• outvec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• The memory reference for buffer is invalid.<br>• The call attempts to write data past the end of the client output vector. |

Definition at line 88 of file psa_interface_sfn.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.355.1.7 tfm_psa_call_pack()

```
psa_status_t tfm_psa_call_pack (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * in_vec,
            psa_outvec * out_vec )
```

Definition at line 37 of file psa_interface_sfn.c.

## 7.356 secure_fw/spm/core/psa_interface_svc.c File Reference

```
#include <stdint.h>
#include "compiler_ext_defs.h"
#include "config_spm.h"
#include "runtime_defs.h"
#include "svc_num.h"
#include "tfm_psa_call_pack.h"
#include "utilities.h"
#include "psa/client.h"
#include "psa/lifecycle.h"
#include "psa/service.h"
```

Include dependency graph for psa_interface_svc.c:

## Functions

- __naked uint32_t psa_framework_version_svc (void)
- __naked uint32_t psa_version_svc (uint32_t sid)
- __naked psa_status_t tfm_psa_call_pack_svc (psa_handle_t handle, uint32_t ctrl_param, const psa_invec *in_vec, psa_outvec *out_vec)
- __naked psa_signal_t psa_wait_svc (psa_signal_t signal_mask, uint32_t timeout)
- __naked psa_status_t psa_get_svc (psa_signal_t signal, psa_msg_t *msg)
- __naked size_t psa_read_svc (psa_handle_t msg_handle, uint32_t invec_idx, void *buffer, size_t num_bytes)
- __naked size_t psa_skip_svc (psa_handle_t msg_handle, uint32_t invec_idx, size_t num_bytes)
- __naked void psa_write_svc (psa_handle_t msg_handle, uint32_t outvec_idx, const void *buffer, size_t num_bytes)
- __naked void psa_reply_svc (psa_handle_t msg_handle, psa_status_t retval)
- __naked void psa_panic_svc (void)
- __naked uint32_t psa_rot_lifecycle_state_svc (void)

## Variables

- const struct psa_api_tbl_t psa_api_svc

## 7.356.1 Function Documentation

### 7.356.1.1 psa_framework_version_svc()

```
__naked uint32_t psa_framework_version_svc (
            void  )
```

Definition at line 22 of file psa_interface_svc.c.

### 7.356.1.2 psa_get_svc()

```
__naked psa_status_t psa_get_svc (
            psa_signal_t signal,
            psa_msg_t * msg )
```

Definition at line 49 of file psa_interface_svc.c.

#### 7.356.1.3 psa_panic_svc()

```
__naked void psa_panic_svc (
            void  )
```

Definition at line 96 of file psa_interface_svc.c.

#### 7.356.1.4 psa_read_svc()

```
__naked size_t psa_read_svc (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            void * buffer,
            size_t num_bytes )
```

Definition at line 55 of file psa_interface_svc.c.

#### 7.356.1.5 psa_reply_svc()

```
__naked void psa_reply_svc (
            psa_handle_t msg_handle,
            psa_status_t retval )
```

Definition at line 76 of file psa_interface_svc.c.

#### 7.356.1.6 psa_rot_lifecycle_state_svc()

```
__naked uint32_t psa_rot_lifecycle_state_svc (
            void  )
```

Definition at line 102 of file psa_interface_svc.c.

#### 7.356.1.7 psa_skip_svc()

```
__naked size_t psa_skip_svc (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            size_t num_bytes )
```

Definition at line 62 of file psa_interface_svc.c.

#### 7.356.1.8 psa_version_svc()

```
__naked uint32_t psa_version_svc (
            uint32_t sid )
```

Definition at line 28 of file psa_interface_svc.c.

#### 7.356.1.9 psa_wait_svc()

```
__naked psa_signal_t psa_wait_svc (
            psa_signal_t signal_mask,
            uint32_t timeout )
```

Definition at line 43 of file psa_interface_svc.c.

#### 7.356.1.10 psa_write_svc()

```
__naked void psa_write_svc (
            psa_handle_t msg_handle,
```

```
            uint32_t outvec_idx,
            const void * buffer,
            size_t num_bytes )
```

Definition at line 69 of file psa_interface_svc.c.

#### 7.356.1.11 tfm_psa_call_pack_svc()

```
__naked psa_status_t tfm_psa_call_pack_svc (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * in_vec,
            psa_outvec * out_vec )
```

Definition at line 34 of file psa_interface_svc.c.

### 7.356.2 Variable Documentation

#### 7.356.2.1 psa_api_svc

```
const struct psa_api_tbl_t psa_api_svc
```

Definition at line 201 of file psa_interface_svc.c.

## 7.357 secure_fw/spm/core/psa_interface_thread_fn_call.c File Reference

```
#include <stdint.h>
#include "compiler_ext_defs.h"
#include "config_spm.h"
#include "ffm/psa_api.h"
#include "spm.h"
#include "svc_num.h"
#include "tfm_psa_call_pack.h"
#include "psa/client.h"
#include "psa/lifecycle.h"
#include "psa/service.h"
#include "runtime_defs.h"
#include "tfm_arch.h"
```

Include dependency graph for psa_interface_thread_fn_call.c:

### Macros

- #define TFM_THREAD_FN_CALL_ENTRY(target_psa_api)

### Functions

- __naked uint32_t psa_framework_version_thread_fn_call (void)
- __naked uint32_t psa_version_thread_fn_call (uint32_t sid)

- __naked psa_status_t tfm_psa_call_pack_thread_fn_call (psa_handle_t handle, uint32_t ctrl_param, const psa_invec *in_vec, psa_outvec *out_vec)
- __naked psa_signal_t psa_wait_thread_fn_call (psa_signal_t signal_mask, uint32_t timeout)
- __naked psa_status_t psa_get_thread_fn_call (psa_signal_t signal, psa_msg_t *msg)
- __naked size_t psa_read_thread_fn_call (psa_handle_t msg_handle, uint32_t invec_idx, void *buffer, size_t num_bytes)
- __naked size_t psa_skip_thread_fn_call (psa_handle_t msg_handle, uint32_t invec_idx, size_t num_bytes)
- __naked void psa_write_thread_fn_call (psa_handle_t msg_handle, uint32_t outvec_idx, const void *buffer, size_t num_bytes)
- __naked void psa_reply_thread_fn_call (psa_handle_t msg_handle, psa_status_t status)
- __naked void psa_panic_thread_fn_call (void)
- __naked uint32_t psa_rot_lifecycle_state_thread_fn_call (void)

## Variables

- const struct psa_api_tbl_t psa_api_thread_fn_call

## 7.357.1 Macro Definition Documentation

### 7.357.1.1 TFM_THREAD_FN_CALL_ENTRY

```
#define TFM_THREAD_FN_CALL_ENTRY(
                target_psa_api )
```

**Value:**

```
    __asm volatile(                                  \
        SYNTAX_UNIFIED                               \
        "push   {r4, lr}                    \n"      \
        "ldr    r4, ="M2S(target_psa_api)" \n"       \
        "mov    r12, r4                     \n"      \
        "bl     tfm_arch_thread_fn_call     \n"      \
        "pop    {r4, pc}                    \n"      \
    )
```

Definition at line 31 of file psa_interface_thread_fn_call.c.

## 7.357.2 Function Documentation

### 7.357.2.1 psa_framework_version_thread_fn_call()

```
__naked uint32_t psa_framework_version_thread_fn_call (
              void  )
```

Definition at line 42 of file psa_interface_thread_fn_call.c.
Here is the call graph for this function:

#### 7.357.2.2 psa_get_thread_fn_call()

```
__naked psa_status_t psa_get_thread_fn_call (
            psa_signal_t signal,
            psa_msg_t * msg )
```

Definition at line 69 of file psa_interface_thread_fn_call.c.

#### 7.357.2.3 psa_panic_thread_fn_call()

```
__naked void psa_panic_thread_fn_call (
            void  )
```

Definition at line 116 of file psa_interface_thread_fn_call.c.
Here is the call graph for this function:

#### 7.357.2.4 psa_read_thread_fn_call()

```
__naked size_t psa_read_thread_fn_call (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            void * buffer,
            size_t num_bytes )
```

Definition at line 75 of file psa_interface_thread_fn_call.c.
Here is the call graph for this function:

#### 7.357.2.5 psa_reply_thread_fn_call()

```
__naked void psa_reply_thread_fn_call (
            psa_handle_t msg_handle,
            psa_status_t status )
```

Definition at line 96 of file psa_interface_thread_fn_call.c.

Here is the call graph for this function:

#### 7.357.2.6 psa_rot_lifecycle_state_thread_fn_call()

```
__naked uint32_t psa_rot_lifecycle_state_thread_fn_call (
            void  )
```

Definition at line 122 of file psa_interface_thread_fn_call.c.
Here is the call graph for this function:

#### 7.357.2.7 psa_skip_thread_fn_call()

```
__naked size_t psa_skip_thread_fn_call (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            size_t num_bytes )
```

Definition at line 82 of file psa_interface_thread_fn_call.c.
Here is the call graph for this function:

#### 7.357.2.8 psa_version_thread_fn_call()

```
__naked uint32_t psa_version_thread_fn_call (
            uint32_t sid )
```

Definition at line 48 of file psa_interface_thread_fn_call.c.

Here is the call graph for this function:

#### 7.357.2.9 psa_wait_thread_fn_call()

```
__naked psa_signal_t psa_wait_thread_fn_call (
            psa_signal_t signal_mask,
            uint32_t timeout )
```

Definition at line 63 of file psa_interface_thread_fn_call.c.

#### 7.357.2.10 psa_write_thread_fn_call()

```
__naked void psa_write_thread_fn_call (
            psa_handle_t msg_handle,
            uint32_t outvec_idx,
            const void * buffer,
            size_t num_bytes )
```

Definition at line 89 of file psa_interface_thread_fn_call.c.
Here is the call graph for this function:

#### 7.357.2.11 tfm_psa_call_pack_thread_fn_call()

```
__naked psa_status_t tfm_psa_call_pack_thread_fn_call (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * in_vec,
            psa_outvec * out_vec )
```

Definition at line 54 of file psa_interface_thread_fn_call.c.

### 7.357.3 Variable Documentation

#### 7.357.3.1 psa_api_thread_fn_call

```
const struct psa_api_tbl_t psa_api_thread_fn_call
```

Definition at line 246 of file psa_interface_thread_fn_call.c.

## 7.358 secure_fw/spm/core/psa_irq_api.c File Reference

```
#include "critical_section.h"
#include "ffm/psa_api.h"
#include "interrupt.h"
#include "spm.h"
#include "tfm_hal_interrupt.h"
#include "utilities.h"
```

Include dependency graph for psa_irq_api.c:

### Functions

- psa_status_t tfm_spm_partition_psa_irq_enable (psa_signal_t irq_signal)
- psa_irq_status_t tfm_spm_partition_psa_irq_disable (psa_signal_t irq_signal)

### 7.358.1 Function Documentation

#### 7.358.1.1 tfm_spm_partition_psa_irq_disable()

```
psa_irq_status_t tfm_spm_partition_psa_irq_disable (
            psa_signal_t irq_signal )
```

Definition at line 35 of file psa_irq_api.c.
Here is the call graph for this function:

#### 7.358.1.2 tfm_spm_partition_psa_irq_enable()

```
psa_status_t tfm_spm_partition_psa_irq_enable (
            psa_signal_t irq_signal )
```

Definition at line 18 of file psa_irq_api.c.

Here is the call graph for this function:

## 7.359 secure_fw/spm/core/psa_mmiovec_api.c File Reference

```
#include "ffm/psa_api.h"
#include "spm.h"
#include "load/service_defs.h"
#include "utilities.h"
#include "tfm_hal_isolation.h"
```

Include dependency graph for psa_mmiovec_api.c:

### Functions

- const void ∗ tfm_spm_partition_psa_map_invec (psa_handle_t msg_handle, uint32_t invec_idx)
- void tfm_spm_partition_psa_unmap_invec (psa_handle_t msg_handle, uint32_t invec_idx)
- void ∗ tfm_spm_partition_psa_map_outvec (psa_handle_t msg_handle, uint32_t outvec_idx)
- void tfm_spm_partition_psa_unmap_outvec (psa_handle_t msg_handle, uint32_t outvec_idx, size_t len)

### 7.359.1 Function Documentation

#### 7.359.1.1 tfm_spm_partition_psa_map_invec()

```
const void* tfm_spm_partition_psa_map_invec (
            psa_handle_t msg_handle,
            uint32_t invec_idx )
```

Definition at line 16 of file psa_mmiovec_api.c.

#### 7.359.1.2 tfm_spm_partition_psa_map_outvec()

```
void* tfm_spm_partition_psa_map_outvec (
            psa_handle_t msg_handle,
            uint32_t outvec_idx )
```

Definition at line 146 of file psa_mmiovec_api.c.

#### 7.359.1.3 tfm_spm_partition_psa_unmap_invec()

```
void tfm_spm_partition_psa_unmap_invec (
            psa_handle_t msg_handle,
            uint32_t invec_idx )
```

Definition at line 92 of file psa_mmiovec_api.c.

#### 7.359.1.4 tfm_spm_partition_psa_unmap_outvec()

```
void tfm_spm_partition_psa_unmap_outvec (
            psa_handle_t msg_handle,
            uint32_t outvec_idx,
            size_t len )
```

Definition at line 220 of file psa_mmiovec_api.c.

## 7.360 secure_fw/spm/core/psa_read_write_skip_api.c File Reference

```
#include "ffm/psa_api.h"
#include "spm.h"
#include "utilities.h"
#include "tfm_hal_isolation.h"
#include "static_checks.h"
#include "tfm_psa_call_pack.h"
```

Include dependency graph for psa_read_write_skip_api.c:

### Functions

- size_t tfm_spm_partition_psa_read (psa_handle_t msg_handle, uint32_t invec_idx, void *buffer, size_t num_bytes)

  *Function body of psa_read.*
- size_t tfm_spm_partition_psa_skip (psa_handle_t msg_handle, uint32_t invec_idx, size_t num_bytes)

  *Function body of psa_skip.*
- psa_status_t tfm_spm_partition_psa_write (psa_handle_t msg_handle, uint32_t outvec_idx, const void *buffer, size_t num_bytes)

  *Function body of psa_write.*

### 7.360.1 Function Documentation

#### 7.360.1.1 tfm_spm_partition_psa_read()

```
size_t tfm_spm_partition_psa_read (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
```

```
            void * buffer,
            size_t num_bytes )
```

Function body of psa_read.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of the input vector to read from. Must be less than PSA_MAX_IOVEC. |
| out | *buffer* | Buffer in the Secure Partition to copy the requested data to. |
| in | *num_bytes* | Maximum number of bytes to be read from the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes copied. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a PSA_IPC_CALL message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• the memory reference for buffer is invalid or not writable. |

Definition at line 17 of file psa_read_write_skip_api.c.
Here is the caller graph for this function:

### 7.360.1.2 tfm_spm_partition_psa_skip()

```
size_t tfm_spm_partition_psa_skip (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            size_t num_bytes )
```

Function body of psa_skip.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of input vector to skip from. Must be less than PSA_MAX_IOVEC. |
| in | *num_bytes* | Maximum number of bytes to skip in the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes skipped. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC. |

Definition at line 88 of file psa_read_write_skip_api.c.
Here is the caller graph for this function:

#### 7.360.1.3 tfm_spm_partition_psa_write()

```
psa_status_t tfm_spm_partition_psa_write (
            psa_handle_t msg_handle,
            uint32_t outvec_idx,
            const void * buffer,
            size_t num_bytes )
```

Function body of psa_write.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| out | *outvec_idx* | Index of output vector in message to write to. Must be less than PSA_MAX_IOVEC. |
| in | *buffer* | Buffer with the data to write. |
| in | *num_bytes* | Number of bytes to write to the client output vector. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |

**Return values**

| | |
|---|---|
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• outvec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• The memory reference for buffer is invalid.<br>• The call attempts to write data past the end of the client output vector. |

Definition at line 148 of file psa_read_write_skip_api.c.
Here is the caller graph for this function:

## 7.361 secure_fw/spm/core/psa_version_api.c File Reference

```
#include "ffm/psa_api.h"
#include "load/service_defs.h"
#include "spm.h"
```

Include dependency graph for psa_version_api.c:

### Functions

- uint32_t tfm_spm_client_psa_framework_version (void)

  *handler for psa_framework_version.*

- uint32_t tfm_spm_client_psa_version (uint32_t sid)

  *handler for psa_version.*

### 7.361.1 Function Documentation

#### 7.361.1.1 tfm_spm_client_psa_framework_version()

```
uint32_t tfm_spm_client_psa_framework_version (
            void  )
```

handler for psa_framework_version.

**Returns**

version The version of the PSA Framework implementation that is providing the runtime services.

Definition at line 15 of file psa_version_api.c.
Here is the caller graph for this function:

#### 7.361.1.2 tfm_spm_client_psa_version()

```
uint32_t tfm_spm_client_psa_version (
            uint32_t sid )
```

handler for psa_version.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | RoT Service identity. |

**Return values**

| | |
|---|---|
| *PSA_VERSION_NONE* | The RoT Service is not implemented, or the caller is not permitted to access the service. |
| > | 0 The version of the implemented RoT Service. |

Definition at line 20 of file psa_version_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.362 secure_fw/spm/core/rom_loader.c File Reference

```
#include <limits.h>
#include <stdint.h>
#include "config_impl.h"
#include "lists.h"
#include "memory_symbols.h"
#include "region_defs.h"
#include "spm.h"
#include "tfm_hal_interrupt.h"
#include "tfm_plat_defs.h"
#include "utilities.h"
#include "ffm/backend.h"
#include "load/partition_defs.h"
#include "load/spm_load_api.h"
#include "load/service_defs.h"
#include "psa/client.h"
```

Include dependency graph for rom_loader.c:

### Functions

- struct partition_t ∗ load_a_partition_assuredly (struct partition_head_t ∗head)
- uint32_t load_services_assuredly (struct partition_t ∗p_partition, struct service_head_t ∗services_listhead, struct service_t ∗∗stateless_services_ref_tbl, size_t ref_tbl_size)
- void load_irqs_assuredly (struct partition_t ∗p_partition)

### 7.362.1 Function Documentation

#### 7.362.1.1 load_a_partition_assuredly()

```
struct partition_t* load_a_partition_assuredly (
            struct partition_head_t * head )
```

Definition at line 63 of file rom_loader.c.

Here is the call graph for this function:

#### 7.362.1.2 load_irqs_assuredly()

```
void load_irqs_assuredly (
            struct partition_t * p_partition )
```

Definition at line 185 of file rom_loader.c.
Here is the call graph for this function:

#### 7.362.1.3 load_services_assuredly()

```
uint32_t load_services_assuredly (
            struct partition_t * p_partition,
            struct service_head_t * services_listhead,
            struct service_t ** stateless_services_ref_tbl,
            size_t ref_tbl_size )
```

Definition at line 131 of file rom_loader.c.

## 7.363 secure_fw/spm/core/spm.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "config_impl.h"
#include "config_spm.h"
#include "current.h"
#include "tfm_arch.h"
#include "lists.h"
#include "runtime_defs.h"
#include "thread.h"
#include "psa/service.h"
#include "load/partition_defs.h"
#include "load/interrupt_defs.h"
```

Include dependency graph for spm.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct connection_t
- struct partition_t
- struct service_t

## Macros

- #define TFM_HANDLE_STATUS_IDLE 0 /∗ Handle created ∗/
- #define TFM_HANDLE_STATUS_ACTIVE 1 /∗ Handle in use ∗/
- #define TFM_HANDLE_STATUS_TO_FREE 2 /∗ Free the handle ∗/
- #define PSA_TIMEOUT_MASK PSA_BLOCK
- #define STATIC_HANDLE_NUM_LIMIT 32
- #define CLIENT_HANDLE_VALUE_MIN 1
- #define STATIC_HANDLE_IDX_BIT_WIDTH 5
- #define STATIC_HANDLE_IDX_MASK (uint32_t)((1UL << STATIC_HANDLE_IDX_BIT_WIDTH) - 1)
- #define GET_INDEX_FROM_STATIC_HANDLE(handle) (uint32_t)((handle) & STATIC_HANDLE_IDX_MASK)
- #define STATIC_HANDLE_VER_BIT_WIDTH 8
- #define STATIC_HANDLE_VER_OFFSET 8
- #define STATIC_HANDLE_VER_MASK (uint32_t)((1UL << STATIC_HANDLE_VER_BIT_WIDTH) - 1)
- #define GET_VERSION_FROM_STATIC_HANDLE(handle) (uint32_t)(((handle) >> STATIC_HANDLE_VER_OFFSET) & STATIC_HANDLE_VER_MASK)
- #define STATIC_HANDLE_INDICATOR_OFFSET 30
- #define IS_STATIC_HANDLE(handle) ((handle) & (1UL << STATIC_HANDLE_INDICATOR_OFFSET))
- #define SPM_INVALID_PARTITION_IDX (∼0U)
- #define GET_THRD_OWNER(x) TO_CONTAINER(x, struct partition_t, thrd)
- #define GET_CTX_OWNER(x) TO_CONTAINER(x, struct partition_t, ctx_ctrl)
- #define TFM_CLIENT_ID_IS_NS(client_id) ((client_id) < 0)
- #define tfm_spm_is_rpc_msg(x) (false)

## Functions

- int32_t tfm_spm_partition_get_running_partition_id (void)

  *Get the running partition ID.*
- void spm_init_connection_space (void)
- struct connection_t ∗ spm_allocate_connection (void)

- psa_status_t spm_validate_connection (const struct connection_t *p_connection)
- void spm_free_connection (struct connection_t *p_connection)
- const struct service_t * tfm_spm_get_service_by_sid (uint32_t sid)

  *Get the service context by service ID.*
- psa_status_t spm_get_idle_connection (struct connection_t **p_connection, psa_handle_t handle, int32_t client_id)

  *Convert the given user handle to an SPM recognised connection and verify that it is a valid idle connection that the caller is authorised to access.*
- struct connection_t * spm_msg_handle_to_connection (psa_handle_t msg_handle)

  *Convert the given message handle to SPM recognised handle and verify it.*
- void spm_init_idle_connection (struct connection_t *p_connection, const struct service_t *service, int32_t client_id)

  *Initialize connection, fill in with the input information and set to idle.*
- psa_status_t spm_associate_call_params (struct connection_t *p_connection, uint32_t ctrl_param, const psa_invec *inptr, psa_outvec *outptr)
- int32_t tfm_spm_check_client_version (const struct service_t *service, uint32_t version)

  *Check the client version according to version policy.*
- int32_t tfm_spm_check_authorization (uint32_t sid, const struct service_t *service, bool ns_caller)

  *Check the client access authorization.*
- bool tfm_spm_is_ns_caller (void)

  *Get the ns_caller info from runtime context.*
- int32_t tfm_spm_get_client_id (bool ns_caller)

  *Get ID of current RoT Service client. This API ensures the caller gets a valid ID.*
- uint64_t ipc_schedule (uint32_t exc_return)
- uint32_t tfm_spm_init (void)

  *SPM initialization implementation.*
- psa_handle_t connection_to_handle (struct connection_t *p_connection)

  *Converts a handle instance into a corresponded user handle.*
- struct connection_t * handle_to_connection (psa_handle_t handle)

  *Converts a user handle into a corresponded handle instance.*
- void update_caller_outvec_len (struct connection_t *handle)

## 7.363.1 Macro Definition Documentation

### 7.363.1.1 CLIENT_HANDLE_VALUE_MIN

```
#define CLIENT_HANDLE_VALUE_MIN 1
```

Definition at line 39 of file spm.h.

### 7.363.1.2 GET_CTX_OWNER

```
#define GET_CTX_OWNER(
            x ) TO_CONTAINER(x, struct partition_t, ctx_ctrl)
```

Definition at line 67 of file spm.h.

### 7.363.1.3 GET_INDEX_FROM_STATIC_HANDLE

```
#define GET_INDEX_FROM_STATIC_HANDLE(
            handle ) (uint32_t)((handle) & STATIC_HANDLE_IDX_MASK)
```

Definition at line 48 of file spm.h.

#### 7.363.1.4 GET_THRD_OWNER

```
#define GET_THRD_OWNER(
              x ) TO_CONTAINER(x, struct partition_t, thrd)
```

Definition at line 66 of file spm.h.

#### 7.363.1.5 GET_VERSION_FROM_STATIC_HANDLE

```
#define GET_VERSION_FROM_STATIC_HANDLE(
              handle ) (uint32_t)(((handle) >> STATIC_HANDLE_VER_OFFSET) & STATIC_HANDLE_VER_MASK)
```

Definition at line 55 of file spm.h.

#### 7.363.1.6 IS_STATIC_HANDLE

```
#define IS_STATIC_HANDLE(
              handle ) ((handle) & (1UL << STATIC_HANDLE_INDICATOR_OFFSET))
```

Definition at line 60 of file spm.h.

#### 7.363.1.7 PSA_TIMEOUT_MASK

```
#define PSA_TIMEOUT_MASK PSA_BLOCK
```

Definition at line 32 of file spm.h.

#### 7.363.1.8 SPM_INVALID_PARTITION_IDX

```
#define SPM_INVALID_PARTITION_IDX (~0U)
```

Definition at line 63 of file spm.h.

#### 7.363.1.9 STATIC_HANDLE_IDX_BIT_WIDTH

```
#define STATIC_HANDLE_IDX_BIT_WIDTH 5
```

Definition at line 45 of file spm.h.

#### 7.363.1.10 STATIC_HANDLE_IDX_MASK

```
#define STATIC_HANDLE_IDX_MASK (uint32_t)((1UL << STATIC_HANDLE_IDX_BIT_WIDTH) - 1)
```

Definition at line 46 of file spm.h.

#### 7.363.1.11 STATIC_HANDLE_INDICATOR_OFFSET

```
#define STATIC_HANDLE_INDICATOR_OFFSET 30
```

Definition at line 59 of file spm.h.

#### 7.363.1.12 STATIC_HANDLE_NUM_LIMIT

```
#define STATIC_HANDLE_NUM_LIMIT 32
```

Definition at line 38 of file spm.h.

#### 7.363.1.13 STATIC_HANDLE_VER_BIT_WIDTH

```
#define STATIC_HANDLE_VER_BIT_WIDTH 8
```

Definition at line 51 of file spm.h.

#### 7.363.1.14 STATIC_HANDLE_VER_MASK

```
#define STATIC_HANDLE_VER_MASK (uint32_t)((1UL << STATIC_HANDLE_VER_BIT_WIDTH) - 1)
```

Definition at line 53 of file spm.h.

#### 7.363.1.15 STATIC_HANDLE_VER_OFFSET

```
#define STATIC_HANDLE_VER_OFFSET 8
```

Definition at line 52 of file spm.h.

#### 7.363.1.16 TFM_CLIENT_ID_IS_NS

```
#define TFM_CLIENT_ID_IS_NS(
            client_id ) ((client_id) < 0)
```

Definition at line 70 of file spm.h.

#### 7.363.1.17 TFM_HANDLE_STATUS_ACTIVE

```
#define TFM_HANDLE_STATUS_ACTIVE 1 /* Handle in use */
```

Definition at line 28 of file spm.h.

#### 7.363.1.18 TFM_HANDLE_STATUS_IDLE

```
#define TFM_HANDLE_STATUS_IDLE 0 /* Handle created */
```

Definition at line 27 of file spm.h.

#### 7.363.1.19 TFM_HANDLE_STATUS_TO_FREE

```
#define TFM_HANDLE_STATUS_TO_FREE 2 /* Free the handle */
```

Definition at line 29 of file spm.h.

#### 7.363.1.20 tfm_spm_is_rpc_msg

```
#define tfm_spm_is_rpc_msg(
            x ) (false)
```

Definition at line 404 of file spm.h.

### 7.363.2 Function Documentation

#### 7.363.2.1 connection_to_handle()

```
psa_handle_t connection_to_handle (
            struct connection_t * p_connection )
```

Converts a handle instance into a corresponded user handle.

Definition at line 63 of file spm_connection_pool.c.

Here is the caller graph for this function:

#### 7.363.2.2 handle_to_connection()

```
struct connection_t* handle_to_connection (
            psa_handle_t handle )
```

Converts a user handle into a corresponded handle instance.
Definition at line 91 of file spm_connection_pool.c.
Here is the caller graph for this function:

#### 7.363.2.3 ipc_schedule()

```
uint64_t ipc_schedule (
            uint32_t exc_return )
```

Definition at line 454 of file backend_ipc.c.
Here is the call graph for this function:

#### 7.363.2.4 spm_allocate_connection()

```
struct connection_t* spm_allocate_connection (
            void  )
```

Definition at line 117 of file spm_connection_pool.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.363.2.5 spm_associate_call_params()

```
psa_status_t spm_associate_call_params (
            struct connection_t * p_connection,
            uint32_t ctrl_param,
            const psa_invec * inptr,
            psa_outvec * outptr )
```

Definition at line 21 of file psa_call_api.c.
Here is the call graph for this function:

#### 7.363.2.6 spm_free_connection()

```
void spm_free_connection (
            struct connection_t * p_connection )
```

Definition at line 135 of file spm_connection_pool.c.
Here is the call graph for this function:

#### 7.363.2.7 spm_get_idle_connection()

```
psa_status_t spm_get_idle_connection (
            struct connection_t ** p_connection,
            psa_handle_t handle,
            int32_t client_id )
```

Convert the given user handle to an SPM recognised connection and verify that it is a valid idle connection that the caller is authorised to access.

**Parameters**

| | | |
|---|---|---|
| out | *p_connection* | The address of connection pointer to be converted from the given user handle. |
| in | *handle* | Either a static handle or a handle to an established connection that was returned by a prior psa_connect call. |
| in | *client_id* | The client ID of the caller. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |
| *PSA_ERROR_CONNECTION_REFUSED* | The SPM or RoT Service has refused the connection. |
| *PSA_ERROR_CONNECTION_BUSY* | The SPM or RoT Service cannot make the connection at the moment. |
| *PSA_ERROR_PROGRAMMER_ERROR* | The handle is invalid, the caller is not authorised to use it or the connection is already handling a request. |

Definition at line 181 of file spm_ipc.c.

#### 7.363.2.8 spm_init_connection_space()

```
void spm_init_connection_space (
            void )
```

Definition at line 107 of file spm_connection_pool.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.363.2.9 spm_init_idle_connection()

```
void spm_init_idle_connection (
            struct connection_t * p_connection,
            const struct service_t * service,
            int32_t client_id )
```

Initialize connection, fill in with the input information and set to idle.

**Parameters**

| | | |
|---|---|---|
| in | *p_connection* | The 'p_connection' to initialize and fill information in. |
| in | *service* | Target service context pointer, which can be obtained by partition management functions |
| in | *client_id* | Partition ID of the sender of the message |

Definition at line 296 of file spm_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.363.2.10 spm_msg_handle_to_connection()

```
struct connection_t* spm_msg_handle_to_connection (
            psa_handle_t msg_handle )
```

Convert the given message handle to SPM recognised handle and verify it.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Message handle which is a reference generated by the SPM to a specific message. |

**Returns**

A SPM recognised handle or NULL. It is NULL when verification of the converted SPM handle fails. connection_t structures

Definition at line 269 of file spm_ipc.c.
Here is the call graph for this function:

#### 7.363.2.11 spm_validate_connection()

```
psa_status_t spm_validate_connection (
            const struct connection_t * p_connection )
```

Definition at line 124 of file spm_connection_pool.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.363.2.12 tfm_spm_check_authorization()

```
int32_t tfm_spm_check_authorization (
            uint32_t sid,
            const struct service_t * service,
            bool ns_caller )
```

Check the client access authorization.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | Target RoT Service identity |
| in | *service* | Target service context pointer, which can be get by partition management functions |
| in | *ns_caller* | Whether from NS caller |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success |
| *SPM_ERROR_GENERIC* | Authorization check failed |

Definition at line 146 of file spm_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.363.2.13 tfm_spm_check_client_version()

```
int32_t tfm_spm_check_client_version (
            const struct service_t * service,
            uint32_t version )
```

Check the client version according to version policy.

**Parameters**

| | | |
|---|---|---|
| in | *service* | Target service context pointer, which can be get by partition management functions |
| in | *version* | Client support version |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success |
| *SPM_ERROR_BAD_PARAMETERS* | Bad parameters input |
| *SPM_ERROR_VERSION* | Check failed |

Definition at line 124 of file spm_ipc.c.
Here is the caller graph for this function:

#### 7.363.2.14 tfm_spm_get_client_id()

```
int32_t tfm_spm_get_client_id (
            bool ns_caller )
```

Get ID of current RoT Service client. This API ensures the caller gets a valid ID.

**Parameters**

| | | |
|---|---|---|
| in | *ns_caller* | If the client is Non-Secure or not. |

**Return values**

| | |
|---|---|
| *The* | client ID |

Definition at line 345 of file spm_ipc.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.363.2.15 tfm_spm_get_service_by_sid()

```
const struct service_t* tfm_spm_get_service_by_sid (
            uint32_t sid )
```

Get the service context by service ID.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | RoT Service identity |

**Return values**

| | |
|---|---|
| *NULL* | Failed |
| *Not NULL* | Target service context pointer, service_t structures |

Definition at line 86 of file spm_ipc.c.
Here is the caller graph for this function:

#### 7.363.2.16 tfm_spm_init()

```
uint32_t tfm_spm_init (
            void  )
```

SPM initialization implementation.
This function must be called under handler mode.

**Return values**

| | |
|---|---|
| *This* | function returns an EXC_RETURN value. Other faults would panic the execution and never returned. |

Definition at line 363 of file spm_ipc.c.
Here is the call graph for this function:

#### 7.363.2.17 tfm_spm_is_ns_caller()

```
bool tfm_spm_is_ns_caller (
            void  )
```

Get the ns_caller info from runtime context.

**Return values**

| | |
|---|---|
| - | true: the PSA API caller is from non-secure<br>• false: the PSA API caller is from secure |

Definition at line 334 of file spm_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.363.2.18 tfm_spm_partition_get_running_partition_id()

```
int32_t tfm_spm_partition_get_running_partition_id (
            void  )
```

Get the running partition ID.

**Returns**

Returns the partition ID

Definition at line 322 of file spm_ipc.c.
Here is the caller graph for this function:

#### 7.363.2.19 update_caller_outvec_len()

```
void update_caller_outvec_len (
            struct connection_t * handle )
```

Definition at line 415 of file spm_ipc.c.

## 7.364 secure_fw/spm/core/spm_connection_pool.c File Reference

```
#include "critical_section.h"
#include "internal_status_code.h"
#include "spm.h"
#include "static_checks.h"
#include "tfm_pools.h"
#include "load/service_defs.h"
```

Include dependency graph for spm_connection_pool.c:

## Macros

- #define CONVERSION_FACTOR_BITOFFSET 3
- #define CONVERSION_FACTOR_VALUE (1 << CONVERSION_FACTOR_BITOFFSET)
- #define CONVERSION_FACTOR_VALUE_MAX 0x20

## Functions

- psa_handle_t connection_to_handle (struct connection_t *p_connection)

  *Converts a handle instance into a corresponded user handle.*
- struct connection_t * handle_to_connection (psa_handle_t handle)

  *Converts a user handle into a corresponded handle instance.*
- void spm_init_connection_space (void)
- struct connection_t * spm_allocate_connection (void)
- psa_status_t spm_validate_connection (const struct connection_t *p_connection)
- void spm_free_connection (struct connection_t *p_connection)

## 7.364.1 Macro Definition Documentation

### 7.364.1.1 CONVERSION_FACTOR_BITOFFSET

```
#define CONVERSION_FACTOR_BITOFFSET 3
```

Definition at line 27 of file spm_connection_pool.c.

### 7.364.1.2 CONVERSION_FACTOR_VALUE

```
#define CONVERSION_FACTOR_VALUE (1 << CONVERSION_FACTOR_BITOFFSET)
```

Definition at line 28 of file spm_connection_pool.c.

### 7.364.1.3 CONVERSION_FACTOR_VALUE_MAX

```
#define CONVERSION_FACTOR_VALUE_MAX 0x20
```

Definition at line 30 of file spm_connection_pool.c.

## 7.364.2 Function Documentation

#### 7.364.2.1 connection_to_handle()

```
psa_handle_t connection_to_handle (
            struct connection_t * p_connection )
```

Converts a handle instance into a corresponded user handle.
Definition at line 63 of file spm_connection_pool.c.

#### 7.364.2.2 handle_to_connection()

```
struct connection_t* handle_to_connection (
            psa_handle_t handle )
```

Converts a user handle into a corresponded handle instance.
Definition at line 91 of file spm_connection_pool.c.

#### 7.364.2.3 spm_allocate_connection()

```
struct connection_t* spm_allocate_connection (
            void  )
```

Definition at line 117 of file spm_connection_pool.c.
Here is the call graph for this function:

#### 7.364.2.4 spm_free_connection()

```
void spm_free_connection (
            struct connection_t * p_connection )
```

Definition at line 135 of file spm_connection_pool.c.
Here is the call graph for this function:

#### 7.364.2.5 spm_init_connection_space()

```
void spm_init_connection_space (
            void  )
```

Definition at line 107 of file spm_connection_pool.c.
Here is the call graph for this function:

#### 7.364.2.6 spm_validate_connection()

```
psa_status_t spm_validate_connection (
            const struct connection_t * p_connection )
```

Definition at line 124 of file spm_connection_pool.c.
Here is the call graph for this function:

## 7.365 secure_fw/spm/core/spm_ipc.c File Reference

```
#include <inttypes.h>
#include <stdbool.h>
#include <stdint.h>
#include "bitops.h"
#include "config_impl.h"
#include "config_spm.h"
#include "critical_section.h"
#include "current.h"
#include "fih.h"
#include "psa/client.h"
#include "psa/service.h"
#include "thread.h"
#include "internal_status_code.h"
#include "tfm_arch.h"
#include "tfm_hal_defs.h"
#include "tfm_hal_interrupt.h"
#include "tfm_hal_isolation.h"
#include "tfm_hal_multi_core.h"
#include "spm.h"
#include "tfm_peripherals_def.h"
#include "tfm_nspm.h"
#include "tfm_core_trustzone.h"
#include "lists.h"
```

```
#include "tfm_pools.h"
#include "region.h"
#include "psa_manifest/pid.h"
#include "ffm/backend.h"
#include "load/partition_defs.h"
#include "load/service_defs.h"
#include "load/asset_defs.h"
#include "load/spm_load_api.h"
```

Include dependency graph for spm_ipc.c:

## Functions

- const struct service_t ∗ tfm_spm_get_service_by_sid (uint32_t sid)

  *Get the service context by service ID.*
- int32_t tfm_spm_check_client_version (const struct service_t ∗service, uint32_t version)

  *Check the client version according to version policy.*
- int32_t tfm_spm_check_authorization (uint32_t sid, const struct service_t ∗service, bool ns_caller)

  *Check the client access authorization.*
- psa_status_t spm_get_idle_connection (struct connection_t ∗∗p_connection, psa_handle_t handle, int32_t client_id)

  *Convert the given user handle to an SPM recognised connection and verify that it is a valid idle connection that the caller is authorised to access.*
- struct connection_t ∗ spm_msg_handle_to_connection (psa_handle_t msg_handle)

  *Convert the given message handle to SPM recognised handle and verify it.*
- void spm_init_idle_connection (struct connection_t ∗p_connection, const struct service_t ∗service, int32_t client_id)

  *Initialize connection, fill in with the input information and set to idle.*
- int32_t tfm_spm_partition_get_running_partition_id (void)

  *Get the running partition ID.*
- bool tfm_spm_is_ns_caller (void)

  *Get the ns_caller info from runtime context.*
- int32_t tfm_spm_get_client_id (bool ns_caller)

  *Get ID of current RoT Service client. This API ensures the caller gets a valid ID.*
- uint32_t tfm_spm_init (void)

  *SPM initialization implementation.*
- void update_caller_outvec_len (struct connection_t ∗handle)

## Variables

- struct service_t ∗ stateless_services_ref_tbl [32]

### 7.365.1 Function Documentation

#### 7.365.1.1 spm_get_idle_connection()

```
psa_status_t spm_get_idle_connection (
            struct connection_t ** p_connection,
            psa_handle_t handle,
            int32_t client_id )
```

Convert the given user handle to an SPM recognised connection and verify that it is a valid idle connection that the caller is authorised to access.

**Parameters**

| | | |
|---|---|---|
| out | *p_connection* | The address of connection pointer to be converted from the given user handle. |
| in | *handle* | Either a static handle or a handle to an established connection that was returned by a prior psa_connect call. |
| in | *client_id* | The client ID of the caller. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |
| *PSA_ERROR_CONNECTION_REFUSED* | The SPM or RoT Service has refused the connection. |
| *PSA_ERROR_CONNECTION_BUSY* | The SPM or RoT Service cannot make the connection at the moment. |
| *PSA_ERROR_PROGRAMMER_ERROR* | The handle is invalid, the caller is not authorised to use it or the connection is already handling a request. |

Definition at line 181 of file spm_ipc.c.

#### 7.365.1.2 spm_init_idle_connection()

```
void spm_init_idle_connection (
            struct connection_t * p_connection,
            const struct service_t * service,
            int32_t client_id )
```

Initialize connection, fill in with the input information and set to idle.

**Parameters**

| | | |
|---|---|---|
| in | *p_connection* | The 'p_connection' to initialize and fill information in. |
| in | *service* | Target service context pointer, which can be obtained by partition management functions |
| in | *client_id* | Partition ID of the sender of the message |

Definition at line 296 of file spm_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.365.1.3 spm_msg_handle_to_connection()

```
struct connection_t* spm_msg_handle_to_connection (
            psa_handle_t msg_handle )
```

Convert the given message handle to SPM recognised handle and verify it.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Message handle which is a reference generated by the SPM to a specific message. |

**Returns**

A SPM recognised handle or NULL. It is NULL when verification of the converted SPM handle fails. connection_t structures

Definition at line 269 of file spm_ipc.c.
Here is the call graph for this function:

#### 7.365.1.4 tfm_spm_check_authorization()

```
int32_t tfm_spm_check_authorization (
            uint32_t sid,
            const struct service_t * service,
            bool ns_caller )
```

Check the client access authorization.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | Target RoT Service identity |
| in | *service* | Target service context pointer, which can be get by partition management functions |
| in | *ns_caller* | Whether from NS caller |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success |
| *SPM_ERROR_GENERIC* | Authorization check failed |

Definition at line 146 of file spm_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.365.1.5 tfm_spm_check_client_version()

```
int32_t tfm_spm_check_client_version (
            const struct service_t * service,
            uint32_t version )
```

Check the client version according to version policy.

**Parameters**

| | | |
|---|---|---|
| in | *service* | Target service context pointer, which can be get by partition management functions |
| in | *version* | Client support version |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success |
| *SPM_ERROR_BAD_PARAMETERS* | Bad parameters input |
| *SPM_ERROR_VERSION* | Check failed |

Definition at line 124 of file spm_ipc.c.

Here is the caller graph for this function:

#### 7.365.1.6 tfm_spm_get_client_id()

```
int32_t tfm_spm_get_client_id (
            bool ns_caller )
```

Get ID of current RoT Service client. This API ensures the caller gets a valid ID.

**Parameters**

| | | |
|---|---|---|
| in | *ns_caller* | If the client is Non-Secure or not. |

**Return values**

| | |
|---|---|
| *The* | client ID |

Definition at line 345 of file spm_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.365.1.7 tfm_spm_get_service_by_sid()

```
const struct service_t* tfm_spm_get_service_by_sid (
            uint32_t sid )
```

Get the service context by service ID.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | RoT Service identity |

**Return values**

| | |
|---|---|
| *NULL* | Failed |
| *Not NULL* | Target service context pointer, service_t structures |

Definition at line 86 of file spm_ipc.c.
Here is the caller graph for this function:

### 7.365.1.8 tfm_spm_init()

```
uint32_t tfm_spm_init (
            void  )
```

SPM initialization implementation.
This function must be called under handler mode.

**Return values**

| | |
|---|---|
| *This* | function returns an EXC_RETURN value. Other faults would panic the execution and never returned. |

Definition at line 363 of file spm_ipc.c.
Here is the call graph for this function:

### 7.365.1.9 tfm_spm_is_ns_caller()

```
bool tfm_spm_is_ns_caller (
            void  )
```

Get the ns_caller info from runtime context.

**Return values**

| | |
|---|---|
| - | true: the PSA API caller is from non-secure<br>• false: the PSA API caller is from secure |

Definition at line 334 of file spm_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.365.1.10 tfm_spm_partition_get_running_partition_id()

```
int32_t tfm_spm_partition_get_running_partition_id (
            void  )
```

Get the running partition ID.

**Returns**

Returns the partition ID

Definition at line 322 of file spm_ipc.c.
Here is the caller graph for this function:

#### 7.365.1.11 update_caller_outvec_len()

```
void update_caller_outvec_len (
            struct connection_t * handle )
```

Definition at line 415 of file spm_ipc.c.

### 7.365.2 Variable Documentation

#### 7.365.2.1 stateless_services_ref_tbl

```
struct service_t* stateless_services_ref_tbl[32]
```

Definition at line 46 of file spm_ipc.c.

## 7.366 secure_fw/spm/core/spm_local_connection.c File Reference

```
#include "compiler_ext_defs.h"
#include "internal_status_code.h"
#include "spm.h"
#include "tfm_arch.h"
```

Include dependency graph for spm_local_connection.c:

### Macros

- #define CONNECTION_SIZE ((sizeof(struct connection_t) + 7) & ~0x7)
- #define FIXED_STATIC_HANDLE ((psa_handle_t)0x1000)

### Functions

- psa_handle_t connection_to_handle (struct connection_t *p_connection)

  *Converts a handle instance into a corresponded user handle.*
- struct connection_t * handle_to_connection (psa_handle_t handle)

  *Converts a user handle into a corresponded handle instance.*
- void spm_init_connection_space (void)
- struct connection_t * spm_allocate_connection (void)
- psa_status_t spm_validate_connection (const struct connection_t *p_connection)
- void spm_free_connection (struct connection_t *p_connection)

### 7.366.1 Macro Definition Documentation

#### 7.366.1.1 CONNECTION_SIZE

```
#define CONNECTION_SIZE ((sizeof(struct connection_t) + 7) & ~0x7)
```

Definition at line 21 of file spm_local_connection.c.

#### 7.366.1.2 FIXED_STATIC_HANDLE

```
#define FIXED_STATIC_HANDLE ((psa_handle_t)0x1000)
```

Definition at line 22 of file spm_local_connection.c.

### 7.366.2 Function Documentation

#### 7.366.2.1 connection_to_handle()

```
psa_handle_t connection_to_handle (
            struct connection_t * p_connection )
```

Converts a handle instance into a corresponded user handle.
Definition at line 51 of file spm_local_connection.c.
Here is the caller graph for this function:

#### 7.366.2.2 handle_to_connection()

```
struct connection_t* handle_to_connection (
            psa_handle_t handle )
```

Converts a user handle into a corresponded handle instance.
Definition at line 58 of file spm_local_connection.c.
Here is the caller graph for this function:

#### 7.366.2.3 spm_allocate_connection()

```
struct connection_t* spm_allocate_connection (
            void )
```

Definition at line 82 of file spm_local_connection.c.

Here is the caller graph for this function:

#### 7.366.2.4 spm_free_connection()

```
void spm_free_connection (
            struct connection_t * p_connection )
```

Definition at line 93 of file spm_local_connection.c.

#### 7.366.2.5 spm_init_connection_space()

```
void spm_init_connection_space (
            void  )
```

Definition at line 73 of file spm_local_connection.c.
Here is the caller graph for this function:

#### 7.366.2.6 spm_validate_connection()

```
psa_status_t spm_validate_connection (
            const struct connection_t * p_connection )
```

Definition at line 87 of file spm_local_connection.c.
Here is the caller graph for this function:

## 7.367 secure_fw/spm/core/spm_log.c File Reference

```
#include "tfm_spm_log.h"
```

Include dependency graph for spm_log.c:

## Macros

- #define MAX_DIGIT_BITS 12 /∗ 8 char for number, 2 for '0x' and 2 for '\r\n' ∗/

## Functions

- int32_t spm_log_msgval (const char ∗msg, size_t len, uint32_t value)

  *SPM output API to convert digit number into HEX string and call the HAL API tfm_hal_output_spm_log.*

## 7.367.1 Macro Definition Documentation

### 7.367.1.1 MAX_DIGIT_BITS

```
#define MAX_DIGIT_BITS 12 /* 8 char for number, 2 for '0x' and 2 for '\r\n' */
```

Definition at line 12 of file spm_log.c.

## 7.367.2 Function Documentation

### 7.367.2.1 spm_log_msgval()

```
int32_t spm_log_msgval (
            const char * msg,
            size_t len,
            uint32_t value )
```

SPM output API to convert digit number into HEX string and call the HAL API tfm_hal_output_spm_log.

**Parameters**

| | | |
|---|---|---|
| in | *msg* | A string message |
| in | *len* | The length of the message |
| in | *value* | A value need to be output |

**Return values**

| | |
|---|---|
| *>=0* | Number of chars output. |

**Return values**

| | |
|---|---|
| *<0* | TFM HAL error code. |

Definition at line 38 of file spm_log.c.
Here is the call graph for this function:

## 7.368 secure_fw/spm/core/stack_watermark.c File Reference

```
#include <stdint.h>
#include "ffm/backend.h"
#include "stack_watermark.h"
#include "lists.h"
#include "load/spm_load_api.h"
#include "spm.h"
#include "tfm_spm_log.h"
```

Include dependency graph for stack_watermark.c:

### Macros

- #define SPMLOG(x) tfm_hal_output_spm_log((x), sizeof(x))
- #define SPMLOG_VAL(x, y) spm_log_msgval((x), sizeof(x), y)
- #define STACK_WATERMARK_VAL 0xdeadbeef

### Functions

- for (int i=0;i< p_pldi->stack_size/4;i++)
- tfm_hal_output_spm_log (("Used stack sizes report\r\n"), sizeof("Used stack sizes report\r\n"))
- UNI_LIST_FOREACH (p_pt, PARTITION_LIST_ADDR, next)

### Variables

- void

### 7.368.1 Macro Definition Documentation

#### 7.368.1.1 SPMLOG

```
#define SPMLOG(
              x ) tfm_hal_output_spm_log((x), sizeof(x))
```

Definition at line 19 of file stack_watermark.c.

#### 7.368.1.2 SPMLOG_VAL

```
#define SPMLOG_VAL(
              x,
              y ) spm_log_msgval((x), sizeof(x), y)
```

Definition at line 20 of file stack_watermark.c.

#### 7.368.1.3 STACK_WATERMARK_VAL

```
#define STACK_WATERMARK_VAL 0xdeadbeef
```

Definition at line 22 of file stack_watermark.c.

### 7.368.2 Function Documentation

#### 7.368.2.1 for()

```
for (
             int i = 0; i < p_pldi->stack_size / 4; i++ )
```

Definition at line 28 of file stack_watermark.c.

Here is the caller graph for this function:

#### 7.368.2.2 tfm_hal_output_spm_log()

```
tfm_hal_output_spm_log (
             ("Used stack sizes report\r\n") ,
             sizeof("Used stack sizes report\r\n")  )
```

Here is the caller graph for this function:

#### 7.368.2.3 UNI_LIST_FOREACH()

```
UNI_LIST_FOREACH (
            p_pt ,
            PARTITION_LIST_ADDR ,
            next  )
```

Definition at line 56 of file stack_watermark.c.

### 7.368.3 Variable Documentation

#### 7.368.3.1 void

```
void
```

**Initial value:**

```
{
    const struct partition_load_info_t *p_pldi = p_pt->p_ldinf
```

Definition at line 25 of file stack_watermark.c.

## 7.369 secure_fw/spm/core/stack_watermark.h File Reference

```
#include "spm.h"
```

Include dependency graph for stack_watermark.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define watermark_stack(p_pt)
- #define dump_used_stacks()

### 7.369.1 Macro Definition Documentation

#### 7.369.1.1 dump_used_stacks

```
#define dump_used_stacks( )
```

Definition at line 18 of file stack_watermark.h.

#### 7.369.1.2 watermark_stack

```
#define watermark_stack(
            p_pt )
```

Definition at line 17 of file stack_watermark.h.

## 7.370 secure_fw/spm/core/tfm_boot_data.c File Reference

```
#include <stdint.h>
#include <string.h>
#include "array.h"
#include "tfm_boot_status.h"
#include "region_defs.h"
#include "psa_manifest/pid.h"
#include "internal_status_code.h"
#include "utilities.h"
#include "psa/service.h"
#include "thread.h"
#include "spm.h"
#include "load/partition_defs.h"
#include "tfm_hal_isolation.h"
#include "static_checks.h"
```

Include dependency graph for tfm_boot_data.c:

### Data Structures

- struct boot_data_access_policy

  *Defines the access policy of secure partitions to data items in shared data area (between bootloader and runtime firmware).*

### Macros

- #define BOOT_DATA_VALID (1u)

  *Indicates that shared data between bootloader and runtime firmware was passed the sanity check with success.*
- #define BOOT_DATA_INVALID (0u)

  *Indicates that shared data between bootloader and runtime firmware was failed on sanity check.*

### Functions

- void tfm_core_validate_boot_data (void)

*Validate the content of shared memory area, which stores the shared data between bootloader and runtime firmware.*

- void tfm_core_get_boot_data_handler (uint32_t args[ ])

  *Retrieve secure partition related data from shared memory area, which stores shared data between bootloader and runtime firmware.*

### 7.370.1 Macro Definition Documentation

#### 7.370.1.1 BOOT_DATA_INVALID

```
#define BOOT_DATA_INVALID (0u)
```

Indicates that shared data between bootloader and runtime firmware was failed on sanity check.
Definition at line 39 of file tfm_boot_data.c.

#### 7.370.1.2 BOOT_DATA_VALID

```
#define BOOT_DATA_VALID (1u)
```

Indicates that shared data between bootloader and runtime firmware was passed the sanity check with success.
Definition at line 31 of file tfm_boot_data.c.

### 7.370.2 Function Documentation

#### 7.370.2.1 tfm_core_get_boot_data_handler()

```
void tfm_core_get_boot_data_handler (
            uint32_t args[] )
```

Retrieve secure partition related data from shared memory area, which stores shared data between bootloader and runtime firmware.

**Parameters**

| | | |
|---|---|---|
| in | *args* | Pointer to stack frame, which carries input parameters. |

Definition at line 148 of file tfm_boot_data.c.

#### 7.370.2.2 tfm_core_validate_boot_data()

```
void tfm_core_validate_boot_data (
            void )
```

Validate the content of shared memory area, which stores the shared data between bootloader and runtime firmware.
Definition at line 133 of file tfm_boot_data.c.

## 7.371 secure_fw/spm/core/tfm_boot_data.h File Reference

```
#include <stdint.h>
```

Include dependency graph for tfm_boot_data.h:

This graph shows which files directly or indirectly include this file:

## Functions

- void tfm_core_get_boot_data_handler (uint32_t args[ ])

  *Retrieve secure partition related data from shared memory area, which stores shared data between bootloader and runtime firmware.*
- void tfm_core_validate_boot_data (void)

  *Validate the content of shared memory area, which stores the shared data between bootloader and runtime firmware.*

### 7.371.1 Function Documentation

#### 7.371.1.1 tfm_core_get_boot_data_handler()

```
void tfm_core_get_boot_data_handler (
            uint32_t args[] )
```

Retrieve secure partition related data from shared memory area, which stores shared data between bootloader and runtime firmware.

**Parameters**

| | | |
|---|---|---|
| in | *args* | Pointer to stack frame, which carries input parameters. |

Definition at line 148 of file tfm_boot_data.c.

#### 7.371.1.2 tfm_core_validate_boot_data()

```
void tfm_core_validate_boot_data (
            void  )
```

Validate the content of shared memory area, which stores the shared data between bootloader and runtime firmware.
Definition at line 133 of file tfm_boot_data.c.

## 7.372 secure_fw/spm/core/tfm_multi_core.h File Reference

```
#include <stdint.h>
```

Include dependency graph for tfm_multi_core.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define CLIENT_ID_OWNER_MAGIC (void *)0xFFFFFFFFU

### Functions

- int32_t tfm_inter_core_comm_init (void)

  *Initialization of the multi core communication.*
- int32_t tfm_inter_core_comm_enable (void)

  *Enable the multi core communication.*
- int32_t tfm_inter_core_comm_disable (void)

  *Disable the multi core communication.*
- int32_t tfm_inter_core_comm_deinit (void)

*De-initialization of the multi core communication.*

- int32_t tfm_multi_core_register_client_id_range (void *owner, int32_t client_id_base, int32_t client_id_limit)

  *Register a non-secure client ID range.*

- int32_t tfm_multi_core_hal_client_id_translate (void *owner, int32_t client_id_in, int32_t *client_id_out)

  *Translate a non-secure client ID range.*

## 7.372.1 Macro Definition Documentation

### 7.372.1.1 CLIENT_ID_OWNER_MAGIC

```
#define CLIENT_ID_OWNER_MAGIC (void *)0xFFFFFFFFU
```

Definition at line 13 of file tfm_multi_core.h.

## 7.372.2 Function Documentation

### 7.372.2.1 tfm_inter_core_comm_deinit()

```
int32_t tfm_inter_core_comm_deinit (
            void  )
```

De-initialization of the multi core communication.
This may be called to return the multi-core communication from the initialised, but not enabled state to the pre-initialised state.

**Return values**

| | |
|---|---|
| *0* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 620 of file tfm_spe_mailbox.c.

### 7.372.2.2 tfm_inter_core_comm_disable()

```
int32_t tfm_inter_core_comm_disable (
            void  )
```

Disable the multi core communication.
This may be called to return the multi core communication to the initialised, but not enabled, state.

**Return values**

| | |
|---|---|
| *0* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 615 of file tfm_spe_mailbox.c.

### 7.372.2.3 tfm_inter_core_comm_enable()

```
int32_t tfm_inter_core_comm_enable (
            void  )
```

Enable the multi core communication.
This is called after tfm_inter_core_init() and may be called again after tfm_inter_core_disable().

**Return values**

| | |
|---|---|
| *0* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 610 of file tfm_spe_mailbox.c.

#### 7.372.2.4 tfm_inter_core_comm_init()

```
int32_t tfm_inter_core_comm_init (
            void  )
```

Initialization of the multi core communication.
This is called once only, after the NS core has booted.

**Return values**

| | |
|---|---|
| *0* | Operation succeeded. |
| *Other* | return code Operation failed with an error code. |

Definition at line 605 of file tfm_spe_mailbox.c.

#### 7.372.2.5 tfm_multi_core_hal_client_id_translate()

```
int32_t tfm_multi_core_hal_client_id_translate (
            void * owner,
            int32_t client_id_in,
            int32_t * client_id_out )
```

Translate a non-secure client ID range.

**Parameters**

| | | |
|---|---|---|
| in | *owner* | Identifier of the non-secure client. |
| in | *client_id_in* | The input client ID. |
| out | *client_id_out* | The translated client ID. Undefined if SPM_ERROR_GENERIC is returned by the function |

**Returns**

SPM_SUCCESS if the translation is successful, SPM_ERROR_GENERIC otherwise.

Definition at line 42 of file tfm_multi_core_client_id.c.

#### 7.372.2.6 tfm_multi_core_register_client_id_range()

```
int32_t tfm_multi_core_register_client_id_range (
            void * owner,
            int32_t client_id_base,
            int32_t client_id_limit )
```

Register a non-secure client ID range.

**Parameters**

| | | |
|---|---|---|
| in | *owner* | Identifier of the non-secure client. |
| in | *client_id_base* | The minimum client ID for this client. |
| in | *client_id_limit* | The maximum client ID for this client. |

**Returns**

SPM_SUCCESS if the registration is successful, SPM_ERROR_GENERIC if owner is null, or no free slots left.

Definition at line 23 of file tfm_multi_core_client_id.c.

## 7.373 secure_fw/spm/core/tfm_pools.c File Reference

```
#include <inttypes.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include "thread.h"
#include "psa/client.h"
#include "psa/service.h"
#include "internal_status_code.h"
#include "cmsis_compiler.h"
#include "utilities.h"
#include "lists.h"
#include "tfm_pools.h"
#include "static_checks.h"
```

Include dependency graph for tfm_pools.c:

### Macros

- #define POOL_MAGIC_ALLOCATED UINT32_C(0xF0F0CCAA)

### Functions

- psa_status_t tfm_pool_init (struct tfm_pool_instance_t *pool, size_t poolsz, size_t chunksz, size_t num)

  *Register a memory pool.*
- void * tfm_pool_alloc (struct tfm_pool_instance_t *pool)

  *Allocate a memory from pool.*
- void tfm_pool_free (struct tfm_pool_instance_t *pool, void *ptr)

  *Free the allocated memory.*
- bool is_valid_chunk_data_in_pool (struct tfm_pool_instance_t *pool, uint8_t *data)

  *Checks whether a pointer points to a valid allocated chunk of data in the pool.*

### 7.373.1 Macro Definition Documentation

#### 7.373.1.1 POOL_MAGIC_ALLOCATED

```
#define POOL_MAGIC_ALLOCATED UINT32_C(0xF0F0CCAA)
```

Definition at line 24 of file tfm_pools.c.

## 7.373.2 Function Documentation

### 7.373.2.1 is_valid_chunk_data_in_pool()

```
bool is_valid_chunk_data_in_pool (
            struct tfm_pool_instance_t * pool,
            uint8_t * data )
```

Checks whether a pointer points to a valid allocated chunk of data in the pool.

**Parameters**

| | | |
|---|---|---|
| in | *pool* | Pointer to memory pool declared by TFM_POOL_DECLARE. |
| in | *data* | The pointer to check. |

**Return values**

| | |
|---|---|
| *true* | Data is a chunk data in the pool. |
| *false* | Data is not a chunk data in the pool. |

Definition at line 103 of file tfm_pools.c.
Here is the caller graph for this function:

### 7.373.2.2 tfm_pool_alloc()

```
void* tfm_pool_alloc (
            struct tfm_pool_instance_t * pool )
```

Allocate a memory from pool.

**Parameters**

| | | |
|---|---|---|
| in | *pool* | pool pointer decleared by TFM_POOL_DECLARE |

**Return values**

| | |
|---|---|
| *buffer* | pointer Success. |
| *NULL* | Failed. |

Definition at line 63 of file tfm_pools.c.

Here is the caller graph for this function:

#### 7.373.2.3 tfm_pool_free()

```
void tfm_pool_free (
            struct tfm_pool_instance_t * pool,
            void * ptr )
```

Free the allocated memory.

**Parameters**

| | | |
|---|---|---|
| in | *pool* | pool pointer decleared by TFM_POOL_DECLARE |
| in | *ptr* | Buffer pointer want to free. |

Definition at line 83 of file tfm_pools.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.373.2.4 tfm_pool_init()

```
psa_status_t tfm_pool_init (
            struct tfm_pool_instance_t * pool,
            size_t poolsz,
            size_t chunksz,
            size_t num )
```

Register a memory pool.

**Parameters**

| | | |
|---|---|---|
| in | *pool* | Pointer to memory pool declared by TFM_POOL_DECLARE |
| in | *poolsz* | Size of the pool buffer. |
| in | *chunksz* | Size of chunks. |
| in | *num* | Number of chunks. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |
| *SPM_ERROR_BAD_PARAMETERS* | Parameters error. |

Definition at line 26 of file tfm_pools.c.
Here is the call graph for this function:

Here is the caller graph for this function:

## 7.374 secure_fw/spm/core/tfm_pools.h File Reference

```
#include <stdbool.h>
#include "psa/error.h"
#include "compiler_ext_defs.h"
```

```
#include "lists.h"
```

Include dependency graph for tfm_pools.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct tfm_pool_chunk_t
- struct tfm_pool_instance_t

## Macros

- #define TFM_POOL_DECLARE(name, chunksz, num)
- #define POOL_HEAD_SIZE
- #define POOL_BUFFER_SIZE(name) sizeof(name##_pool_buf)

## Functions

- psa_status_t tfm_pool_init (struct tfm_pool_instance_t *pool, size_t poolsz, size_t chunksz, size_t num)

  *Register a memory pool.*
- void * tfm_pool_alloc (struct tfm_pool_instance_t *pool)

  *Allocate a memory from pool.*
- void tfm_pool_free (struct tfm_pool_instance_t *pool, void *ptr)

  *Free the allocated memory.*
- bool is_valid_chunk_data_in_pool (struct tfm_pool_instance_t *pool, uint8_t *data)

  *Checks whether a pointer points to a valid allocated chunk of data in the pool.*

## 7.374.1 Macro Definition Documentation

### 7.374.1.1 POOL_BUFFER_SIZE

```
#define POOL_BUFFER_SIZE(
            name ) sizeof(name##_pool_buf)
```

Definition at line 52 of file tfm_pools.h.

### 7.374.1.2 POOL_HEAD_SIZE

```
#define POOL_HEAD_SIZE
```

**Value:**

```
                    (sizeof(struct tfm_pool_instance_t) +                \
                    sizeof(struct tfm_pool_chunk_t))
```

Definition at line 48 of file tfm_pools.h.

### 7.374.1.3 TFM_POOL_DECLARE

```
#define TFM_POOL_DECLARE(
            name,
            chunksz,
            num )
```

**Value:**

```
    static uint8_t name##_pool_buf[((chunksz) +                          \
                                    sizeof(struct tfm_pool_chunk_t)) * (num) \
                                    + sizeof(struct tfm_pool_instance_t)]    \
                                    __aligned(4);                            \
    static struct tfm_pool_instance_t *name =                                \
                            (struct tfm_pool_instance_t *)name##_pool_buf
```

Definition at line 39 of file tfm_pools.h.

## 7.374.2 Function Documentation

### 7.374.2.1 is_valid_chunk_data_in_pool()

```
bool is_valid_chunk_data_in_pool (
            struct tfm_pool_instance_t * pool,
            uint8_t * data )
```

Checks whether a pointer points to a valid allocated chunk of data in the pool.

**Parameters**

| | | |
|---|---|---|
| in | *pool* | Pointer to memory pool declared by TFM_POOL_DECLARE. |
| in | *data* | The pointer to check. |

**Return values**

| | |
|---|---|
| *true* | Data is a chunk data in the pool. |
| *false* | Data is not a chunk data in the pool. |

Definition at line 103 of file tfm_pools.c.

Here is the caller graph for this function:

#### 7.374.2.2 tfm_pool_alloc()

```
void* tfm_pool_alloc (
            struct tfm_pool_instance_t * pool )
```

Allocate a memory from pool.

**Parameters**

| | | |
|---|---|---|
| in | *pool* | pool pointer decleared by TFM_POOL_DECLARE |

**Return values**

| | |
|---|---|
| *buffer* | pointer Success. |
| *NULL* | Failed. |

Definition at line 63 of file tfm_pools.c.
Here is the caller graph for this function:

#### 7.374.2.3 tfm_pool_free()

```
void tfm_pool_free (
            struct tfm_pool_instance_t * pool,
            void * ptr )
```

Free the allocated memory.

**Parameters**

| | | |
|---|---|---|
| in | *pool* | pool pointer decleared by TFM_POOL_DECLARE |
| in | *ptr* | Buffer pointer want to free. |

Definition at line 83 of file tfm_pools.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.374.2.4 tfm_pool_init()

```
psa_status_t tfm_pool_init (
            struct tfm_pool_instance_t * pool,
            size_t poolsz,
            size_t chunksz,
            size_t num )
```

Register a memory pool.

**Parameters**

| | | |
|---|---|---|
| in | *pool* | Pointer to memory pool declared by TFM_POOL_DECLARE |
| in | *poolsz* | Size of the pool buffer. |
| in | *chunksz* | Size of chunks. |
| in | *num* | Number of chunks. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |
| *SPM_ERROR_BAD_PARAMETERS* | Parameters error. |

Definition at line 26 of file tfm_pools.c.

Here is the call graph for this function:

Here is the caller graph for this function:

## 7.375 secure_fw/spm/core/tfm_rpc.c File Reference

```
#include "async.h"
#include "config_impl.h"
#include "spm.h"
#include "static_checks.h"
#include "ffm/mailbox_agent_api.h"
#include "ffm/psa_api.h"
#include "tfm_rpc.h"
#include "utilities.h"
#include "load/partition_defs.h"
#include "tfm_psa_call_pack.h"
```

Include dependency graph for tfm_rpc.c:

### Functions

- uint32_t tfm_rpc_psa_framework_version (void)
- uint32_t tfm_rpc_psa_version (uint32_t sid)
- psa_status_t tfm_rpc_psa_call (psa_handle_t handle, uint32_t control, const struct client_params_t ∗params, const void ∗client_data_stateless)

- int32_t tfm_rpc_register_ops (const struct tfm_rpc_ops_t *ops_ptr)
- void tfm_rpc_unregister_ops (void)
- void tfm_rpc_client_call_handler (void)

## 7.375.1 Function Documentation

### 7.375.1.1 tfm_rpc_client_call_handler()

```
void tfm_rpc_client_call_handler (
            void  )
```

Definition at line 104 of file tfm_rpc.c.

### 7.375.1.2 tfm_rpc_psa_call()

```
psa_status_t tfm_rpc_psa_call (
            psa_handle_t handle,
            uint32_t control,
            const struct client_params_t * params,
            const void * client_data_stateless )
```

Definition at line 46 of file tfm_rpc.c.

### 7.375.1.3 tfm_rpc_psa_framework_version()

```
uint32_t tfm_rpc_psa_framework_version (
            void  )
```

Definition at line 36 of file tfm_rpc.c.
Here is the call graph for this function:

### 7.375.1.4 tfm_rpc_psa_version()

```
uint32_t tfm_rpc_psa_version (
            uint32_t sid )
```

Definition at line 41 of file tfm_rpc.c.
Here is the call graph for this function:

#### 7.375.1.5 tfm_rpc_register_ops()

```
int32_t tfm_rpc_register_ops (
            const struct tfm_rpc_ops_t * ops_ptr )
```

Definition at line 76 of file tfm_rpc.c.

#### 7.375.1.6 tfm_rpc_unregister_ops()

```
void tfm_rpc_unregister_ops (
            void  )
```

Definition at line 98 of file tfm_rpc.c.

## 7.376 secure_fw/spm/core/tfm_rpc.h File Reference

This graph shows which files directly or indirectly include this file:

## 7.377 secure_fw/spm/core/tfm_svcalls.c File Reference

```
#include <string.h>
#include <stdint.h>
#include "aapcs_local.h"
#include "config_spm.h"
#include "interrupt.h"
#include "internal_status_code.h"
#include "memory_symbols.h"
#include "spm.h"
#include "static_checks.h"
#include "svc_num.h"
#include "tfm_arch.h"
#include "tfm_svcalls.h"
#include "tfm_boot_data.h"
#include "tfm_hal_platform.h"
#include "tfm_hal_isolation.h"
#include "tfm_hal_spm_logdev.h"
#include "tfm_core_trustzone.h"
#include "utilities.h"
#include "ffm/backend.h"
#include "ffm/psa_api.h"
#include "load/spm_load_api.h"
#include "load/partition_defs.h"
```

```
#include "psa/client.h"
```

Include dependency graph for tfm_svcalls.c:

## Macros

- #define set_ns_ctx_mgr NULL
- #define INVALID_PSP_VALUE 0xFFFFFFFFU

## Functions

- uint32_t spm_svc_handler (uint32_t *msp, uint32_t exc_return, uint32_t *psp)

  *The C source of SVCall handlers.*
- void tfm_svc_thread_mode_spm_return (psa_status_t result)

  *Return from PSA API function executed in Thread mode. Trigger svc handler to restore thread context before entering PSA API function.*

## 7.377.1 Macro Definition Documentation

### 7.377.1.1 INVALID_PSP_VALUE

```
#define INVALID_PSP_VALUE 0xFFFFFFFFU
```

Definition at line 45 of file tfm_svcalls.c.

### 7.377.1.2 set_ns_ctx_mgr

```
#define set_ns_ctx_mgr NULL
```

Definition at line 42 of file tfm_svcalls.c.

## 7.377.2 Function Documentation

### 7.377.2.1 spm_svc_handler()

```
uint32_t spm_svc_handler (
            uint32_t * msp,
            uint32_t exc_return,
            uint32_t * psp )
```

The C source of SVCall handlers.

**Parameters**

| | | |
|---|---|---|
| in | *msp* | MSP at SVCall entry. |
| in | *exc_return* | EXC_RETURN value of the SVC. |
| in | *psp* | PSP at SVCall entry. |

**Returns**

EXC_RETURN value indicates where to return.

Definition at line 325 of file tfm_svcalls.c.
Here is the call graph for this function:

#### 7.377.2.2 tfm_svc_thread_mode_spm_return()

```
void tfm_svc_thread_mode_spm_return (
            psa_status_t result )
```

Return from PSA API function executed in Thread mode. Trigger svc handler to restore thread context before entering PSA API function.

**Parameters**

| | | |
|---|---|---|
| in | *result* | PSA API function return value. |

**Returns**

none.

Definition at line 390 of file tfm_svcalls.c.

## 7.378 secure_fw/spm/core/tfm_svcalls.h File Reference

```
#include <stdint.h>
```

Include dependency graph for tfm_svcalls.h:

This graph shows which files directly or indirectly include this file:

## Functions

- uint32_t spm_svc_handler (uint32_t ∗msp, uint32_t exc_return, uint32_t ∗psp)

  *The C source of SVCall handlers.*
- void tfm_svc_thread_mode_spm_return (psa_status_t result)

  *Return from PSA API function executed in Thread mode. Trigger svc handler to restore thread context before entering PSA API function.*

## 7.378.1 Function Documentation

### 7.378.1.1 spm_svc_handler()

```
uint32_t spm_svc_handler (
            uint32_t * msp,
            uint32_t exc_return,
            uint32_t * psp )
```

The C source of SVCall handlers.

**Parameters**

| | | |
|---|---|---|
| in | *msp* | MSP at SVCall entry. |
| in | *exc_return* | EXC_RETURN value of the SVC. |
| in | *psp* | PSP at SVCall entry. |

**Returns**

EXC_RETURN value indicates where to return.

Definition at line 325 of file tfm_svcalls.c.
Here is the call graph for this function:

#### 7.378.1.2 tfm_svc_thread_mode_spm_return()

```
void tfm_svc_thread_mode_spm_return (
            psa_status_t result )
```

Return from PSA API function executed in Thread mode. Trigger svc handler to restore thread context before entering PSA API function.

**Parameters**

| | | |
|---|---|---|
| in | *result* | PSA API function return value. |

**Returns**

none.

Definition at line 390 of file tfm_svcalls.c.

## 7.379 secure_fw/spm/core/thread.c File Reference

```
#include <stdint.h>
#include "thread.h"
#include "tfm_arch.h"
#include "utilities.h"
#include "critical_section.h"
```

Include dependency graph for thread.c:

### Macros

- #define LIST_HEAD p_thrd_head
- #define RNBL_HEAD p_rnbl_head

### Functions

- void thrd_set_query_callback (thrd_query_state_t fn)
- struct thread_t ∗ thrd_next (void)
- void thrd_start (struct thread_t ∗p_thrd, thrd_fn_t fn, thrd_fn_t exit_fn, void ∗param)
- void thrd_set_state (struct thread_t ∗p_thrd, uint32_t new_state)
- uint32_t thrd_start_scheduler (struct thread_t ∗∗ppth)

### Variables

- struct thread_t ∗ p_curr_thrd

### 7.379.1 Macro Definition Documentation

#### 7.379.1.1 LIST_HEAD

```
#define LIST_HEAD p_thrd_head
```

Definition at line 25 of file thread.c.

#### 7.379.1.2 RNBL_HEAD

```
#define RNBL_HEAD p_rnbl_head
```

Definition at line 26 of file thread.c.

### 7.379.2 Function Documentation

#### 7.379.2.1 thrd_next()

```
struct thread_t* thrd_next (
            void  )
```

Definition at line 36 of file thread.c.

Here is the caller graph for this function:

#### 7.379.2.2 thrd_set_query_callback()

```
void thrd_set_query_callback (
            thrd_query_state_t fn )
```

Definition at line 31 of file thread.c.

Here is the caller graph for this function:

#### 7.379.2.3 thrd_set_state()

```
void thrd_set_state (
            struct thread_t * p_thrd,
            uint32_t new_state )
```

Definition at line 99 of file thread.c.

#### 7.379.2.4 thrd_start()

```
void thrd_start (
            struct thread_t * p_thrd,
            thrd_fn_t fn,
            thrd_fn_t exit_fn,
            void * param )
```

Definition at line 84 of file thread.c.

#### 7.379.2.5 thrd_start_scheduler()

```
uint32_t thrd_start_scheduler (
            struct thread_t ** ppth )
```

Definition at line 117 of file thread.c.
Here is the call graph for this function:

### 7.379.3 Variable Documentation

#### 7.379.3.1 p_curr_thrd

```
struct thread_t* p_curr_thrd
```

Definition at line 18 of file thread.c.

## 7.380 secure_fw/spm/core/thread.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "tfm_arch.h"
```

Include dependency graph for thread.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct thread_t

## Macros

- #define THRD_STATE_CREATING 0
- #define THRD_STATE_RUNNABLE 1
- #define THRD_STATE_BLOCK 2
- #define THRD_STATE_DETACH 3
- #define THRD_STATE_INVALID 4
- #define THRD_STATE_RET_VAL_AVAIL 5
- #define THRD_PRIOR_HIGHEST 0x0
- #define THRD_PRIOR_HIGH 0xF
- #define THRD_PRIOR_MEDIUM 0x1F
- #define THRD_PRIOR_LOW 0x7F
- #define THRD_PRIOR_LOWEST 0xFF
- #define THRD_SUCCESS 0
- #define THRD_ERR_GENERIC 1
- #define THRD_GENERAL_EXIT ((thrd_fn_t)(0xFFFFFFFE))
- #define CURRENT_THREAD p_curr_thrd
- #define THRD_INIT(p_thrd, p_ctx_ctrl, prio)
- #define THRD_SET_PRIORITY(p_thrd, priority) p_thrd->priority = (uint8_t)(priority)
- #define THRD_UPDATE_CUR_CTXCTRL(x) CURRENT_THREAD->p_context_ctrl = (x)

## Typedefs

- typedef void(* thrd_fn_t) (void *)
- typedef uint32_t(* thrd_query_state_t) (struct thread_t *p_thrd, uint32_t *p_retval)

## Functions

- void thrd_set_query_callback (thrd_query_state_t fn)
- void thrd_set_state (struct thread_t *p_thrd, uint32_t new_state)
- void thrd_start (struct thread_t *p_thrd, thrd_fn_t fn, thrd_fn_t exit_fn, void *param)
- struct thread_t * thrd_next (void)
- uint32_t thrd_start_scheduler (struct thread_t **ppth)

## Variables

- struct thread_t * p_curr_thrd

## 7.380.1 Macro Definition Documentation

### 7.380.1.1 CURRENT_THREAD

```
#define CURRENT_THREAD p_curr_thrd
```

Definition at line 61 of file thread.h.

### 7.380.1.2 THRD_ERR_GENERIC

```
#define THRD_ERR_GENERIC 1
```

Definition at line 36 of file thread.h.

### 7.380.1.3 THRD_GENERAL_EXIT

```
#define THRD_GENERAL_EXIT ((thrd_fn_t)(0xFFFFFFFE))
```

Definition at line 42 of file thread.h.

### 7.380.1.4 THRD_INIT

```
#define THRD_INIT(
            p_thrd,
            p_ctx_ctrl,
            prio )
```

**Value:**

```
        do {                                    \
        (p_thrd)->priority       = (uint8_t)(prio);       \
        (p_thrd)->state          = THRD_STATE_CREATING;   \
        (p_thrd)->flags          = 0;                     \
        (p_thrd)->p_context_ctrl = p_ctx_ctrl;            \
    } while (0)
```

Definition at line 71 of file thread.h.

### 7.380.1.5 THRD_PRIOR_HIGH

```
#define THRD_PRIOR_HIGH 0xF
```

Definition at line 29 of file thread.h.

### 7.380.1.6 THRD_PRIOR_HIGHEST

```
#define THRD_PRIOR_HIGHEST 0x0
```

Definition at line 28 of file thread.h.

#### 7.380.1.7 THRD_PRIOR_LOW

```
#define THRD_PRIOR_LOW 0x7F
```

Definition at line 31 of file thread.h.

#### 7.380.1.8 THRD_PRIOR_LOWEST

```
#define THRD_PRIOR_LOWEST 0xFF
```

Definition at line 32 of file thread.h.

#### 7.380.1.9 THRD_PRIOR_MEDIUM

```
#define THRD_PRIOR_MEDIUM 0x1F
```

Definition at line 30 of file thread.h.

#### 7.380.1.10 THRD_SET_PRIORITY

```
#define THRD_SET_PRIORITY(
            p_thrd,
            priority ) p_thrd->priority = (uint8_t)(priority)
```

Definition at line 88 of file thread.h.

#### 7.380.1.11 THRD_STATE_BLOCK

```
#define THRD_STATE_BLOCK 2
```

Definition at line 22 of file thread.h.

#### 7.380.1.12 THRD_STATE_CREATING

```
#define THRD_STATE_CREATING 0
```

Definition at line 20 of file thread.h.

#### 7.380.1.13 THRD_STATE_DETACH

```
#define THRD_STATE_DETACH 3
```

Definition at line 23 of file thread.h.

#### 7.380.1.14 THRD_STATE_INVALID

```
#define THRD_STATE_INVALID 4
```

Definition at line 24 of file thread.h.

#### 7.380.1.15 THRD_STATE_RET_VAL_AVAIL

```
#define THRD_STATE_RET_VAL_AVAIL 5
```

Definition at line 25 of file thread.h.

#### 7.380.1.16 THRD_STATE_RUNNABLE

```
#define THRD_STATE_RUNNABLE 1
```

Definition at line 21 of file thread.h.

#### 7.380.1.17 THRD_SUCCESS

```
#define THRD_SUCCESS 0
```

Definition at line 35 of file thread.h.

#### 7.380.1.18 THRD_UPDATE_CUR_CTXCTRL

```
#define THRD_UPDATE_CUR_CTXCTRL(
            x ) CURRENT_THREAD->p_context_ctrl = (x)
```

Definition at line 97 of file thread.h.

### 7.380.2 Typedef Documentation

#### 7.380.2.1 thrd_fn_t

```
typedef void(* thrd_fn_t) (void *)
```

Definition at line 39 of file thread.h.

#### 7.380.2.2 thrd_query_state_t

```
typedef uint32_t(* thrd_query_state_t) (struct thread_t *p_thrd, uint32_t *p_retval)
```

Definition at line 54 of file thread.h.

### 7.380.3 Function Documentation

#### 7.380.3.1 thrd_next()

```
struct thread_t* thrd_next (
            void )
```

Definition at line 36 of file thread.c.
Here is the caller graph for this function:

#### 7.380.3.2 thrd_set_query_callback()

```
void thrd_set_query_callback (
            thrd_query_state_t fn )
```

Definition at line 31 of file thread.c.

Here is the caller graph for this function:

#### 7.380.3.3 thrd_set_state()

```
void thrd_set_state (
            struct thread_t * p_thrd,
            uint32_t new_state )
```

Definition at line 99 of file thread.c.

#### 7.380.3.4 thrd_start()

```
void thrd_start (
            struct thread_t * p_thrd,
            thrd_fn_t fn,
            thrd_fn_t exit_fn,
            void * param )
```

Definition at line 84 of file thread.c.

#### 7.380.3.5 thrd_start_scheduler()

```
uint32_t thrd_start_scheduler (
            struct thread_t ** ppth )
```

Definition at line 117 of file thread.c.
Here is the call graph for this function:

### 7.380.4 Variable Documentation

#### 7.380.4.1 p_curr_thrd

```
struct thread_t* p_curr_thrd
```

Definition at line 18 of file thread.c.

## 7.381 platform/ext/target/infineon/common/nspe/os_wrapper/thread.h File Reference

`#include "os_wrapper/common.h"`

Include dependency graph for thread.h:

This graph shows which files directly or indirectly include this file:

### Typedefs

- typedef void(∗ os_wrapper_thread_func) (void ∗argument)

### Functions

- void ∗ os_wrapper_thread_new (const char ∗name, int32_t stack_size, os_wrapper_thread_func func, void ∗arg, uint32_t priority)

  *Creates a new thread.*
- void ∗ os_wrapper_thread_get_handle (void)

  *Gets current thread handle.*
- uint32_t os_wrapper_thread_get_priority (void ∗handle, uint32_t ∗priority)

  *Gets thread priority.*
- void os_wrapper_thread_exit (void)

  *Exits the calling thread.*
- uint32_t os_wrapper_thread_set_flag (void ∗handle, uint32_t flags)

  *Set the event flags for synchronizing a thread specified by handle.*
- uint32_t os_wrapper_thread_set_flag_isr (void ∗handle, uint32_t flags)

  *Set the event flags in an interrupt handler for synchronizing a thread specified by handle.*
- uint32_t os_wrapper_thread_wait_flag (uint32_t flags, uint32_t timeout)

  *Wait for the event flags for synchronizing threads.*

## 7.381.1 Typedef Documentation

### 7.381.1.1 os_wrapper_thread_func

```
typedef void(* os_wrapper_thread_func) (void *argument)
```

Definition at line 20 of file thread.h.

## 7.381.2 Function Documentation

### 7.381.2.1 os_wrapper_thread_exit()

```
void os_wrapper_thread_exit (
            void  )
```

Exits the calling thread.

### 7.381.2.2 os_wrapper_thread_get_handle()

```
void* os_wrapper_thread_get_handle (
            void  )
```

Gets current thread handle.

**Returns**

Returns the thread handle, or NULL in case of error

Here is the caller graph for this function:

### 7.381.2.3 os_wrapper_thread_get_priority()

```
uint32_t os_wrapper_thread_get_priority (
            void * handle,
            uint32_t * priority )
```

Gets thread priority.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Thread handle |
| out | *priority* | The priority of the thread |

**Returns**

Returns OS_WRAPPER_SUCCESS on success, or OS_WRAPPER_ERROR in case of error

#### 7.381.2.4 os_wrapper_thread_new()

```
void* os_wrapper_thread_new (
            const char * name,
            int32_t stack_size,
            os_wrapper_thread_func func,
            void * arg,
            uint32_t priority )
```

Creates a new thread.

**Parameters**

| | | |
|---|---|---|
| in | *name* | Name of the thread |
| in | *stack_size* | Size of stack to be allocated for this thread. It can be OS_WRAPPER_DEFAULT_STACK_SIZE to use the default value provided by the underlying RTOS |
| in | *func* | Pointer to the function invoked by thread |
| in | *arg* | Argument to pass to the function invoked by thread |
| in | *priority* | Initial thread priority |

**Returns**

Returns the thread handle created, or NULL in case of error

#### 7.381.2.5 os_wrapper_thread_set_flag()

```
uint32_t os_wrapper_thread_set_flag (
            void * handle,
            uint32_t flags )
```

Set the event flags for synchronizing a thread specified by handle.

**Note**

This function may not be allowed to be called from Interrupt Service Routines.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Thread handle to be notified |
| in | *flags* | Event flags value |

**Returns**

Returns OS_WRAPPER_SUCCESS on success, or OS_WRAPPER_ERROR in case of error

#### 7.381.2.6 os_wrapper_thread_set_flag_isr()

```
uint32_t os_wrapper_thread_set_flag_isr (
            void * handle,
            uint32_t flags )
```

Set the event flags in an interrupt handler for synchronizing a thread specified by handle.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Thread handle to be notified |
| in | *flags* | Event flags value |

**Returns**

Returns OS_WRAPPER_SUCCESS on success, or OS_WRAPPER_ERROR in case of error

Here is the caller graph for this function:

#### 7.381.2.7 os_wrapper_thread_wait_flag()

```
uint32_t os_wrapper_thread_wait_flag (
            uint32_t flags,
            uint32_t timeout )
```

Wait for the event flags for synchronizing threads.

**Note**

This function may not be allowed to be called from Interrupt Service Routines.

**Parameters**

| | | |
|---|---|---|
| in | *flags* | Specify the flags to wait for |
| in | *timeout* | Timeout value |

**Returns**

Returns OS_WRAPPER_SUCCESS on success, or OS_WRAPPER_ERROR in case of error

Here is the caller graph for this function:

## 7.382 secure_fw/spm/core/utilities.c File Reference

```
#include <inttypes.h>
#include "config_spm.h"
#include "fih.h"
#include "utilities.h"
#include "tfm_hal_platform.h"
#include "tfm_spm_log.h"
```

```
#include "static_checks.h"
```

Include dependency graph for utilities.c:

## Functions

- void tfm_core_panic (void)

## 7.382.1 Function Documentation

### 7.382.1.1 tfm_core_panic()

```
void tfm_core_panic (
            void  )
```

Definition at line 18 of file utilities.c.

Here is the call graph for this function:

Here is the caller graph for this function:

## 7.383 secure_fw/spm/include/aapcs_local.h File Reference

```
#include <stdint.h>
```

Include dependency graph for aapcs_local.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- union u64_in_u32_regs_t

## Macros

- #define AAPCS_DUAL_U32_T union u64_in_u32_regs_t
- #define AAPCS_DUAL_U32_SET(v, a0, a1)
- #define AAPCS_DUAL_U32_SET_A0(v, a0) (v).u32_regs.r0 = (uint32_t)(a0)
- #define AAPCS_DUAL_U32_SET_A1(v, a1) (v).u32_regs.r1 = (uint32_t)(a1)
- #define AAPCS_DUAL_U32_AS_U64(v) (v).u64_val

## 7.383.1 Macro Definition Documentation

### 7.383.1.1 AAPCS_DUAL_U32_AS_U64

```
#define AAPCS_DUAL_U32_AS_U64(
            v ) (v).u64_val
```

Definition at line 60 of file aapcs_local.h.

### 7.383.1.2 AAPCS_DUAL_U32_SET

```
#define AAPCS_DUAL_U32_SET(
            v,
            a0,
            a1 )
```

**Value:**

```
                do {                                    \
                    (v).u32_regs.r0 = (uint32_t)(a0);   \
```

```
                    (v).u32_regs.r1 = (uint32_t)(a1);    \
                } while (0)
```

Definition at line 48 of file aapcs_local.h.

#### 7.383.1.3 AAPCS_DUAL_U32_SET_A0

```
#define AAPCS_DUAL_U32_SET_A0(
            v,
            a0 ) (v).u32_regs.r0 = (uint32_t)(a0)
```

Definition at line 54 of file aapcs_local.h.

#### 7.383.1.4 AAPCS_DUAL_U32_SET_A1

```
#define AAPCS_DUAL_U32_SET_A1(
            v,
            a1 ) (v).u32_regs.r1 = (uint32_t)(a1)
```

Definition at line 57 of file aapcs_local.h.

#### 7.383.1.5 AAPCS_DUAL_U32_T

```
#define AAPCS_DUAL_U32_T union u64_in_u32_regs_t
```

Definition at line 46 of file aapcs_local.h.

## 7.384 secure_fw/spm/include/bitops.h File Reference

```
#include <stdint.h>
```

Include dependency graph for bitops.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define IS_ONLY_ONE_BIT_IN_UINT32(n) ((uint32_t)(n) && !((uint32_t)(n) & ((uint32_t)(n)-1)))

### 7.384.1 Macro Definition Documentation

#### 7.384.1.1 IS_ONLY_ONE_BIT_IN_UINT32

```
#define IS_ONLY_ONE_BIT_IN_UINT32(
            n ) ((uint32_t)(n) && !((uint32_t)(n) & ((uint32_t)(n)-1)))
```

Definition at line 13 of file bitops.h.

## 7.385 secure_fw/spm/include/boot/tfm_boot_status.h File Reference

```
#include <stdint.h>
#include <stddef.h>
```

Include dependency graph for tfm_boot_status.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct shared_data_tlv_header
- struct shared_data_tlv_entry
- struct tfm_boot_data

  *Store the data for the runtime SW.*

## Macros

- #define TLV_MAJOR_CORE 0x0
- #define TLV_MAJOR_IAS 0x1
- #define TLV_MAJOR_FWU 0x2
- #define TLV_MAJOR_MBS 0x3
- #define TLV_MAJOR_INVALID 0xF
- #define SW_GENERAL 0x00
- #define SW_BL2 0x01
- #define SW_PROT 0x02
- #define SW_AROT 0x03
- #define SW_SPE 0x04
- #define SW_NSPE 0x05
- #define SW_S_NS 0x06
- #define SW_MAX 0x07
- #define SW_VERSION 0x00
- #define SW_SIGNER_ID 0x01
- #define SW_TYPE 0x03
- #define SW_MEASURE_VALUE 0x08
- #define SW_MEASURE_TYPE 0x09
- #define SW_MEASURE_METADATA 0x0A
- #define SW_MEASURE_VALUE_NON_EXTENDABLE 0x0B
- #define SW_BOOT_RECORD 0x3F
- #define MAJOR_MASK 0xF /∗ 4 bit ∗/
- #define MAJOR_POS 12 /∗ 12 bit ∗/
- #define MINOR_MASK 0xFFF /∗ 12 bit ∗/
- #define MINOR_POS 0
- #define MASK_LEFT_SHIFT(data, mask, position) (((data) & (mask)) << (position))
- #define MASK_RIGHT_SHIFT(data, mask, position) ((uint16_t)((data) & (mask)) >> (position))
- #define SET_TLV_TYPE(major, minor)
- #define GET_MAJOR(tlv_type) ((tlv_type) >> MAJOR_POS)
- #define GET_MINOR(tlv_type) ((tlv_type) & MINOR_MASK)
- #define MODULE_POS 6 /∗ 6 bit ∗/
- #define MODULE_MASK 0x3F /∗ 6 bit ∗/
- #define CLAIM_POS 0
- #define CLAIM_MASK 0x3F /∗ 6 bit ∗/
- #define MEASUREMENT_CLAIM_POS 3 /∗ 3 bit ∗/
- #define GET_IAS_MODULE(tlv_type) MASK_RIGHT_SHIFT(tlv_type, MINOR_MASK, MODULE_POS)

- #define GET_IAS_CLAIM(tlv_type) MASK_RIGHT_SHIFT(tlv_type, CLAIM_MASK, CLAIM_POS)
- #define GET_IAS_MEASUREMENT_CLAIM(ias_claim) MASK_RIGHT_SHIFT(ias_claim, CLAIM_MASK, MEASUREMENT_CLAIM_POS)
- #define SET_IAS_MINOR(sw_module, claim)
- #define SLOT_ID_POS 6
- #define SLOT_ID_MASK 0x3F /* 6 bit */
- #define GET_MBS_SLOT(tlv_type) MASK_RIGHT_SHIFT(tlv_type, MINOR_MASK, SLOT_ID_POS)
- #define GET_MBS_CLAIM(tlv_type) MASK_RIGHT_SHIFT(tlv_type, CLAIM_MASK, CLAIM_POS)
- #define SET_MBS_MINOR(slot_index, claim)
- #define GET_FWU_MODULE(tlv_type) MASK_RIGHT_SHIFT(tlv_type, MINOR_MASK, MODULE_POS)
- #define GET_FWU_CLAIM(tlv_type) MASK_RIGHT_SHIFT(tlv_type, CLAIM_MASK, CLAIM_POS)
- #define SET_FWU_MINOR(sw_module, claim)
- #define SHARED_DATA_TLV_INFO_MAGIC 0x2016
- #define SHARED_DATA_HEADER_SIZE sizeof(struct shared_data_tlv_header)
- #define SHARED_DATA_ENTRY_HEADER_SIZE sizeof(struct shared_data_tlv_entry)
- #define SHARED_DATA_ENTRY_SIZE(size) (size + SHARED_DATA_ENTRY_HEADER_SIZE)

## 7.385.1 Macro Definition Documentation

### 7.385.1.1 CLAIM_MASK

```
#define CLAIM_MASK 0x3F /* 6 bit */
```

Definition at line 117 of file tfm_boot_status.h.

### 7.385.1.2 CLAIM_POS

```
#define CLAIM_POS 0
```

Definition at line 116 of file tfm_boot_status.h.

### 7.385.1.3 GET_FWU_CLAIM

```
#define GET_FWU_CLAIM(
            tlv_type ) MASK_RIGHT_SHIFT(tlv_type, CLAIM_MASK, CLAIM_POS)
```

Definition at line 147 of file tfm_boot_status.h.

### 7.385.1.4 GET_FWU_MODULE

```
#define GET_FWU_MODULE(
            tlv_type ) MASK_RIGHT_SHIFT(tlv_type, MINOR_MASK, MODULE_POS)
```

Definition at line 145 of file tfm_boot_status.h.

### 7.385.1.5 GET_IAS_CLAIM

```
#define GET_IAS_CLAIM(
            tlv_type ) MASK_RIGHT_SHIFT(tlv_type, CLAIM_MASK, CLAIM_POS)
```

Definition at line 124 of file tfm_boot_status.h.

### 7.385.1.6 GET_IAS_MEASUREMENT_CLAIM

```
#define GET_IAS_MEASUREMENT_CLAIM(
            ias_claim ) MASK_RIGHT_SHIFT(ias_claim, CLAIM_MASK, MEASUREMENT_CLAIM_POS)
```

Definition at line 126 of file tfm_boot_status.h.

#### 7.385.1.7 GET_IAS_MODULE

```
#define GET_IAS_MODULE(
            tlv_type ) MASK_RIGHT_SHIFT(tlv_type, MINOR_MASK, MODULE_POS)
```

Definition at line 122 of file tfm_boot_status.h.

#### 7.385.1.8 GET_MAJOR

```
#define GET_MAJOR(
            tlv_type ) ((tlv_type) >> MAJOR_POS)
```

Definition at line 110 of file tfm_boot_status.h.

#### 7.385.1.9 GET_MBS_CLAIM

```
#define GET_MBS_CLAIM(
            tlv_type ) MASK_RIGHT_SHIFT(tlv_type, CLAIM_MASK, CLAIM_POS)
```

Definition at line 138 of file tfm_boot_status.h.

#### 7.385.1.10 GET_MBS_SLOT

```
#define GET_MBS_SLOT(
            tlv_type ) MASK_RIGHT_SHIFT(tlv_type, MINOR_MASK, SLOT_ID_POS)
```

Definition at line 136 of file tfm_boot_status.h.

#### 7.385.1.11 GET_MINOR

```
#define GET_MINOR(
            tlv_type ) ((tlv_type) & MINOR_MASK)
```

Definition at line 111 of file tfm_boot_status.h.

#### 7.385.1.12 MAJOR_MASK

```
#define MAJOR_MASK 0xF /* 4 bit */
```

Definition at line 97 of file tfm_boot_status.h.

#### 7.385.1.13 MAJOR_POS

```
#define MAJOR_POS 12 /* 12 bit */
```

Definition at line 98 of file tfm_boot_status.h.

#### 7.385.1.14 MASK_LEFT_SHIFT

```
#define MASK_LEFT_SHIFT(
            data,
            mask,
            position ) (((data) & (mask)) << (position))
```

Definition at line 102 of file tfm_boot_status.h.

#### 7.385.1.15 MASK_RIGHT_SHIFT

```
#define MASK_RIGHT_SHIFT(
            data,
```

```
                mask,
                position ) ((uint16_t)((data) & (mask)) >> (position))
```

Definition at line 104 of file tfm_boot_status.h.

#### 7.385.1.16 MEASUREMENT_CLAIM_POS

```
#define MEASUREMENT_CLAIM_POS 3 /* 3 bit */
```

Definition at line 120 of file tfm_boot_status.h.

#### 7.385.1.17 MINOR_MASK

```
#define MINOR_MASK 0xFFF /* 12 bit */
```

Definition at line 99 of file tfm_boot_status.h.

#### 7.385.1.18 MINOR_POS

```
#define MINOR_POS 0
```

Definition at line 100 of file tfm_boot_status.h.

#### 7.385.1.19 MODULE_MASK

```
#define MODULE_MASK 0x3F /* 6 bit */
```

Definition at line 115 of file tfm_boot_status.h.

#### 7.385.1.20 MODULE_POS

```
#define MODULE_POS 6 /* 6 bit */
```

Definition at line 114 of file tfm_boot_status.h.

#### 7.385.1.21 SET_FWU_MINOR

```
#define SET_FWU_MINOR(
                sw_module,
                claim )
```

**Value:**

```
    (MASK_LEFT_SHIFT(sw_module, MODULE_MASK, MODULE_POS) | \
     MASK_LEFT_SHIFT(claim, CLAIM_MASK, CLAIM_POS))
```

Definition at line 149 of file tfm_boot_status.h.

#### 7.385.1.22 SET_IAS_MINOR

```
#define SET_IAS_MINOR(
                sw_module,
                claim )
```

**Value:**

```
  (MASK_LEFT_SHIFT(sw_module, MODULE_MASK, MODULE_POS) | \
   MASK_LEFT_SHIFT(claim, CLAIM_MASK, CLAIM_POS))
```

Definition at line 128 of file tfm_boot_status.h.

#### 7.385.1.23 SET_MBS_MINOR

```
#define SET_MBS_MINOR(
            slot_index,
            claim )
```

**Value:**

```
(MASK_LEFT_SHIFT(slot_index, SLOT_ID_MASK, SLOT_ID_POS) | \
 MASK_LEFT_SHIFT(claim, CLAIM_MASK, CLAIM_POS))
```

Definition at line 140 of file tfm_boot_status.h.

#### 7.385.1.24 SET_TLV_TYPE

```
#define SET_TLV_TYPE(
            major,
            minor )
```

**Value:**

```
(MASK_LEFT_SHIFT(major, MAJOR_MASK, MAJOR_POS) | \
 MASK_LEFT_SHIFT(minor, MINOR_MASK, MINOR_POS))
```

Definition at line 107 of file tfm_boot_status.h.

#### 7.385.1.25 SHARED_DATA_ENTRY_HEADER_SIZE

```
#define SHARED_DATA_ENTRY_HEADER_SIZE sizeof(struct shared_data_tlv_entry)
```

Definition at line 194 of file tfm_boot_status.h.

#### 7.385.1.26 SHARED_DATA_ENTRY_SIZE

```
#define SHARED_DATA_ENTRY_SIZE(
            size ) (size + SHARED_DATA_ENTRY_HEADER_SIZE)
```

Definition at line 195 of file tfm_boot_status.h.

#### 7.385.1.27 SHARED_DATA_HEADER_SIZE

```
#define SHARED_DATA_HEADER_SIZE sizeof(struct shared_data_tlv_header)
```

Definition at line 168 of file tfm_boot_status.h.

#### 7.385.1.28 SHARED_DATA_TLV_INFO_MAGIC

```
#define SHARED_DATA_TLV_INFO_MAGIC 0x2016
```

Definition at line 154 of file tfm_boot_status.h.

#### 7.385.1.29 SLOT_ID_MASK

```
#define SLOT_ID_MASK 0x3F /* 6 bit */
```

Definition at line 134 of file tfm_boot_status.h.

#### 7.385.1.30 SLOT_ID_POS

```
#define SLOT_ID_POS 6
```

Definition at line 133 of file tfm_boot_status.h.

#### 7.385.1.31 SW_AROT

```
#define SW_AROT 0x03
```

Definition at line 77 of file tfm_boot_status.h.

#### 7.385.1.32 SW_BL2

```
#define SW_BL2 0x01
```

Definition at line 75 of file tfm_boot_status.h.

#### 7.385.1.33 SW_BOOT_RECORD

```
#define SW_BOOT_RECORD 0x3F
```

Definition at line 94 of file tfm_boot_status.h.

#### 7.385.1.34 SW_GENERAL

```
#define SW_GENERAL 0x00
```

The shared data between boot loader and runtime SW is TLV encoded. The shared data is stored in a well known location in secure memory and this is a contract between boot loader and runtime SW.
The structure of shared data must be the following:

- At the beginning there must be a header: struct shared_data_tlv_header This contains a magic number and a size field which covers the entire size of the shared data area including this header.
- After the header there come the entries which are composed from an entry header structure: struct shared_data_tlv_entry and the data. In the entry header is a type field (tlv_type) which identify the consumer of the entry in the runtime SW and specify the subtype of that data item. There is a size field (tlv_len) which covers the length of the data (not including the entry header structure). After this structure comes the actual data.
- Arbitrary number and size of data entry can be in the shared memory area.

This table gives of overview about the tlv_type field in the entry header. The tlv_type always composed from a major and minor number. Major number identifies the addressee in runtime SW, who should process the data entry. Minor number used to encode more info about the data entry. The actual definition of minor number could change per major number. In case of boot status data, which is going to be processed by initial attestation service the minor number is split further to two part: sw_module and claim. The sw_module identifies the SW component in the system which the data item belongs to and the claim part identifies the exact type of the data.
|----------—| | tlv_type (16) | |----------—| | tlv_major(4)| tlv_minor(12) | |----------—| | MAJOR_IAS | sw_module(6) | claim(6) | |----------—| | MAJOR_MBS | slot ID (6) | claim(6) | |----------—| | MAJOR_CORE | TBD | |----------—|
Definition at line 74 of file tfm_boot_status.h.

#### 7.385.1.35 SW_MAX

```
#define SW_MAX 0x07
```

Definition at line 81 of file tfm_boot_status.h.

#### 7.385.1.36 SW_MEASURE_METADATA

```
#define SW_MEASURE_METADATA 0x0A
```

Definition at line 92 of file tfm_boot_status.h.

#### 7.385.1.37 SW_MEASURE_TYPE

```
#define SW_MEASURE_TYPE 0x09
```

Definition at line 91 of file tfm_boot_status.h.

#### 7.385.1.38 SW_MEASURE_VALUE

```
#define SW_MEASURE_VALUE 0x08
```

Definition at line 90 of file tfm_boot_status.h.

#### 7.385.1.39 SW_MEASURE_VALUE_NON_EXTENDABLE

```
#define SW_MEASURE_VALUE_NON_EXTENDABLE 0x0B
```

Definition at line 93 of file tfm_boot_status.h.

#### 7.385.1.40 SW_NSPE

```
#define SW_NSPE 0x05
```

Definition at line 79 of file tfm_boot_status.h.

#### 7.385.1.41 SW_PROT

```
#define SW_PROT 0x02
```

Definition at line 76 of file tfm_boot_status.h.

#### 7.385.1.42 SW_S_NS

```
#define SW_S_NS 0x06
```

Definition at line 80 of file tfm_boot_status.h.

#### 7.385.1.43 SW_SIGNER_ID

```
#define SW_SIGNER_ID 0x01
```

Definition at line 86 of file tfm_boot_status.h.

#### 7.385.1.44 SW_SPE

```
#define SW_SPE 0x04
```

Definition at line 78 of file tfm_boot_status.h.

#### 7.385.1.45 SW_TYPE

```
#define SW_TYPE 0x03
```

Definition at line 88 of file tfm_boot_status.h.

#### 7.385.1.46 SW_VERSION

```
#define SW_VERSION 0x00
```

Definition at line 85 of file tfm_boot_status.h.

#### 7.385.1.47 TLV_MAJOR_CORE

```
#define TLV_MAJOR_CORE 0x0
```

Definition at line 22 of file tfm_boot_status.h.

#### 7.385.1.48 TLV_MAJOR_FWU

```
#define TLV_MAJOR_FWU 0x2
```

Definition at line 24 of file tfm_boot_status.h.

#### 7.385.1.49 TLV_MAJOR_IAS

```
#define TLV_MAJOR_IAS 0x1
```

Definition at line 23 of file tfm_boot_status.h.

#### 7.385.1.50 TLV_MAJOR_INVALID

```
#define TLV_MAJOR_INVALID 0xF
```

Definition at line 26 of file tfm_boot_status.h.

#### 7.385.1.51 TLV_MAJOR_MBS

```
#define TLV_MAJOR_MBS 0x3
```

Definition at line 25 of file tfm_boot_status.h.

## 7.386 secure_fw/spm/include/config_spm.h File Reference

```
#include "config_impl.h"
#include "config_tfm.h"
```

Include dependency graph for config_spm.h:

This graph shows which files directly or indirectly include this file:

## 7.387 secure_fw/spm/include/critical_section.h File Reference

```
#include <stdint.h>
#include "tfm_arch.h"
```

Include dependency graph for critical_section.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct critical_section_t

### Macros

- #define CRITICAL_SECTION_STATIC_INIT {.state = 0,}
- #define CRITICAL_SECTION_INIT(cs) (cs).state = (0)
- #define CRITICAL_SECTION_ENTER(cs) (cs).state = __save_disable_irq()
- #define CRITICAL_SECTION_LEAVE(cs) __restore_irq((cs).state)

### 7.387.1 Macro Definition Documentation

#### 7.387.1.1 CRITICAL_SECTION_ENTER

```
#define CRITICAL_SECTION_ENTER(
            cs ) (cs).state = __save_disable_irq()
```

Definition at line 20 of file critical_section.h.

#### 7.387.1.2 CRITICAL_SECTION_INIT

```
#define CRITICAL_SECTION_INIT(
            cs ) (cs).state = (0)
```

Definition at line 19 of file critical_section.h.

#### 7.387.1.3 CRITICAL_SECTION_LEAVE

```
#define CRITICAL_SECTION_LEAVE(
            cs ) __restore_irq((cs).state)
```

Definition at line 21 of file critical_section.h.

#### 7.387.1.4 CRITICAL_SECTION_STATIC_INIT

```
#define CRITICAL_SECTION_STATIC_INIT {.state = 0,}
```

Definition at line 18 of file critical_section.h.

## 7.388 secure_fw/spm/include/current.h File Reference

```
#include "thread.h"
#include "spm.h"
```

Include dependency graph for current.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define GET_CURRENT_COMPONENT() GET_CTX_OWNER(CURRENT_THREAD->p_context_ctrl)
- #define SET_CURRENT_COMPONENT(p) THRD_UPDATE_CUR_CTXCTRL(&(p)->ctx_ctrl)

### 7.388.1 Macro Definition Documentation

#### 7.388.1.1 GET_CURRENT_COMPONENT

```
#define GET_CURRENT_COMPONENT( ) GET_CTX_OWNER(CURRENT_THREAD->p_context_ctrl)
```

Definition at line 15 of file current.h.

#### 7.388.1.2 SET_CURRENT_COMPONENT

```
#define SET_CURRENT_COMPONENT(
            p ) THRD_UPDATE_CUR_CTXCTRL(&(p)->ctx_ctrl)
```

Definition at line 17 of file current.h.

## 7.389 secure_fw/spm/include/ffm/backend.h File Reference

```
#include <stdint.h>
#include "spm.h"
#include "tfm_arch.h"
#include "load/spm_load_api.h"
#include "psa/error.h"
```

Include dependency graph for backend.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define PARTITION_LIST_ADDR (&partition_listhead)
- #define SPM_THREAD_CONTEXT p_spm_thread_context

### Functions

- void backend_init_comp_assuredly (struct partition_t *p_pt, uint32_t service_setting)
- uint32_t backend_system_run (void)
- psa_status_t backend_messaging (struct connection_t *p_connection)
- psa_status_t backend_replying (struct connection_t *handle, int32_t status)
- psa_signal_t backend_wait_signals (struct partition_t *p_pt, psa_signal_t signals)

  *Set the wait signal pattern in current partition.*
- psa_status_t backend_assert_signal (struct partition_t *p_pt, psa_signal_t signal)

  *Set the asserted signal pattern in current partition.*

### Variables

- struct partition_head_t partition_listhead
- struct context_ctrl_t * p_spm_thread_context

### 7.389.1 Macro Definition Documentation

#### 7.389.1.1 PARTITION_LIST_ADDR

```
#define PARTITION_LIST_ADDR (&partition_listhead)
```

Definition at line 60 of file backend.h.

#### 7.389.1.2 SPM_THREAD_CONTEXT

```
#define SPM_THREAD_CONTEXT p_spm_thread_context
```

Definition at line 64 of file backend.h.

### 7.389.2 Function Documentation

#### 7.389.2.1 backend_assert_signal()

```
psa_status_t backend_assert_signal (
            struct partition_t * p_pt,
            psa_signal_t signal )
```

Set the asserted signal pattern in current partition.
Definition at line 385 of file backend_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.389.2.2 backend_init_comp_assuredly()

```
void backend_init_comp_assuredly (
            struct partition_t * p_pt,
            uint32_t service_setting )
```

Definition at line 304 of file backend_ipc.c.

Here is the call graph for this function:

#### 7.389.2.3 backend_messaging()

```
psa_status_t backend_messaging (
            struct connection_t * p_connection )
```

Definition at line 174 of file backend_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.389.2.4 backend_replying()

```
psa_status_t backend_replying (
            struct connection_t * handle,
            int32_t status )
```

Definition at line 212 of file backend_ipc.c.

#### 7.389.2.5 backend_system_run()

```
uint32_t backend_system_run (
            void  )
```

Definition at line 329 of file backend_ipc.c.
Here is the call graph for this function:

#### 7.389.2.6 backend_wait_signals()

```
psa_signal_t backend_wait_signals (
            struct partition_t * p_pt,
            psa_signal_t signals )
```

Set the wait signal pattern in current partition.
Definition at line 364 of file backend_ipc.c.
Here is the call graph for this function:

Here is the caller graph for this function:

### 7.389.3 Variable Documentation

#### 7.389.3.1 p_spm_thread_context

```
struct context_ctrl_t* p_spm_thread_context
```

Definition at line 53 of file backend_ipc.c.

#### 7.389.3.2 partition_listhead

```
struct partition_head_t partition_listhead
```

Definition at line 37 of file backend_ipc.c.

## 7.390 secure_fw/spm/include/ffm/backend_ipc.h File Reference

```
#include <stdint.h>
#include "svc_num.h"
```

Include dependency graph for backend_ipc.h:

### Macros

- #define BACKEND_SERVICE_SET(set, p_service) ((set) |= (p_service)->signal)
- #define BACKEND_SPM_INIT()

### Functions

- uint64_t backend_abi_entering_spm (void)
- uint32_t backend_abi_leaving_spm (uint32_t result)

### 7.390.1 Macro Definition Documentation

#### 7.390.1.1 BACKEND_SERVICE_SET

```
#define BACKEND_SERVICE_SET(
            set,
            p_service ) ((set) |= (p_service)->signal)
```

Definition at line 15 of file backend_ipc.h.

#### 7.390.1.2 BACKEND_SPM_INIT

```
#define BACKEND_SPM_INIT( )
```

**Value:**

```
__ASM volatile("SVC %0          \n"         \
"BX LR            \n"         \
:  :  "I" (TFM_SVC_SPM_INIT))
```

Definition at line 18 of file backend_ipc.h.

### 7.390.2 Function Documentation

#### 7.390.2.1 backend_abi_entering_spm()

```
uint64_t backend_abi_entering_spm (
            void  )
```

Definition at line 405 of file backend_ipc.c.
Here is the call graph for this function:

#### 7.390.2.2 backend_abi_leaving_spm()

```
uint32_t backend_abi_leaving_spm (
            uint32_t result )
```

Definition at line 436 of file backend_ipc.c.
Here is the call graph for this function:

## 7.391 secure_fw/spm/include/ffm/backend_sfn.h File Reference

### Macros

- #define BACKEND_SERVICE_SET(set, p_service)
- #define BACKEND_SPM_INIT() tfm_spm_init()

### 7.391.1 Macro Definition Documentation

#### 7.391.1.1 BACKEND_SERVICE_SET

```
#define BACKEND_SERVICE_SET(
            set,
            p_service )
```

Definition at line 12 of file backend_sfn.h.

#### 7.391.1.2 BACKEND_SPM_INIT

```
#define BACKEND_SPM_INIT( ) tfm_spm_init()
```

Definition at line 14 of file backend_sfn.h.

## 7.392 secure_fw/spm/include/ffm/mailbox_agent_api.h File Reference

```
#include <stdint.h>
#include "config_impl.h"
#include "psa/client.h"
#include "psa/service.h"
```

Include dependency graph for mailbox_agent_api.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct client_params_t

### Macros

- #define agent_psa_connect NULL
- #define agent_psa_close NULL

## Functions

- psa_status_t agent_psa_call (psa_handle_t handle, uint32_t control, const struct client_params_t ∗params, const void ∗client_data_stateless)

  *Specific psa_call() variants for agents.*

## 7.392.1 Macro Definition Documentation

### 7.392.1.1 agent_psa_close

```
#define agent_psa_close NULL
```

Definition at line 92 of file mailbox_agent_api.h.

### 7.392.1.2 agent_psa_connect

```
#define agent_psa_connect NULL
```

Definition at line 91 of file mailbox_agent_api.h.

## 7.392.2 Function Documentation

### 7.392.2.1 agent_psa_call()

```
psa_status_t agent_psa_call (
            psa_handle_t handle,
            uint32_t control,
            const struct client_params_t * params,
            const void * client_data_stateless )
```

Specific psa_call() variants for agents.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Handle to the service being accessed. |
| in | *control* | A composited uint32_t value for controlling purpose, containing call types, numbers of in/out vectors and attributes of vectors. |
| in | *params* | Combines the psa_invec and psa_outvec params for the psa_call() to be made, as well as NS agent's client identifier, which is ignored for connection-based services. |
| in | *client_data_stateless* | Client data, treated as opaque by SPM. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |
| *Does not return* | The call is invalid, one or more of the following are true:<br>• An invalid handle was passed.<br>• The connection is already handling a request.<br>• An invalid memory reference was provided.<br>• in_num + out_num > PSA_MAX_IOVEC.<br>• The message is unrecognized by the RoT Service or incorrectly formatted. |

## 7.393 secure_fw/spm/include/ffm/original_vector_api.h File Reference

```
#include <stdint.h>
#include "config_impl.h"
#include "psa/client.h"
```

Include dependency graph for original_vector_api.h:

### Data Structures

- struct tfm_original_iovec_t

### Typedefs

- typedef struct tfm_original_iovec_t tfm_original_iovec_t

### Functions

- psa_status_t original_iovec (psa_handle_t msg_handle, tfm_original_iovec_t *io_vec)

  *Retrieve original invec and outvec as passed to psa_call()*

### 7.393.1 Typedef Documentation

#### 7.393.1.1 tfm_original_iovec_t

```
typedef struct tfm_original_iovec_t tfm_original_iovec_t
```

Structure to hold original input and output vectors and their counts

### 7.393.2 Function Documentation

#### 7.393.2.1 original_iovec()

```
psa_status_t original_iovec (
            psa_handle_t msg_handle,
            tfm_original_iovec_t * io_vec )
```

Retrieve original invec and outvec as passed to psa_call()

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| out | *io_vec* | Pointer to tfm_original_iovec_t to be filled. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |

Here is the caller graph for this function:

## 7.394 secure_fw/spm/include/ffm/psa_api.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "config_spm.h"
#include "psa/client.h"
#include "psa/service.h"
```

Include dependency graph for psa_api.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define tfm_spm_agent_psa_connect NULL
- #define tfm_spm_agent_psa_close NULL
- #define tfm_spm_agent_psa_call NULL
- #define tfm_spm_client_psa_connect NULL
- #define tfm_spm_client_psa_close NULL
- #define tfm_spm_partition_psa_notify NULL
- #define tfm_spm_partition_psa_clear NULL
- #define tfm_spm_partition_psa_set_rhandle NULL
- #define tfm_spm_partition_psa_irq_enable NULL
- #define tfm_spm_partition_psa_irq_disable NULL
- #define tfm_spm_partition_psa_reset_signal NULL
- #define tfm_spm_partition_psa_eoi NULL
- #define tfm_spm_partition_original_iovec NULL

## Functions

- void spm_handle_programmer_errors (psa_status_t status)

  *This function handles the specific programmer error cases.*
- uint32_t tfm_spm_get_lifecycle_state (void)

  *This function get the current PSA RoT lifecycle state.*
- uint32_t tfm_spm_client_psa_framework_version (void)

  *handler for psa_framework_version.*
- uint32_t tfm_spm_client_psa_version (uint32_t sid)

  *handler for psa_version.*
- psa_status_t tfm_spm_client_psa_call (psa_handle_t handle, uint32_t ctrl_param, const psa_invec *inptr, psa_outvec *outptr)

  *handler for psa_call.*
- size_t tfm_spm_partition_psa_read (psa_handle_t msg_handle, uint32_t invec_idx, void *buffer, size_t num_bytes)

  *Function body of psa_read.*
- size_t tfm_spm_partition_psa_skip (psa_handle_t msg_handle, uint32_t invec_idx, size_t num_bytes)

  *Function body of psa_skip.*
- psa_status_t tfm_spm_partition_psa_write (psa_handle_t msg_handle, uint32_t outvec_idx, const void *buffer, size_t num_bytes)

  *Function body of psa_write.*
- int32_t tfm_spm_partition_psa_reply (psa_handle_t msg_handle, psa_status_t status)

  *Function body of psa_reply.*
- psa_status_t tfm_spm_partition_psa_panic (void)

  *Function body of psa_panic.*

### 7.394.1 Macro Definition Documentation

#### 7.394.1.1 tfm_spm_agent_psa_call

```
#define tfm_spm_agent_psa_call NULL
```

Definition at line 157 of file psa_api.h.

#### 7.394.1.2 tfm_spm_agent_psa_close

```
#define tfm_spm_agent_psa_close NULL
```

Definition at line 156 of file psa_api.h.

#### 7.394.1.3 tfm_spm_agent_psa_connect

```
#define tfm_spm_agent_psa_connect NULL
```

Definition at line 155 of file psa_api.h.

#### 7.394.1.4 tfm_spm_client_psa_close

```
#define tfm_spm_client_psa_close NULL
```

Definition at line 269 of file psa_api.h.

#### 7.394.1.5 tfm_spm_client_psa_connect

```
#define tfm_spm_client_psa_connect NULL
```

Definition at line 268 of file psa_api.h.

#### 7.394.1.6 tfm_spm_partition_original_iovec

```
#define tfm_spm_partition_original_iovec NULL
```

Definition at line 576 of file psa_api.h.

#### 7.394.1.7 tfm_spm_partition_psa_clear

```
#define tfm_spm_partition_psa_clear NULL
```

Definition at line 432 of file psa_api.h.

#### 7.394.1.8 tfm_spm_partition_psa_eoi

```
#define tfm_spm_partition_psa_eoi NULL
```

Definition at line 539 of file psa_api.h.

#### 7.394.1.9 tfm_spm_partition_psa_irq_disable

```
#define tfm_spm_partition_psa_irq_disable NULL
```

Definition at line 496 of file psa_api.h.

#### 7.394.1.10 tfm_spm_partition_psa_irq_enable

```
#define tfm_spm_partition_psa_irq_enable NULL
```

Definition at line 495 of file psa_api.h.

#### 7.394.1.11 tfm_spm_partition_psa_notify

```
#define tfm_spm_partition_psa_notify NULL
```

Definition at line 431 of file psa_api.h.

#### 7.394.1.12 tfm_spm_partition_psa_reset_signal

```
#define tfm_spm_partition_psa_reset_signal NULL
```

Definition at line 519 of file psa_api.h.

#### 7.394.1.13 tfm_spm_partition_psa_set_rhandle

```
#define tfm_spm_partition_psa_set_rhandle NULL
```

Definition at line 458 of file psa_api.h.

### 7.394.2 Function Documentation

#### 7.394.2.1 spm_handle_programmer_errors()

```
void spm_handle_programmer_errors (
            psa_status_t status )
```

This function handles the specific programmer error cases.

**Parameters**

| | | |
|---|---|---|
| in | *status* | Standard error codes for the SPM. |

**Return values**

| | |
|---|---|
| *void* | Status will not cause SPM panic |
| *SPM panic* | Following programmer errors are triggered by SP:<br>• PSA_ERROR_PROGRAMMER_ERROR<br>• PSA_ERROR_CONNECTION_REFUSED<br>• PSA_ERROR_CONNECTION_BUSY |

Definition at line 34 of file psa_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.394.2.2 tfm_spm_client_psa_call()

```
psa_status_t tfm_spm_client_psa_call (
            psa_handle_t handle,
            uint32_t ctrl_param,
            const psa_invec * inptr,
            psa_outvec * outptr )
```

handler for psa_call.

**Parameters**

| | | |
|---|---|---|
| in | *handle* | Service handle to the established connection, psa_handle_t |
| in | *ctrl_param* | Parameters combined in uint32_t, includes request type, in_num and out_num. |
| in | *inptr* | Array of input psa_invec structures. psa_invec |
| in | *outptr* | Array of output psa_outvec structures. psa_outvec |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |
| *Does not return* | The call is invalid, one or more of the following are true:<br>• An invalid handle was passed.<br>• The connection is already handling a request.<br>• An invalid memory reference was provided.<br>• in_num + out_num > PSA_MAX_IOVEC.<br>• The message is unrecognized by the RoT Service or incorrectly formatted. |

Definition at line 162 of file psa_call_api.c.

#### 7.394.2.3 tfm_spm_client_psa_framework_version()

```
uint32_t tfm_spm_client_psa_framework_version (
            void )
```

handler for psa_framework_version.

**Returns**

version The version of the PSA Framework implementation that is providing the runtime services.

Definition at line 15 of file psa_version_api.c.
Here is the caller graph for this function:

### 7.394.2.4 tfm_spm_client_psa_version()

```
uint32_t tfm_spm_client_psa_version (
            uint32_t sid )
```

handler for psa_version.

**Parameters**

| | | |
|---|---|---|
| in | *sid* | RoT Service identity. |

**Return values**

| | |
|---|---|
| *PSA_VERSION_NONE* | The RoT Service is not implemented, or the caller is not permitted to access the service. |
| > | 0 The version of the implemented RoT Service. |

Definition at line 20 of file psa_version_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.394.2.5 tfm_spm_get_lifecycle_state()

```
uint32_t tfm_spm_get_lifecycle_state (
            void  )
```

This function get the current PSA RoT lifecycle state.

**Returns**

state The current security lifecycle state of the PSA RoT. The PSA state and implementation state are encoded as follows:

- state[15:8] – PSA lifecycle state
- state[7:0] – IMPLEMENTATION DEFINED state

Definition at line 44 of file psa_api.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.394.2.6 tfm_spm_partition_psa_panic()

```
psa_status_t tfm_spm_partition_psa_panic (
            void  )
```

Function body of psa_panic.

**Return values**

| | |
|---|---|
| *Should not return* | |

Definition at line 358 of file psa_api.c.
Here is the caller graph for this function:

### 7.394.2.7 tfm_spm_partition_psa_read()

```
size_t tfm_spm_partition_psa_read (
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            void * buffer,
            size_t num_bytes )
```

Function body of psa_read.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of the input vector to read from. Must be less than PSA_MAX_IOVEC. |
| out | *buffer* | Buffer in the Secure Partition to copy the requested data to. |
| in | *num_bytes* | Maximum number of bytes to be read from the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes copied. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a PSA_IPC_CALL message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• the memory reference for buffer is invalid or not writable. |

Definition at line 17 of file psa_read_write_skip_api.c.

Here is the caller graph for this function:

#### 7.394.2.8 tfm_spm_partition_psa_reply()

```
int32_t tfm_spm_partition_psa_reply (
            psa_handle_t msg_handle,
            psa_status_t status )
```

Function body of psa_reply.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *status* | Message result value to be reported to the client. |

**Return values**

| | |
|---|---|
| *Positive* | integer Success, the connection handle. |
| *PSA_SUCCESS* | Success |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• An invalid status code is specified for the type of message. |

Definition at line 201 of file psa_api.c.
Here is the caller graph for this function:

#### 7.394.2.9 tfm_spm_partition_psa_skip()

```
size_t tfm_spm_partition_psa_skip (
```

```
            psa_handle_t msg_handle,
            uint32_t invec_idx,
            size_t num_bytes )
```

Function body of psa_skip.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| in | *invec_idx* | Index of input vector to skip from. Must be less than PSA_MAX_IOVEC. |
| in | *num_bytes* | Maximum number of bytes to skip in the client input vector. |

**Return values**

| | |
|---|---|
| *>0* | Number of bytes skipped. |
| *0* | There was no remaining data in this input vector. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• invec_idx is equal to or greater than PSA_MAX_IOVEC. |

Definition at line 88 of file psa_read_write_skip_api.c.
Here is the caller graph for this function:

### 7.394.2.10 tfm_spm_partition_psa_write()

```
psa_status_t tfm_spm_partition_psa_write (
            psa_handle_t msg_handle,
            uint32_t outvec_idx,
            const void * buffer,
            size_t num_bytes )
```

Function body of psa_write.

**Parameters**

| | | |
|---|---|---|
| in | *msg_handle* | Handle for the client's message. |
| out | *outvec_idx* | Index of output vector in message to write to. Must be less than PSA_MAX_IOVEC. |
| in | *buffer* | Buffer with the data to write. |
| in | *num_bytes* | Number of bytes to write to the client output vector. |

**Return values**

| | |
|---|---|
| *PSA_SUCCESS* | Success. |
| *PROGRAMMER ERROR* | The call is invalid, one or more of the following are true:<br>• msg_handle is invalid.<br>• msg_handle does not refer to a request message.<br>• outvec_idx is equal to or greater than PSA_MAX_IOVEC.<br>• The memory reference for buffer is invalid.<br>• The call attempts to write data past the end of the client output vector. |

Definition at line 148 of file psa_read_write_skip_api.c.
Here is the caller graph for this function:

## 7.395 secure_fw/spm/include/interface/runtime_defs.h File Reference

```
#include <stdint.h>
#include "config_impl.h"
#include "psa/client.h"
#include "psa/error.h"
#include "psa/service.h"
#include "ffm/mailbox_agent_api.h"
```

Include dependency graph for runtime_defs.h:

This graph shows which files directly or indirectly include this file:

## Typedefs

- typedef psa_status_t(∗ service_fn_t) (psa_msg_t ∗msg)
- typedef psa_status_t(∗ sfn_init_fn_t) (void ∗param)

### 7.395.1 Typedef Documentation

#### 7.395.1.1 service_fn_t

```
typedef psa_status_t(* service_fn_t) (psa_msg_t *msg)
```

Definition at line 26 of file runtime_defs.h.

#### 7.395.1.2 sfn_init_fn_t

```
typedef psa_status_t(* sfn_init_fn_t) (void *param)
```

Definition at line 27 of file runtime_defs.h.

## 7.396 secure_fw/spm/include/interface/svc_num.h File Reference

This graph shows which files directly or indirectly include this file:

## Macros

- #define TFM_SVC_NUM_PLATFORM_POS 7U
- #define TFM_SVC_NUM_PLATFORM_MSK (1U << TFM_SVC_NUM_PLATFORM_POS)
- #define TFM_SVC_NUM_HANDLER_MODE_POS 6U
- #define TFM_SVC_NUM_HANDLER_MODE_MSK (1U << TFM_SVC_NUM_HANDLER_MODE_POS)
- #define TFM_SVC_NUM_PSA_API_POS 5U
- #define TFM_SVC_NUM_PSA_API_MSK (1U << TFM_SVC_NUM_PSA_API_POS)
- #define TFM_SVC_NUM_INDEX_MSK (0x1FU)
- #define TFM_SVC_NUM_SPM_THREAD(index) ((index) & TFM_SVC_NUM_INDEX_MSK)
- #define TFM_SVC_NUM_SPM_HANDLER(index)
- #define TFM_SVC_NUM_PSA_API_THREAD(index)
- #define TFM_SVC_NUM_PLATFORM_THREAD(index)
- #define TFM_SVC_NUM_PLATFORM_HANDLER(index)
- #define TFM_SVC_SPM_INIT TFM_SVC_NUM_SPM_THREAD(0)
- #define TFM_SVC_FLIH_FUNC_RETURN TFM_SVC_NUM_SPM_THREAD(1)
- #define TFM_SVC_OUTPUT_UNPRIV_STRING TFM_SVC_NUM_SPM_THREAD(2)
- #define TFM_SVC_GET_BOOT_DATA TFM_SVC_NUM_SPM_THREAD(3)
- #define TFM_SVC_THREAD_MODE_SPM_RETURN TFM_SVC_NUM_SPM_THREAD(4)

- #define TFM_SVC_PREPARE_DEPRIV_FLIH TFM_SVC_NUM_SPM_HANDLER(0)
- #define TFM_SVC_PSA_FRAMEWORK_VERSION TFM_SVC_NUM_PSA_API_THREAD(0)
- #define TFM_SVC_PSA_VERSION TFM_SVC_NUM_PSA_API_THREAD(1)
- #define TFM_SVC_PSA_CALL TFM_SVC_NUM_PSA_API_THREAD(2)
- #define TFM_SVC_PSA_CONNECT TFM_SVC_NUM_PSA_API_THREAD(3)
- #define TFM_SVC_PSA_CLOSE TFM_SVC_NUM_PSA_API_THREAD(4)
- #define TFM_SVC_PSA_WAIT TFM_SVC_NUM_PSA_API_THREAD(5)
- #define TFM_SVC_PSA_GET TFM_SVC_NUM_PSA_API_THREAD(6)
- #define TFM_SVC_PSA_SET_RHANDLE TFM_SVC_NUM_PSA_API_THREAD(7)
- #define TFM_SVC_PSA_READ TFM_SVC_NUM_PSA_API_THREAD(8)
- #define TFM_SVC_PSA_SKIP TFM_SVC_NUM_PSA_API_THREAD(9)
- #define TFM_SVC_PSA_WRITE TFM_SVC_NUM_PSA_API_THREAD(10)
- #define TFM_SVC_PSA_REPLY TFM_SVC_NUM_PSA_API_THREAD(11)
- #define TFM_SVC_PSA_NOTIFY TFM_SVC_NUM_PSA_API_THREAD(12)
- #define TFM_SVC_PSA_CLEAR TFM_SVC_NUM_PSA_API_THREAD(13)
- #define TFM_SVC_PSA_EOI TFM_SVC_NUM_PSA_API_THREAD(14)
- #define TFM_SVC_PSA_PANIC TFM_SVC_NUM_PSA_API_THREAD(15)
- #define TFM_SVC_PSA_LIFECYCLE TFM_SVC_NUM_PSA_API_THREAD(16)
- #define TFM_SVC_PSA_IRQ_ENABLE TFM_SVC_NUM_PSA_API_THREAD(17)
- #define TFM_SVC_PSA_IRQ_DISABLE TFM_SVC_NUM_PSA_API_THREAD(18)
- #define TFM_SVC_PSA_RESET_SIGNAL TFM_SVC_NUM_PSA_API_THREAD(19)
- #define TFM_SVC_AGENT_PSA_CALL TFM_SVC_NUM_PSA_API_THREAD(20)
- #define TFM_SVC_AGENT_PSA_CONNECT TFM_SVC_NUM_PSA_API_THREAD(21)
- #define TFM_SVC_AGENT_PSA_CLOSE TFM_SVC_NUM_PSA_API_THREAD(22)
- #define TFM_SVC_NSID_SHM_INIT TFM_SVC_NUM_PSA_API_THREAD(23)
- #define TFM_SVC_ORIGINAL_IOVEC TFM_SVC_NUM_PSA_API_THREAD(24)
- #define TFM_SVC_IS_PLATFORM(svc_num) (!!((svc_num) & TFM_SVC_NUM_PLATFORM_MSK))
- #define TFM_SVC_IS_HANDLER_MODE(svc_num) (!!((svc_num) & TFM_SVC_NUM_HANDLER_MODE_MSK))
- #define TFM_SVC_IS_PSA_API(svc_num)
- #define TFM_SVC_IS_SPM(svc_num)

## 7.396.1 Macro Definition Documentation

### 7.396.1.1 TFM_SVC_AGENT_PSA_CALL

```
#define TFM_SVC_AGENT_PSA_CALL TFM_SVC_NUM_PSA_API_THREAD(20)
```

Definition at line 105 of file svc_num.h.

### 7.396.1.2 TFM_SVC_AGENT_PSA_CLOSE

```
#define TFM_SVC_AGENT_PSA_CLOSE TFM_SVC_NUM_PSA_API_THREAD(22)
```

Definition at line 107 of file svc_num.h.

### 7.396.1.3 TFM_SVC_AGENT_PSA_CONNECT

```
#define TFM_SVC_AGENT_PSA_CONNECT TFM_SVC_NUM_PSA_API_THREAD(21)
```

Definition at line 106 of file svc_num.h.

### 7.396.1.4 TFM_SVC_FLIH_FUNC_RETURN

```
#define TFM_SVC_FLIH_FUNC_RETURN TFM_SVC_NUM_SPM_THREAD(1)
```

Definition at line 74 of file svc_num.h.

#### 7.396.1.5 TFM_SVC_GET_BOOT_DATA

```
#define TFM_SVC_GET_BOOT_DATA TFM_SVC_NUM_SPM_THREAD(3)
```

Definition at line 76 of file svc_num.h.

#### 7.396.1.6 TFM_SVC_IS_HANDLER_MODE

```
#define TFM_SVC_IS_HANDLER_MODE(
            svc_num ) (!!((svc_num) & TFM_SVC_NUM_HANDLER_MODE_MSK))
```

Definition at line 112 of file svc_num.h.

#### 7.396.1.7 TFM_SVC_IS_PLATFORM

```
#define TFM_SVC_IS_PLATFORM(
            svc_num ) (!!((svc_num) & TFM_SVC_NUM_PLATFORM_MSK))
```

Definition at line 111 of file svc_num.h.

#### 7.396.1.8 TFM_SVC_IS_PSA_API

```
#define TFM_SVC_IS_PSA_API(
            svc_num )
```

**Value:**

```
(((svc_num) & (TFM_SVC_NUM_PLATFORM_MSK | \
TFM_SVC_NUM_PSA_API_MSK)) \
            == TFM_SVC_NUM_PSA_API_MSK)
```

Definition at line 113 of file svc_num.h.

#### 7.396.1.9 TFM_SVC_IS_SPM

```
#define TFM_SVC_IS_SPM(
            svc_num )
```

**Value:**

```
(((svc_num) & (TFM_SVC_NUM_PLATFORM_MSK | \
TFM_SVC_NUM_PSA_API_MSK)) == 0)
```

Definition at line 116 of file svc_num.h.

#### 7.396.1.10 TFM_SVC_NSID_SHM_INIT

```
#define TFM_SVC_NSID_SHM_INIT TFM_SVC_NUM_PSA_API_THREAD(23)
```

Definition at line 108 of file svc_num.h.

#### 7.396.1.11 TFM_SVC_NUM_HANDLER_MODE_MSK

```
#define TFM_SVC_NUM_HANDLER_MODE_MSK (1U << TFM_SVC_NUM_HANDLER_MODE_POS)
```

Definition at line 47 of file svc_num.h.

#### 7.396.1.12 TFM_SVC_NUM_HANDLER_MODE_POS

```
#define TFM_SVC_NUM_HANDLER_MODE_POS 6U
```

Definition at line 46 of file svc_num.h.

#### 7.396.1.13 TFM_SVC_NUM_INDEX_MSK

```
#define TFM_SVC_NUM_INDEX_MSK (0x1FU)
```

Definition at line 52 of file svc_num.h.

#### 7.396.1.14 TFM_SVC_NUM_PLATFORM_HANDLER

```
#define TFM_SVC_NUM_PLATFORM_HANDLER(
            index )
```

**Value:**

```
(TFM_SVC_NUM_PLATFORM_MSK | \
TFM_SVC_NUM_HANDLER_MODE_MSK |\
((index) & TFM_SVC_NUM_INDEX_MSK))
```

Definition at line 68 of file svc_num.h.

#### 7.396.1.15 TFM_SVC_NUM_PLATFORM_MSK

```
#define TFM_SVC_NUM_PLATFORM_MSK (1U << TFM_SVC_NUM_PLATFORM_POS)
```

Definition at line 44 of file svc_num.h.

#### 7.396.1.16 TFM_SVC_NUM_PLATFORM_POS

```
#define TFM_SVC_NUM_PLATFORM_POS 7U
```

Definition at line 43 of file svc_num.h.

#### 7.396.1.17 TFM_SVC_NUM_PLATFORM_THREAD

```
#define TFM_SVC_NUM_PLATFORM_THREAD(
            index )
```

**Value:**

```
(TFM_SVC_NUM_PLATFORM_MSK | \
((index) & TFM_SVC_NUM_INDEX_MSK))
```

Definition at line 65 of file svc_num.h.

#### 7.396.1.18 TFM_SVC_NUM_PSA_API_MSK

```
#define TFM_SVC_NUM_PSA_API_MSK (1U << TFM_SVC_NUM_PSA_API_POS)
```

Definition at line 50 of file svc_num.h.

#### 7.396.1.19 TFM_SVC_NUM_PSA_API_POS

```
#define TFM_SVC_NUM_PSA_API_POS 5U
```

Definition at line 49 of file svc_num.h.

#### 7.396.1.20 TFM_SVC_NUM_PSA_API_THREAD

```
#define TFM_SVC_NUM_PSA_API_THREAD(
            index )
```

**Value:**

```
(TFM_SVC_NUM_PSA_API_MSK | \
((index) & TFM_SVC_NUM_INDEX_MSK))
```

Definition at line 62 of file svc_num.h.

#### 7.396.1.21 TFM_SVC_NUM_SPM_HANDLER

```
#define TFM_SVC_NUM_SPM_HANDLER(
            index )
```

**Value:**

```
(TFM_SVC_NUM_HANDLER_MODE_MSK | \
((index) & TFM_SVC_NUM_INDEX_MSK))
```

Definition at line 59 of file svc_num.h.

#### 7.396.1.22 TFM_SVC_NUM_SPM_THREAD

```
#define TFM_SVC_NUM_SPM_THREAD(
            index ) ((index) & TFM_SVC_NUM_INDEX_MSK)
```

Definition at line 57 of file svc_num.h.

#### 7.396.1.23 TFM_SVC_ORIGINAL_IOVEC

```
#define TFM_SVC_ORIGINAL_IOVEC TFM_SVC_NUM_PSA_API_THREAD(24)
```

Definition at line 109 of file svc_num.h.

#### 7.396.1.24 TFM_SVC_OUTPUT_UNPRIV_STRING

```
#define TFM_SVC_OUTPUT_UNPRIV_STRING TFM_SVC_NUM_SPM_THREAD(2)
```

Definition at line 75 of file svc_num.h.

#### 7.396.1.25 TFM_SVC_PREPARE_DEPRIV_FLIH

```
#define TFM_SVC_PREPARE_DEPRIV_FLIH TFM_SVC_NUM_SPM_HANDLER(0)
```

Definition at line 80 of file svc_num.h.

#### 7.396.1.26 TFM_SVC_PSA_CALL

```
#define TFM_SVC_PSA_CALL TFM_SVC_NUM_PSA_API_THREAD(2)
```

Definition at line 86 of file svc_num.h.

#### 7.396.1.27 TFM_SVC_PSA_CLEAR

```
#define TFM_SVC_PSA_CLEAR TFM_SVC_NUM_PSA_API_THREAD(13)
```

Definition at line 98 of file svc_num.h.

#### 7.396.1.28 TFM_SVC_PSA_CLOSE

```
#define TFM_SVC_PSA_CLOSE TFM_SVC_NUM_PSA_API_THREAD(4)
```

Definition at line 88 of file svc_num.h.

#### 7.396.1.29 TFM_SVC_PSA_CONNECT

```
#define TFM_SVC_PSA_CONNECT TFM_SVC_NUM_PSA_API_THREAD(3)
```

Definition at line 87 of file svc_num.h.

#### 7.396.1.30 TFM_SVC_PSA_EOI

```
#define TFM_SVC_PSA_EOI TFM_SVC_NUM_PSA_API_THREAD(14)
```

Definition at line 99 of file svc_num.h.

#### 7.396.1.31 TFM_SVC_PSA_FRAMEWORK_VERSION

```
#define TFM_SVC_PSA_FRAMEWORK_VERSION TFM_SVC_NUM_PSA_API_THREAD(0)
```

Definition at line 84 of file svc_num.h.

#### 7.396.1.32 TFM_SVC_PSA_GET

```
#define TFM_SVC_PSA_GET TFM_SVC_NUM_PSA_API_THREAD(6)
```

Definition at line 91 of file svc_num.h.

#### 7.396.1.33 TFM_SVC_PSA_IRQ_DISABLE

```
#define TFM_SVC_PSA_IRQ_DISABLE TFM_SVC_NUM_PSA_API_THREAD(18)
```

Definition at line 103 of file svc_num.h.

#### 7.396.1.34 TFM_SVC_PSA_IRQ_ENABLE

```
#define TFM_SVC_PSA_IRQ_ENABLE TFM_SVC_NUM_PSA_API_THREAD(17)
```

Definition at line 102 of file svc_num.h.

#### 7.396.1.35 TFM_SVC_PSA_LIFECYCLE

```
#define TFM_SVC_PSA_LIFECYCLE TFM_SVC_NUM_PSA_API_THREAD(16)
```

Definition at line 101 of file svc_num.h.

#### 7.396.1.36 TFM_SVC_PSA_NOTIFY

```
#define TFM_SVC_PSA_NOTIFY TFM_SVC_NUM_PSA_API_THREAD(12)
```

Definition at line 97 of file svc_num.h.

#### 7.396.1.37 TFM_SVC_PSA_PANIC

```
#define TFM_SVC_PSA_PANIC TFM_SVC_NUM_PSA_API_THREAD(15)
```

Definition at line 100 of file svc_num.h.

#### 7.396.1.38 TFM_SVC_PSA_READ

```
#define TFM_SVC_PSA_READ TFM_SVC_NUM_PSA_API_THREAD(8)
```

Definition at line 93 of file svc_num.h.

#### 7.396.1.39 TFM_SVC_PSA_REPLY

```
#define TFM_SVC_PSA_REPLY TFM_SVC_NUM_PSA_API_THREAD(11)
```

Definition at line 96 of file svc_num.h.

#### 7.396.1.40 TFM_SVC_PSA_RESET_SIGNAL

```
#define TFM_SVC_PSA_RESET_SIGNAL TFM_SVC_NUM_PSA_API_THREAD(19)
```

Definition at line 104 of file svc_num.h.

#### 7.396.1.41 TFM_SVC_PSA_SET_RHANDLE

```
#define TFM_SVC_PSA_SET_RHANDLE TFM_SVC_NUM_PSA_API_THREAD(7)
```

Definition at line 92 of file svc_num.h.

#### 7.396.1.42 TFM_SVC_PSA_SKIP

```
#define TFM_SVC_PSA_SKIP TFM_SVC_NUM_PSA_API_THREAD(9)
```

Definition at line 94 of file svc_num.h.

#### 7.396.1.43 TFM_SVC_PSA_VERSION

```
#define TFM_SVC_PSA_VERSION TFM_SVC_NUM_PSA_API_THREAD(1)
```

Definition at line 85 of file svc_num.h.

#### 7.396.1.44 TFM_SVC_PSA_WAIT

```
#define TFM_SVC_PSA_WAIT TFM_SVC_NUM_PSA_API_THREAD(5)
```

Definition at line 90 of file svc_num.h.

#### 7.396.1.45 TFM_SVC_PSA_WRITE

```
#define TFM_SVC_PSA_WRITE TFM_SVC_NUM_PSA_API_THREAD(10)
```

Definition at line 95 of file svc_num.h.

#### 7.396.1.46 TFM_SVC_SPM_INIT

```
#define TFM_SVC_SPM_INIT TFM_SVC_NUM_SPM_THREAD(0)
```

Definition at line 73 of file svc_num.h.

#### 7.396.1.47 TFM_SVC_THREAD_MODE_SPM_RETURN

```
#define TFM_SVC_THREAD_MODE_SPM_RETURN TFM_SVC_NUM_SPM_THREAD(4)
```

Definition at line 77 of file svc_num.h.

## 7.397 secure_fw/spm/include/lists.h File Reference

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct bi_list_node_t

## Macros

- #define BI_LIST_INIT_NODE(node)
- #define BI_LIST_INSERT_AFTER(curr, node)
- #define BI_LIST_INSERT_BEFORE(curr, node)
- #define BI_LIST_REMOVE_NODE(node)
- #define BI_LIST_IS_EMPTY(head) ((head)->bnext == (head))
- #define BI_LIST_NEXT_NODE(node) ((node)->bnext)
- #define BI_LIST_FOR_EACH(node, head) for (node = (head)->bnext; node != head; node = (node)->bnext)
- #define UNI_LISI_INIT_NODE(head, link)
- #define UNI_LIST_INSERT_AFTER(posi, node, link)
- #define UNI_LIST_IS_EMPTY(node, link) ((node == NULL) || ((node)->link == NULL))
- #define UNI_LIST_NEXT_NODE(node, link) ((node)->link)
- #define UNI_LIST_REMOVE_NODE(prev, node, link)
- #define UNI_LIST_REMOVE_NODE_BY_PNODE(pnode, link) *(pnode) = (*(pnode))->link
- #define UNI_LIST_MOVE_AFTER(posi, prev, node, link)
- #define UNI_LIST_FOREACH(node, head, link) for (node = (head)->link; node != NULL; node = (node)->link)
- #define UNI_LIST_FOREACH_NODE_PREV(prev, node, head, link)
- #define UNI_LIST_FOREACH_NODE_PNODE(pnode, node, head, link)

### 7.397.1 Macro Definition Documentation

#### 7.397.1.1 BI_LIST_FOR_EACH

```
#define BI_LIST_FOR_EACH(
            node,
            head ) for (node = (head)->bnext; node != head; node = (node)->bnext)
```

Definition at line 52 of file lists.h.

#### 7.397.1.2 BI_LIST_INIT_NODE

```
#define BI_LIST_INIT_NODE(
            node )
```

**Value:**

```
    do {            \
    (node)->bnext = node;                   \
    (node)->bprev = node;                   \
} while (0)
```

Definition at line 18 of file lists.h.

#### 7.397.1.3 BI_LIST_INSERT_AFTER

```
#define BI_LIST_INSERT_AFTER(
            curr,
            node )
```

**Value:**

```
    do {    \
    (node)->bnext = (curr)->bnext;          \
    (node)->bprev = curr;                   \
    (curr)->bnext->bprev = node;            \
    (curr)->bnext = node;                   \
} while (0)
```

Definition at line 24 of file lists.h.

#### 7.397.1.4 BI_LIST_INSERT_BEFORE

```
#define BI_LIST_INSERT_BEFORE(
                curr,
                node )
```

**Value:**

```
    do {    \
    (curr)->bprev->bnext = node;                \
    (node)->bprev = (curr)->bprev;              \
    (curr)->bprev = node;                       \
    (node)->bnext = curr;                       \
} while (0)
```

Definition at line 32 of file lists.h.

#### 7.397.1.5 BI_LIST_IS_EMPTY

```
#define BI_LIST_IS_EMPTY(
                head ) ((head)->bnext == (head))
```

Definition at line 46 of file lists.h.

#### 7.397.1.6 BI_LIST_NEXT_NODE

```
#define BI_LIST_NEXT_NODE(
                node ) ((node)->bnext)
```

Definition at line 49 of file lists.h.

#### 7.397.1.7 BI_LIST_REMOVE_NODE

```
#define BI_LIST_REMOVE_NODE(
                node )
```

**Value:**

```
    do {            \
    (node)->bprev->bnext = (node)->bnext;       \
    (node)->bnext->bprev = (node)->bprev;       \
} while (0)
```

Definition at line 40 of file lists.h.

#### 7.397.1.8 UNI_LISI_INIT_NODE

```
#define UNI_LISI_INIT_NODE(
                head,
                link )
```

**Value:**

```
    do {            \
    if ((head) != NULL) {                       \
        (head)->link = NULL;                    \
    }                                           \
} while (0)
```

Definition at line 57 of file lists.h.

#### 7.397.1.9 UNI_LIST_FOREACH

```
#define UNI_LIST_FOREACH(
                node,
                head,
                link ) for (node = (head)->link; node != NULL; node = (node)->link)
```

Definition at line 96 of file lists.h.

#### 7.397.1.10 UNI_LIST_FOREACH_NODE_PNODE

```
#define UNI_LIST_FOREACH_NODE_PNODE(
                pnode,
                node,
                head,
                link )
```

**Value:**

```
    for (pnode = &(head)->link, node = (head)->link;          \
         node != NULL;                                        \
         pnode = &(node)->link, node = (node)->link)
```

Definition at line 105 of file lists.h.

#### 7.397.1.11 UNI_LIST_FOREACH_NODE_PREV

```
#define UNI_LIST_FOREACH_NODE_PREV(
                prev,
                node,
                head,
                link )
```

**Value:**

```
    for (prev = NULL, node = (head)->link;                    \
                 node != NULL; prev = node, node = (prev)->link)
```

Definition at line 100 of file lists.h.

#### 7.397.1.12 UNI_LIST_INSERT_AFTER

```
#define UNI_LIST_INSERT_AFTER(
                posi,
                node,
                link )
```

**Value:**

```
    do {    \
    (node)->link = (posi)->link;                          \
    (posi)->link = node;                                  \
} while (0)
```

Definition at line 64 of file lists.h.

#### 7.397.1.13 UNI_LIST_IS_EMPTY

```
#define UNI_LIST_IS_EMPTY(
                node,
                link ) ((node == NULL) || ((node)->link == NULL))
```

Definition at line 70 of file lists.h.

#### 7.397.1.14 UNI_LIST_MOVE_AFTER

```
#define UNI_LIST_MOVE_AFTER(
                posi,
                prev,
                node,
                link )
```

**Value:**

```
    do {\
    if (prev != NULL) {                                   \
        (prev)->link = (node)->link;                      \
        (node)->link = (posi)->link;                      \
        (posi)->link = node;                              \
    }                                                     \
} while (0)
```

Definition at line 87 of file lists.h.

#### 7.397.1.15 UNI_LIST_NEXT_NODE

```
#define UNI_LIST_NEXT_NODE(
            node,
            link ) ((node)->link)
```

Definition at line 74 of file lists.h.

#### 7.397.1.16 UNI_LIST_REMOVE_NODE

```
#define UNI_LIST_REMOVE_NODE(
            prev,
            node,
            link )
```

**Value:**

```
    do {      \
    (prev)->link = (node)->link;                    \
    (node)->link = NULL;                            \
} while (0)
```

Definition at line 77 of file lists.h.

#### 7.397.1.17 UNI_LIST_REMOVE_NODE_BY_PNODE

```
#define UNI_LIST_REMOVE_NODE_BY_PNODE(
            pnode,
            link ) *(pnode) = (*(pnode))->link
```

Definition at line 83 of file lists.h.

## 7.398 secure_fw/spm/include/load/asset_defs.h File Reference

```
#include <stdint.h>
```

Include dependency graph for asset_defs.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct asset_desc_t

## Macros

- #define ASSET_ATTR_READ_ONLY (1U << 0) /* 1: Read-only MMIO */
- #define ASSET_ATTR_READ_WRITE (1U << 1) /* 1: Read-write MMIO */
- #define ASSET_ATTR_PPB (1U << 2) /* 1: PPB indicator */
- #define ASSET_ATTR_NAMED_MMIO (1U << 3) /* 1: Named mmio object */
- #define ASSET_ATTR_NUMBERED_MMIO (1U << 4) /* 1: Numbered mmio object */
- #define ASSET_ATTR_MMIO

## 7.398.1 Macro Definition Documentation

### 7.398.1.1 ASSET_ATTR_MMIO

```
#define ASSET_ATTR_MMIO
```

**Value:**

```
(ASSET_ATTR_NAMED_MMIO | \
ASSET_ATTR_NUMBERED_MMIO)
```

Definition at line 19 of file asset_defs.h.

### 7.398.1.2 ASSET_ATTR_NAMED_MMIO

```
#define ASSET_ATTR_NAMED_MMIO (1U << 3) /* 1:  Named mmio object */
```

Definition at line 17 of file asset_defs.h.

### 7.398.1.3 ASSET_ATTR_NUMBERED_MMIO

```
#define ASSET_ATTR_NUMBERED_MMIO (1U << 4) /* 1:  Numbered mmio object */
```

Definition at line 18 of file asset_defs.h.

### 7.398.1.4 ASSET_ATTR_PPB

```
#define ASSET_ATTR_PPB (1U << 2) /* 1:  PPB indicator */
```

Definition at line 15 of file asset_defs.h.

### 7.398.1.5 ASSET_ATTR_READ_ONLY

```
#define ASSET_ATTR_READ_ONLY (1U << 0) /* 1:  Read-only MMIO */
```

Definition at line 13 of file asset_defs.h.

### 7.398.1.6 ASSET_ATTR_READ_WRITE

```
#define ASSET_ATTR_READ_WRITE (1U << 1) /* 1:  Read-write MMIO */
```

Definition at line 14 of file asset_defs.h.

## 7.399 secure_fw/spm/include/load/interrupt_defs.h File Reference

```
#include <stdint.h>
#include "tfm_hal_defs.h"
#include "psa/service.h"
```

Include dependency graph for interrupt_defs.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct irq_load_info_t
- struct irq_t

## 7.400 secure_fw/spm/include/load/partition_defs.h File Reference

```
#include <stddef.h>
#include <stdint.h>
```

Include dependency graph for partition_defs.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct partition_load_info_t

## Macros

- #define TFM_SP_IDLE (0x5555003f)
- #define TFM_SP_TZ_AGENT (0x555501c7)
- #define INVALID_PARTITION_ID (0U)
- #define PARTITION_INFO_VERSION_MASK (0x0000FFFF)
- #define PARTITION_INFO_MAGIC_MASK (0xFFFF0000)
- #define PARTITION_INFO_MAGIC (0x5F5F0000)
- #define PARTITION_PRI_HIGHEST (0x0)
- #define PARTITION_PRI_HIGH (0xF)
- #define PARTITION_PRI_NORMAL (0x1F)
- #define PARTITION_PRI_LOW (0x7F)
- #define PARTITION_PRI_LOWEST (0xFF)
- #define PARTITION_PRI_MASK (0xFF)
- #define PARTITION_MODEL_PSA_ROT (1UL << 8)
- #define PARTITION_MODEL_IPC (1UL << 9)
- #define PARTITION_NS_AGENT_MB (1UL << 10)
- #define PARTITION_NS_AGENT_TZ (1UL << 11)
- #define PARTITION_PRIORITY(flag) ((flag) & PARTITION_PRI_MASK)
- #define TO_THREAD_PRIORITY(x) (x)
- #define ENTRY_TO_POSITION(x) (uintptr_t)(x)
- #define POSITION_TO_ENTRY(x, t) (t)(x)
- #define PTR_TO_REFERENCE(x) (uintptr_t)(x)
- #define REFERENCE_TO_PTR(x, t) (t)(x)
- #define IS_PSA_ROT(pldi)
- #define IS_IPC_MODEL(pldi)
- #define IS_NS_AGENT(pldi)

- #define IS_NS_AGENT_TZ(pldi) ((void)pldi, false)
- #define IS_NS_AGENT_MAILBOX(pldi) ((void)pldi, false)
- #define PARTITION_TYPE_TO_INDEX(type) (!!((type) & PARTITION_NS_AGENT_TZ))

## 7.400.1 Macro Definition Documentation

### 7.400.1.1 ENTRY_TO_POSITION

```
#define ENTRY_TO_POSITION(
            x ) (uintptr_t)(x)
```

Definition at line 59 of file partition_defs.h.

### 7.400.1.2 INVALID_PARTITION_ID

```
#define INVALID_PARTITION_ID (0U)
```

Definition at line 20 of file partition_defs.h.

### 7.400.1.3 IS_IPC_MODEL

```
#define IS_IPC_MODEL(
            pldi )
```

**Value:**

```
(!!((pldi)->flags \
& PARTITION_MODEL_IPC))
```

Definition at line 67 of file partition_defs.h.

### 7.400.1.4 IS_NS_AGENT

```
#define IS_NS_AGENT(
            pldi )
```

**Value:**

```
(!!((pldi)->flags \
& (PARTITION_NS_AGENT_MB | PARTITION_NS_AGENT_TZ)))
```

Definition at line 69 of file partition_defs.h.

### 7.400.1.5 IS_NS_AGENT_MAILBOX

```
#define IS_NS_AGENT_MAILBOX(
            pldi ) ((void)pldi, false)
```

Definition at line 79 of file partition_defs.h.

### 7.400.1.6 IS_NS_AGENT_TZ

```
#define IS_NS_AGENT_TZ(
            pldi ) ((void)pldi, false)
```

Definition at line 74 of file partition_defs.h.

### 7.400.1.7 IS_PSA_ROT

```
#define IS_PSA_ROT(
            pldi )
```

**Value:**

```
(!!((pldi)->flags \
```

```
& PARTITION_MODEL_PSA_ROT))
```

Definition at line 65 of file partition_defs.h.

#### 7.400.1.8 PARTITION_INFO_MAGIC

```
#define PARTITION_INFO_MAGIC (0x5F5F0000)
```

Definition at line 25 of file partition_defs.h.

#### 7.400.1.9 PARTITION_INFO_MAGIC_MASK

```
#define PARTITION_INFO_MAGIC_MASK (0xFFFF0000)
```

Definition at line 24 of file partition_defs.h.

#### 7.400.1.10 PARTITION_INFO_VERSION_MASK

```
#define PARTITION_INFO_VERSION_MASK (0x0000FFFF)
```

Definition at line 23 of file partition_defs.h.

#### 7.400.1.11 PARTITION_MODEL_IPC

```
#define PARTITION_MODEL_IPC (1UL << 9)
```

Definition at line 51 of file partition_defs.h.

#### 7.400.1.12 PARTITION_MODEL_PSA_ROT

```
#define PARTITION_MODEL_PSA_ROT (1UL << 8)
```

Definition at line 50 of file partition_defs.h.

#### 7.400.1.13 PARTITION_NS_AGENT_MB

```
#define PARTITION_NS_AGENT_MB (1UL << 10)
```

Definition at line 53 of file partition_defs.h.

#### 7.400.1.14 PARTITION_NS_AGENT_TZ

```
#define PARTITION_NS_AGENT_TZ (1UL << 11)
```

Definition at line 54 of file partition_defs.h.

#### 7.400.1.15 PARTITION_PRI_HIGH

```
#define PARTITION_PRI_HIGH (0xF)
```

Definition at line 44 of file partition_defs.h.

#### 7.400.1.16 PARTITION_PRI_HIGHEST

```
#define PARTITION_PRI_HIGHEST (0x0)
```

Definition at line 43 of file partition_defs.h.

#### 7.400.1.17 PARTITION_PRI_LOW

```
#define PARTITION_PRI_LOW (0x7F)
```

Definition at line 46 of file partition_defs.h.

#### 7.400.1.18 PARTITION_PRI_LOWEST

```
#define PARTITION_PRI_LOWEST (0xFF)
```

Definition at line 47 of file partition_defs.h.

#### 7.400.1.19 PARTITION_PRI_MASK

```
#define PARTITION_PRI_MASK (0xFF)
```

Definition at line 48 of file partition_defs.h.

#### 7.400.1.20 PARTITION_PRI_NORMAL

```
#define PARTITION_PRI_NORMAL (0x1F)
```

Definition at line 45 of file partition_defs.h.

#### 7.400.1.21 PARTITION_PRIORITY

```
#define PARTITION_PRIORITY(
            flag ) ((flag) & PARTITION_PRI_MASK)
```

Definition at line 56 of file partition_defs.h.

#### 7.400.1.22 PARTITION_TYPE_TO_INDEX

```
#define PARTITION_TYPE_TO_INDEX(
            type ) (!!((type) & PARTITION_NS_AGENT_TZ))
```

Definition at line 82 of file partition_defs.h.

#### 7.400.1.23 POSITION_TO_ENTRY

```
#define POSITION_TO_ENTRY(
            x,
            t ) (t)(x)
```

Definition at line 60 of file partition_defs.h.

#### 7.400.1.24 PTR_TO_REFERENCE

```
#define PTR_TO_REFERENCE(
            x ) (uintptr_t)(x)
```

Definition at line 62 of file partition_defs.h.

#### 7.400.1.25 REFERENCE_TO_PTR

```
#define REFERENCE_TO_PTR(
            x,
            t ) (t)(x)
```

Definition at line 63 of file partition_defs.h.

#### 7.400.1.26 TFM_SP_IDLE

```
#define TFM_SP_IDLE (0x5555003f)
```

Definition at line 18 of file partition_defs.h.

#### 7.400.1.27 TFM_SP_TZ_AGENT

```
#define TFM_SP_TZ_AGENT (0x555501c7)
```

Definition at line 19 of file partition_defs.h.

#### 7.400.1.28 TO_THREAD_PRIORITY

```
#define TO_THREAD_PRIORITY(
            x ) (x)
```

Definition at line 57 of file partition_defs.h.

## 7.401 secure_fw/spm/include/load/service_defs.h File Reference

```
#include <stdint.h>
#include "config_impl.h"
#include "psa/service.h"
```

Include dependency graph for service_defs.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct service_load_info_t

## Macros

- #define SERVICE_FLAG_STATELESS_HINDEX_MASK (0xFF)
- #define SERVICE_FLAG_NS_ACCESSIBLE (1UL << 8)
- #define SERVICE_FLAG_STATELESS (1UL << 9)
- #define SERVICE_FLAG_VERSION_POLICY_BIT (1UL << 10)
- #define SERVICE_VERSION_POLICY_RELAXED (0UL << 10)
- #define SERVICE_VERSION_POLICY_STRICT (1UL << 10)
- #define SERVICE_FLAG_MM_IOVEC (1UL << 11)
- #define SERVICE_GET_STATELESS_HINDEX(flag) ((flag) & SERVICE_FLAG_STATELESS_HINDEX_MASK)
- #define SERVICE_IS_NS_ACCESSIBLE(flag) ((flag) & SERVICE_FLAG_NS_ACCESSIBLE)
- #define SERVICE_IS_STATELESS(flag) ((flag) & SERVICE_FLAG_STATELESS)
- #define SERVICE_GET_VERSION_POLICY(flag) ((flag) & SERVICE_FLAG_VERSION_POLICY_BIT)
- #define SERVICE_ENABLED_MM_IOVEC(flag) ((flag) & SERVICE_FLAG_MM_IOVEC)
- #define STRID_TO_STRING_PTR(strid) (const char *)(strid)
- #define STRING_PTR_TO_STRID(str) (uintptr_t)(str)

### 7.401.1 Macro Definition Documentation

#### 7.401.1.1 SERVICE_ENABLED_MM_IOVEC

```
#define SERVICE_ENABLED_MM_IOVEC(
            flag ) ((flag) & SERVICE_FLAG_MM_IOVEC)
```

Definition at line 40 of file service_defs.h.

#### 7.401.1.2 SERVICE_FLAG_MM_IOVEC

```
#define SERVICE_FLAG_MM_IOVEC (1UL << 11)
```

Definition at line 30 of file service_defs.h.

#### 7.401.1.3 SERVICE_FLAG_NS_ACCESSIBLE

```
#define SERVICE_FLAG_NS_ACCESSIBLE (1UL << 8)
```

Definition at line 24 of file service_defs.h.

#### 7.401.1.4 SERVICE_FLAG_STATELESS

```
#define SERVICE_FLAG_STATELESS (1UL << 9)
```

Definition at line 25 of file service_defs.h.

#### 7.401.1.5 SERVICE_FLAG_STATELESS_HINDEX_MASK

```
#define SERVICE_FLAG_STATELESS_HINDEX_MASK (0xFF)
```

Definition at line 23 of file service_defs.h.

#### 7.401.1.6 SERVICE_FLAG_VERSION_POLICY_BIT

```
#define SERVICE_FLAG_VERSION_POLICY_BIT (1UL << 10)
```

Definition at line 27 of file service_defs.h.

#### 7.401.1.7 SERVICE_GET_STATELESS_HINDEX

```
#define SERVICE_GET_STATELESS_HINDEX(
            flag ) ((flag) & SERVICE_FLAG_STATELESS_HINDEX_MASK)
```

Definition at line 32 of file service_defs.h.

#### 7.401.1.8 SERVICE_GET_VERSION_POLICY

```
#define SERVICE_GET_VERSION_POLICY(
            flag ) ((flag) & SERVICE_FLAG_VERSION_POLICY_BIT)
```

Definition at line 38 of file service_defs.h.

#### 7.401.1.9 SERVICE_IS_NS_ACCESSIBLE

```
#define SERVICE_IS_NS_ACCESSIBLE(
            flag ) ((flag) & SERVICE_FLAG_NS_ACCESSIBLE)
```

Definition at line 34 of file service_defs.h.

#### 7.401.1.10 SERVICE_IS_STATELESS

```
#define SERVICE_IS_STATELESS(
            flag ) ((flag) & SERVICE_FLAG_STATELESS)
```

Definition at line 36 of file service_defs.h.

#### 7.401.1.11 SERVICE_VERSION_POLICY_RELAXED

```
#define SERVICE_VERSION_POLICY_RELAXED (0UL << 10)
```

Definition at line 28 of file service_defs.h.

#### 7.401.1.12 SERVICE_VERSION_POLICY_STRICT

```
#define SERVICE_VERSION_POLICY_STRICT (1UL << 10)
```

Definition at line 29 of file service_defs.h.

#### 7.401.1.13 STRID_TO_STRING_PTR

```
#define STRID_TO_STRING_PTR(
            strid ) (const char *)(strid)
```

Definition at line 43 of file service_defs.h.

#### 7.401.1.14 STRING_PTR_TO_STRID

```
#define STRING_PTR_TO_STRID(
            str ) (uintptr_t)(str)
```

Definition at line 44 of file service_defs.h.

## 7.402 secure_fw/spm/include/load/spm_load_api.h File Reference

```
#include "asset_defs.h"
#include "interrupt_defs.h"
#include "partition_defs.h"
#include "service_defs.h"
```

```
#include "spm.h"
```

Include dependency graph for spm_load_api.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct partition_head_t
- struct service_head_t

## Macros

- #define NO_MORE_PARTITION NULL
- #define LOAD_INFO_EXT_LENGTH (2U)
- #define LOAD_INFSZ_BYTES(pldinf)
- #define LOAD_ALLOCED_STACK_ADDR(pldinf) (∗((uintptr_t ∗)((pldinf) + 1)))
- #define LOAD_INFO_DEPS(pldinf) ((const uint32_t ∗)((uintptr_t)((pldinf) + 1) + ((LOAD_INFO_EXT_LENGTH) ∗ sizeof(uintptr_t))))
- #define LOAD_INFO_SERVICE(pldinf)
- #define LOAD_INFO_ASSET(pldinf)
- #define LOAD_INFO_IRQ(pldinf)

## Functions

- struct partition_t ∗ load_a_partition_assuredly (struct partition_head_t ∗head)
- uint32_t load_services_assuredly (struct partition_t ∗p_partition, struct service_head_t ∗services_listhead, struct service_t ∗∗stateless_services_ref_tbl, size_t ref_tbl_size)
- void load_irqs_assuredly (struct partition_t ∗p_partition)

### 7.402.1 Macro Definition Documentation

#### 7.402.1.1 LOAD_ALLOCED_STACK_ADDR

```
#define LOAD_ALLOCED_STACK_ADDR(
            pldinf ) (*((uintptr_t *)((pldinf) + 1)))
```

Definition at line 35 of file spm_load_api.h.

#### 7.402.1.2 LOAD_INFO_ASSET

```
#define LOAD_INFO_ASSET(
            pldinf )
```

**Value:**

```
    ((const struct asset_desc_t *)((uintptr_t)LOAD_INFO_SERVICE(pldinf) + \
     ((pldinf)->nservices * sizeof(struct service_load_info_t))))
```

Definition at line 42 of file spm_load_api.h.

#### 7.402.1.3 LOAD_INFO_DEPS

```
#define LOAD_INFO_DEPS(
            pldinf ) ((const uint32_t *)((uintptr_t)((pldinf) + 1) + ((LOAD_INFO_EXT_LENGTH)
* sizeof(uintptr_t))))
```

Definition at line 37 of file spm_load_api.h.

#### 7.402.1.4 LOAD_INFO_EXT_LENGTH

```
#define LOAD_INFO_EXT_LENGTH (2U)
```

Definition at line 25 of file spm_load_api.h.

#### 7.402.1.5 LOAD_INFO_IRQ

```
#define LOAD_INFO_IRQ(
            pldinf )
```

**Value:**

```
    ((const struct irq_load_info_t *)((uintptr_t)LOAD_INFO_ASSET(pldinf) + \
     (pldinf)->nassets * sizeof(struct asset_desc_t)))
```

Definition at line 45 of file spm_load_api.h.

#### 7.402.1.6 LOAD_INFO_SERVICE

```
#define LOAD_INFO_SERVICE(
            pldinf )
```

**Value:**

```
    ((const struct service_load_info_t *)((uintptr_t)(LOAD_INFO_DEPS(pldinf)) + \
     ((pldinf)->ndeps * sizeof(uint32_t))))
```

Definition at line 39 of file spm_load_api.h.

#### 7.402.1.7 LOAD_INFSZ_BYTES

```
#define LOAD_INFSZ_BYTES(
            pldinf )
```

**Value:**

```
    (sizeof(*(pldinf)) + LOAD_INFO_EXT_LENGTH * sizeof(uintptr_t) +    \
     (pldinf)->ndeps * sizeof(uint32_t) +                              \
     (pldinf)->nservices * sizeof(struct service_load_info_t) +        \
     (pldinf)->nassets * sizeof(struct asset_desc_t) +                 \
     (pldinf)->nirqs * sizeof(struct irq_load_info_t))
```

Definition at line 27 of file spm_load_api.h.

#### 7.402.1.8 NO_MORE_PARTITION

```
#define NO_MORE_PARTITION NULL
```

Definition at line 22 of file spm_load_api.h.

## 7.402.2 Function Documentation

### 7.402.2.1 load_a_partition_assuredly()

```
struct partition_t* load_a_partition_assuredly (
            struct partition_head_t * head )
```

Definition at line 63 of file rom_loader.c.
Here is the call graph for this function:

### 7.402.2.2 load_irqs_assuredly()

```
void load_irqs_assuredly (
            struct partition_t * p_partition )
```

Definition at line 185 of file rom_loader.c.
Here is the call graph for this function:

### 7.402.2.3 load_services_assuredly()

```
uint32_t load_services_assuredly (
            struct partition_t * p_partition,
            struct service_head_t * services_listhead,
            struct service_t ** stateless_services_ref_tbl,
            size_t ref_tbl_size )
```

Definition at line 131 of file rom_loader.c.

## 7.403 secure_fw/spm/include/ns_agent_mailbox_defs.h File Reference

This graph shows which files directly or indirectly include this file:

### Macros

- #define BOOT_NS_CORE 1001
- #define DISABLE_MAILBOX 1002
- #define ENABLE_MAILBOX 1003
- #define NS_CORE_SHUTDOWN 1004

### 7.403.1 Macro Definition Documentation

#### 7.403.1.1 BOOT_NS_CORE

```
#define BOOT_NS_CORE 1001
```

Definition at line 13 of file ns_agent_mailbox_defs.h.

#### 7.403.1.2 DISABLE_MAILBOX

```
#define DISABLE_MAILBOX 1002
```

Definition at line 14 of file ns_agent_mailbox_defs.h.

#### 7.403.1.3 ENABLE_MAILBOX

```
#define ENABLE_MAILBOX 1003
```

Definition at line 15 of file ns_agent_mailbox_defs.h.

#### 7.403.1.4 NS_CORE_SHUTDOWN

```
#define NS_CORE_SHUTDOWN 1004
```

Definition at line 16 of file ns_agent_mailbox_defs.h.

## 7.404 secure_fw/spm/include/ns_agent_mailbox_interface.h File Reference

```
#include <stdint.h>
#include "psa/error.h"
```

Include dependency graph for ns_agent_mailbox_interface.h:

This graph shows which files directly or indirectly include this file:

### Functions

- psa_status_t ns_agent_boot_ns_core (void)

  *Boot the non-secure core.*
- psa_status_t ns_agent_mailbox_enable (void)

  *Enable the mailbox.*
- psa_status_t ns_agent_mailbox_disable (void)

  *Disable the mailbox.*
- psa_status_t ns_agent_ns_core_shutdown (void)

  *Inform ns_agent_mailbox that the NS core has shut down.*

## 7.404.1 Function Documentation

### 7.404.1.1 ns_agent_boot_ns_core()

```
psa_status_t ns_agent_boot_ns_core (
            void  )
```

Boot the non-secure core.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 16 of file ns_agent_mailbox_interface.c.
Here is the call graph for this function:

### 7.404.1.2 ns_agent_mailbox_disable()

```
psa_status_t ns_agent_mailbox_disable (
            void  )
```

Disable the mailbox.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 26 of file ns_agent_mailbox_interface.c.
Here is the call graph for this function:

### 7.404.1.3 ns_agent_mailbox_enable()

```
psa_status_t ns_agent_mailbox_enable (
            void  )
```

Enable the mailbox.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 21 of file ns_agent_mailbox_interface.c.
Here is the call graph for this function:

#### 7.404.1.4 ns_agent_ns_core_shutdown()

```
psa_status_t ns_agent_ns_core_shutdown (
            void  )
```

Inform ns_agent_mailbox that the NS core has shut down.
Before re-enabling the mailbox, ns_agent_boot_ns_core() must be called.

**Returns**

Returns values as specified by the psa_status_t

Definition at line 31 of file ns_agent_mailbox_interface.c.
Here is the call graph for this function:

## 7.405 secure_fw/spm/include/tfm_arch.h File Reference

```
#include <stdbool.h>
#include <stddef.h>
#include <inttypes.h>
#include "fih.h"
#include "tfm_hal_device_header.h"
#include "tfm_exc_num.h"
#include "cmsis_compiler.h"
```

Include dependency graph for tfm_arch.h:

This graph shows which files directly or indirectly include this file:

## Data Structures

- struct tfm_state_context_t
- struct tfm_additional_context_t
- struct full_context_t
- struct context_ctrl_t
- struct context_flih_ret_t

## Macros

- #define SCHEDULER_ATTEMPTED 2 /∗ Schedule attempt when scheduler is locked. ∗/
- #define SCHEDULER_LOCKED 1
- #define SCHEDULER_UNLOCKED 0
- #define XPSR_T32 0x01000000
- #define PENDSV_PRIO_FOR_SCHED ((1 << __NVIC_PRIO_BITS) - 1)
- #define TFM_FPU_CONTEXT_SIZE 0
- #define ARCH_CTXCTRL_INIT(x, buf, sz)
- #define ARCH_CTXCTRL_ALLOCATE_STACK(x, size) ((x)->sp -= ((size) + 7) & ∼0x7)
- #define ARCH_CTXCTRL_ALLOCATED_PTR(x) ((x)->sp)
- #define ARCH_CTXCTRL_EXCRET_PATTERN(x, param0, param1, param2, param3, pfn, pfnlr)
- #define ARCH_STATE_CTX_SET_R0(x, r0_val) ((x)->r0 = (uint32_t)(r0_val))
- #define ARCH_CLAIM_CTXCTRL_INSTANCE(name, stack_buf, stack_size)
- #define ARCH_FLUSH_FP_CONTEXT()

## Functions

- __STATIC_INLINE uint32_t __save_disable_irq (void)
- __STATIC_INLINE void __restore_irq (uint32_t status)
- __STATIC_INLINE void __set_CONTROL_nPRIV (uint32_t nPRIV)
- __STATIC_INLINE bool tfm_arch_is_priv (void)

  *Whether in privileged level.*
- void tfm_arch_set_secure_exception_priorities (void)
- void tfm_arch_config_extensions (void)
- void tfm_arch_free_msp_and_exc_ret (uint32_t msp_base, uint32_t exc_return)
- void tfm_arch_set_context_ret_code (const struct context_ctrl_t ∗p_ctx_ctrl, uint32_t ret_code)

- void tfm_arch_init_context (struct context_ctrl_t *p_ctx_ctrl, uintptr_t pfn, void *param, uintptr_t pfnlr)
- uint32_t tfm_arch_refresh_hardware_context (const struct context_ctrl_t *p_ctx_ctrl)
- void arch_acquire_sched_lock (void)
- uint32_t arch_release_sched_lock (void)
- uint32_t arch_attempt_schedule (void)
- void tfm_arch_thread_fn_call (uint32_t a0, uint32_t a1, uint32_t a2, uint32_t a3)
- void arch_clean_stack_and_launch (void *param, uintptr_t spm_init_func, uintptr_t ns_agent_entry, uint32_t msp_base)

## 7.405.1 Macro Definition Documentation

### 7.405.1.1 ARCH_CLAIM_CTXCTRL_INSTANCE

```
#define ARCH_CLAIM_CTXCTRL_INSTANCE(
            name,
            stack_buf,
            stack_size )
```

**Value:**

```
struct context_ctrl_t name = {                                  \
    .sp        = (uint32_t)&stack_buf[stack_size],              \
    .sp_base   = (uint32_t)&stack_buf[stack_size],              \
    .sp_limit  = (uint32_t)stack_buf,                           \
    .exc_ret   = 0,                                             \
}
```

Definition at line 203 of file tfm_arch.h.

### 7.405.1.2 ARCH_CTXCTRL_ALLOCATE_STACK

```
#define ARCH_CTXCTRL_ALLOCATE_STACK(
            x,
            size ) ((x)->sp -= ((size) + 7) & ~0x7)
```

Definition at line 177 of file tfm_arch.h.

### 7.405.1.3 ARCH_CTXCTRL_ALLOCATED_PTR

```
#define ARCH_CTXCTRL_ALLOCATED_PTR(
            x ) ((x)->sp)
```

Definition at line 181 of file tfm_arch.h.

### 7.405.1.4 ARCH_CTXCTRL_EXCRET_PATTERN

```
#define ARCH_CTXCTRL_EXCRET_PATTERN(
            x,
            param0,
            param1,
            param2,
            param3,
            pfn,
            pfnlr )
```

**Value:**

```
do { \
(x)->r0 = (uint32_t)(param0);                                   \
(x)->r1 = (uint32_t)(param1);                                   \
(x)->r2 = (uint32_t)(param2);                                   \
(x)->r3 = (uint32_t)(param3);                                   \
(x)->ra = (uint32_t)(pfn);                                      \
(x)->lr = (uint32_t)(pfnlr);                                    \
(x)->xpsr = XPSR_T32;                                           \
```

```
        } while (0)
```

Definition at line 184 of file tfm_arch.h.

#### 7.405.1.5 ARCH_CTXCTRL_INIT

```
#define ARCH_CTXCTRL_INIT(
                x,
                buf,
                sz )
```

**Value:**

```
            do {                                        \
            (x)->sp             = ((uint32_t)(buf) + (uint32_t)(sz)) & ~0x7; \
            (x)->sp_limit       = ((uint32_t)(buf) + 7) & ~0x7;              \
            (x)->sp_base        = (x)->sp;                                   \
            (x)->exc_ret        = 0;                                         \
        } while (0)
```

Definition at line 169 of file tfm_arch.h.

#### 7.405.1.6 ARCH_FLUSH_FP_CONTEXT

```
#define ARCH_FLUSH_FP_CONTEXT( )
```

Definition at line 262 of file tfm_arch.h.

#### 7.405.1.7 ARCH_STATE_CTX_SET_R0

```
#define ARCH_STATE_CTX_SET_R0(
                x,
                r0_val ) ((x)->r0 = (uint32_t)(r0_val))
```

Definition at line 195 of file tfm_arch.h.

#### 7.405.1.8 PENDSV_PRIO_FOR_SCHED

```
#define PENDSV_PRIO_FOR_SCHED ((1 << __NVIC_PRIO_BITS) - 1)
```

Definition at line 87 of file tfm_arch.h.

#### 7.405.1.9 SCHEDULER_ATTEMPTED

```
#define SCHEDULER_ATTEMPTED 2 /* Schedule attempt when scheduler is locked.  */
```

Definition at line 33 of file tfm_arch.h.

#### 7.405.1.10 SCHEDULER_LOCKED

```
#define SCHEDULER_LOCKED 1
```

Definition at line 34 of file tfm_arch.h.

#### 7.405.1.11 SCHEDULER_UNLOCKED

```
#define SCHEDULER_UNLOCKED 0
```

Definition at line 35 of file tfm_arch.h.

#### 7.405.1.12 TFM_FPU_CONTEXT_SIZE

```
#define TFM_FPU_CONTEXT_SIZE 0
```

Definition at line 121 of file tfm_arch.h.

#### 7.405.1.13 XPSR_T32

```
#define XPSR_T32 0x01000000
```

Definition at line 37 of file tfm_arch.h.

### 7.405.2 Function Documentation

#### 7.405.2.1 __restore_irq()

```
__STATIC_INLINE void __restore_irq (
            uint32_t status )
```

Definition at line 219 of file tfm_arch.h.

#### 7.405.2.2 __save_disable_irq()

```
__STATIC_INLINE uint32_t __save_disable_irq (
            void  )
```

Definition at line 211 of file tfm_arch.h.

#### 7.405.2.3 __set_CONTROL_nPRIV()

```
__STATIC_INLINE void __set_CONTROL_nPRIV (
            uint32_t nPRIV )
```

Definition at line 225 of file tfm_arch.h.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.405.2.4 arch_acquire_sched_lock()

```
void arch_acquire_sched_lock (
            void  )
```

Here is the caller graph for this function:

#### 7.405.2.5 arch_attempt_schedule()

```
uint32_t arch_attempt_schedule (
            void  )
```

Here is the caller graph for this function:

#### 7.405.2.6 arch_clean_stack_and_launch()

```
void arch_clean_stack_and_launch (
            void * param,
            uintptr_t spm_init_func,
            uintptr_t ns_agent_entry,
            uint32_t msp_base )
```

Here is the caller graph for this function:

#### 7.405.2.7 arch_release_sched_lock()

```
uint32_t arch_release_sched_lock (
            void  )
```

Here is the caller graph for this function:

#### 7.405.2.8 tfm_arch_config_extensions()

```
void tfm_arch_config_extensions (
            void  )
```

Definition at line 246 of file tfm_arch_v6m_v7m.c.
Here is the call graph for this function:

#### 7.405.2.9 tfm_arch_free_msp_and_exc_ret()

```
void tfm_arch_free_msp_and_exc_ret (
            uint32_t msp_base,
            uint32_t exc_return )
```

Definition at line 22 of file tfm_arch.c.

#### 7.405.2.10 tfm_arch_init_context()

```
void tfm_arch_init_context (
            struct context_ctrl_t * p_ctx_ctrl,
            uintptr_t pfn,
            void * param,
            uintptr_t pfnlr )
```

Definition at line 138 of file tfm_arch.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.405.2.11 tfm_arch_is_priv()

```
__STATIC_INLINE bool tfm_arch_is_priv (
            void  )
```

Whether in privileged level.

**Return values**

| | |
|---|---|
| *true* | If current execution runs in privileged level. |
| *false* | If current execution runs in unprivileged level. |

Definition at line 241 of file tfm_arch.h.

#### 7.405.2.12 tfm_arch_refresh_hardware_context()

```
uint32_t tfm_arch_refresh_hardware_context (
            const struct context_ctrl_t * p_ctx_ctrl )
```

Definition at line 173 of file tfm_arch.c.

Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.405.2.13 tfm_arch_set_context_ret_code()

```
void tfm_arch_set_context_ret_code (
            const struct context_ctrl_t * p_ctx_ctrl,
            uint32_t ret_code )
```

#### 7.405.2.14 tfm_arch_set_secure_exception_priorities()

```
void tfm_arch_set_secure_exception_priorities (
            void  )
```

Definition at line 207 of file tfm_arch_v6m_v7m.c.

#### 7.405.2.15 tfm_arch_thread_fn_call()

```
void tfm_arch_thread_fn_call (
            uint32_t a0,
            uint32_t a1,
            uint32_t a2,
            uint32_t a3 )
```

## 7.406 secure_fw/spm/include/tfm_arch_v8m.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "cmsis_compiler.h"
#include "fih.h"
#include "tfm_core_trustzone.h"
```

```
#include "utilities.h"
```

Include dependency graph for tfm_arch_v8m.h:

## Macros

- #define EXC_RETURN_RES1 (0x1FFFFUL << 7)
- #define EXC_RETURN_THREAD_PSP
- #define EXC_RETURN_THREAD_MSP
- #define EXC_RETURN_HANDLER
- #define SCB_ICSR_ADDR (0xE000ED04)
- #define SCB_ICSR_PENDSVSET_BIT (0x10000000)
- #define TFM_NS_EXC_DISABLE() __TZ_set_PRIMASK_NS(1)
- #define TFM_NS_EXC_ENABLE() __TZ_set_PRIMASK_NS(0)

## Functions

- __STATIC_INLINE bool is_return_secure_stack (uint32_t lr)

  *Check whether Secure or Non-secure stack is used to restore stack frame on exception return.*
- __STATIC_INLINE bool is_default_stacking_rules_apply (uint32_t lr)

  *Check whether the default stacking rules apply, or whether the Additional state context, also known as callee registers, are already on the stack. DCRS bit is only present from V8M and above. If DCRS is 1 then Stack contains: r0, r1, r2, r3, r12, r14 (lr), the return address and xPSR.*
- __STATIC_INLINE bool is_stack_alloc_fp_space (uint32_t lr)

  *Check whether the stack frame for this exception has space allocated for Floating Point(FP) state information.*
- __STATIC_INLINE uint32_t tfm_arch_get_psplim (void)

  *Get value of PSPLIM register.*
- __STATIC_INLINE void tfm_arch_set_psplim (uint32_t psplim)

  *Set PSPLIM register.*
- __STATIC_INLINE void tfm_arch_set_msplim (uint32_t msplim)

  *Set MSP limit value.*
- __STATIC_INLINE uintptr_t arch_seal_thread_stack (uintptr_t stk)

  *Seal the thread stack.*
- __STATIC_INLINE void tfm_arch_check_msp_sealing (void)

  *Check MSP sealing.*
- __STATIC_INLINE void arch_update_process_sp (uint32_t bottom, uint32_t toplimit)

## Variables

- uint64_t __STACK_SEAL

## 7.406.1 Macro Definition Documentation

### 7.406.1.1 EXC_RETURN_HANDLER

```
#define EXC_RETURN_HANDLER
```

**Value:**

```
        EXC_RETURN_PREFIX | EXC_RETURN_RES1 |                    \
        EXC_RETURN_S | EXC_RETURN_DCRS |                         \
        EXC_RETURN_FTYPE | EXC_RETURN_ES
```

Definition at line 35 of file tfm_arch_v8m.h.

### 7.406.1.2 EXC_RETURN_RES1

```
#define EXC_RETURN_RES1 (0x1FFFFUL << 7)
```

Definition at line 20 of file tfm_arch_v8m.h.

### 7.406.1.3 EXC_RETURN_THREAD_MSP

```
#define EXC_RETURN_THREAD_MSP
```

**Value:**

```
        EXC_RETURN_PREFIX | EXC_RETURN_RES1 |                    \
        EXC_RETURN_S | EXC_RETURN_DCRS |                         \
        EXC_RETURN_FTYPE | EXC_RETURN_MODE |                     \
        EXC_RETURN_ES
```

Definition at line 29 of file tfm_arch_v8m.h.

### 7.406.1.4 EXC_RETURN_THREAD_PSP

```
#define EXC_RETURN_THREAD_PSP
```

**Value:**

```
        EXC_RETURN_PREFIX | EXC_RETURN_RES1 |                    \
        EXC_RETURN_S | EXC_RETURN_DCRS |                         \
        EXC_RETURN_FTYPE | EXC_RETURN_MODE |                     \
        EXC_RETURN_SPSEL | EXC_RETURN_ES
```

Definition at line 23 of file tfm_arch_v8m.h.

### 7.406.1.5 SCB_ICSR_ADDR

```
#define SCB_ICSR_ADDR (0xE000ED04)
```

Definition at line 40 of file tfm_arch_v8m.h.

### 7.406.1.6 SCB_ICSR_PENDSVSET_BIT

```
#define SCB_ICSR_PENDSVSET_BIT (0x10000000)
```

Definition at line 41 of file tfm_arch_v8m.h.

### 7.406.1.7 TFM_NS_EXC_DISABLE

```
#define TFM_NS_EXC_DISABLE( ) __TZ_set_PRIMASK_NS(1)
```

Definition at line 44 of file tfm_arch_v8m.h.

### 7.406.1.8 TFM_NS_EXC_ENABLE

```
#define TFM_NS_EXC_ENABLE( ) __TZ_set_PRIMASK_NS(0)
```

Definition at line 46 of file tfm_arch_v8m.h.

## 7.406.2 Function Documentation

### 7.406.2.1 arch_seal_thread_stack()

```
__STATIC_INLINE uintptr_t arch_seal_thread_stack (
            uintptr_t stk )
```

Seal the thread stack.
This function must be called only when the caller is using MSP.

**Parameters**

| | | |
|---|---|---|
| in | *stk* | Thread stack address. |

**Return values**

| | |
|---|---|
| *stack* | Updated thread stack address. |

Definition at line 149 of file tfm_arch_v8m.h.

### 7.406.2.2 arch_update_process_sp()

```
__STATIC_INLINE void arch_update_process_sp (
            uint32_t bottom,
            uint32_t toplimit )
```

Definition at line 178 of file tfm_arch_v8m.h.
Here is the call graph for this function:

### 7.406.2.3 is_default_stacking_rules_apply()

```
__STATIC_INLINE bool is_default_stacking_rules_apply (
            uint32_t lr )
```

Check whether the default stacking rules apply, or whether the Additional state context, also known as callee registers, are already on the stack. DCRS bit is only present from V8M and above. If DCRS is 1 then Stack contains: r0, r1, r2, r3, r12, r14 (lr), the return address and xPSR.
If DCRS is 0 then the stack contains the following too before the caller-saved registers: Integrity signature, res, r4, r5, r6, r7, r8, r9, r10, r11

**Parameters**

| | | |
|---|---|---|
| in | *lr* | LR register containing the EXC_RETURN value. |

**Return values**

| | |
|---|---|
| *true* | Default rules for stacking the Additional state context registers followed. |
| *false* | Stacking of the Additional state context registers skipped. |

Definition at line 85 of file tfm_arch_v8m.h.

#### 7.406.2.4 is_return_secure_stack()

```
__STATIC_INLINE bool is_return_secure_stack (
            uint32_t lr )
```

Check whether Secure or Non-secure stack is used to restore stack frame on exception return.

**Parameters**

| | | |
|---|---|---|
| in | *lr* | LR register containing the EXC_RETURN value. |

**Return values**

| | |
|---|---|
| *true* | Secure stack is used to restore stack frame on exception return. |
| *false* | Non-secure stack is used to restore stack frame on exception return. |

Definition at line 61 of file tfm_arch_v8m.h.

#### 7.406.2.5 is_stack_alloc_fp_space()

```
__STATIC_INLINE bool is_stack_alloc_fp_space (
            uint32_t lr )
```

Check whether the stack frame for this exception has space allocated for Floating Point(FP) state information.

**Parameters**

| | | |
|---|---|---|
| in | *lr* | LR register containing the EXC_RETURN value. |

**Return values**

| | |
|---|---|
| *true* | The stack allocates space for FP information |
| *false* | The stack doesn't allocate space for FP information |

Definition at line 99 of file tfm_arch_v8m.h.

#### 7.406.2.6 tfm_arch_check_msp_sealing()

```
__STATIC_INLINE void tfm_arch_check_msp_sealing (
            void  )
```

Check MSP sealing.
Sealing must be done in the Reset_Handler() on a 8 byte region (__STACK_SEAL) defined in the linker scripts. (It is a CMSIS recommendation)
Definition at line 168 of file tfm_arch_v8m.h.

#### 7.406.2.7 tfm_arch_get_psplim()

```
__STATIC_INLINE uint32_t tfm_arch_get_psplim (
            void  )
```

Get value of PSPLIM register.

**Return values**

| | |
|---|---|
| *psplim* | Register value in PSPLIM register. |

Definition at line 109 of file tfm_arch_v8m.h.

#### 7.406.2.8 tfm_arch_set_msplim()

```
__STATIC_INLINE void tfm_arch_set_msplim (
            uint32_t msplim )
```

Set MSP limit value.

**Parameters**

| | | |
|---|---|---|
| in | *msplim* | MSP limit value to be written. |

Definition at line 129 of file tfm_arch_v8m.h.

#### 7.406.2.9 tfm_arch_set_psplim()

```
__STATIC_INLINE void tfm_arch_set_psplim (
            uint32_t psplim )
```

Set PSPLIM register.

**Parameters**

| | | |
|---|---|---|
| in | *psplim* | Register value to be written into PSPLIM. |

Definition at line 119 of file tfm_arch_v8m.h.

### 7.406.3 Variable Documentation

#### 7.406.3.1 __STACK_SEAL

```
uint64_t __STACK_SEAL
```

## 7.407 secure_fw/spm/include/tfm_core_trustzone.h File Reference

`#include "security_defs.h"`
Include dependency graph for tfm_core_trustzone.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define TFM_STACK_SEALED_SIZE 8
- #define TFM_STACK_SEAL_VALUE STACK_SEAL_PATTERN
- #define TFM_STACK_SEAL_VALUE_64 (uint64_t)0xFEF5EDA5FEF5EDA5
- #define TFM_BASIC_FP_CONTEXT_WORDS 18
- #define TFM_ADDTIONAL_FP_CONTEXT_WORDS 16
- #define TFM_VENEER_LR_BIT0_MASK 1

### 7.407.1 Macro Definition Documentation

#### 7.407.1.1 TFM_ADDTIONAL_FP_CONTEXT_WORDS

`#define TFM_ADDTIONAL_FP_CONTEXT_WORDS 16`
Definition at line 35 of file tfm_core_trustzone.h.

#### 7.407.1.2 TFM_BASIC_FP_CONTEXT_WORDS

`#define TFM_BASIC_FP_CONTEXT_WORDS 18`
Definition at line 29 of file tfm_core_trustzone.h.

#### 7.407.1.3 TFM_STACK_SEAL_VALUE

```
#define TFM_STACK_SEAL_VALUE STACK_SEAL_PATTERN
```

Definition at line 22 of file tfm_core_trustzone.h.

#### 7.407.1.4 TFM_STACK_SEAL_VALUE_64

```
#define TFM_STACK_SEAL_VALUE_64 (uint64_t)0xFEF5EDA5FEF5EDA5
```

Definition at line 23 of file tfm_core_trustzone.h.

#### 7.407.1.5 TFM_STACK_SEALED_SIZE

```
#define TFM_STACK_SEALED_SIZE 8
```

Definition at line 21 of file tfm_core_trustzone.h.

#### 7.407.1.6 TFM_VENEER_LR_BIT0_MASK

```
#define TFM_VENEER_LR_BIT0_MASK 1
```

Definition at line 41 of file tfm_core_trustzone.h.

## 7.408 secure_fw/spm/include/tfm_exc_num.h File Reference

```
#include <stdint.h>
#include "cmsis_compiler.h"
```

Include dependency graph for tfm_exc_num.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define EXC_NUM_THREAD_MODE (0)
- #define EXC_NUM_SVCALL (11)
- #define EXC_NUM_PENDSV (14)

## Functions

- __STATIC_INLINE uint32_t __get_active_exc_num (void)

## 7.408.1 Macro Definition Documentation

### 7.408.1.1 EXC_NUM_PENDSV

```
#define EXC_NUM_PENDSV (14)
```

Definition at line 23 of file tfm_exc_num.h.

### 7.408.1.2 EXC_NUM_SVCALL

```
#define EXC_NUM_SVCALL (11)
```

Definition at line 22 of file tfm_exc_num.h.

### 7.408.1.3 EXC_NUM_THREAD_MODE

```
#define EXC_NUM_THREAD_MODE (0)
```

Definition at line 21 of file tfm_exc_num.h.

## 7.408.2 Function Documentation

### 7.408.2.1 __get_active_exc_num()

```
__STATIC_INLINE uint32_t __get_active_exc_num (
            void  )
```

Definition at line 26 of file tfm_exc_num.h.

Here is the caller graph for this function:

## 7.409 secure_fw/spm/include/tfm_nspm.h File Reference

```
#include <stdint.h>
#include "security_defs.h"
```

Include dependency graph for tfm_nspm.h:

This graph shows which files directly or indirectly include this file:

## Macros

- #define TFM_NS_CLIENT_INVALID_ID ((int32_t)0)
- #define __tz_c_veneer

## Functions

- void tfm_nspm_ctx_init (void)

  *initialise the NS context database*

- int32_t tfm_nspm_get_current_client_id (void)

  *Get the client ID of the current NS client.*

- void tz_ns_agent_register_client_id_range (int32_t client_id_base, int32_t client_id_limit)

  *Register a non-secure client ID range.*

## 7.409.1 Macro Definition Documentation

### 7.409.1.1 __tz_c_veneer

```
#define __tz_c_veneer
```

Definition at line 31 of file tfm_nspm.h.

### 7.409.1.2 TFM_NS_CLIENT_INVALID_ID

```
#define TFM_NS_CLIENT_INVALID_ID ((int32_t)0)
```

Definition at line 21 of file tfm_nspm.h.

## 7.409.2 Function Documentation

### 7.409.2.1 tfm_nspm_ctx_init()

```
void tfm_nspm_ctx_init (
            void  )
```

initialise the NS context database

Definition at line 94 of file tfm_spm_ns_ctx.c.

Here is the call graph for this function:

#### 7.409.2.2 tfm_nspm_get_current_client_id()

```
int32_t tfm_nspm_get_current_client_id (
            void  )
```

Get the client ID of the current NS client.

**Returns**

The client id of the current NS client. 0 (invalid client id) is returned in case of error.

Definition at line 79 of file tfm_spm_ns_ctx.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.409.2.3 tz_ns_agent_register_client_id_range()

```
void tz_ns_agent_register_client_id_range (
            int32_t client_id_base,
            int32_t client_id_limit )
```

Register a non-secure client ID range.

**Parameters**

| | | |
|---|---|---|
| in | *client_id_base* | The minimum client ID for this client. |
| in | *client_id_limit* | The maximum client ID for this client. |

Definition at line 30 of file tfm_spm_ns_ctx.c.
Here is the caller graph for this function:

## 7.410 secure_fw/spm/include/tfm_spm_log.h File Reference

```
#include <stddef.h>
#include <stdint.h>
#include "tfm_hal_defs.h"
#include "tfm_hal_spm_logdev.h"
```

Include dependency graph for tfm_spm_log.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define TFM_SPM_LOG_LEVEL_DEBUG 3 /∗ All log APIs output ∗/
- #define TFM_SPM_LOG_LEVEL_INFO
- #define TFM_SPM_LOG_LEVEL_ERROR
- #define TFM_SPM_LOG_LEVEL_SILENCE 0 /∗ All log APIs are suppressed ∗/
- #define SPMLOG_DBGMSGVAL(msg, val)
- #define SPMLOG_DBGMSG(msg)
- #define SPMLOG_INFMSGVAL(msg, val)
- #define SPMLOG_INFMSG(msg)
- #define SPMLOG_ERRMSGVAL(msg, val)
- #define SPMLOG_ERRMSG(msg)

### Functions

- int32_t spm_log_msgval (const char ∗msg, size_t len, uint32_t value)

  *SPM output API to convert digit number into HEX string and call the HAL API tfm_hal_output_spm_log.*

### 7.410.1 Macro Definition Documentation

#### 7.410.1.1 SPMLOG_DBGMSG

```
#define SPMLOG_DBGMSG(
              msg )
```

Definition at line 38 of file tfm_spm_log.h.

#### 7.410.1.2 SPMLOG_DBGMSGVAL

```
#define SPMLOG_DBGMSGVAL(
              msg,
              val )
```

Definition at line 37 of file tfm_spm_log.h.

#### 7.410.1.3 SPMLOG_ERRMSG

```
#define SPMLOG_ERRMSG(
              msg )
```

Definition at line 54 of file tfm_spm_log.h.

#### 7.410.1.4 SPMLOG_ERRMSGVAL

```
#define SPMLOG_ERRMSGVAL(
              msg,
              val )
```

Definition at line 53 of file tfm_spm_log.h.

#### 7.410.1.5 SPMLOG_INFMSG

```
#define SPMLOG_INFMSG(
              msg )
```

Definition at line 46 of file tfm_spm_log.h.

#### 7.410.1.6 SPMLOG_INFMSGVAL

```
#define SPMLOG_INFMSGVAL(
              msg,
              val )
```

Definition at line 45 of file tfm_spm_log.h.

#### 7.410.1.7 TFM_SPM_LOG_LEVEL_DEBUG

```
#define TFM_SPM_LOG_LEVEL_DEBUG 3 /* All log APIs output */
```

Definition at line 19 of file tfm_spm_log.h.

#### 7.410.1.8 TFM_SPM_LOG_LEVEL_ERROR

```
#define TFM_SPM_LOG_LEVEL_ERROR
```

**Value:**

```
                                    1  /*
* Only SPMLOG_ERRMSG and SPMLOG_ERRMSGVAL
```

```
* APIs output.
*/
```

Definition at line 21 of file tfm_spm_log.h.

#### 7.410.1.9 TFM_SPM_LOG_LEVEL_INFO

```
#define TFM_SPM_LOG_LEVEL_INFO
```

**Value:**

```
                                        2  /*
* All log APIs output except SPMLOG_DBG
* and SPMLOG_DBGMSGVAL
*/
```

Definition at line 20 of file tfm_spm_log.h.

#### 7.410.1.10 TFM_SPM_LOG_LEVEL_SILENCE

```
#define TFM_SPM_LOG_LEVEL_SILENCE 0 /* All log APIs are suppressed */
```

Definition at line 22 of file tfm_spm_log.h.

### 7.410.2 Function Documentation

#### 7.410.2.1 spm_log_msgval()

```
int32_t spm_log_msgval (
            const char * msg,
            size_t len,
            uint32_t value )
```

SPM output API to convert digit number into HEX string and call the HAL API tfm_hal_output_spm_log.

**Parameters**

| | | |
|---|---|---|
| in | *msg* | A string message |
| in | *len* | The length of the message |
| in | *value* | A value need to be output |

**Return values**

| | |
|---|---|
| *>=0* | Number of chars output. |
| *<0* | TFM HAL error code. |

Definition at line 38 of file spm_log.c.
Here is the call graph for this function:

## 7.411 secure_fw/spm/include/utilities.h File Reference

```
#include <string.h>
#include "tfm_spm_log.h"
#include "cmsis_compiler.h"
```

Include dependency graph for utilities.h:

This graph shows which files directly or indirectly include this file:

### Macros

- #define SPM_ASSERT(cond)
- #define TO_CONTAINER(ptr, type, member) ((type ∗)((uint32_t)(ptr) - offsetof(type, member)))
- #define ERROR_MSG(msg) SPMLOG_ERRMSG(msg "\r\n")
- #define STRINGIFY_EXPAND(x) #x
- #define M2S(m) STRINGIFY_EXPAND(m)
- #define spm_memcpy memcpy
- #define spm_memset memset

### Functions

- __NO_RETURN void tfm_core_panic (void)

### 7.411.1 Macro Definition Documentation

#### 7.411.1.1 ERROR_MSG

```
#define ERROR_MSG(
            msg ) SPMLOG_ERRMSG(msg "\r\n")
```

Definition at line 43 of file utilities.h.

#### 7.411.1.2 M2S

```
#define M2S(
            m ) STRINGIFY_EXPAND(m)
```

Definition at line 47 of file utilities.h.

#### 7.411.1.3 SPM_ASSERT

```
#define SPM_ASSERT(
            cond )
```

**Value:**

```
        do {                                                    \
            if (!(cond)) {                                      \
                SPMLOG_INFMSG("Assert:");                       \
                SPMLOG_INFMSG(__func__);                        \
                SPMLOG_INFMSGVAL(",", __LINE__);                \
                while (1)                                       \
                    ;                                           \
            }                                                   \
        } while (0)
```

Definition at line 24 of file utilities.h.

#### 7.411.1.4 spm_memcpy

```
#define spm_memcpy memcpy
```

Definition at line 51 of file utilities.h.

#### 7.411.1.5 spm_memset

```
#define spm_memset memset
```

Definition at line 57 of file utilities.h.

#### 7.411.1.6 STRINGIFY_EXPAND

```
#define STRINGIFY_EXPAND(
            x ) #x
```

Definition at line 46 of file utilities.h.

#### 7.411.1.7 TO_CONTAINER

```
#define TO_CONTAINER(
            ptr,
            type,
            member ) ((type *)((uint32_t)(ptr) - offsetof(type, member)))
```

Definition at line 39 of file utilities.h.

### 7.411.2 Function Documentation

#### 7.411.2.1 tfm_core_panic()

```
__NO_RETURN void tfm_core_panic (
            void )
```

Definition at line 18 of file utilities.c.

## 7.412 secure_fw/spm/ns_client_ext/tfm_ns_client_ext.c File Reference

```
#include "cmsis.h"
#include "tfm_exc_num.h"
#include "tfm_nspm.h"
#include "tfm_ns_client_ext.h"
#include "tfm_ns_ctx.h"
```

```
#include "tfm_ns_ctx_shm.h"
```

Include dependency graph for tfm_ns_client_ext.c:

## Macros

- #define MAKE_NS_CLIENT_TOKEN(tid, gid, idx)
- #define IS_INVALID_TOKEN(token) ((token) & 0xff000000)
- #define NS_CLIENT_TOKEN_TO_CTX_IDX(token) (((token) >> 16) & 0xff)
- #define NS_CLIENT_TOKEN_TO_GID(token) (((token) >> 8) & 0xff)
- #define NS_CLIENT_TOKEN_TO_TID(token) ((token) & 0xff)

## Functions

- __tz_c_veneer uint32_t tfm_nsce_init (uint32_t ctx_requested)

  *Initialize the non-secure client extension.*
- __tz_c_veneer uint32_t tfm_nsce_acquire_ctx (uint8_t group_id, uint8_t thread_id)

  *Acquire the context for a non-secure client.*
- __tz_c_veneer uint32_t tfm_nsce_release_ctx (uint32_t token)

  *Release the context for the non-secure client.*
- __tz_c_veneer uint32_t tfm_nsce_load_ctx (uint32_t token, int32_t nsid)

  *Load the context for the non-secure client.*
- __tz_c_veneer uint32_t tfm_nsce_save_ctx (uint32_t token)

  *Save the context for the non-secure client.*

## Variables

- struct tfm_ns_ctx_mgr_t * ns_ctx_mgr

## 7.412.1 Macro Definition Documentation

### 7.412.1.1 IS_INVALID_TOKEN

```
#define IS_INVALID_TOKEN(
            token ) ((token) & 0xff000000)
```

Definition at line 28 of file tfm_ns_client_ext.c.

### 7.412.1.2 MAKE_NS_CLIENT_TOKEN

```
#define MAKE_NS_CLIENT_TOKEN(
            tid,
            gid,
            idx )
```

**Value:**

```
(uint32_t)((((uint32_t)tid & 0xff)      \
| (((uint32_t)gid & 0xff) « 8)         \
| (((uint32_t)idx & 0xff) « 16))       \
& 0x00ffffff)
```

Definition at line 23 of file tfm_ns_client_ext.c.

#### 7.412.1.3 NS_CLIENT_TOKEN_TO_CTX_IDX

```
#define NS_CLIENT_TOKEN_TO_CTX_IDX(
              token ) (((token) >> 16) & 0xff)
```

Definition at line 29 of file tfm_ns_client_ext.c.

#### 7.412.1.4 NS_CLIENT_TOKEN_TO_GID

```
#define NS_CLIENT_TOKEN_TO_GID(
              token ) (((token) >> 8) & 0xff)
```

Definition at line 30 of file tfm_ns_client_ext.c.

#### 7.412.1.5 NS_CLIENT_TOKEN_TO_TID

```
#define NS_CLIENT_TOKEN_TO_TID(
              token ) ((token) & 0xff)
```

Definition at line 31 of file tfm_ns_client_ext.c.

### 7.412.2 Function Documentation

#### 7.412.2.1 tfm_nsce_acquire_ctx()

```
__tz_c_veneer uint32_t tfm_nsce_acquire_ctx (
            uint8_t group_id,
            uint8_t thread_id )
```

Acquire the context for a non-secure client.

This function should be called before a non-secure client calling the PSA API into TF-M. It is to request the allocation of the context for the upcoming service call from that non-secure client. The non-secure clients in one group share the same context. The thread ID is used to identify the different non-secure clients.

**Parameters**

| | | |
|---|---|---|
| in | *group_id* | The group ID of the non-secure client |
| in | *thread↩ _id* | The thread ID of the non-secure client |

**Returns**

Returns the token of the allocated context. 0xFFFFFFFF means the allocation failed and the token is invalid.

Definition at line 54 of file tfm_ns_client_ext.c.
Here is the call graph for this function:

#### 7.412.2.2 tfm_nsce_init()

```
__tz_c_veneer uint32_t tfm_nsce_init (
            uint32_t ctx_requested )
```

Initialize the non-secure client extension.
This function should be called before any other non-secure client APIs. It gives NSPE the opportunity to initialize the non-secure client extension in TF-M. Also, NSPE can get the number of allocated non-secure client context slots in the return value. That is useful if NSPE wants to decide the group (context) assignment at runtime.

**Parameters**

| | | |
|---|---|---|
| in | *ctx_requested* | The number of non-secure context requested from the NS entity. If request maximum available context, then set it to 0. |

**Returns**

Returns the number of non-secure context allocated to the NS entity. The allocated context number <= maximum supported context number. If the initialization is failed, then 0 is returned.

Definition at line 36 of file tfm_ns_client_ext.c.
Here is the call graph for this function:

#### 7.412.2.3 tfm_nsce_load_ctx()

```
__tz_c_veneer uint32_t tfm_nsce_load_ctx (
```

```
uint32_t token,
int32_t nsid )
```

Load the context for the non-secure client.

This function should be called when a non-secure client is going to be scheduled in at the non-secure side. The caller is usually the scheduler of the RTOS. The non-secure client ID is managed by the non-secure world and passed to TF-M as the input parameter of TF-M.

**Parameters**

| | | |
|---|---|---|
| in | *token* | The token returned by tfm_nsce_acquire_ctx |
| in | *nsid* | The non-secure client ID for this client |

**Returns**

Returns the error code.

Definition at line 102 of file tfm_ns_client_ext.c.

Here is the call graph for this function:

#### 7.412.2.4 tfm_nsce_release_ctx()

```
__tz_c_veneer uint32_t tfm_nsce_release_ctx (
uint32_t token )
```

Release the context for the non-secure client.

This function should be called when a non-secure client is going to be terminated or will not call TF-M secure services in the future. It is to release the context allocated for the calling non-secure client. If the calling non-secure client is the only thread in the group, then the context will be deallocated. Otherwise, the context will still be taken for the other threads in the group.

**Parameters**

| | | |
|---|---|---|
| in | *token* | The token returned by tfm_nsce_acquire_ctx |

**Returns**

Returns the error code.

Definition at line 73 of file tfm_ns_client_ext.c.
Here is the call graph for this function:

#### 7.412.2.5 tfm_nsce_save_ctx()

```
__tz_c_veneer uint32_t tfm_nsce_save_ctx (
            uint32_t token )
```

Save the context for the non-secure client.
This function should be called when a non-secure client is going to be scheduled out at the non-secure side. The caller is usually the scheduler of the RTOS.

**Parameters**

| | | |
|---|---|---|
| in | *token* | The token returned by tfm_nsce_acquire_ctx |

**Returns**

Returns the error code.

Definition at line 135 of file tfm_ns_client_ext.c.
Here is the call graph for this function:

### 7.412.3 Variable Documentation

#### 7.412.3.1 ns_ctx_mgr

```
struct tfm_ns_ctx_mgr_t* ns_ctx_mgr
```

Definition at line 33 of file tfm_ns_ctx.c.

## 7.413 secure_fw/spm/ns_client_ext/tfm_ns_ctx.c File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "tfm_hal_device_header.h"
#include "tfm_ns_ctx.h"
#include "tfm_ns_ctx_shm.h"
```

Include dependency graph for tfm_ns_ctx.c:

### Macros

- #define TFM_NS_CONTEXT_MAX_TID 0xff
- #define TFM_NS_CONTEXT_MAX 1

### Functions

- bool acquire_ns_ctx (uint8_t gid, uint8_t *idx)
- bool release_ns_ctx (uint8_t gid, uint8_t tid, uint8_t idx)
- bool load_ns_ctx (uint8_t gid, uint8_t tid, int32_t nsid, uint8_t idx)
- bool save_ns_ctx (uint8_t gid, uint8_t tid, uint8_t idx)

### Variables

- struct tfm_ns_ctx_mgr_t * ns_ctx_mgr = &tfm_ns_ctx_mgr

### 7.413.1 Macro Definition Documentation

#### 7.413.1.1 TFM_NS_CONTEXT_MAX

```
#define TFM_NS_CONTEXT_MAX 1
```

Definition at line 19 of file tfm_ns_ctx.c.

#### 7.413.1.2 TFM_NS_CONTEXT_MAX_TID

```
#define TFM_NS_CONTEXT_MAX_TID 0xff
```

Definition at line 16 of file tfm_ns_ctx.c.

### 7.413.2 Function Documentation

#### 7.413.2.1 acquire_ns_ctx()

```
bool acquire_ns_ctx (
            uint8_t gid,
            uint8_t * idx )
```

Definition at line 40 of file tfm_ns_ctx.c.
Here is the caller graph for this function:

#### 7.413.2.2 load_ns_ctx()

```
bool load_ns_ctx (
            uint8_t gid,
            uint8_t tid,
            int32_t nsid,
            uint8_t idx )
```

Definition at line 143 of file tfm_ns_ctx.c.
Here is the caller graph for this function:

#### 7.413.2.3 release_ns_ctx()

```
bool release_ns_ctx (
            uint8_t gid,
            uint8_t tid,
            uint8_t idx )
```

Definition at line 95 of file tfm_ns_ctx.c.

Here is the caller graph for this function:

#### 7.413.2.4 save_ns_ctx()

```
bool save_ns_ctx (
            uint8_t gid,
            uint8_t tid,
            uint8_t idx )
```

Definition at line 165 of file tfm_ns_ctx.c.
Here is the caller graph for this function:

### 7.413.3 Variable Documentation

#### 7.413.3.1 ns_ctx_mgr

```
struct tfm_ns_ctx_mgr_t* ns_ctx_mgr = &tfm_ns_ctx_mgr
```

Definition at line 33 of file tfm_ns_ctx.c.

## 7.414 secure_fw/spm/ns_client_ext/tfm_ns_ctx.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
```

Include dependency graph for tfm_ns_ctx.h:

This graph shows which files directly or indirectly include this file:

## Functions

- bool init_ns_ctx (void)
- bool acquire_ns_ctx (uint8_t gid, uint8_t *idx)
- bool release_ns_ctx (uint8_t gid, uint8_t tid, uint8_t idx)
- bool load_ns_ctx (uint8_t gid, uint8_t tid, int32_t nsid, uint8_t idx)
- bool save_ns_ctx (uint8_t gid, uint8_t tid, uint8_t idx)
- int32_t get_nsid_from_active_ns_ctx (void)

## 7.414.1 Function Documentation

### 7.414.1.1 acquire_ns_ctx()

```
bool acquire_ns_ctx (
            uint8_t gid,
            uint8_t * idx )
```

Definition at line 40 of file tfm_ns_ctx.c.

Here is the caller graph for this function:

#### 7.414.1.2 get_nsid_from_active_ns_ctx()

```
int32_t get_nsid_from_active_ns_ctx (
            void  )
```

Definition at line 77 of file tfm_ns_ctx_s.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.414.1.3 init_ns_ctx()

```
bool init_ns_ctx (
            void  )
```

Definition at line 72 of file tfm_ns_ctx_s.c.

Here is the caller graph for this function:

#### 7.414.1.4 load_ns_ctx()

```
bool load_ns_ctx (
            uint8_t gid,
            uint8_t tid,
            int32_t nsid,
            uint8_t idx )
```

Definition at line 143 of file tfm_ns_ctx.c.
Here is the caller graph for this function:

#### 7.414.1.5 release_ns_ctx()

```
bool release_ns_ctx (
            uint8_t gid,
            uint8_t tid,
            uint8_t idx )
```

Definition at line 95 of file tfm_ns_ctx.c.
Here is the caller graph for this function:

#### 7.414.1.6 save_ns_ctx()

```
bool save_ns_ctx (
            uint8_t gid,
            uint8_t tid,
            uint8_t idx )
```

Definition at line 165 of file tfm_ns_ctx.c.
Here is the caller graph for this function:

## 7.415 secure_fw/spm/ns_client_ext/tfm_ns_ctx_s.c File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "cmsis.h"
#include "security_defs.h"
#include "tfm_ns_ctx.h"
#include "tfm_nspm.h"
#include "tfm_ns_ctx_shm.h"
```

Include dependency graph for tfm_ns_ctx_s.c:

### Functions

- bool init_ns_ctx (void)
- int32_t get_nsid_from_active_ns_ctx (void)

### Variables

- struct tfm_ns_ctx_mgr_t * ns_ctx_mgr

### 7.415.1 Function Documentation

#### 7.415.1.1 get_nsid_from_active_ns_ctx()

```
int32_t get_nsid_from_active_ns_ctx (
            void  )
```

Definition at line 77 of file tfm_ns_ctx_s.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.415.1.2 init_ns_ctx()

```
bool init_ns_ctx (
            void  )
```

Definition at line 72 of file tfm_ns_ctx_s.c.
Here is the caller graph for this function:

### 7.415.2 Variable Documentation

#### 7.415.2.1 ns_ctx_mgr

```
struct tfm_ns_ctx_mgr_t* ns_ctx_mgr
```

Definition at line 33 of file tfm_ns_ctx.c.

## 7.416 secure_fw/spm/ns_client_ext/tfm_ns_ctx_shm.h File Reference

```
#include <stdint.h>
```

Include dependency graph for tfm_ns_ctx_shm.h:

This graph shows which files directly or indirectly include this file:

### Data Structures

- struct tfm_ns_ctx_t
- struct tfm_ns_ctx_mgr_t

## 7.417 secure_fw/spm/ns_client_ext/tfm_spm_ns_ctx.c File Reference

```
#include "config_spm.h"
#include "current.h"
#include "load/partition_defs.h"
#include "tfm_nspm.h"
#include "tfm_ns_ctx.h"
#include "tfm_ns_client_ext.h"
#include "utilities.h"
#include "tfm_arch.h"
#include "tfm_hal_platform.h"
```

Include dependency graph for tfm_spm_ns_ctx.c:

## Data Structures

- struct client_id_region_t

## Macros

- #define DEFAULT_NS_CLIENT_ID ((int32_t)-1)

## Functions

- void tz_ns_agent_register_client_id_range (int32_t client_id_base, int32_t client_id_limit)

  *Register a non-secure client ID range.*

- int32_t tfm_nspm_get_current_client_id (void)

  *Get the client ID of the current NS client.*

- void tfm_nspm_ctx_init (void)

  *initialise the NS context database*

### 7.417.1 Macro Definition Documentation

#### 7.417.1.1 DEFAULT_NS_CLIENT_ID

```
#define DEFAULT_NS_CLIENT_ID ((int32_t)-1)
```

Definition at line 21 of file tfm_spm_ns_ctx.c.

### 7.417.2 Function Documentation

#### 7.417.2.1 tfm_nspm_ctx_init()

```
void tfm_nspm_ctx_init (
            void  )
```

initialise the NS context database

Definition at line 94 of file tfm_spm_ns_ctx.c.

Here is the call graph for this function:

#### 7.417.2.2 tfm_nspm_get_current_client_id()

```
int32_t tfm_nspm_get_current_client_id (
            void  )
```

Get the client ID of the current NS client.

**Returns**

The client id of the current NS client. 0 (invalid client id) is returned in case of error.

Definition at line 79 of file tfm_spm_ns_ctx.c.
Here is the call graph for this function:

Here is the caller graph for this function:

#### 7.417.2.3 tz_ns_agent_register_client_id_range()

```
void tz_ns_agent_register_client_id_range (
            int32_t client_id_base,
            int32_t client_id_limit )
```

Register a non-secure client ID range.

**Parameters**

| | | |
|---|---|---|
| in | *client_id_base* | The minimum client ID for this client. |
| in | *client_id_limit* | The maximum client ID for this client. |

Definition at line 30 of file tfm_spm_ns_ctx.c.
Here is the caller graph for this function: