WiSeConnect 3

- rsi_ble_req_long_att_value_s
  - dev_addr
  - handle
  - offset
- rsi_ble_set_att_val_s
  - dev_addr
  - handle
  - length
  - att_value
- rsi_ble_set_att_cmd_s
  - dev_addr
  - handle
  - length
  - att_value
- rsi_ble_set_long_att_val_s
  - dev_addr
  - handle
  - offset
  - length
  - att_value
- rsi_ble_req_prepare_write_s
  - dev_addr
  - handle
  - offset
  - length
  - att_value
- rsi_ble_req_execute_write_s
  - dev_addr
  - flag
- rsi_ble_cmd_conn_param_resp
  - dev_addr
  - status
- rsi_ble_req_add_serv_s
  - service_uuid
  - num_of_attributes
  - total_att_datasize
- rsi_ble_set_local_att_value_s
  - handle
  - data_len
  - data
- rsi_ble_notify_att_value_s
  - dev_addr
  - handle
  - data_len

- RSI_BLE_ATT_READ_BY_GROUP_TYPE_REQUEST
- RSI_BLE_ATT_WRITE_REQUEST
- RSI_BLE_ATT_PREPARE_WRITE_REQUEST
- RSI_BLE_ATT_EXECUTE_WRITE_REQUEST
- SUPPORTED_SCNNING_PHYS
- __attribute__

Event Types

- rsi_ble_event_adv_report_s
  - dev_addr_type
  - dev_addr
  - adv_data_len
  - adv_data
  - rssi
  - report_type
- rsi_ble_event_conn_status_s
  - dev_addr_type
  - dev_addr
  - status
- rsi_ble_event_enhnace_conn_status_s
  - dev_addr_type
  - dev_addr
  - local_resolvlable_addr
  - peer_resolvlable_addr
  - role
  - conn_interval
  - conn_latency
  - supervision_timeout
  - master_clock_accuracy
  - status
- rsi_ble_event_disconnect_s
  - dev_addr
  - dev_type
- rsi_ble_event_le_ping_time_expired_s
  - dev_addr
- rsi_bt_event_le_ltk_request_s
  - dev_addr
  - localediv
  - localrand
  - dev_addr_type
- rsi_bt_event_le_security_keys_s
  - dev_addr
  - local_irk
  - remote_irk
  - remote_ediv

- SL_SI91X_TCP_IP_FEAT_SNTP_CLIENT
- SL_SI91X_TCP_IP_FEAT_IPV6
- SL_SI91X_TCP_IP_FEAT_RAW_DATA
- SL_SI91X_TCP_IP_FEAT_MDNSD
- SL_SI91X_TCP_IP_FEAT_SMTP_CLIENT
- SL_SI91X_TCP_IP_TOTAL_SOCKETS
- SL_SI91X_TCP_IP_FEAT_SINGLE_SSL_SOCKET
- SL_SI91X_TCP_IP_FEAT_LOAD_PUBLIC_PRIVATE_CERTS
- SL_SI91X_TCP_IP_FEAT_LOAD_CERTS_INTO_RAM
- SL_SI91X_TCP_IP_FEAT_POP3_CLIENT
- SL_SI91X_TCP_IP_FEAT_OTAF
- SL_SI91X_TCP_IP_FEAT_EXTENSION_VALID

Extended TCP/IP Feature Bitmap

- SL_SI91X_EXT_TCP_FEAT_DHCP_OPT77
- SL_SI91X_EXT_TCP_IP_HTTP_SERVER_BYPASS
- SL_SI91X_EXT_TCP_IP_BI_DIR_ACK_UPDATE
- SL_SI91X_EXT_TCP_IP_WINDOW_DIV
- SL_SI91X_EXT_TCP_IP_CERT_BYPASS
- SL_SI91X_EXT_TCP_IP_SSL_16K_RECORD
- SL_SI91X_EXT_TCP_IP_DNS_CLIENT_BYPASS
- SL_SI91X_EXT_TCP_IP_WINDOW_SCALING
- SL_SI91X_EXT_TCP_IP_DUAL_MODE_ENABLE
- SL_SI91X_EXT_TCP_IP_ETH_WIFI_BRIDGE
- SL_SI91X_EXT_DYNAMIC_COEX_MEMORY
- SL_SI91X_EXT_TCP_IP_TOTAL_SELECTS
- SL_SI91X_EXT_TCP_IP_WAIT_FOR_SOCKET_CLOSE
- SL_SI91X_EXT_EMB_MQTT_ENABLE
- SL_SI91X_EXT_TCP_IP_FEAT_SSL_HIGH_PERFORMANCE
- SL_SI91X_EXT_TCP_DYNAMIC_WINDOW_UPDATE_FROM_HOST
- SL_SI91X_EXT_TCP_MAX_RECV_LENGTH
- SL_SI91X_EXT_TCP_IP_FEAT_SSL_THREE_SOCKETS
- SL_SI91X_EXT_TCP_IP_FEAT_SSL_MEMORY_CLOUD
- SL_SI91X_CONFIG_FEAT_EXTENTION_VALID

BLE Feature Bitmap

- SL_SI91X_BLE_MAX_NBR_ATT_REC
- SL_SI91X_BLE_MAX_NBR_ATT_SERV
- SL_SI91X_BLE_MAX_NBR_PERIPHERALS
- SL_SI91X_BLE_PWR_INX
- SL_SI91X_BLE_PWR_SAVE_OPTIONS
- SL_SI91X_BLE_MAX_NBR_CENTRALS
- SL_SI91X_BLE_GATT_ASYNC_ENABLE
- SL_SI91X_916_BLE_COMPATIBLE_FEAT_ENABLE
- SL_SI91X_FEAT_BLE_CUSTOM_FEAT_EXTENTION_VALID

Extended BLE Custom Feature Bitmap

- SL_SI91X_BLE_NUM_CONN_EVENTS
- SL_SI91X_BLE_NUM_REC_BYTES
- SL_SI91X_BLE_GATT_INIT
- SL_SI91X_BLE_INDICATE_CONFIRMATION_FROM_HOST
- SL_SI91X_BLE_MTU_EXCHANGE_FROM_HOST
- SL_SI91X_BLE_SET_SCAN_RESP_DATA_FROM_HOST
- SL_SI91X_BLE_DISABLE_CODED_PHY_FROM_HOST
- SL_SI91X_BLE_ENABLE_ADV_EXTN
- SL_SI91X_BLE_AE_MAX_ADV_SETS

Bluetooth Feature Bitmap

- SL_SI91X_BT_BDR_MODE_ENABLE
- SL_SI91X_BT_BDR_MODE_LP_CHAIN_ENABLE
- SL_SI91X_BT_PWR_CTRL_DISABLE
- SL_SI91X_BT_EDR_3MBPS_DISABLE
- SL_SI91X_BT_EDR_2MBPS_DISABLE
- SL_SI91X_BT_5_SLOT_PACKETS_DISABLE
- SL_SI91X_BT_3_SLOT_PACKETS_DISABLE
- SL_SI91X_TA_BASED_ENCODER_ENABLE
- SL_SI91X_HFP_PROFILE_ENABLE
- SL_SI91X_A2DP_PROFILE_ENABLE
- SL_SI91X_A2DP_SOURCE_ROLE_ENABLE
- SL_SI91X_A2DP_ACCELERATOR_MODE_ENABLE
- SL_SI91X_BT_BLE_CP_BUFF_SIZE
- SL_SI91X_BT_ATT_OVER_CLASSIC_ACL
- SL_SI91X_BT_RF_TYPE
- SL_SI91X_ENABLE_BLE_PROTOCOL

Device Feature Bitmap

- SL_SI91X_FEAT_SECURITY_OPEN
- SL_SI91X_FEAT_SECURITY_PSK
- SL_SI91X_FEAT_AGGREGATION
- SL_SI91X_FEAT_LP_GPIO_BASED_HANDSHAKE
- SL_SI91X_FEAT_ULP_GPIO_BASED_HANDSHAKE
- SL_SI91X_FEAT_DEV_TO_HOST_ULP_GPIO_1
- SL_SI91X_FEAT_RF_SUPPLY_VOL_3_3_VOLT
- SL_SI91X_FEAT_WPS_DISABLE
- SL_SI91X_FEAT_EAP_LEAP_IN_COEX
- SL_SI91X_FEAT_HIDE_PSK_CREDENTIALS
- SL_SI91X_FEAT_SSL_HIGH_STREAMING_BIT
- SL_SI91X_FEAT_SECURE_ATTESTATION

Calibration Flags

- SL_SI91X_BURN_GAIN_OFFSET
- SL_SI91X_BURN_FREQ_OFFSET
- SL_SI91X_SW_XO_CTUNE_VALID

- SL_SI91X_BURN_XO_FAST_DISABLE
- Burn Target Options
  - SL_SI91X_BURN_INTO_EFUSE
  - SL_SI91X_BURN_INTO_FLASH
- Configuration Feature Bitmap
  - SL_SI91X_FEAT_SLEEP_GPIO_SEL_BITMAP
  - SL_SI91X_FEAT_DVS_SEL_CONFIG_1
  - SL_SI91X_FEAT_DVS_SEL_CONFIG_2
  - SL_SI91X_FEAT_DVS_SEL_CONFIG_3
  - SL_SI91X_FEAT_DVS_SEL_CONFIG_4
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_100us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_200us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_300us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_400us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_500us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_600us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_700us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_800us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_900us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1000us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1100us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1200us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1300us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1400us
  - SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1500us
  - SL_SI91X_FEAT_EXTERNAL_LDO_SEL
  - SL_SI91X_FEAT_EXTERNAL_LDO_VOL
  - SL_SI91X_FEAT_EAP_TLS_V1P0
  - SL_SI91X_FEAT_EAP_TLS_V1P2
  - SL_SI91X_FEAT_CONC_STA_AP_DYN_SWITCH_SEL
  - SL_SI91X_ULP_GPIO9_FOR_UART2_TX
  - SL_SI91X_FEAT_DISABLE_MCS_5_6_7_DATARATES
  - SL_SI91X_FEAT_DISABLE_SHORT_GI
  - SL_SI91X_PTA_3WIRE_EN
  - SL_SI91X_PTA_3WIRE_CONFIG_SEL
  - SL_SI91X_XTAL_GOODTIME_1000us
  - SL_SI91X_XTAL_GOODTIME_2000us
  - SL_SI91X_XTAL_GOODTIME_3000us
  - SL_SI91X_XTAL_GOODTIME_600us
  - SL_SI91X_ENABLE_ENHANCED_MAX_PSP
  - SL_SI91X_ENABLE_DEBUG_BBP_TEST_PINS
- Custom Feature Bitmap
  - SL_SI91X_CUSTOM_FEAT_DISABLE_GATEWAY_IN_RSI_AP
  - SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_160MHZ

- SL_SI91X_CUSTOM_FEAT_AP_IN_HIDDEN_MODE
- SL_SI91X_CUSTOM_FEAT_DNS_SERVER_IN_DHCP_OFFER
- SL_SI91X_CUSTOM_FEAT_DFS_CHANNEL_SUPPORT
- SL_SI91X_CUSTOM_FEAT_LED_FEATURE
- SL_SI91X_CUSTOM_FEAT_ASYNC_CONNECTION_STATUS
- SL_SI91X_CUSTOM_FEAT_WAKE_ON_WIRELESS
- SL_SI91X_CUSTOM_FEAT_ENABLE_AP_BLACKLIST
- SL_SI91X_CUSTOM_FEAT_MAX_NUM_OF_CLIENTS
- SL_SI91X_CUSTOM_FEAT_ROAM_WITH_DEAUTH_OR_NULL_DATA
- SL_SI91X_CUSTOM_FEAT_TRIGGER_AUTO_CONFIG
- SL_SI91X_CUSTOM_FEAT_LIMIT_PACKETS_PER_STA
- SL_SI91X_CUSTOM_FEAT_HTTP_HTTPS_AUTH
- SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_120MHZ
- SL_SI91X_CUSTOM_FEAT_HTTP_SERVER_CRED_TO_HOST
- SL_SI91X_CUSTOM_FEAT_REJECT_CONNECT_REQ_IMMEDIATELY
- SL_SI91X_CUSTOM_FEAT_DUAL_BAND_ROAM_VCSAFD
- SL_SI91X_CUSTOM_FEAT_RTC_FROM_HOST
- SL_SI91X_CUSTOM_FEAT_BT_IAP
- SL_SI91X_CUSTOM_FEAT_EXTENTION_VALID

Extended Custom Feature Bitmap

- SL_SI91X_EXT_FEAT_RSA_KEY_WITH_4096_SUPPORT
- SL_SI91X_EXT_FEAT_TELEC_SUPPORT
- SL_SI91X_EXT_FEAT_SSL_CERT_WITH_4096_KEY_SUPPORT
- SL_SI91X_EXT_FEAT_AP_BROADCAST_PKT_SND_B4_DTIM
- SL_SI91X_EXT_FEAT_FCC_LOW_PWR
- SL_SI91X_EXT_FEAT_PUF
- SL_SI91X_EXT_FEAT_SPECTRAL_MASK_NOKIA
- SL_SI91X_EXT_HTTP_SKIP_DEFAULT_LEADING_CHARACTER
- SL_SI91X_EXT_FEAT_PUF_PRIVATE_KEY
- SL_SI91X_EXT_FEAT_ENABLE_11R_OTA
- SL_SI91X_EXT_FEAT_IEEE_80211J
- SL_SI91X_EXT_FEAT_IEEE_80211W
- SL_SI91X_EXT_FEAT_SSL_VERSIONS_SUPPORT
- SL_SI91X_EXT_FEAT_16th_STATION_IN_AP_MODE
- SL_SI91X_EXT_FEAT_ENABLE_11R_ODS
- SL_SI91X_EXT_FEAT_HTTP_OTAF_SUPPORT
- SL_SI91X_EXT_FEAT_LOW_POWER_MODE
- SL_SI91X_EXT_FEAT_256K_MODE
- SL_SI91X_RAM_LEVEL_NWP_MEDIUM_MCU_MEDIUM
- SL_SI91X_EXT_FEAT_320K_MODE
- SL_SI91X_RAM_LEVEL_NWP_ADV_MCU_BASIC
- SL_SI91X_EXT_FEAT_384K_MODE
- SL_SI91X_RAM_LEVEL_NWP_ALL_MCU_ZERO
- SL_SI91X_EXT_FEAT_XTAL_CLK_ENABLE

- is_counter_mode_32bit_enabled
- is_counter0_soft_reset_enabled
- is_counter0_periodic_enabled
- is_counter0_trigger_enabled
- is_counter0_sync_trigger_enabled
- is_counter0_buffer_enabled
- is_counter1_soft_reset_enabled
- is_counter1_periodic_enabled
- is_counter1_trigger_enabled
- is_counter1_sync_trigger_enabled
- is_counter1_buffer_enabled
- counter0_direction
- counter1_direction

sl_config_timer_ocu_config_t
- is_counter0_ocu_output_enabled
- is_counter0_ocu_dma_enabled
- is_counter0_ocu_8bit_mode_enabled
- is_counter0_ocu_sync_enabled
- is_counter1_ocu_output_enabled
- is_counter1_ocu_dma_enabled
- is_counter1_ocu_mode_enabled
- is_counter1_ocu_sync_enabled
- is_counter0_toggle_output_high_enabled
- is_counter0_toggle_output_low_enabled
- is_counter1_toggle_output_high_enabled
- is_counter1_toggle_output_low_enabled

sl_config_timer_ocu_control_t
- is_counter_number_1
- is_dma_state_enabled
- params
- callback

sl_config_action_event_t
- action
- and_event_counter0
- or_event_counter0
- and_event_valid_bits_counter0
- or_event_valid_bits_counter0
- and_event_counter1
- or_event_counter1
- and_event_valid_bits_counter1
- or_event_valid_bits_counter1

sl_config_timer_interrupt_flags_t
- is_counter0_event_interrupt_enabled
- is_counter0_fifo_full_interrupt_enabled

- sl_i2c_driver_send_data_blocking
- sl_i2c_driver_send_data_non_blocking
- sl_i2c_driver_receive_data_blocking
- sl_i2c_driver_receive_data_non_blocking
- sl_i2c_driver_transfer_data
- sl_i2c_driver_deinit
- sl_si91x_i2c_pin_init
- SL_I2C0_DMA_TX_CHANNEL
- SL_I2C0_DMA_RX_CHANNEL
- SL_I2C1_DMA_TX_CHANNEL
- SL_I2C1_DMA_RX_CHANNEL
- SL_I2C2_DMA_TX_CHANNEL
- SL_I2C2_DMA_RX_CHANNEL

I2S

- sl_i2s_version_t
  - release
  - major
  - minor
- sl_i2s_xfer_config_t
  - mode
  - sync
  - protocol
  - resolution
  - data_size
  - sampling_rate
  - transfer_type
- i2s_event_typedef_t
- sl_i2s_power_state_t
- sl_i2s_mode_t
- sl_sai_protocol_t
- sl_i2s_sync_t
- sl_i2s_xfer_type_t
- sl_i2s_xfer_size_t
- sl_i2s_data_resolution_t
- sl_i2s_sampling_rate_t
- sl_i2s_signal_event_t
- sl_i2s_status_t
- sl_i2s_driver_t
- sl_i2s_handle_t
- sl_si91x_i2s_init
- sl_si91x_i2s_deinit
- sl_si91x_i2s_configure_power_mode
- sl_si91x_i2s_config_transmit_receive
- sl_si91x_i2s_transmit_data

- tXHS_US
- tHS_US

PWM

- sl_pwm_version_t
  - release
  - major
  - minor
- sl_pwm_config_t
  - channel
  - frequency
  - is_polarity_low
  - is_polarity_high
  - is_mode
  - base_time_counter_initial_value
  - duty_cycle
  - base_timer_mode
  - channel_timer_selection
- sl_pwm_init_t
  - port_l
  - pin_l
  - port_h
  - pin_h
  - mux_l
  - mux_h
  - pad_l
  - pad_h
- sl_pwm_fault_init_t
  - port
  - pin
  - mux
  - pad
- sl_pwm_fault_t
- sl_pwm_dead_time_t
- sl_pwm_duty_cycle_t
- sl_pwm_override_t
- sl_pwm_output_fault_t
- sl_pwm_event_t
- sl_pwm_channel_t
- sl_pwm_timer_t
- sl_pwm_polarity_low_t
- sl_pwm_polarity_high_t
- sl_pwm_mode_t
- sl_pwm_base_timer_mode_t
- sl_pwm_output_t

- M4_HOST_INTR_STATUS_REG
- M4_HOST_INTR_CLEAR
- MISC_CFG_HOST_CTRL
- SL_SDIO_WR_INT_EN
- SL_SDIO_RD_INT_EN
- SL_SDIO_CSA_INT_EN
- SL_SDIO_CMD52_INT_EN
- SL_SDIO_PWR_LEV_INT_EN
- SL_SDIO_CRC_ERR_INT_EN
- SL_SDIO_ABORT_INT_EN
- SL_SDIO_TOUT_INT_EN
- SL_SDIO_WR_INT_MSK
- SL_SDIO_RD_INT_MSK
- SL_SDIO_CSA_INT_MSK
- SL_SDIO_CMD52_INT_MSK
- SL_SDIO_PWR_LEV_INT_MSK
- SL_SDIO_CRC_ERR_INT_MSK
- SL_SDIO_ABORT_INT_MSK
- SL_SDIO_TOUT_INT_MSK
- SL_SDIO_WR_INT_UNMSK
- SL_SDIO_RD_INT_UNMSK
- SL_SDIO_CSA_INT_UNMSK
- SL_SDIO_CMD52_INT_UNMSK
- SL_SDIO_PWR_LEV_INT_UNMSK
- SL_SDIO_CRC_ERR_INT_UNMSK
- SL_SDIO_ABORT_INT_UNMSK
- SL_SDIO_TOUT_INT_UNMSK

Serial Input-Output

- sl_sio_version_t
  - release
  - major
  - minor
- sl_sio_spi_t
  - spi_cs_port
  - spi_cs_pin
  - spi_cs_mux
  - spi_cs_pad
  - spi_clk_port
  - spi_clk_pin
  - spi_clk_mux
  - spi_clk_pad
  - spi_mosi_port
  - spi_mosi_pin
  - spi_mosi_mux

sli_si91x_power_manager_set_ram_retention_configuration
sli_si91x_power_manager_configure_clock
sli_si91x_power_manager_init_hardware
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS4
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS4
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS3
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS3
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS2
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS2
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS1
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_SLEEP
SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_STANDBY
SL_SI91X_POWER_MANAGER_DST_WAKEUP
SL_SI91X_POWER_MANAGER_HOST_WAKEUP
SL_SI91X_POWER_MANAGER_WIRELESS_WAKEUP
SL_SI91X_POWER_MANAGER_M4_PROCESSOR_WAKEUP
SL_SI91X_POWER_MANAGER_GPIO_WAKEUP
SL_SI91X_POWER_MANAGER_COMPARATOR_WAKEUP
SL_SI91X_POWER_MANAGER_SYSRTC_WAKEUP
SL_SI91X_POWER_MANAGER_ULPSS_WAKEUP
SL_SI91X_POWER_MANAGER_SDCSS_WAKEUP
SL_SI91X_POWER_MANAGER_ALARM_WAKEUP
SL_SI91X_POWER_MANAGER_SEC_WAKEUP
SL_SI91X_POWER_MANAGER_MSEC_WAKEUP
SL_SI91X_POWER_MANAGER_WDT_WAKEUP
SL_SI91X_POWER_MANAGER_HPSRAM_RETENTION_ULP_ENABLE
SL_SI91X_POWER_MANAGER_M4SS_RAM_RETENTION_ENABLE
SL_SI91X_POWER_MANAGER_M4ULP_RAM_RETENTION_ENABLE
SL_SI91X_POWER_MANAGER_TA_RAM_RETENTION_ENABLE
SL_SI91X_POWER_MANAGER_ULPSS_RAM_RETENTION_ENABLE
SL_SI91X_POWER_MANAGER_M4ULP_RAM16K_RETENTION_ENABLE
SL_SI91X_POWER_MANAGER_M4SS_PG_EFUSE
SL_SI91X_POWER_MANAGER_M4SS_PG_RPDMA
SL_SI91X_POWER_MANAGER_M4SS_PG_SDIO_SPI
SL_SI91X_POWER_MANAGER_M4SS_PG_QSPI
SL_SI91X_POWER_MANAGER_M4SS_PG_IID
SL_SI91X_POWER_MANAGER_M4SS_PG_M4_DEBUG
SL_SI91X_POWER_MANAGER_M4SS_PG_M4_CORE
SL_SI91X_POWER_MANAGER_M4SS_PG_EXTERNAL_ROM
SL_SI91X_POWER_MANAGER_ULPSS_PG_MISC
SL_SI91X_POWER_MANAGER_ULPSS_PG_CAP
SL_SI91X_POWER_MANAGER_ULPSS_PG_UART
SL_SI91X_POWER_MANAGER_ULPSS_PG_SSI

- SL_SI91X_POWER_MANAGER_ULPSS_PG_I2S
- SL_SI91X_POWER_MANAGER_ULPSS_PG_I2C
- SL_SI91X_POWER_MANAGER_ULPSS_PG_AUX
- SL_SI91X_POWER_MANAGER_ULPSS_PG_IR
- SL_SI91X_POWER_MANAGER_ULPSS_PG_UDMA
- SL_SI91X_POWER_MANAGER_ULPSS_PG_FIM
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCUBFFS
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCUFSM
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCURTC
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCUWDT
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCUPS
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCUTS
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE1
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE2
- SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE3
- SL_SI91X_POWER_MANAGER_NPSS_PG_TIMEPERIOD
- SL_SI91X_POWER_MANAGER_NPSS_PG_NWPAPB_MCU_CTRL
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_1
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_2
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_3
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_4
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_5
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_6
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_7
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_8
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_9
- SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_10
- SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_1
- SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_2
- SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_3
- SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_4

Sensor Hub

- sl_sensorhub_errors_t
  - i2c
  - spi
  - adc
  - sensor_global_status
- sl_data_deliver_type_t
  - data_mode
  - threshold
  - timeout
  - numofsamples
  - @1
- sl_sensor_info_t

- sensor_name
- sensor_intr_type
- sampling_intr_req_pin
- sampling_interval
- address
- channel
- @3
- sensor_id
- sensor_bus
- sensor_mode
- sensor_range
- data_deliver
- sensor_data_ptr

sl_sensor_handle_t
- sensor_handle
- ctrl_handle
- sensor_event_bit
- event_ack
- max_samples
- config_st
- sensor_impl
- sensor_status
- timer_handle

sl_sensor_list_t
- sensor_index
- sl_sensors_st

sl_intr_list_t
- sensor_list_index
- intr
- adc_intr_channel

sl_intr_list_map_t
- map_index
- map_table

sl_em_event_t
- em_sensor_data
- sensor_id
- event

sl_sensor_cb_info_t
- cb_event
- cb_event_ack

sl_i2c_config_t
- i2c_id
- i2c_power_state
- i2c_control_mode

SL_STATUS_BOOTLOADER_VERSION_MISMATCH
SL_STATUS_WAITING_FOR_BOARD_READY
SL_STATUS_VALID_FIRMWARE_NOT_PRESENT
SL_STATUS_INVALID_OPTION
SL_STATUS_SPI_BUSY
SL_STATUS_CARD_READY_TIMEOUT
SL_STATUS_FW_LOAD_OR_UPGRADE_TIMEOUT
SL_STATUS_WIFI_DOES_NOT_EXIST
SL_STATUS_WIFI_NOT_AUTHENTICATED
SL_STATUS_WIFI_NOT_KEYED
SL_STATUS_WIFI_IOCTL_FAIL
SL_STATUS_WIFI_BUFFER_UNAVAILABLE_TEMPORARY
SL_STATUS_WIFI_BUFFER_UNAVAILABLE_PERMANENT
SL_STATUS_WIFI_WPS_PBC_OVERLAP
SL_STATUS_WIFI_CONNECTION_LOST
SL_STATUS_WIFI_OUT_OF_EVENT_HANDLER_SPACE
SL_STATUS_WIFI_SEMAPHORE_ERROR
SL_STATUS_WIFI_FLOW_CONTROLLED
SL_STATUS_WIFI_NO_CREDITS
SL_STATUS_WIFI_NO_PACKET_TO_SEND
SL_STATUS_WIFI_CORE_CLOCK_NOT_ENABLED
SL_STATUS_WIFI_CORE_IN_RESET
SL_STATUS_WIFI_UNSUPPORTED
SL_STATUS_WIFI_BUS_WRITE_REGISTER_ERROR
SL_STATUS_WIFI_SDIO_BUS_UP_FAIL
SL_STATUS_WIFI_JOIN_IN_PROGRESS
SL_STATUS_WIFI_NETWORK_NOT_FOUND
SL_STATUS_WIFI_INVALID_JOIN_STATUS
SL_STATUS_WIFI_UNKNOWN_INTERFACE
SL_STATUS_WIFI_SDIO_RX_FAIL
SL_STATUS_WIFI_HWTAG_MISMATCH
SL_STATUS_WIFI_RX_BUFFER_ALLOC_FAIL
SL_STATUS_WIFI_BUS_READ_REGISTER_ERROR
SL_STATUS_WIFI_THREAD_CREATE_FAILED
SL_STATUS_WIFI_QUEUE_ERROR
SL_STATUS_WIFI_BUFFER_POINTER_MOVE_ERROR
SL_STATUS_WIFI_BUFFER_SIZE_SET_ERROR
SL_STATUS_WIFI_THREAD_STACK_NULL
SL_STATUS_WIFI_THREAD_DELETE_FAIL
SL_STATUS_WIFI_SLEEP_ERROR
SL_STATUS_WIFI_BUFFER_ALLOC_FAIL
SL_STATUS_WIFI_INTERFACE_NOT_UP
SL_STATUS_WIFI_DELAY_TOO_LONG

- SL_STATUS_WIFI_INVALID_DUTY_CYCLE
- SL_STATUS_WIFI_PMK_WRONG_LENGTH
- SL_STATUS_WIFI_UNKNOWN_SECURITY_TYPE
- SL_STATUS_WIFI_WEP_NOT_ALLOWED
- SL_STATUS_WIFI_WPA_KEYLEN_BAD
- SL_STATUS_WIFI_FILTER_NOT_FOUND
- SL_STATUS_WIFI_SPI_ID_READ_FAIL
- SL_STATUS_WIFI_SPI_SIZE_MISMATCH
- SL_STATUS_WIFI_ADDRESS_ALREADY_REGISTERED
- SL_STATUS_WIFI_SDIO_RETRIES_EXCEEDED
- SL_STATUS_WIFI_NULL_PTR_ARG
- SL_STATUS_WIFI_THREAD_FINISH_FAIL
- SL_STATUS_WIFI_WAIT_ABORTED
- SL_STATUS_WIFI_QUEUE_MESSAGE_UNALIGNED
- SL_STATUS_WIFI_MUTEX_ERROR
- SL_STATUS_WIFI_SECURE_LINK_DECRYPT_ERROR
- SL_STATUS_WIFI_SECURE_LINK_KEY_RENEGOTIATION_ERROR
- SL_STATUS_WIFI_INVALID_OPERMODE
- SL_STATUS_WIFI_INVALID_ENCRYPTION_METHOD
- SL_STATUS_TRNG_DUPLICATE_ENTROPY
- SL_STATUS_CRYPTO_INVALID_PARAMETER
- SL_STATUS_CRYPTO_INVALID_SIGNATURE
- SL_STATUS_SI91X_SCAN_ISSUED_IN_ASSOCIATED_STATE
- SL_STATUS_SI91X_NO_AP_FOUND
- SL_STATUS_SI91X_INVALID_PSK_IN_WEP_SECURITY
- SL_STATUS_SI91X_INVALID_BAND
- SL_STATUS_SI91X_UNASSOCIATED
- SL_STATUS_SI91X_DEAUTHENTICATION_RECEIVED_FROM_AP
- SL_STATUS_SI91X_ASSOCIATION_FAILED
- SL_STATUS_SI91X_INVALID_CHANNEL
- SL_STATUS_SI91X_JOIN_AUTHENTICATION_FAILED
- SL_STATUS_SI91X_BEACON_MISSED_FROM_AP_DURING_JOIN
- SL_STATUS_SI91X_INVALID_MAC_SUPPLIED
- SL_STATUS_SI91X_EAP_CONFIG_NOT_DONE
- SL_STATUS_SI91X_MEMORY_FAILED_FROM_MODULE
- SL_STATUS_SI91X_INSUFFICIENT_INFO
- SL_STATUS_SI91X_NOT_AP_INTERFACE
- SL_STATUS_SI91X_INVALID_PUSH_BUTTON_SEQUENCE
- SL_STATUS_SI91X_REJOIN_FAILURE
- SL_STATUS_SI91X_FREQUENCY_NOT_SUPPORTED
- SL_STATUS_SI91X_INVALID_OPERMODE
- SL_STATUS_SI91X_EAP_CONFIG_FAILED
- SL_STATUS_SI91X_P2P_CONFIG_FAILED

- SL_STATUS_SI91X_GROUP_OWNER_NEGOTIATION_FAILED
- SL_STATUS_SI91X_JOIN_TIMEOUT
- SL_STATUS_SI91X_COMMAND_GIVEN_IN_INVALID_STATE
- SL_STATUS_SI91X_INVALID_QUERY_GO_PARAMS
- SL_STATUS_SI91X_ACCESS_POINT_FAILED
- SL_STATUS_SI91X_INVALID_SCAN_INFO
- SL_STATUS_SI91X_COMMAND_ISSUED_IN_REJOIN_STATE
- SL_STATUS_SI91X_WRONG_PARAMETERS
- SL_STATUS_SI91X_PROVISION_DISCOVERY_FAILED_IN_P2P
- SL_STATUS_SI91X_INVALID_PSK_LENGTH
- SL_STATUS_SI91X_FAILED_TO_CLEAR_OR_SET_EAP_CERTIFICATE
- SL_STATUS_SI91X_P2P_GO_NEGOTIATED_FAILED
- SL_STATUS_SI91X_ASSOCIATION_TIMEOUT_IN_P2P_WPS_MODE
- SL_STATUS_SI91X_COMMAND_ISSUED_WHILE_INTERNAL_OPERATION
- SL_STATUS_SI91X_INVALID_WEP_KEY_LEN
- SL_STATUS_SI91X_ICMP_REQUEST_TIMEOUT_ERROR
- SL_STATUS_SI91X_ICMP_DATA_SIZE_EXCEED_MAX_LIMIT
- SL_STATUS_SI91X_SEND_DATA_PACKET_EXCEED_LIMIT
- SL_STATUS_SI91X_ARP_CACHE_NOT_FOUND
- SL_STATUS_SI91X_UART_COMMAND_TIMEOUT
- SL_STATUS_SI91X_FIXED_DATA_RATE_NOT_SUPPORTED_BY_AP
- SL_STATUS_SI91X_USERNAME_PASSWORD_CLIENTID_TOPIC_MAX_LEN
- SL_STATUS_SI91X_INVALID_WPS_PIN
- SL_STATUS_SI91X_INVALID_WPS_PIN_LEN
- SL_STATUS_SI91X_INVALID_PMK_LEN
- SL_STATUS_SI91X_SSID_NOT_PRESENT_FOR_PMK_GENERATION
- SL_STATUS_SI91X_SSID_INCORRECT_PMK_GENERATION
- SL_STATUS_SI91X_BAND_NOT_SUPPORTED
- SL_STATUS_SI91X_INVALID_USR_STORE_CONFIGURATION_LEN
- SL_STATUS_SI91X_INVALID_COMMAND_LEN
- SL_STATUS_SI91X_DATA_PACKET_DROPPED
- SL_STATUS_SI91X_WEP_KEY_NOT_GIVEN
- SL_STATUS_SI91X_INVALID_STORE_CONFIG_PROFILE
- SL_STATUS_SI91X_MISSING_PSK_OR_PMK
- SL_STATUS_SI91X_INVALID_SECURITY_MODE_IN_JOIN_COMMAND
- SL_STATUS_SI91X_MAX_BEACON_MISCOUNT
- SL_STATUS_SI91X_DEAUTH_REQUEST_FROM_SUPPLICANT
- SL_STATUS_SI91X_DEAUTH_REQUEST_FROM_FROM_AP
- SL_STATUS_SI91X_MISSED_SYNCHRONIZATION
- SL_STATUS_SI91X_AUTHENTICATION_TIMEOUT
- SL_STATUS_SI91X_ASSOCIATION_TIMEOUT
- SL_STATUS_SI91X_BG_SCAN_NOT_ALLOWED
- SL_STATUS_SI91X_SSID_MISMATCH

- SL_STATUS_SI91X_CLIENT_MAX_SUPPORTED_EXCEEDED
- SL_STATUS_SI91X_HT_CAPABILITIES_NOT_SUPPORTED
- SL_STATUS_SI91X_UART_FLOW_NOT_SUPPORTED
- SL_STATUS_SI91X_ZB_BT_BLE_PKT_RECEIVED
- SL_STATUS_SI91X_MGMT_PKT_DROPPED
- SL_STATUS_SI91X_INVALID_RF_CURRENT_MODE
- SL_STATUS_SI91X_POWER_SAVE_NOT_SUPPORTED
- SL_STATUS_SI91X_CONCURRENT_AP_IN_CONNECTED_STATE
- SL_STATUS_SI91X_CONNECTED_AP_OR_STATION_CHANNEL_MISMATCH
- SL_STATUS_SI91X_IAP_COPROCESSOR_ERROR
- SL_STATUS_SI91X_WPS_NOT_SUPPORTED
- SL_STATUS_SI91X_CONCURRENT_AP_CHANNEL_MISMATCH
- SL_STATUS_SI91X_PBC_SESSION_OVERLAP
- SL_STATUS_SI91X_BT_FEATURE_BITMAP_INVALID
- SL_STATUS_SI91X_FOUR_WAY_HANDSHAKE_FAILED
- SL_STATUS_SI91X_MAC_ADDRESS_NOT_PRESENT_IN_MAC_JOIN
- SL_STATUS_SI91X_CONCURRENT_MODE_DOWN
- SL_STATUS_SI91X_CERTIFICATE_LOAD_NOT_ALLOWED_IN_FLASH
- SL_STATUS_SI91X_CERTIFICATE_LOAD_NOT_ALLOWED_IN_RAM
- SL_STATUS_SI91X_WRONG_CERTIFICATE_LOAD_INDEX
- SL_STATUS_SI91X_AP_HT_CAPS_NOT_ENABLED
- SL_STATUS_SI91X_ADDRESS_FAMILY_NOT_SUPPORTED
- SL_STATUS_SI91X_INVALID_BEACON_INTERVAL_OR_DTM_PERIOD
- SL_STATUS_SI91X_INVALID_CONFIG_RANGE_PROVIDED
- SL_STATUS_SI91X_INVALID_CONFIG_TYPE
- SL_STATUS_SI91X_ERROR_WITH_MQTT_COMMAND
- SL_STATUS_SI91X_HIGHER_LISTEN_INTERVAL
- SL_STATUS_SI91X_WLAN_RADIO_DEREGISTERED
- SL_STATUS_SI91X_SAE_FAILURE_DUE_TO_MULTIPLE_CONFIRM_FRAMES_FROM_AP
- SL_STATUS_SI91X_EC_GROUP_STATION_UNSUPPORTED_BY_AP
- SL_STATUS_TWT_SUPPORT_NOT_ENABLED_ERR
- SL_STATUS_TWT_SETUP_ERR_SESSION_ACTIVE
- SL_STATUS_TWT_TEARDOWN_ERR_FLOWID_NOT_MATCHED
- SL_STATUS_TWT_TEARDOWN_ERR_NOACTIVE_SESS
- SL_STATUS_TWT_SESSION_NOT_FEASIBLE
- SL_STATUS_SI91X_RESCHEDULE_TWT_NOT_SUPPORTED
- SL_STATUS_SI91X_RESCHEDULE_TWT_ERR_NOACTIVE_SESS
- SL_STATUS_SI91X_TWT_RESCHEDULING_IN_PROGRESS
- SL_STATUS_SI91X_RESCHEDULE_TWT_PACKET_CREATION_FAILED
- SL_STATUS_SI91X_MQTT_ERROR_UNACCEPTABLE_PROTOCOL
- SL_STATUS_SI91X_MQTT_ERROR_IDENTIFIER_REJECTED
- SL_STATUS_SI91X_MQTT_ERROR_SERVER_UNAVAILABLE
- SL_STATUS_SI91X_MQTT_ERROR_BAD_USERNAME_PASSWORD

- SL_STATUS_SI91X_MQTT_ERROR_NOT_AUTHORIZED
- SL_STATUS_SI91X_DUPLICATE_ENTRY_EXISTS_IN_DNS_SERVER_TABLE
- SL_STATUS_SI91X_NO_MEM_AVAILABLE
- SL_STATUS_SI91X_INVALID_CHARACTERS_IN_JSON_OBJECT
- SL_STATUS_SI91X_NO_KEY_FOUND
- SL_STATUS_SI91X_NO_FILE_FOUND
- SL_STATUS_SI91X_NO_WEB_PAGE_EXISTS_WITH_SAME_FILENAME
- SL_STATUS_SI91X_SPACE_UNAVAILABLE_FOR_NEW_FILE
- SL_STATUS_SI91X_INVALID_INPUT_DATA
- SL_STATUS_SI91X_NO_SPACE_AVAILABLE_FOR_NEW_FILE
- SL_STATUS_SI91X_EXISTING_FILE_OVERWRITE
- SL_STATUS_SI91X_NO_SUCH_FILE_FOUND
- SL_STATUS_SI91X_MEMORY_ERROR
- SL_STATUS_SI91X_RECEIVED_MORE_WEB_PAGE_DATA
- SL_STATUS_SI91X_SET_REGION_ERROR
- SL_STATUS_SI91X_INVALID_WEBPAGE_CURRENT_CHUNK_LEN
- SL_STATUS_SI91X_AP_SET_REGION_COMMAND_ERROR
- SL_STATUS_SI91X_AP_SET_REGION_COMMAND_PARAMETERS_ERROR
- SL_STATUS_SI91X_REGION_CODE_NOT_SUPPORTED
- SL_STATUS_SI91X_EXTRACTING_COUNTRY_REGION_FROM_BEACON_FAILED
- SL_STATUS_SI91X_SELECTED_REGION_NOT_SUPPORTED
- SL_STATUS_SI91X_SSL_TLS_CONTEXT_CREATION_FAILED
- SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_FAIL
- SL_STATUS_SI91X_SSL_TLS_MAX_SOCKETS_REACHED
- SL_STATUS_SI91X_FTP_CLIENT_NOT_CONNECTED
- SL_STATUS_SI91X_CIPHER_SET_FAILED
- SL_STATUS_SI91X_HTTP_CREDENTIALS_MAX_LEN_EXCEEDED
- SL_STATUS_SI91X_FEATURE_NOT_SUPPORTED
- SL_STATUS_SI91X_FLASH_WRITE_OR_FLASH_DATA_VERIFICATION_FAILED
- SL_STATUS_SI91X_CALIBRATION_DATA_VERIFICATION_FAILED
- SL_STATUS_SI91X_SNMP_INTERNAL_ERROR
- SL_STATUS_SI91X_SNMP_INVALID_IP_PROTOCOL
- SL_STATUS_SI91X_NO_DATA_RECEIVED_OR_RECEIVE_TIMEOUT
- SL_STATUS_SI91X_INSUFFICIENT_DATA_FOR_TIME_CONVERSION
- SL_STATUS_SI91X_INVALID_SNTP_SERVER_ADDRESS
- SL_STATUS_SI91X_SNTP_CLIENT_NOT_STARTED
- SL_STATUS_SI91X_SNTP_SERVER_UNAVAILABLE
- SL_STATUS_SI91X_SNTP_SERVER_AUTHENTICATION_FAILED
- SL_STATUS_SI91X_INTERNAL_ERROR
- SL_STATUS_SI91X_MULTICAST_IP_ADDRESS_ENTRY_NOT_FOUND
- SL_STATUS_SI91X_MULTICAST_NO_ENTRIES_FOUND
- SL_STATUS_SI91X_IP_ADDRESS_ERROR
- SL_STATUS_SI91X_SOCKET_ALREADY_BOUND

- SL_STATUS_SI91X_PORT_UNAVAILABLE
- SL_STATUS_SI91X_SOCKET_NOT_CREATED
- SL_STATUS_SI91X_ICMP_REQUEST_FAILED
- SL_STATUS_SI91X_MAX_LISTEN_SOCKETS_REACHED
- SL_STATUS_SI91X_DHCP_DUPLICATE_LISTEN
- SL_STATUS_SI91X_PORT_NOT_IN_CLOSE_STATE
- SL_STATUS_SI91X_SOCKET_CLOSED
- SL_STATUS_SI91X_PROCESS_IN_PROGRESS
- SL_STATUS_SI91X_CONNECT_TO_NON_EXISTING_TCP_SERVER_SOCKET
- SL_STATUS_SI91X_ERROR_IN_LEN_OF_THE_COMMAND
- SL_STATUS_SI91X_WRONG_PACKET_INFO
- SL_STATUS_SI91X_SOCKET_STILL_BOUND
- SL_STATUS_SI91X_NO_FREE_PORT
- SL_STATUS_SI91X_INVALID_PORT
- SL_STATUS_SI91X_FEATURE_UNSUPPORTED
- SL_STATUS_SI91X_SOCKET_IN_UNCONNECTED_STATE
- SL_STATUS_SI91X_POP3_SESSION_CREATION_FAILED
- SL_STATUS_SI91X_DHCPV6_HANDSHAKE_FAIL
- SL_STATUS_SI91X_DHCP_INVALID_IP_RESPONSE
- SL_STATUS_SI91X_SMTP_AUTHENTICATION_ERROR
- SL_STATUS_SI91X_SMTP_OVER_SIZE_MAIL_DATA
- SL_STATUS_SI91X_SMTP_INVALID_SERVER_REPLY
- SL_STATUS_SI91X_SMTP_DNS_QUERY_FAILED
- SL_STATUS_SI91X_SMTP_BAD_DNS_ADDRESS
- SL_STATUS_SI91X_SMTP_INVALID_PARAMETERS
- SL_STATUS_SI91X_SMTP_PACKET_ALLOCATION_FAILED
- SL_STATUS_SI91X_SMTP_GREET_REPLY_FAILED
- SL_STATUS_SI91X_SMTP_PARAMETER_ERROR
- SL_STATUS_SI91X_SMTP_MAIL_REPLY_ERROR
- SL_STATUS_SI91X_SMTP_RCPT_REPLY_ERROR
- SL_STATUS_SI91X_SMTP_MESSAGE_REPLY_ERROR
- SL_STATUS_SI91X_SMTP_DATA_REPLY_ERROR
- SL_STATUS_SI91X_SMTP_AUTH_REPLY_ERROR
- SL_STATUS_SI91X_SMTP_SERVER_REPLY_ERROR
- SL_STATUS_SI91X_DNS_DUPLICATE_ENTRY
- SL_STATUS_SI91X_SMTP_OVERSIZE_SERVER_REPLY
- SL_STATUS_SI91X_SMTP_CLIENT_NOT_INITIALIZED
- SL_STATUS_SI91X_DNS_IPV6_NOT_SUPPORTED
- SL_STATUS_SI91X_INVALID_MAIL_INDEX_FOR_POP3_MAIL_RETRIEVE_COMMAND
- SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_FAILED
- SL_STATUS_SI91X_FTP_CLIENT_DISCONNECTED
- SL_STATUS_SI91X_FTP_CLIENT_NOT_DISCONNECTED
- SL_STATUS_SI91X_FTP_FILE_NOT_OPENED

SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_TIMEOUT_OR_FTP_FILE_NOT_CLOSED
SL_STATUS_SI91X_FTP_EXPECTED_1XX_RESPONSE_NOT_RECEIVED
SL_STATUS_SI91X_FTP_EXPECTED_2XX_RESPONSE_NOT_RECEIVED
SL_STATUS_SI91X_FTP_EXPECTED_22X_RESPONSE_NOT_RECEIVED
SL_STATUS_SI91X_FTP_EXPECTED_23X_RESPONSE_NOT_RECEIVED
SL_STATUS_SI91X_FTP_EXPECTED_3XX_RESPONSE_NOT_RECEIVED
SL_STATUS_SI91X_FTP_EXPECTED_33X_RESPONSE_NOT_RECEIVED
SL_STATUS_SI91X_HTTP_TIMEOUT
SL_STATUS_SI91X_HTTP_FAILED
SL_STATUS_SI91X_HTTP_PUT_CLIENT_TIMEOUT
SL_STATUS_SI91X_AUTHENTICATION_ERROR
SL_STATUS_SI91X_INVALID_PACKET_LENGTH
SL_STATUS_SI91X_SERVER_RESPONDS_BEFORE_REQUEST_COMPLETE
SL_STATUS_SI91X_HTTP_PASSWORD_TOO_LONG
SL_STATUS_SI91X_MQTT_PING_TIMEOUT
SL_STATUS_SI91X_MQTT_COMMAND_SENT_IN_INCORRECT_STATE
SL_STATUS_SI91X_MQTT_ACK_TIMEOUT
SL_STATUS_SI91X_POP3_INVALID_MAIL_INDEX
SL_STATUS_SI91X_SOCKET_NOT_CONNECTED
SL_STATUS_SI91X_SOCKET_LIMIT_EXCEEDED
SL_STATUS_SI91X_HTTP_OTAF_INVALID_PACKET
SL_STATUS_SI91X_TCP_IP_INIT_FAILED
SL_STATUS_SI91X_CONCURRENT_IP_CREATION_ERROR
SL_STATUS_SI91X_HTTP_OTAF_INCOMPLETE_PACKET
SL_STATUS_SI91X_INVALID_STORE_CONFIGURATION_PROFILE
SL_STATUS_SI91X_MQTT_REMOTE_TERMINATE_ERROR
SL_STATUS_SI91X_BYTE_STUFFING_ERROR_IN_AT_MODE
SL_STATUS_SI91X_INVALID_COMMAND_OR_OPERATION
SL_STATUS_SI91X_HTTP_OTAF_NO_PACKET
SL_STATUS_SI91X_TCP_SOCKET_NOT_CONNECTED
SL_STATUS_SI91X_MAX_STATION_COUNT_EXCEEDED
SL_STATUS_SI91X_UNABLE_TO_SEND_TCP_DATA
SL_STATUS_SI91X_SOCKET_BUFFER_TOO_SMALL
SL_STATUS_SI91X_INVALID_CONTENT_IN_DNS_RESPONSE
SL_STATUS_SI91X_DNS_CLASS_ERROR_IN_DNS_RESPONSE
SL_STATUS_SI91X_DNS_COUNT_ERROR_IN_DNS_RESPONSE
SL_STATUS_SI91X_DNS_RETURN_CODE_ERROR_IN_DNS_RESPONSE
SL_STATUS_SI91X_DNS_OPCODE_ERROR_IN_DNS_RESPONSE
SL_STATUS_SI91X_DNS_ID_MISMATCH
SL_STATUS_SI91X_INVALID_INPUT_IN_DNS_QUERY
SL_STATUS_SI91X_DNS_RESPONSE_TIMEOUT
SL_STATUS_SI91X_ARP_REQUEST_FAILURE
SL_STATUS_SI91X_UNABLE_TO_UPDATE_TCP_WINDOW

- SL_STATUS_SI91X_DHCP_LEASE_EXPIRED
- SL_STATUS_SI91X_DHCP_HANDSHAKE_FAILURE
- SL_STATUS_SI91X_WEBSOCKET_CREATION_FAILED
- SL_STATUS_SI91X_TRYING_TO_CONNECT_NON_EXISTENT_TCP_SERVER_SOCKET
- SL_STATUS_SI91X_TRYING_TO_CLOSE_NON_EXISTENT_SOCKET
- SL_STATUS_SI91X_INVALID_SOCKET_PARAMETERS
- SL_STATUS_SI91X_FEATURE_NOT_AVAILABLE
- SL_STATUS_SI91X_SOCKET_ALREADY_OPEN
- SL_STATUS_SI91X_MAX_SOCKETS_EXCEEDED
- SL_STATUS_SI91X_DATA_LENGTH_EXCEEDS_MSS
- SL_STATUS_SI91X_IP_CONFLICT_ERROR
- SL_STATUS_SI91X_FEATURE_NOT_ENABLED
- SL_STATUS_SI91X_DHCP_SERVER_NOT_SET
- SL_STATUS_SI91X_AP_SET_REGION_PARAM_ERROR
- SL_STATUS_SI91X_SSL_TLS_NOT_SUPPORTED
- SL_STATUS_SI91X_JSON_NOT_SUPPORTED
- SL_STATUS_SI91X_INVALID_OPERATING_MODE
- SL_STATUS_SI91X_INVALID_SOCKET_CONFIG_PARAMS
- SL_STATUS_SI91X_WEBSOCKET_CREATION_TIMEOUT
- SL_STATUS_SI91X_PARAM_MAX_VALUE_EXCEEDED
- SL_STATUS_SI91X_SOCKET_READ_TIMEOUT
- SL_STATUS_SI91X_INVALID_COMMAND_SEQUENCE
- SL_STATUS_SI91X_DNS_RESPONSE_TIMEOUT_ERROR
- SL_STATUS_SI91X_HTTP_SOCKET_CREATION_FAILED
- SL_STATUS_SI91X_TCP_CLOSE_BEFORE_RESPONSE_ERROR
- SL_STATUS_SI91X_WAIT_ON_HOST_FEATURE_NOT_ENABLED
- SL_STATUS_SI91X_STORE_CONFIG_CHECKSUM_INVALID
- SL_STATUS_SI91X_TCP_KEEP_ALIVE_TIMEOUT
- SL_STATUS_SI91X_TCP_ACK_FAILED_FOR_SYN_ACK
- SL_STATUS_SI91X_MEMORY_LIMIT_EXCEEDED
- SL_STATUS_SI91X_MEMORY_LIMIT_EXCEEDED_DURING_AUTO_JOIN
- SL_STATUS_SI91X_PUF_OPERATION_BLOCKED
- SL_STATUS_SI91X_PUF_ACTIVATION_CODE_INVALID
- SL_STATUS_SI91X_PUF_INPUT_PARAMETERS_INVALID
- SL_STATUS_SI91X_PUF_IN_ERROR_STATE
- SL_STATUS_SI91X_PUF_OPERATION_NOT_ALLOWED
- SL_STATUS_SI91X_PUF_OPERATION_FAILED
- SL_STATUS_SI91X_AUTO_JOIN_IN_PROGRESS
- SL_STATUS_SI91X_RSNIE_FROM_AP_INVALID
- SL_STATUS_SI91X_SNTP_MAX_ATTEMPTS_REACHED
- SL_STATUS_SI91X_FREQUENCY_OFFSET_ZER0
- SL_STATUS_SI91X_FREQUENCY_OFFSET_OUT_OF_LIMITS

Examples

Resources

BRD4338A Radio Board User Guide

SiWx917 Hardware Reference

Developing with WiSeConnect 3 SDK

Developing with the WiSeConnect™ v3.x SDK

This documentation provides a development guide and reference for the SiWx91x™ chipset family using version 3.x (3.0.0 and later) of the WiSeConnect™ software development kit (SDK), a simple-to-use application programming interface (API) for Internet Protocol (IP) networking and connectivity.

Version 3.x of the WiSeConnect SDK (also known as Simple API or SAPI) is the next-generation Wi-Fi SDK for Silicon Labs customers that replaces WiSeConnect SDK v2.x (before 3.0.0) with a modular design and an organization of features into configurable components.

SiWx917 is a Wi-Fi system-on-chip (SoC) consisting of:

- an application processor for running applications
- a network processor running the full Wi-Fi and Bluetooth Low Energy (BLE) networking stacks, along with certain protocols like Message Queue Telemetry Transport (MQTT) and HyperText Transfer Protocol (HTTP).

To get started using SiWx91x, refer to the Getting Started guides in this documentation.

To learn more about the WiSeConnect SDK v3.x, refer to the SDK Changes between this version and version 2.x.

To learn more about the SiWx91x firmware, see the SiWx917 Software Reference.

To explore our available application examples, see the Examples section.

Overview

Getting Started with the WiSeConnect™ SDK v3.x

To get started with using the WiSeConnect™ SDK v3.x, download the latest version of Simplicity Studio and select one of the getting started guides in this section:

- **System-on-chip (SoC) mode** - both the application and connectivity stack run on the SiWx91x™ chipset. See the Starting with SoC Mode section.
- **Network co-processor (NCP) mode** - application runs on external microcontroller unit (MCU) host, connectivity stack runs on the SiWx91x chipset. See the Starting with EFR32™ in NCP Mode section.

> **Note**: If you are using an external MCU host other than EFR32, stay tuned. We will publish more guides soon.

Starting with SoC Mode

Getting Started with WiSeConnect™ SDK v3.x in SoC Mode

This guide describes how to get started with running the out-of-box demo for the SiWx91x™ chipset family using the WiSeConnect™ SDK v3.x in System-on-chip (SoC) mode, where both the application and the connectivity stack run on the SiWx91x chipset.

To get started with compiling and running an application, see the Developing for SiWx91x Host page.

Check Prerequisites

Hardware

- Wi-Fi Access Point (802.11 ax/b/g/n)
- **BRD4002A** - Wireless Pro Kit Mainboard (hereafter referred to as **WPK board**).
- One of the following SiWx917 SoC Radio Boards (hereafter referred to as **SiWx917 radio board**):
  - BRD4338A - SiWx917 Wi-Fi 6 and Bluetooth LE 8MB Flash Radio Board
  - **BRD4342A** - SiWx917 Wi-Fi 6 and Bluetooth LE 8MB Flash + 8MB ext PSRAM Radio Board
- USB power source (for e.g., a computer)
- Type C USB cable compatible with the USB power source (for e.g., type C to type A in case of a computer with a type A USB port).
- Android smartphone or iPhone

> **Note**: The following SiWx917 pro kits come with a WPK board and SiWx917 radio board:
>
> - SiWx917-PK6031A (WPK board + BRD4338A) - SiWx917 Wi-Fi 6 and Bluetooth LE 8MB Flash Pro-Kit
> - SiWx917-PK6032A (WPK board + **BRD4342A**) - SiWx917 Wi-Fi 6 and Bluetooth LE 8MB Flash + ext 8MB PSRAM Pro-Kit

Software

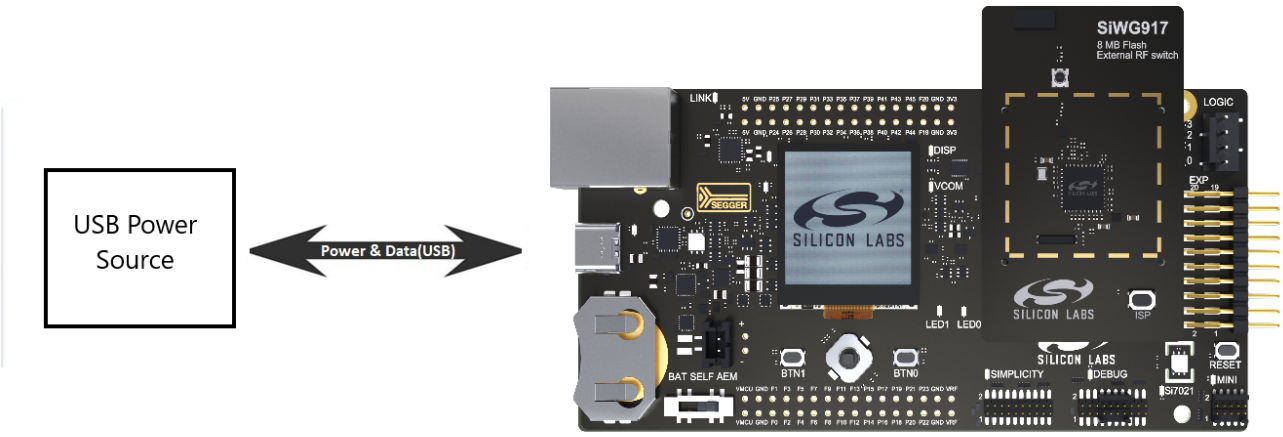
- Silicon Labs EFR Connect App for Android or iPhone

Run the Demo

The out-of-box demo is pre-flashed on your SiWx917 radio board.

In this demo, you use the the Silicon Labs EFR Connect App to connect to your SiWx91x device over Bluetooth Low Energy (BLE), commission your SiWx91x device to a Wi-Fi network, and exchange data with your SiWx91x device.

1. Plug the SiWx917 radio board into the radio board connectors of the WPK board as shown below.
2. Power on the WPK board by connecting it to a USB power source such as a computer.



> **Note**: Image is for illustration only. Details may not match what you see.

> **Note**: If you have an older SiWx917 SoC radio board, you may not have the out-of-box demo pre-flashed on your board. To verify this, check for messages on the board's liquid crystal display (LCD) screen when you plug the radio board into a power source.
>
> > Wireless interface initialized successfully
> >
> > BLE advertising started
> >
> > Device name: BLE_CONFIGURATOR
>
> If the expected messages don't appear, flash the latest software to your board before running the demo:
>
> - Download the out-of-box demo binary from the **WiseConnect SDK v3.x** github repo.
> - Follow the instructions on the Developing for SiWx91x Host page to:
>   - Install Simplicity Studio
>   - Connect the SiWx917 board to your computer
>   - Upgrade the SiWx917 connectivity firmware
>   - Flash the out-of-box demo binary as described in the Flash an Application Binary section.

3. Run the demo by following the instructions in the **README** page of the **out-of-box-demo** example in the **WiSeConnect SDK v3.x** github repo starting from the Run the Application section.

Starting with EFR32 in NCP Mode

Getting Started with WiSeConnect™ SDK v3.x and EFR32™ Host in NCP Mode

This guide describes how to get started with developing an application for the SiWx91x™ chipset family using the WiSeConnect™ SDK v3.x with an EFR32™ host in Network Co-Processor (NCP) mode, where the application runs on the EFR32 host and the connectivity stack runs on the SiWx91x chipset.

> **Note**: The output images in this guide are for illustration purposes only. Details such as board names and version numbers may not exactly match the product.

Check Prerequisites

Software

- Simplicity Studio
- Gecko software development kit (GSDK)
- Tera Term version 4.106 or later

> **Note**:
>
> - We recommend using the latest GSDK version.
> - Refer to the Release Notes for the GSDK version tested with this release.

Hardware

- Wi-Fi Access Point (802.11 ax/b/g/n)
- **BRD4002A** - Wireless Pro Kit Mainboard (hereafter referred to as **WPK board**).
- EFR32xG24 - Wireless 2.4 GHz +10 dBm Radio Board (hereafter referred to as **EFR32 radio board**).
- **BRD4346A** - SiWx917 Wi-Fi 6 and Bluetooth LE 4MB Flash Co-Processor Radio Board (hereafter referred to as **SiWx917 radio board**).
- **BRD8045A** - EXP Adapter Board for SiWx917 Co-Processors (hereafter referred to as **adapter board**).
- Windows/Linux/MacOS computer with a USB port
- Type C USB cable compatible with the computer's USB port (for e.g., type C to type A if the computer has a type A USB port).

> **Note**: The **SiWx917 Wi-Fi 6 and Bluetooth LE Co-Processor EXP Expansion Kit** (SiWx917-EB4346A) comes with the SiWx917 radio board and adapter board mentioned above.

Setup Software

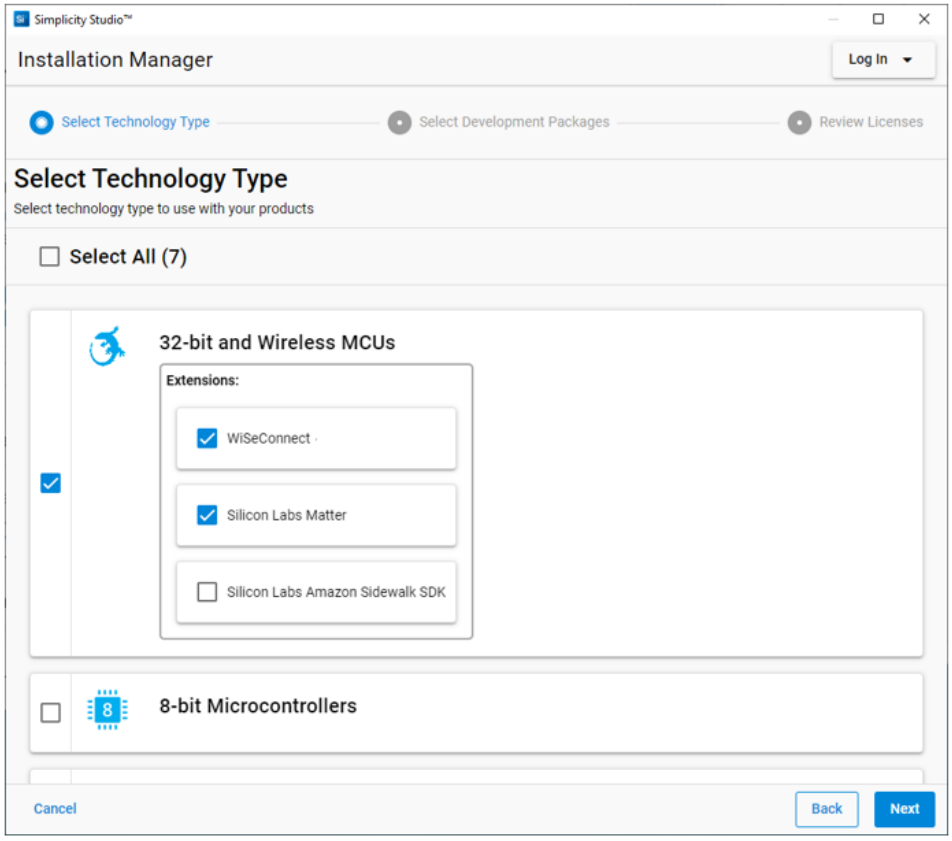
You may setup the following software in this section while waiting to receive the hardware:

- Simplicity Studio
- WiSeConnect 3 Extension

Install Simplicity Studio

Download the latest version of Simplicity Studio and follow the installation instructions. During the installation:

- make sure you log in to Simplicity Studio in the **Installation Manager** window,
- select **Install by technology type**, and
- select the **WiSeConnect** extension under **32-bit and Wireless MCUs**.



Install the WiSeConnect 3 Extension

If you already selected the **WiSeConnect** extension in the Install Simplicity Studio section, you may skip this section.

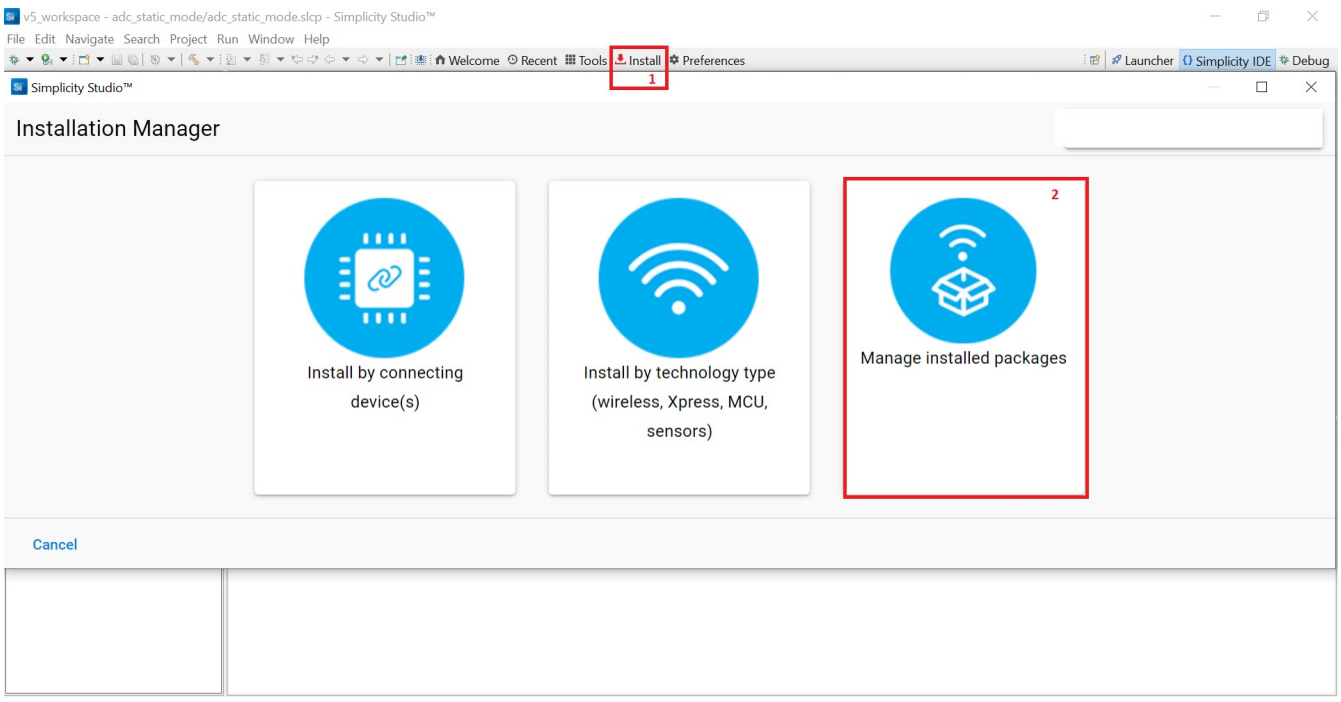
Before installing the **WiSeConnect 3** extension, upgrade to a compatible GSDK version if not already done. See the Prerequisites section for the supported GSDK versions.

You may install **WiSeConnect 3** through one of the following alternative paths:

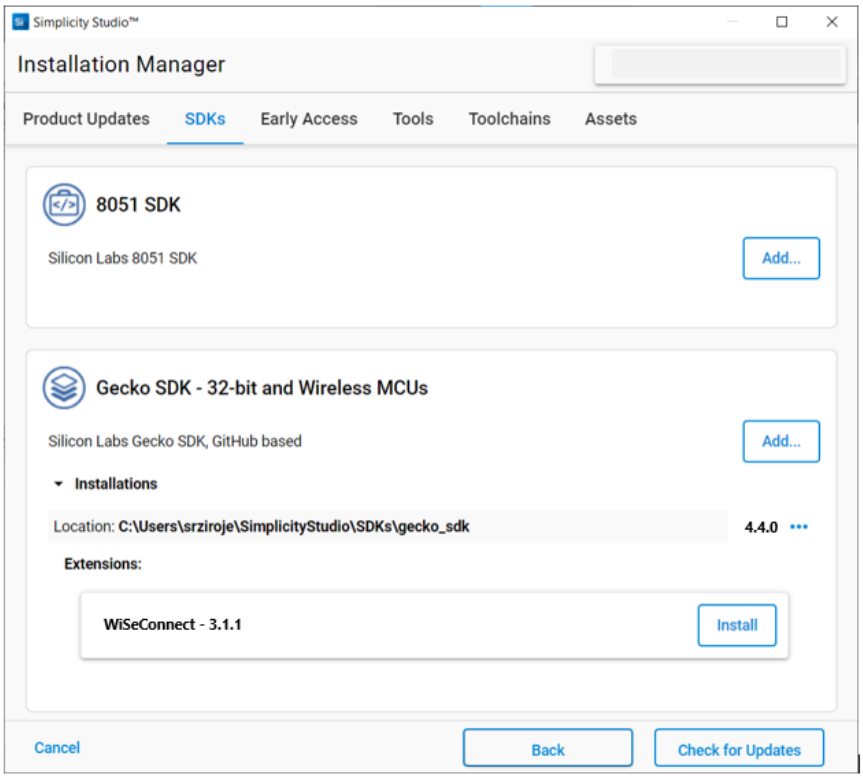
- Installation Manager (**recommended**)
- Manage SDKs Window (**hardware required**)

Install WiSeConnect 3 through the Installation Manager

1. Log in to Simplicity Studio if not already done.
2. In the Simplicity Studio home page, select **Install > Manage installed packages**.



3. Select the **SDKs** tab.
4. Next to the **WiSeConnect - 3.x.x** extension, click **Install**.



Install WiSeConnect 3 through the Manage SDKs Window

> **Note**: You must have the hardware available before using these steps to install the **WiSeConnect 3** extension.

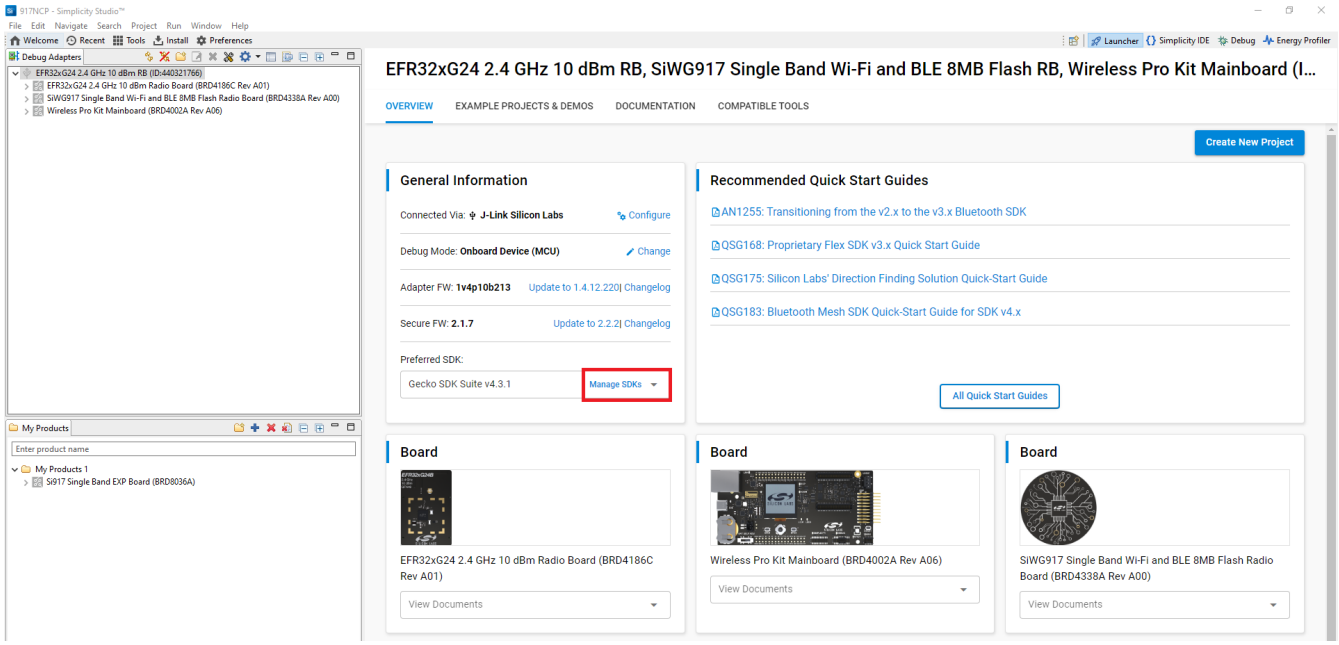
1. Download the WiSeConnect v3.x source code from the following URL after substituting **3.x.x** with the desired release version:

   ```
   https://github.com/SiliconLabs/wiseconnect/archive/refs/tags/v3.x.x.zip
   ```

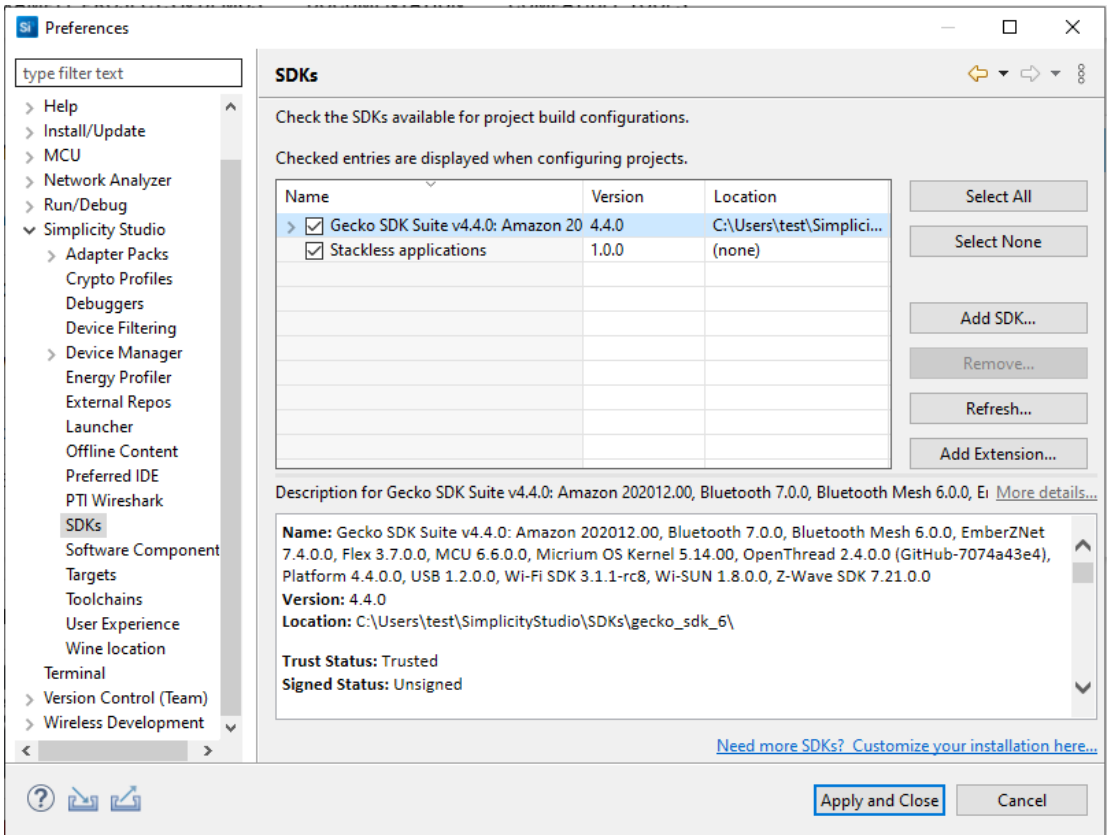
   - If you don't know your release version, go to the github releases page and select the version to download.
2. Unzip the downloaded `wiseconnect-3.x.x.zip` file. It will be extracted into a folder structure similar to the following:

   ```
   wiseconnect-3.x.x
   |--- wiseconnect-3.x.x
   |------ <source code>
   ```

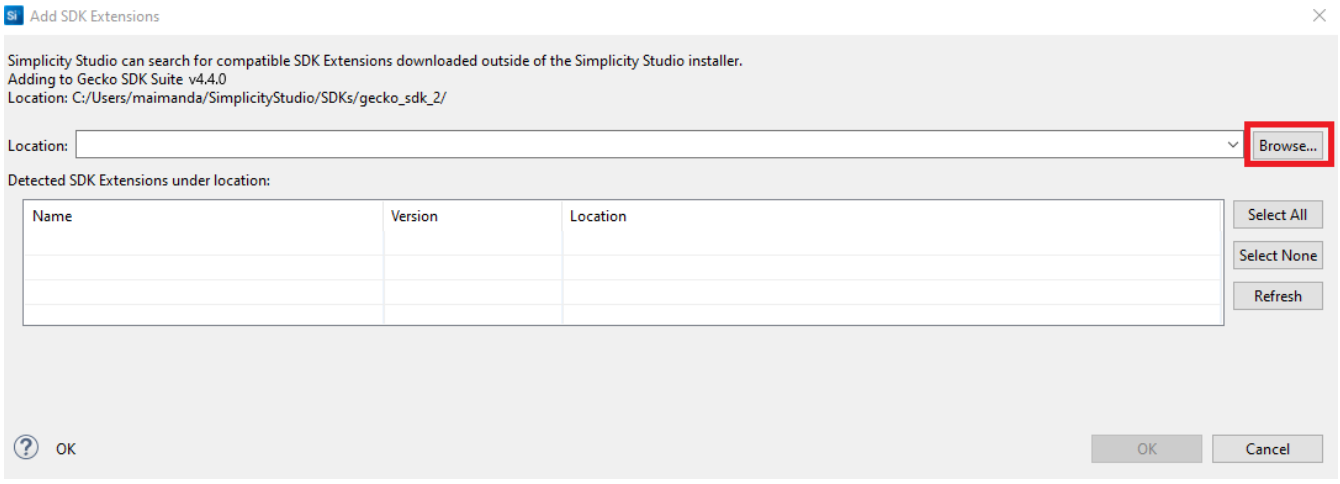
3. Launch **Simplicity Studio** and log in.
4. Connect the EFR32 and SiWx917 boards to your computer
5. In the **Debug Adapters** pane, select your radio board.
6. In the **General Information** section, click **Manage SDKs**.



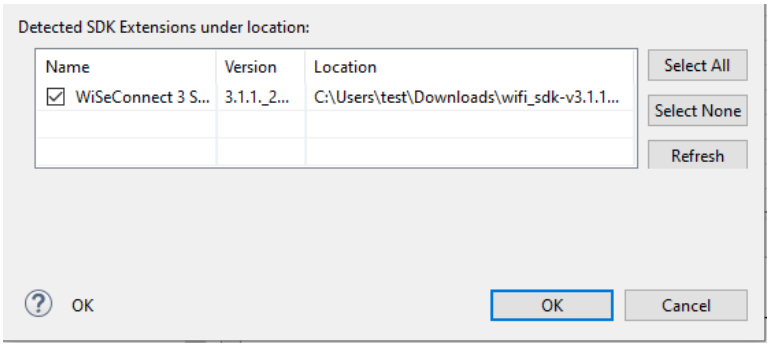
7. The **Preferences** window will be opened in the **SDKs** section.
8. Select **Gecko SDK Suite vx.x.x** and click **Add Extension**.



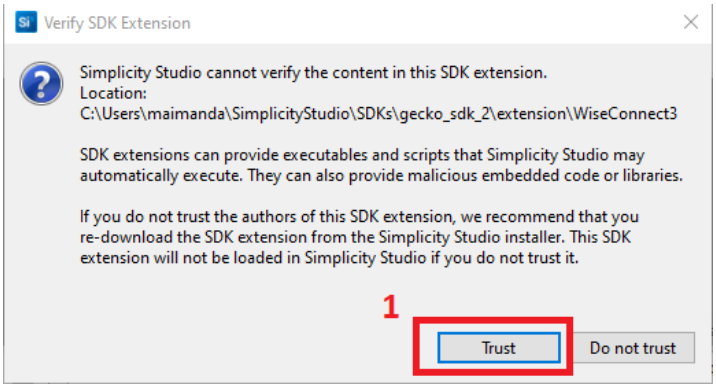
9. In the **Add SDK Extensions** window, click **Browse**.



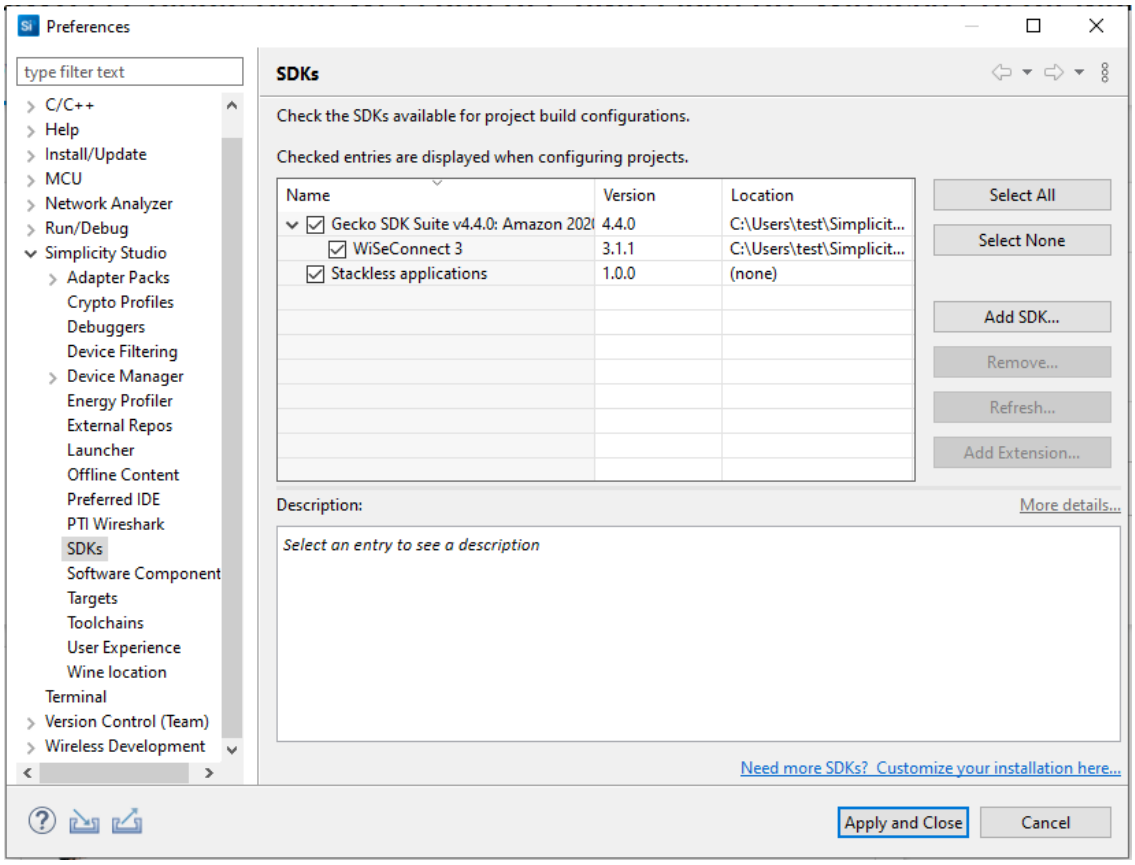
10. Locate and select the `wiseconnect-3.x.x` sub-folder extracted in step 2 above which contains the source code.
11. Studio will detect the **WiSeConnect 3 SDK** extension.
12. Select the detected extension and click **OK**.



13. If a **Verify SDK Extension** popup is displayed, click **Trust**.

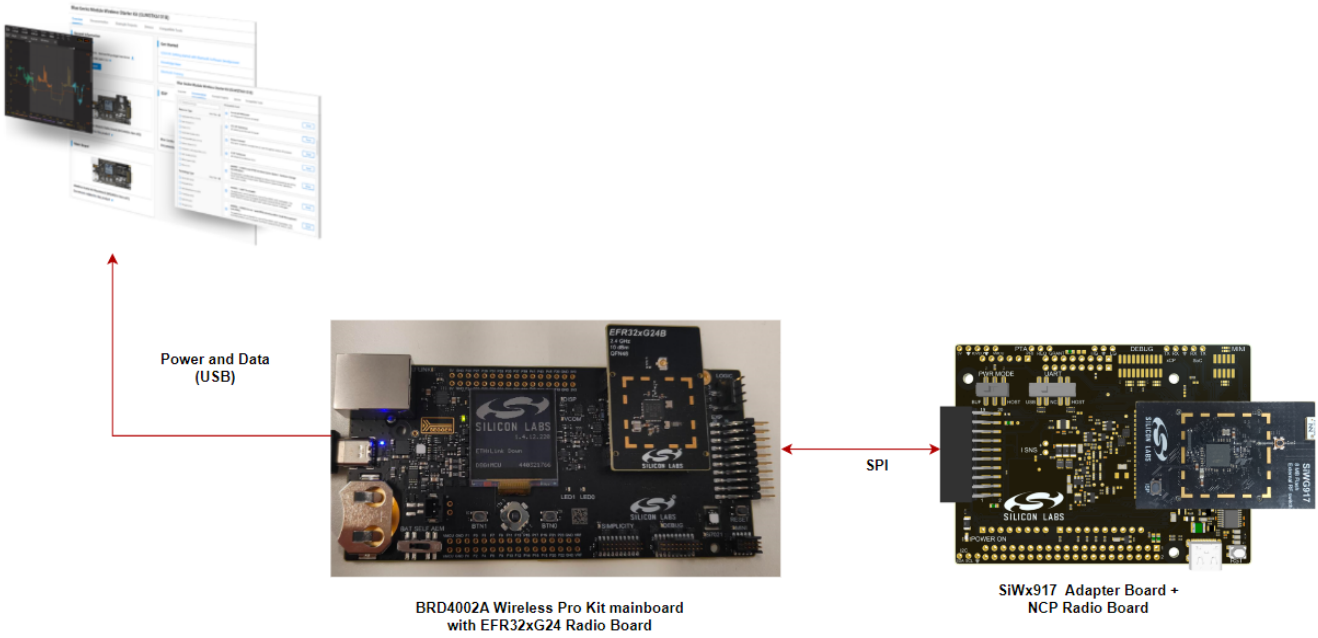


14. The selected **WiSeConnect 3** extension will be displayed.
15. Click **Apply and Close**.



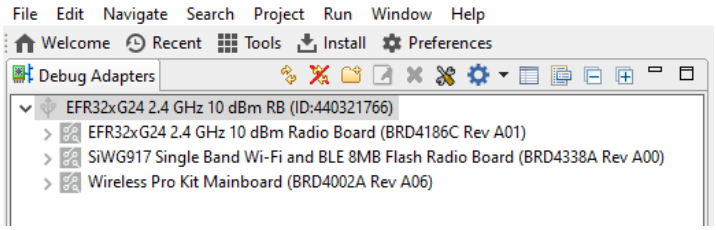
Connect the Boards to a Computer

1. Plug the EFR32 radio board into the radio board connectors of the WPK board as shown below.
2. Plug the SiWx917 radio board into the radio board connectors of the adapter board as shown below.
3. Connect the adapter board to the EXP header on the EFR32 WPK board.
4. Make sure the **UART** switch on the adapter board is in the **USB** position.
5. Make sure the **PWR MODE** switch on the adapter board is in the **BUF** position.
6. Connect the EFR32 WPK board to your computer using a type C USB cable.



> **Note**: The SiWx917 adapter + radio board image above is for illustration only and the radio board number **SiWG917** in the image is incorrect. Please use radio board number **SiWN917** instead.

7. Simplicity Studio will detect and display your EFR32 radio board.



Troubleshoot Board Detection Failure

If Simplicity Studio does not detect your EFR32 radio board, try the following:

- In the **Debug Adapters** pane, Click the **Refresh** button (having an icon of two looping arrows).
- Reset both the WPK board and adapter board by pressing the **RESET** button and **RST** button on the corresponding boards.
- Power-cycle the EFR32 radio board by disconnecting and reconnecting the USB cable.

Upgrade the SiWx91x Connectivity Firmware

We recommend that you upgrade the SiWx917 connectivity firmware to the latest available version when:

- you first receive an SiWx917 EXP expansion kit.
- you first receive an SiWx917 co-processor radio board.
- you upgrade to a new version of the **WiSeConnect 3** extension

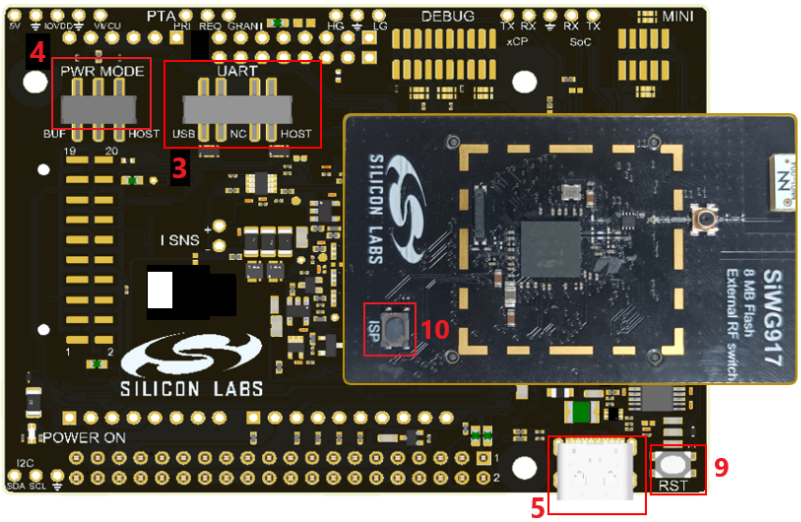
1. Disconnect the adapter board from the EFR32 WPK board, so that the host MCU connections do not interfere with the firmware upgrade process.

2. Plug the SiWx917 radio board into the radio board connectors of the adapter board as shown below.
3. Make sure the **UART** switch on the adapter board is in the **USB** position.
4. Make sure the **PWR MODE** switch on the adapter board is in the **BUF** position.
5. Connect the **USB** port of the adapter board to your computer's USB port using a type C USB cable.
6. Open the `teraterm.ini` file present in the Tera Term installation path (for example, `C:\Program Files (x86)\teraterm`).
7. Find the `KmtLongPacket` setting and update it to `on` if not already set.

```
KmtLongPacket=on
```

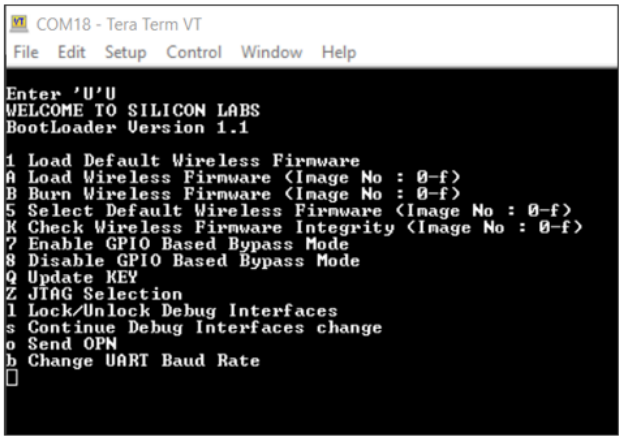
> **Note**: Enabling the **Transmit/Receive Extended-length Packets** feature (`KmtLongPacket`) reduces the firmware update time from approximately 7 minutes 15 seconds to approximately 1 minute 35 seconds along with the baud rate settings described in the following steps.

8. Set up Tera Term. The instructions for setting this up are the same as those for RS9116.
9. Hold down the **RST** button on the adapter board.
10. Hold down the **ISP** button on the SiWx917 radio board.
11. Release the **RST** button followed by the **ISP** button.

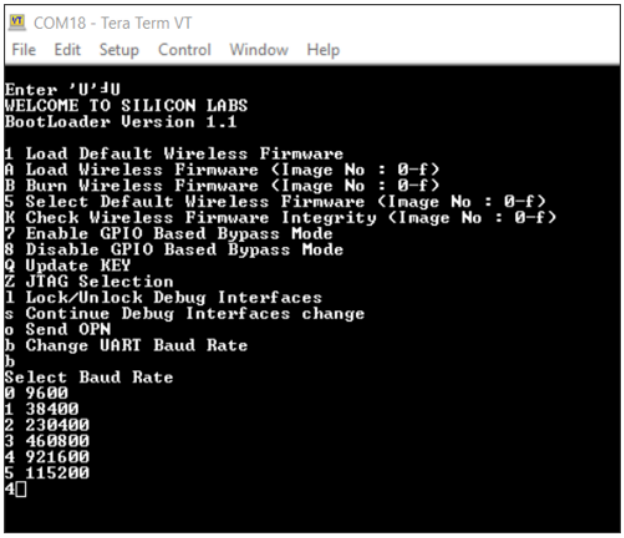


> **Note**: The image is for illustration only and the radio board number **SiWG917** in the image is incorrect. Please use radio board number **SiWN917** instead.

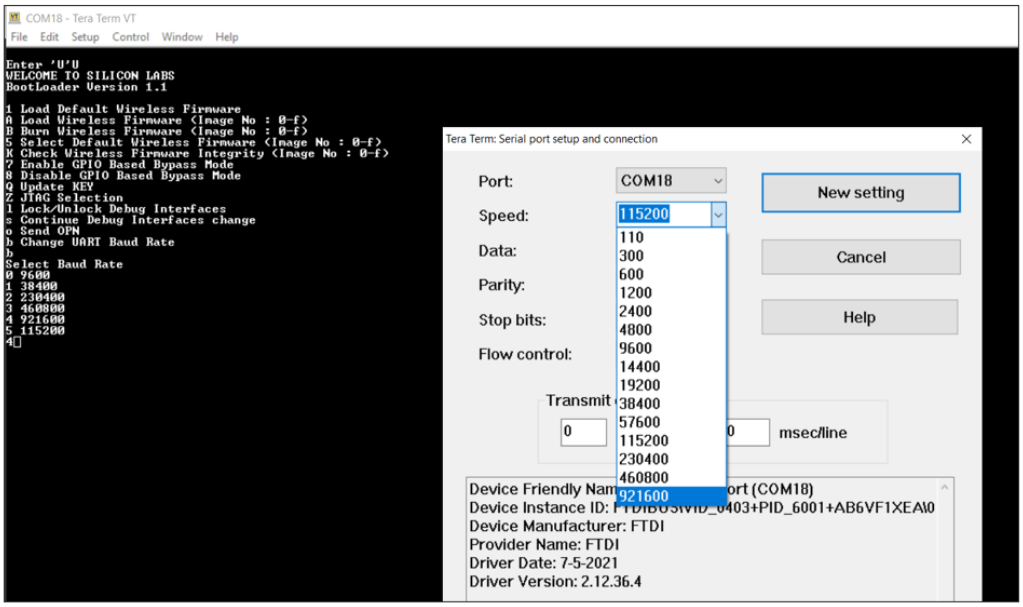
12. Enter the characters **|U** while holding down the **Shift** button.
13. The bootloader menu is displayed.



14. Press the **b** key to change the baud rate of the SiWx917 UART interface.
15. A menu of available baud rates is displayed.
16. Press **4** to change the baud rate to **921600**.



17. Select "**Setup > Serial port...**" from the Tera Term menu.
18. Change the **Speed** to **921600**.
19. Click **New setting**.

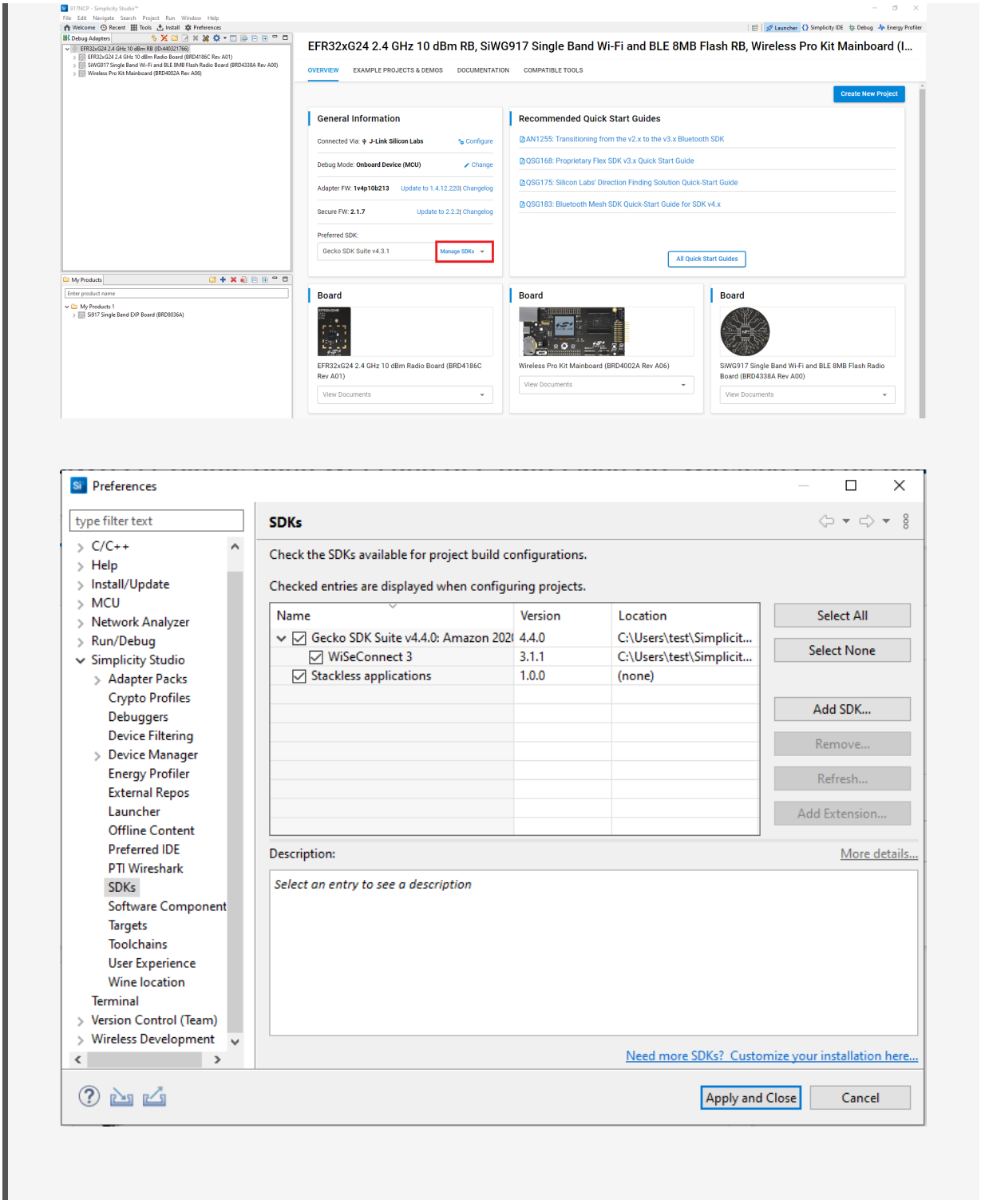


20. The dialog will close and you will be returned to the terminal screen.
21. Press **U**.
22. The following message will be displayed: "Baud Rate was updated successfully!"

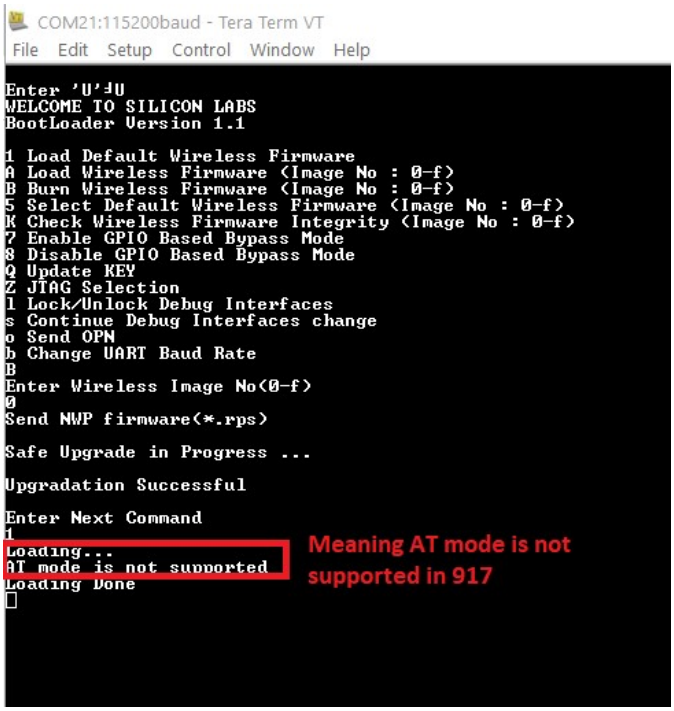
> **Note:** With the increased baud rate, the firmware will be flashed and loaded in approximately 1 minute 35 seconds, with a firmware file of approximately 1.6 MB.

23. From here the steps are *almost* the same as the RS9116 firmware update instructions from **STEP 2** onwards, except for the steps noted below:
    - In **STEP 4** of the RS9116 instructions, locate and select the firmware file to flash from within the `connectivity_firmware` sub-folder of the **WiSeConnect 3** extension path.

> Note:
>
> - If you don't see any files in the sub-folder, select **All files** in the file filter field in the Browse dialog.
> - The **WiSeConnect 3** extension path is the one where the extension was downloaded during installation. If you're not sure what the path is, you may refer to the location in the **Preferences > SDKs** page shown on clicking **Manage SDKs**.

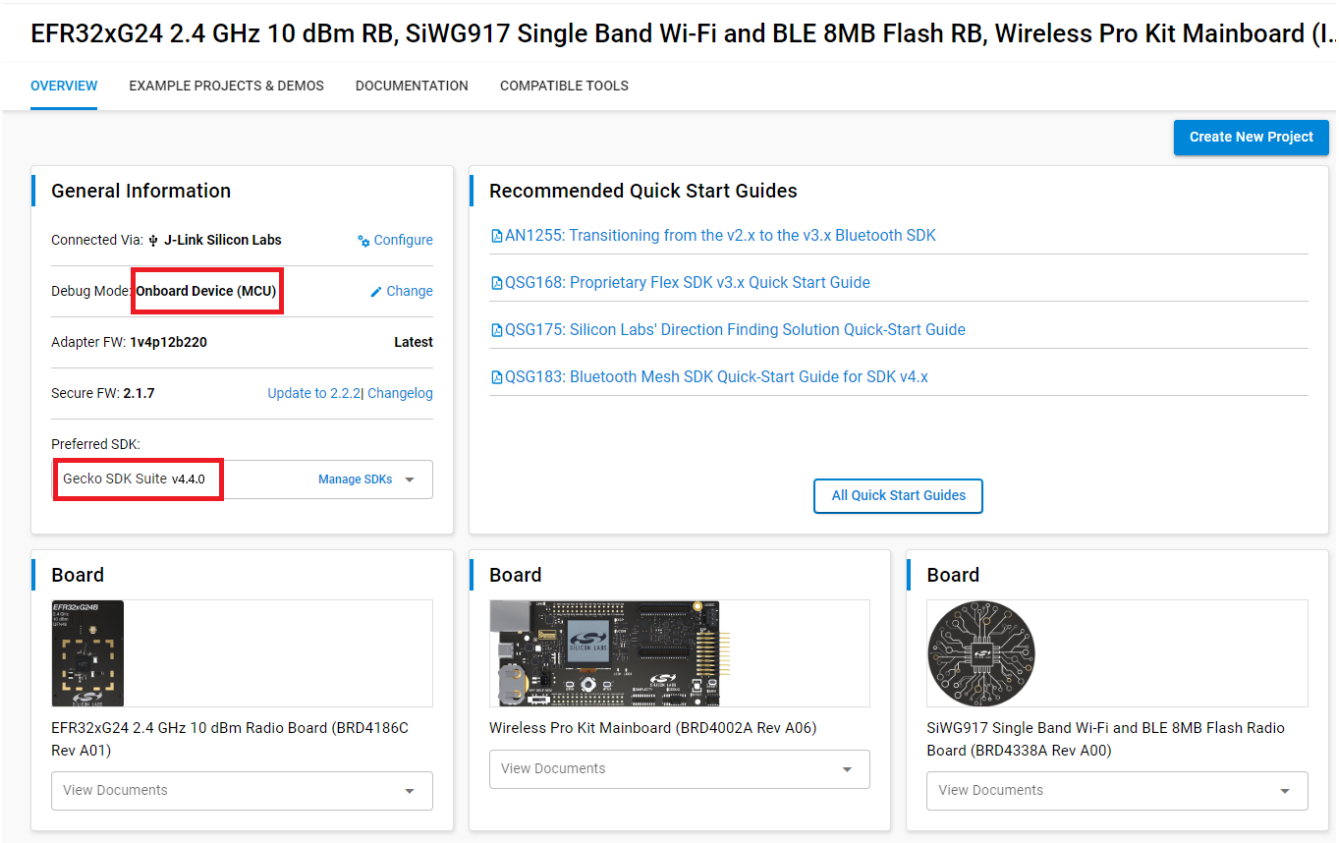


- When the upgrade is complete, the following messages will be seen.



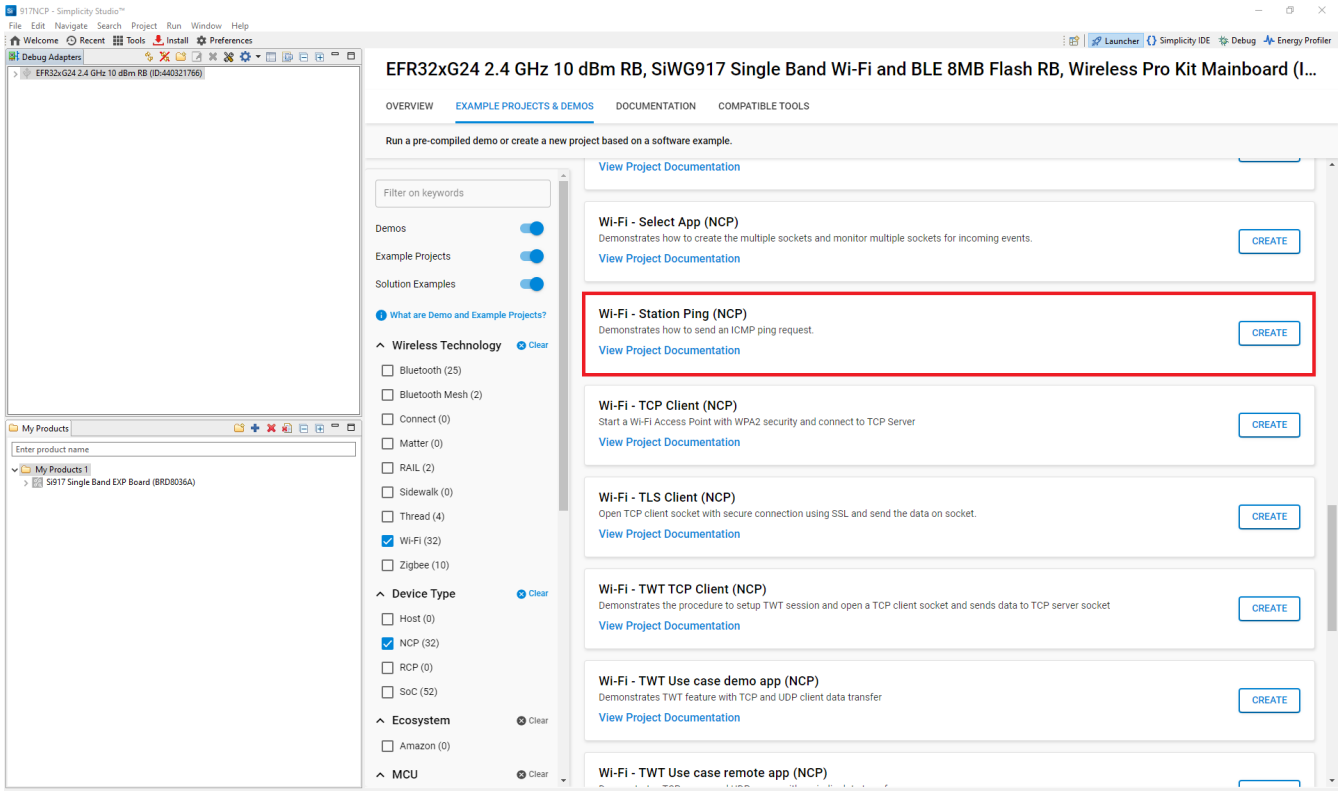
Create a Project

1. Log in to Simplicity Studio and connect the EFR32 and SiWx917 boards to your computer.
2. Go to the **Debug Adapters** section.
3. Select your radio board from the displayed list.
4. The **Launcher** page will display the selected radio board's details.
5. Select the **OVERVIEW** tab.
6. Verify the following in the **General Information** section:
    - The **Debug Mode** is **Onboard Device (MCU)**.
    - The **Preferred SDK** is the version you selected earlier.



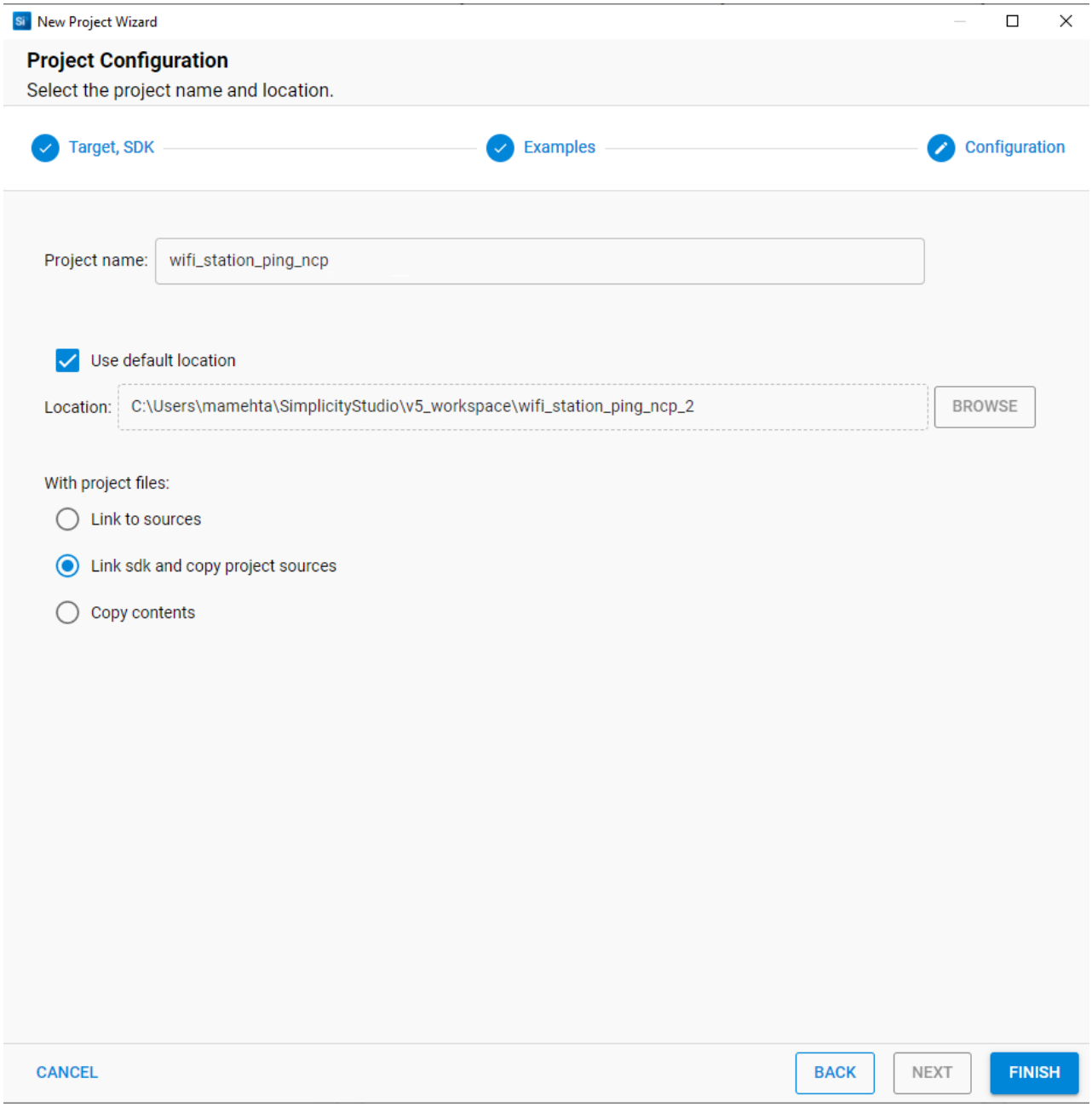
> **Note**: The image above is for illustration only and the Gecko SDK version shown is outdated. See the Software section for the currently supported GSDK versions.

7. Select the **EXAMPLE PROJECTS AND DEMOS** tab.
8. Locate the example you want and click **CREATE**.

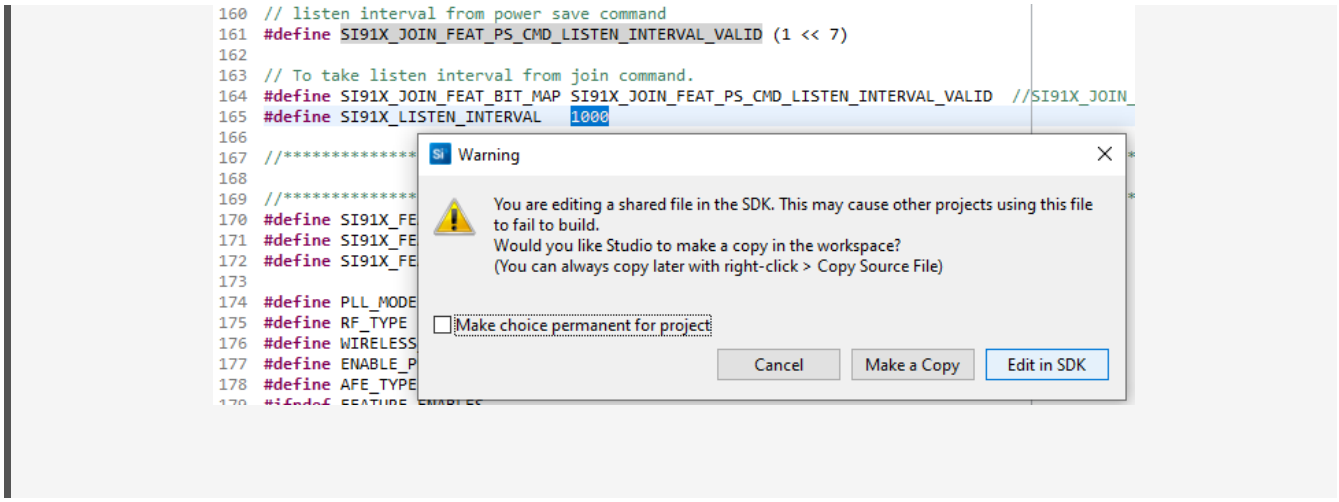


9. In the **New Project Wizard** window, click **FINISH**.

> **Note**: The **Copy contents** option is not currently supported.



> **Note**: The **Make a Copy** option is not currently supported while editing a shared SDK file, when prompted to either make a copy of the file or edit it in the SDK itself.



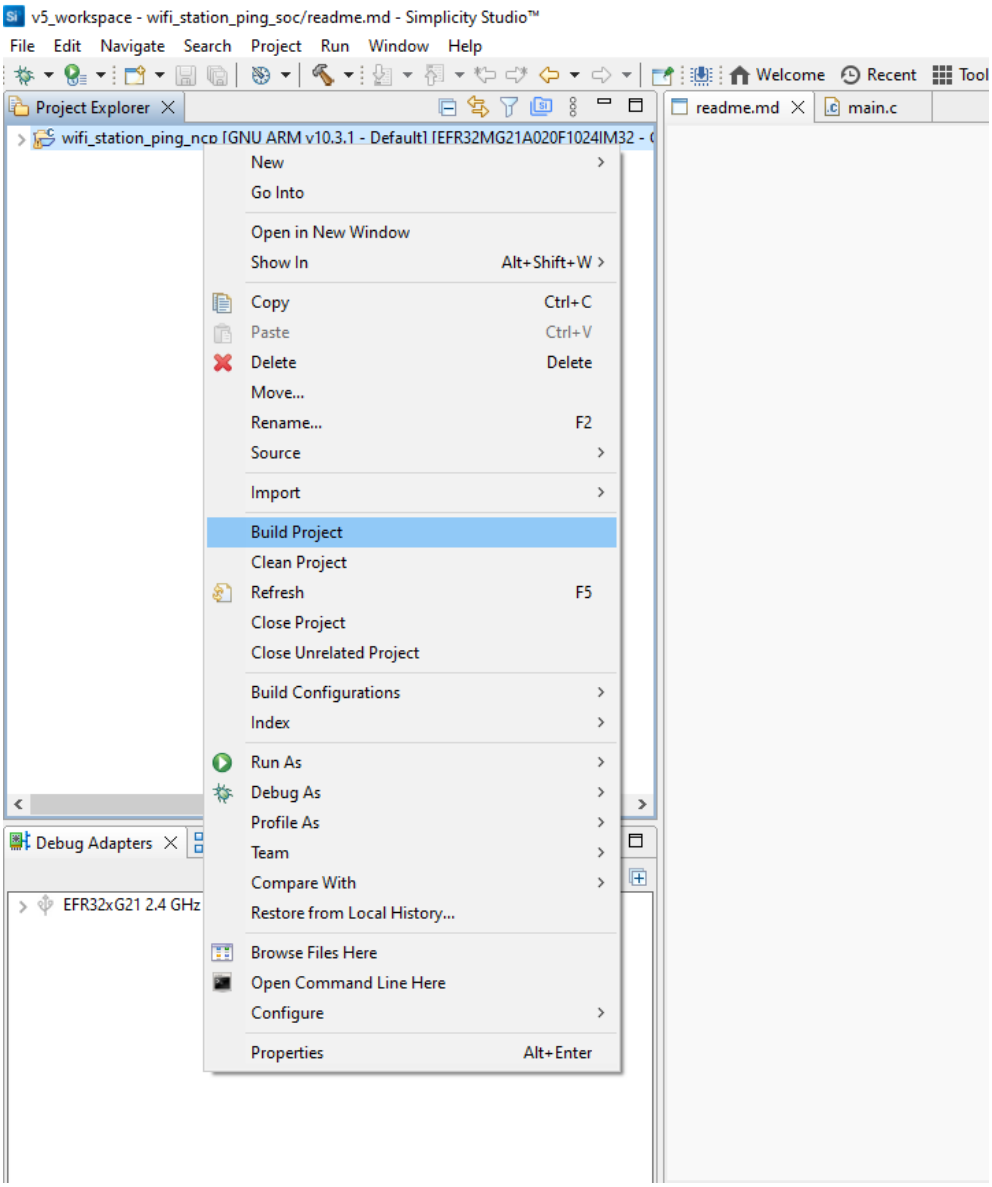
Configure an Application

Configure the settings for your example. For **Wi-Fi - STATION PING (NCP)** (the recommended example for this guide), see the Application Build Environment section in the **README** page for configuration instructions.

You may use the Component Editor to configure the components in your example.

Build an Application

1. Launch Simplicity Studio and log in.
2. In the **Project Explorer** pane, right-click the project name and select **Build Project**.
    - You may also click the **Build** button with a hammer icon on the Simplicity IDE perspective toolbar.



Flash an Application

> **Note**: Make sure the bootloader image is flashed separately on to the EFR32 host MCU. Refer to the Gecko SDK Platform Bootloading Reference for more information.

There are two alternative methods to flash an application to the application processor of the SiWx91x device:

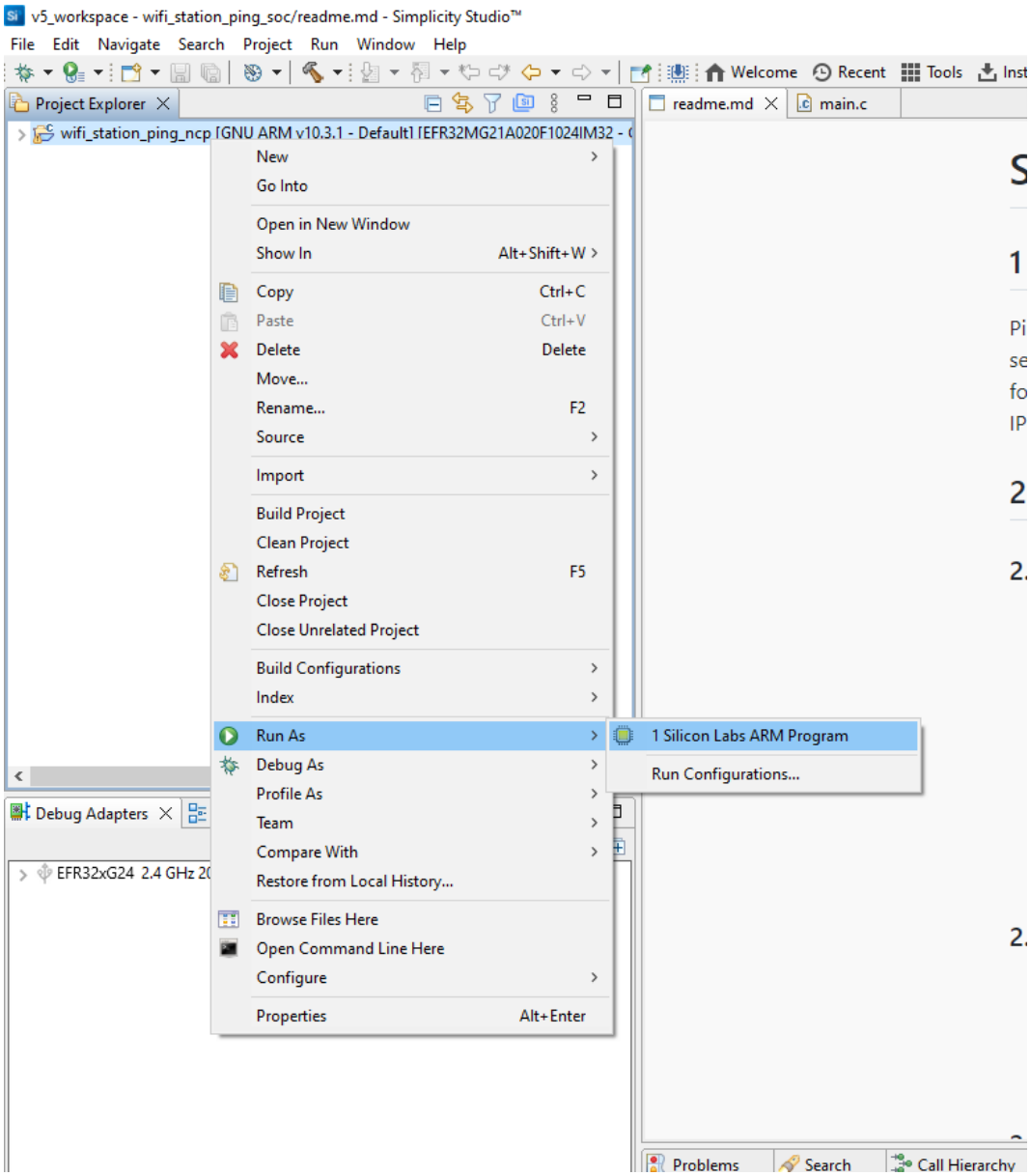
- Flash an Application Built using Studio
- Flash a Binary that is Already Available

Once you flash the **Wi-Fi - STATION PING (NCP)** example (the recommended example for this guide), you may refer to the Test the Application section of its **README** page to explore its output. The other sections of the **README** like the Purpose/Scope section and **Overview** section (not present in some **README**'s) provide more information about the example.

See the Examples page to explore all available examples and view their **README** pages.

Flash an Application Built Using Simplicity Studio

1. Build the application as described in the Build an Application section.
2. In the **Project Explorer** pane, right-click on your project name and select **Run As > 1 Silicon Labs ARM Program**.

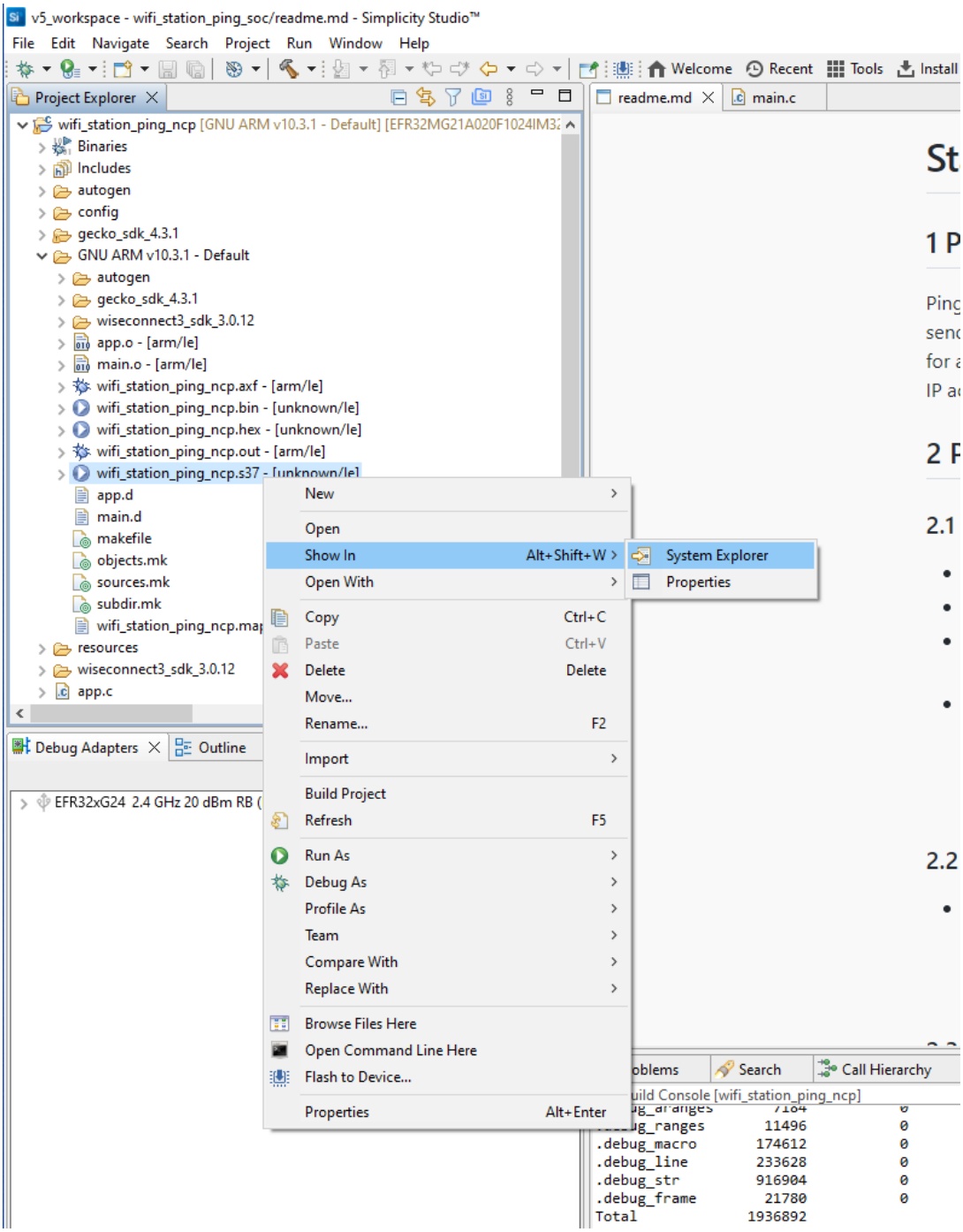


3. The application binary will be flashed on the radio board and the application will start running.
4. View the standard output or enter input data as needed. See the Console Input and Output section.

> **Note**: See the troubleshooting section in case the application fails to flash.

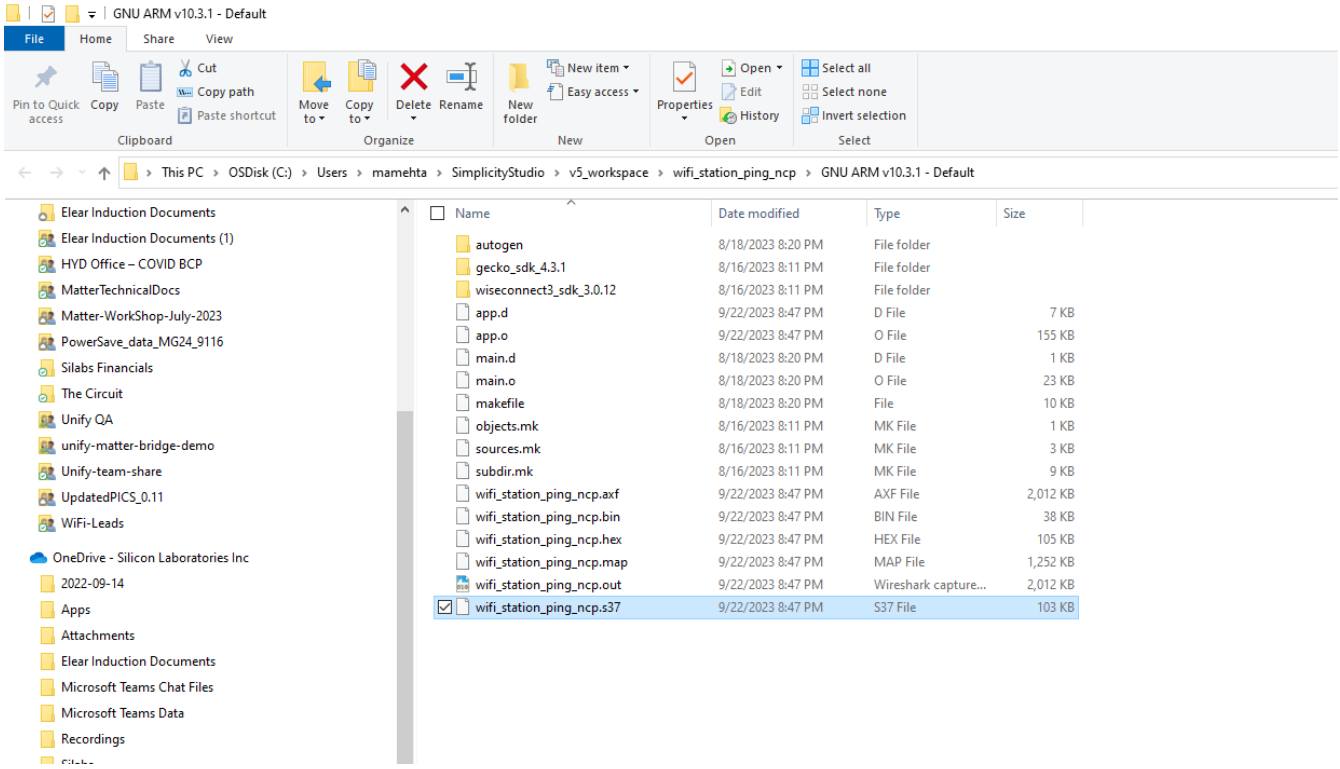
Flash the Application Binary

1. If the binary was built as described in the Build an Application section, a file with a `.s37` extension is generated. This file will be available under **GNU ARM vXX.x.x - Default** in the users workspace. To see its location in your computer's file system, right-click on the `.s37` file and select **Show In > System Explorer**.
   - Alternatively, you may have obtained a `.s37` file through another means such as someone else building and providing the binary to you.



2. The instructions to flash the application binary are *almost* the same as those for flashing an SiWx91x firmware file, except for the point noted below:
   - Instead of choosing the SiWx91x firmware file, select the `.s37` you obtained in step 1 above.

> **Note**: If you don't see `.s37` files in the sub-folder, select **All files** in the file filter field in the Browse dialog.



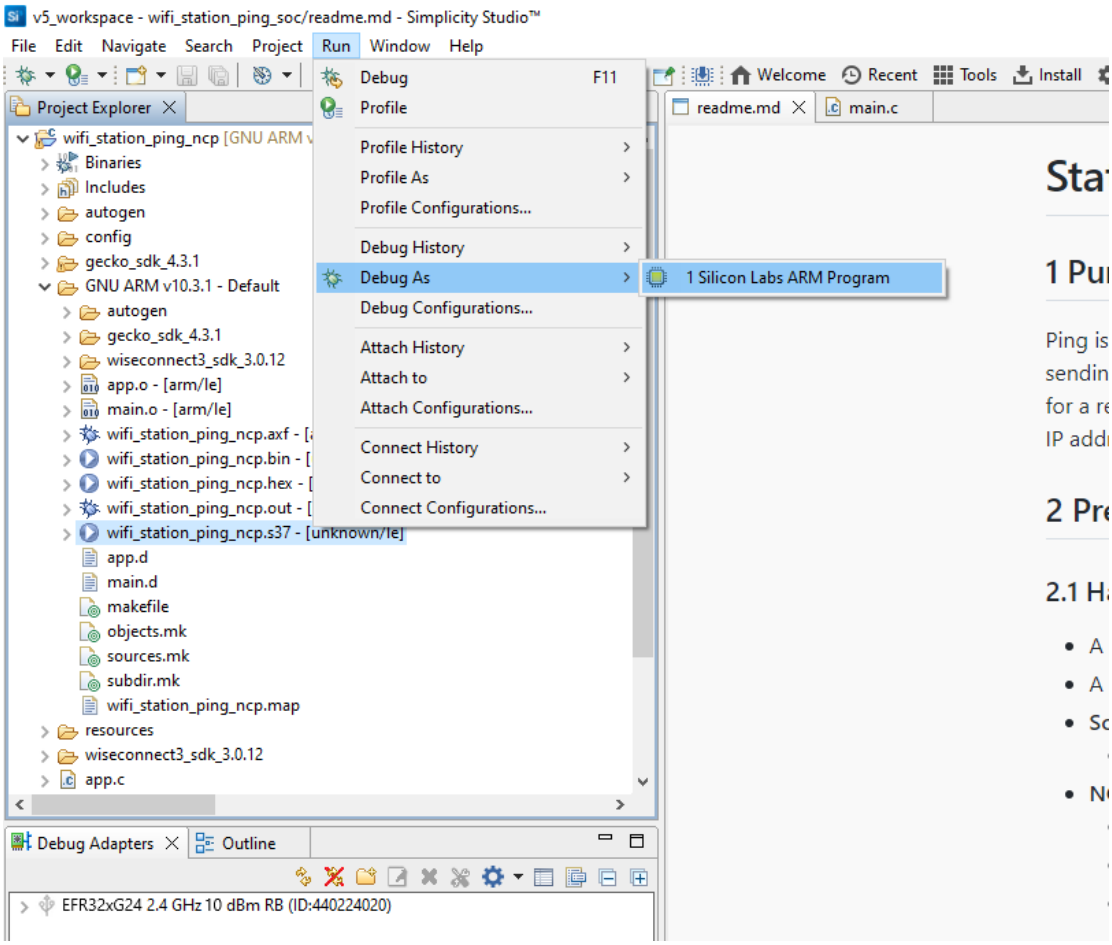
Troubleshoot an Application Flash Failure

The application may fail to flash for one of the following reasons:

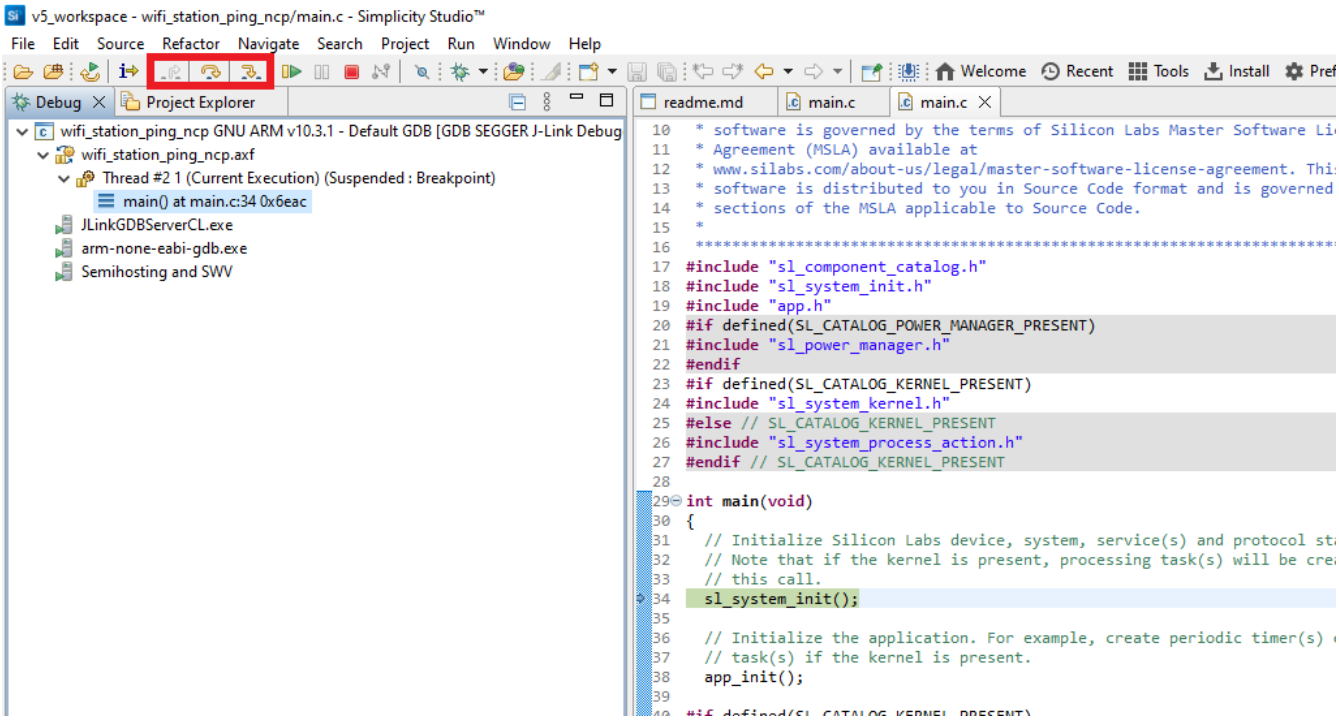
- Error code **102** is displayed in the logs, indicating that **ISP** mode is enabled. Try the following steps:
  - Press and hold the **ISP** button on the radio board.
  - Press and release the **RESET** button on the WPK board.
  - Release the **ISP** button on the radio board.
  - Retry flashing the application.
- "Could not connect debugger. Could not connect to target device" is displayed in the logs, indicating that the application processor is in a low power state with no flash access. Try the same steps as those described above for error **102**.
- Unknown reason - try re-launching Simplicity Studio.
- Studio failed to detect your board. See the Troubleshoot Board Detection Failure section.

Debug an Application

1. In the **Project Explorer** pane, select your project name.
2. From the menu, select **Run > Debug As > 1 Silicon Labs ARM Program**.



3. Studio will switch to debug mode and halt execution at the `main()` function in your application.
4. Add a break point in the desired location of the code and click the **Resume** button (having an icon with a rectangular bar and play button).
5. Execution will halt at the break point.
6. Use the following debug functions to direct the execution of the code:
    - **Step In** button (having an icon with a arrow pointing between two dots).
    - **Step Over** button (having an icon with an arrow going over a dot).
    - **Step Out** button (having an icon with an arrow pointing out from between two dots).

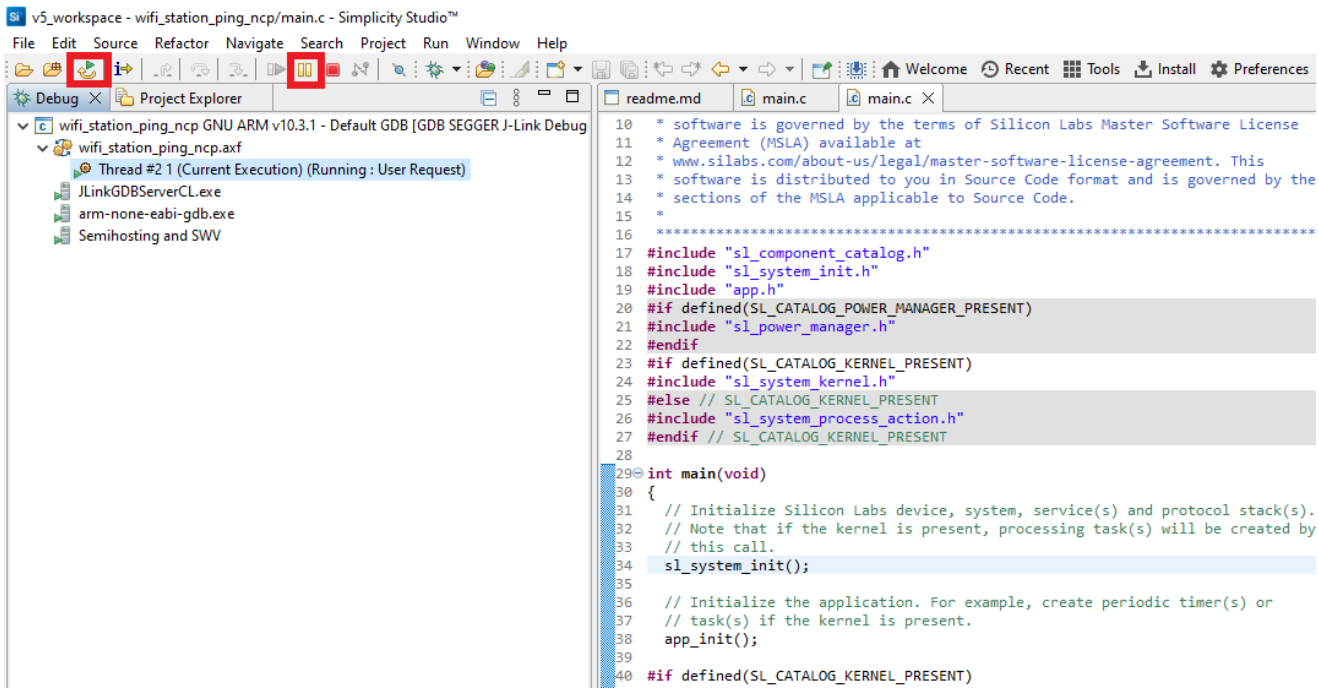


7. View the standard output or enter input data as needed. See the Console Input and Output section.

Troubleshoot an Application Debug Failure

The application may fail to enter the debug mode due to one of the following reasons:

- Studio failed to halt execution at the `main()` function. Try the following steps:
  - Click the **Suspend** button (having a pause button icon).
  - Click the **Reset** button (having an icon of a play button with an arrow underneath).

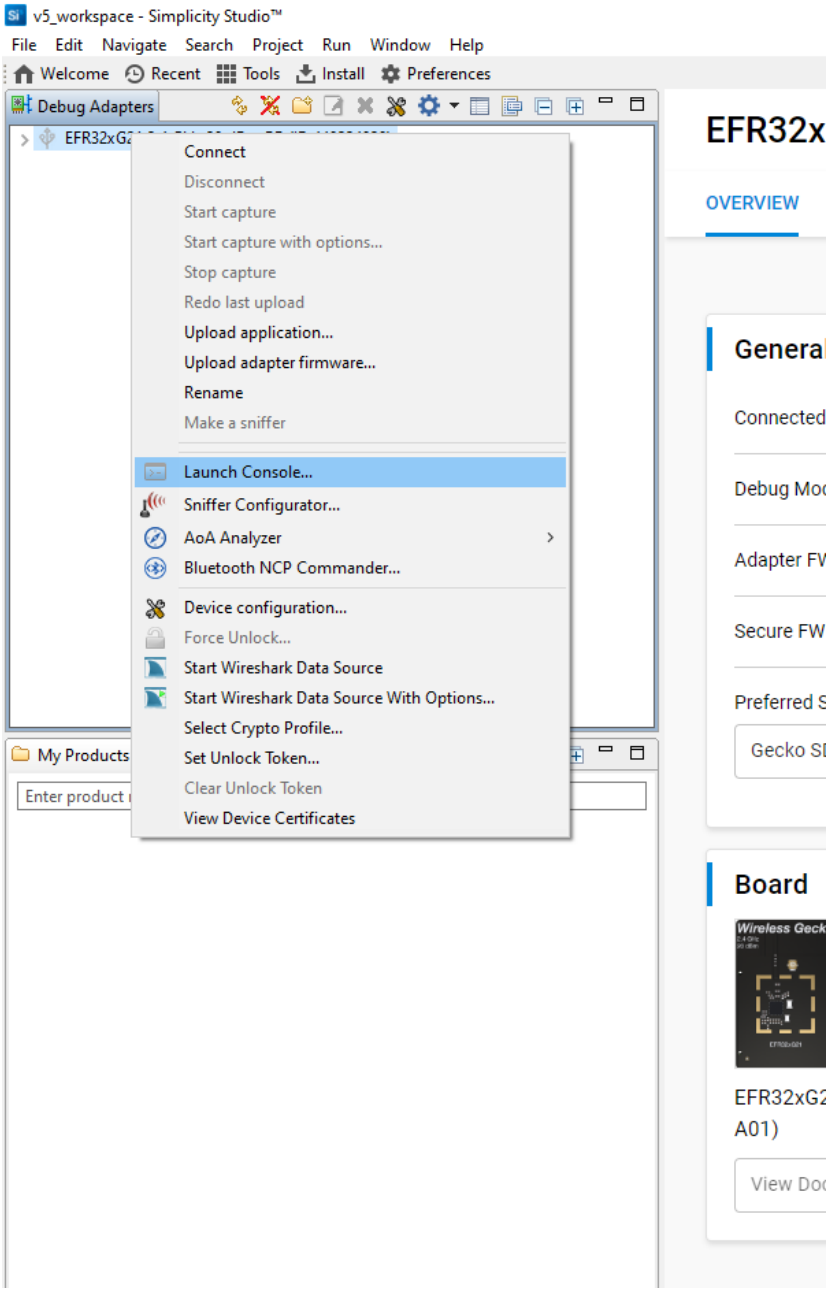


  - Retry debugging the application.

- For additional tips, see the Troubleshoot an Application Flash Failure section. Follow the suggested steps and then, instead of retying the flashing of the application, retry debugging the application.

Console Input and Output

1. Connect your EFR32 and SiWx91x boards to your computer.
2. Open Simplicity Studio.
3. In the **Debug Adapters** pane, right-click on your radio board and click **Launch Console**.



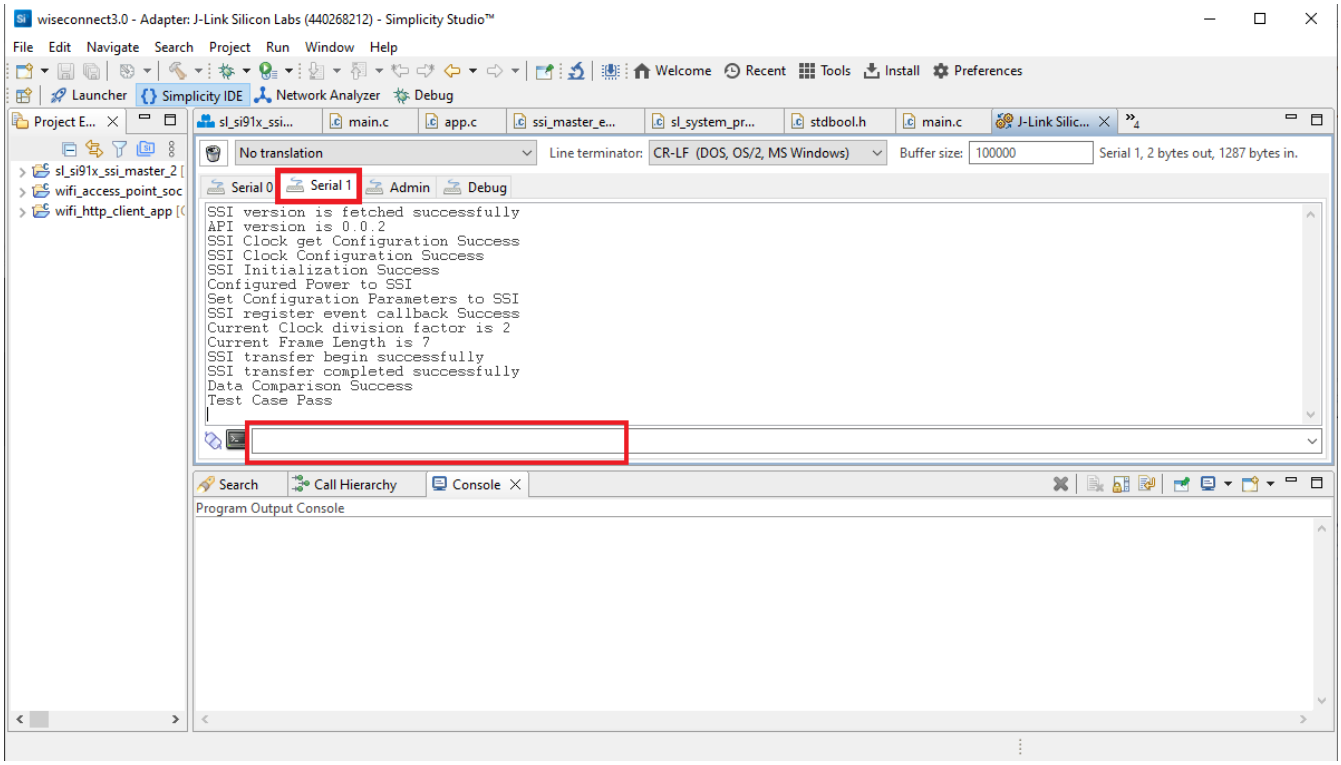
> **Note**: The image is for illustration purposes only. Board names and other details may not exactly match the product.

4. Select the **Admin** tab and press **Enter** to see the `WPK>` prompt.

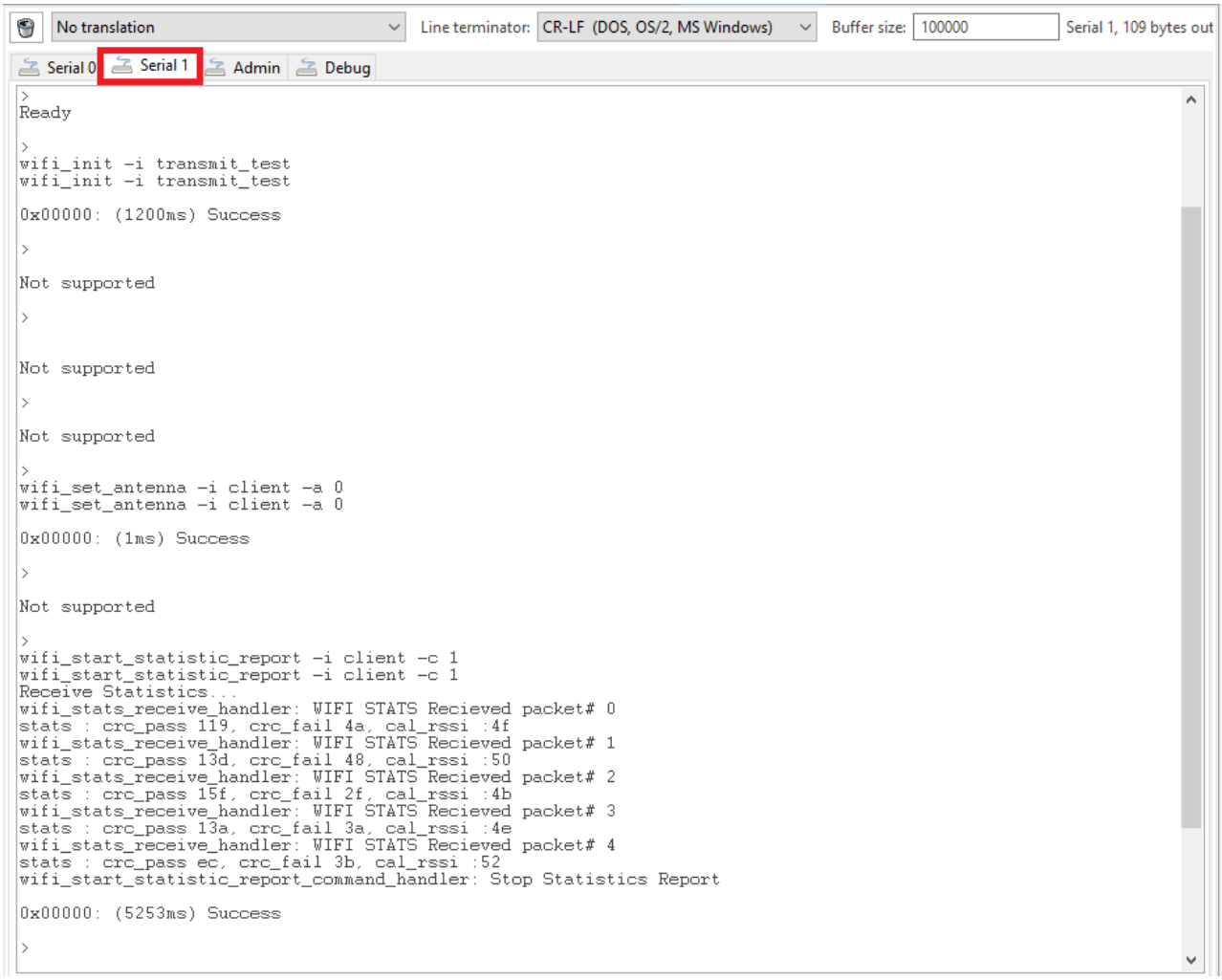
5. Enter the following command to set the baud rate:

```
pti config 0 vuart 115200
```

6. Select the **Serial 1** tab.
7. Place the cursor inside the text input field and hit **Enter**.
8. Console output will start getting displayed in the **Serial 1** tab.



9. Console input can be entered and sent to the device.



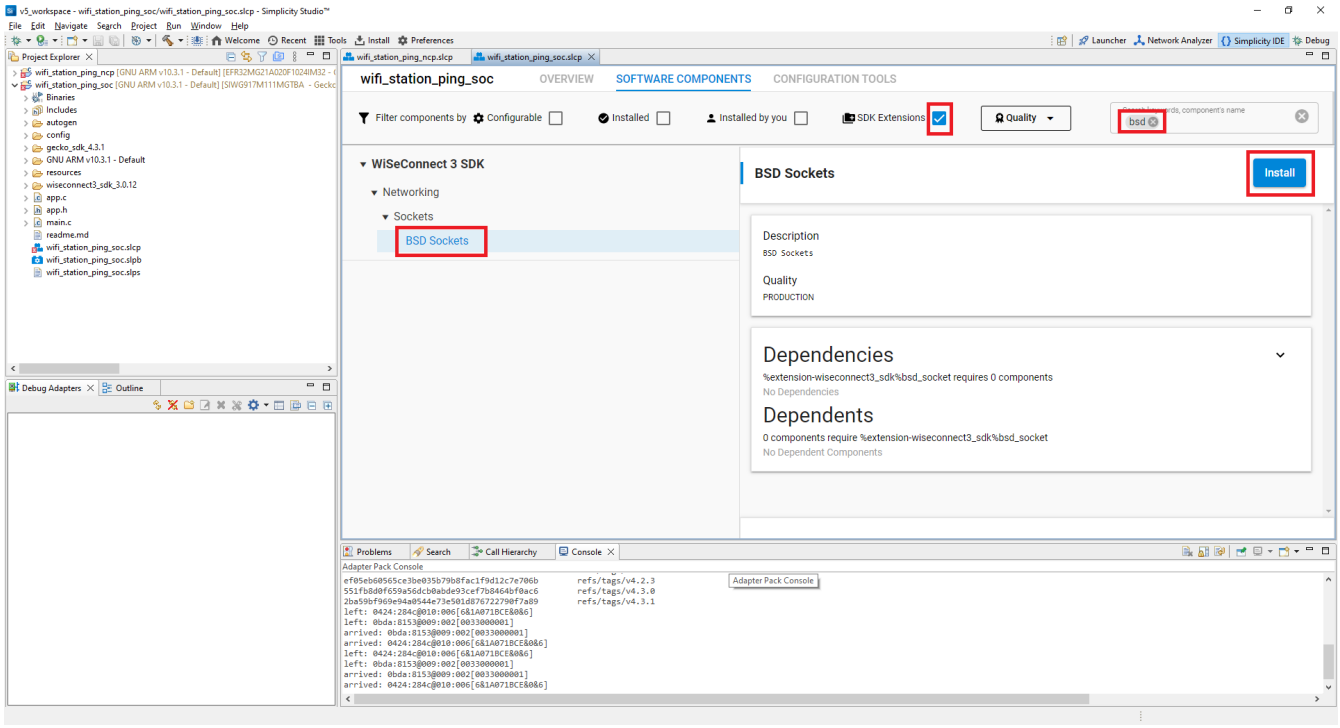
Customize Application Components

Simplicity Studio allows you to add or remove functional components in your application, such as BSD Sockets.

> **Note**: For information about the functional components available with WiSeConnect SDK v3.x, see the Application Components section.

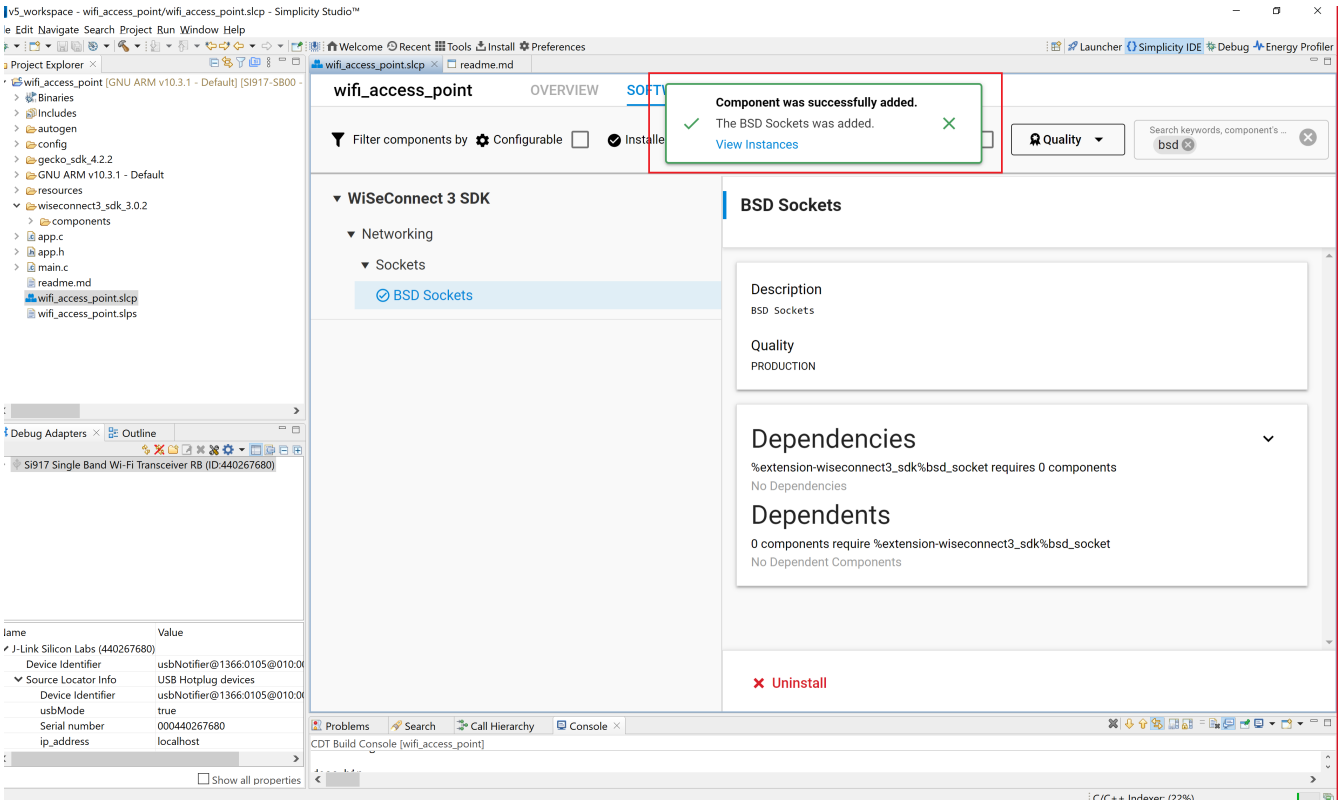
Add a Component

1. In the **Project Explorer** pane, double-click the **project_name.slcp** file.
2. Select the **SOFTWARE COMPONENTS** tab.
3. Select the **SDK Extensions** filter.
4. Browse or search for and select the component that you want to install.
5. Click **Install**.



> **Note**: The image is for illustration purposes only. Component and other details may not exactly match the product.

6. Studio will add the component and display a success message.

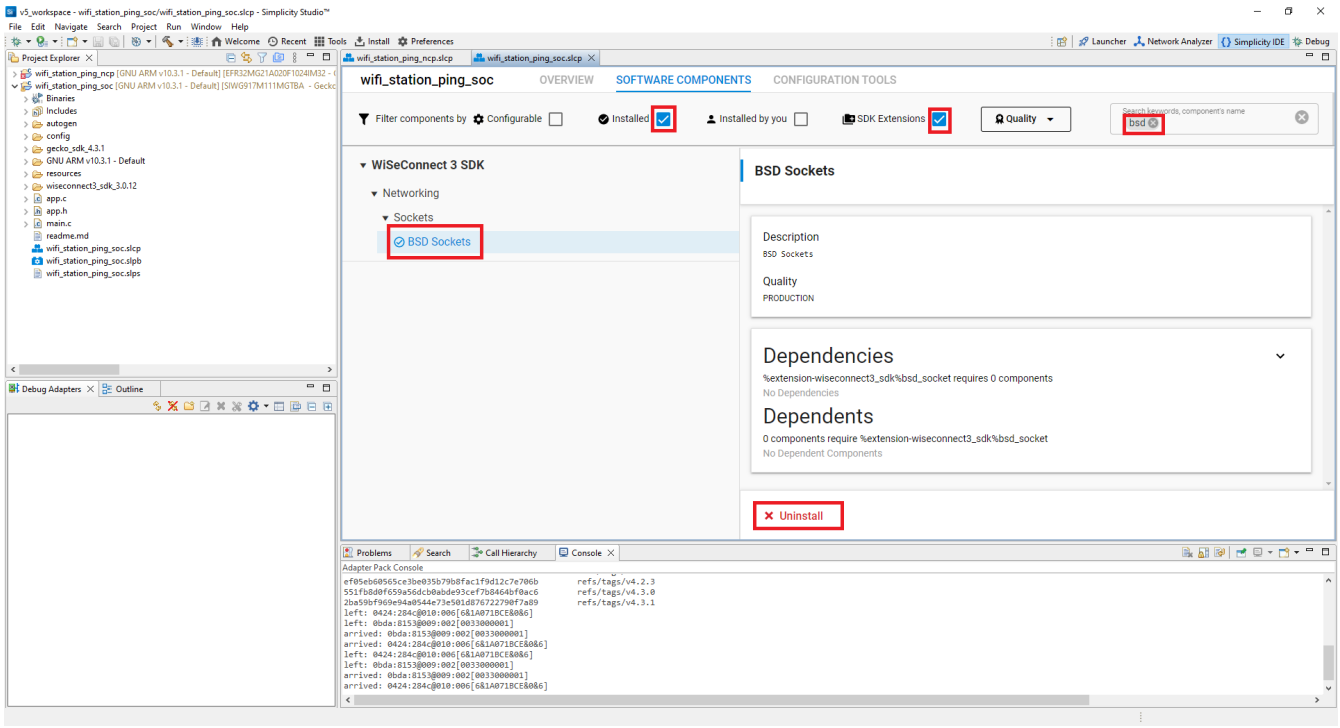


> **Note**: The image is for illustration purposes only. Component and other details may not exactly match the product.

> **Note**: You may use the Component Editor to configure a component after adding it or to configure other components in your example.

Remove a Component

1. View the list of **WiSeConnect 3** extension components by following steps 1-3 of the previous section.
2. Select the **Installed** filter to view the components you have installed.
3. Browse or search for and select the component you want to remove.
4. Select the component and click **Uninstall**.



> **Note**: The image is for illustration purposes only. Component and other details may not exactly match the product.

Developing for SiWx91x Host

Developing with WiSeConnect™ SDK v3.x with the SiWx91x Host™

This guide describes how to develop applications for the SiWx91x™ chipset family using the WiSeConnect™ SDK v3.x in System-on-chip (SoC) mode, where both the application and the connectivity stack run on the SiWx91x chipset.

> **Note**: The output images in this guide are for illustration purposes only. Details such as board names and version numbers may not exactlyb match the product.

Check Prerequisites

Software

- Simplicity Studio
- Gecko software development kit (GSDK)

> Note:
>
> - We recommend using the latest GSDK version.
> - Refer to the Release Notes for the GSDK version tested with this release.

Hardware

- Wi-Fi Access Point (802.11 ax/b/g/n)
- SiWx917 SoC development boards or kits. See the Check Prerequisites > Hardware section on the **Starting in SoC Mode** page.
- Windows/Linux/MacOS computer with a USB port
- Type C USB cable compatible with the computer's USB port (for e.g., type C to type A if the computer has a type A USB port).

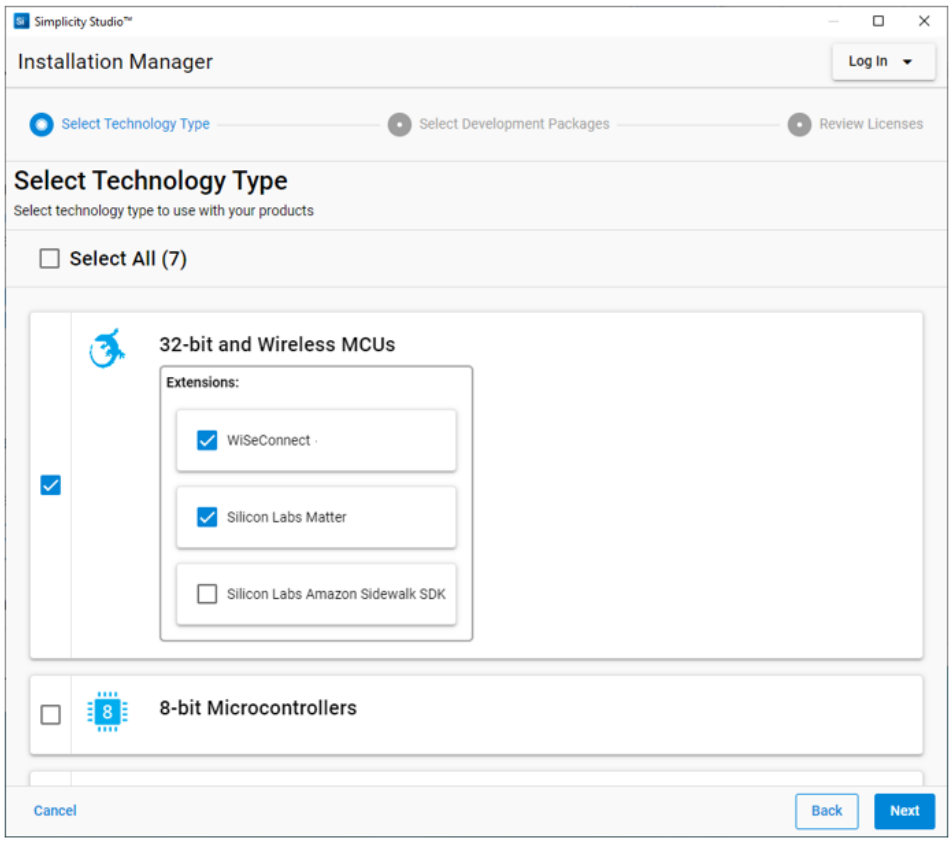
Setup Software

You may setup the following software in this section while waiting to receive the hardware:

- Simplicity Studio
- WiSeConnect 3 Extension

Install Simplicity Studio

1. Download the latest version of Simplicity Studio and follow the installation instructions. During the installation:
   - make sure you log in to Simplicity Studio in the **Installation Manager** window,
   - select **Install by technology type**, and
   - select the **WiSeConnect** extension under **32-bit and Wireless MCUs**.



Install the WiSeConnect 3 Extension

If you already selected the **WiSeConnect** extension in the Install Simplicity Studio section, you may skip this section.

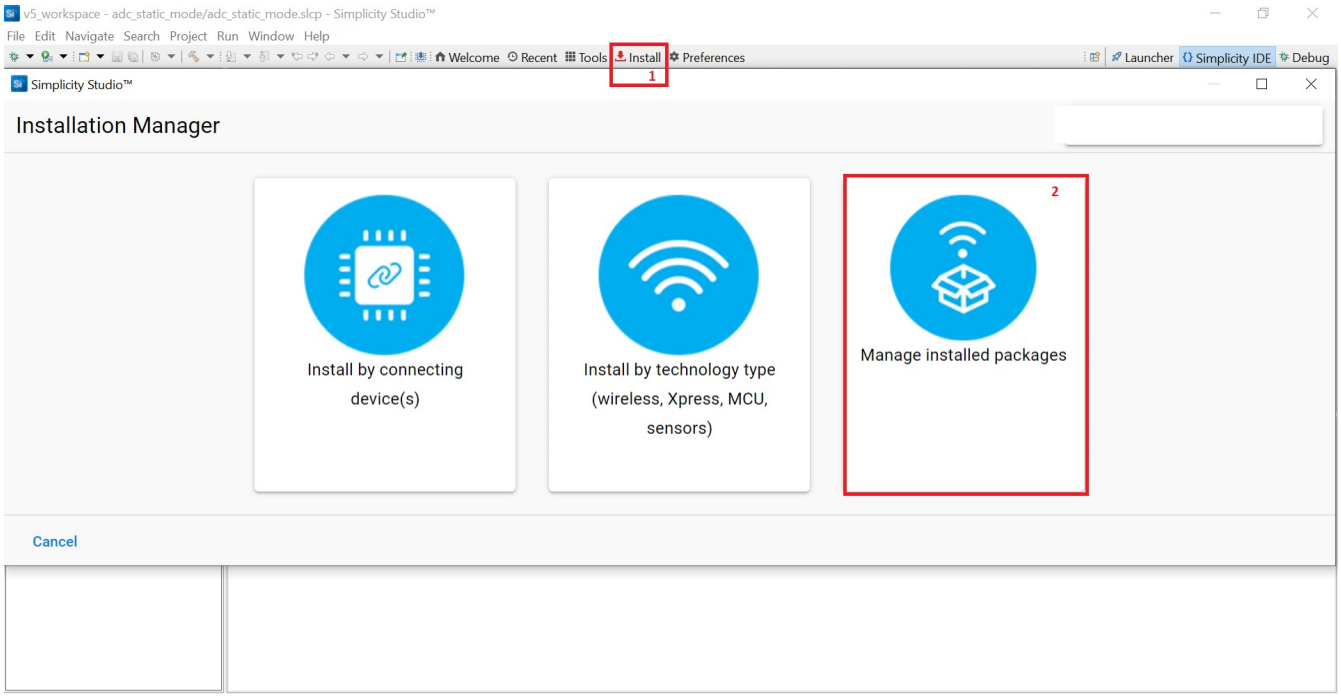
Before installing the **WiSeConnect 3** extension, upgrade to a compatible GSDK version if not already done. See the Prerequisites section for the supported GSDK versions.

You may install **WiSeConnect 3** through one of the following alternative paths:

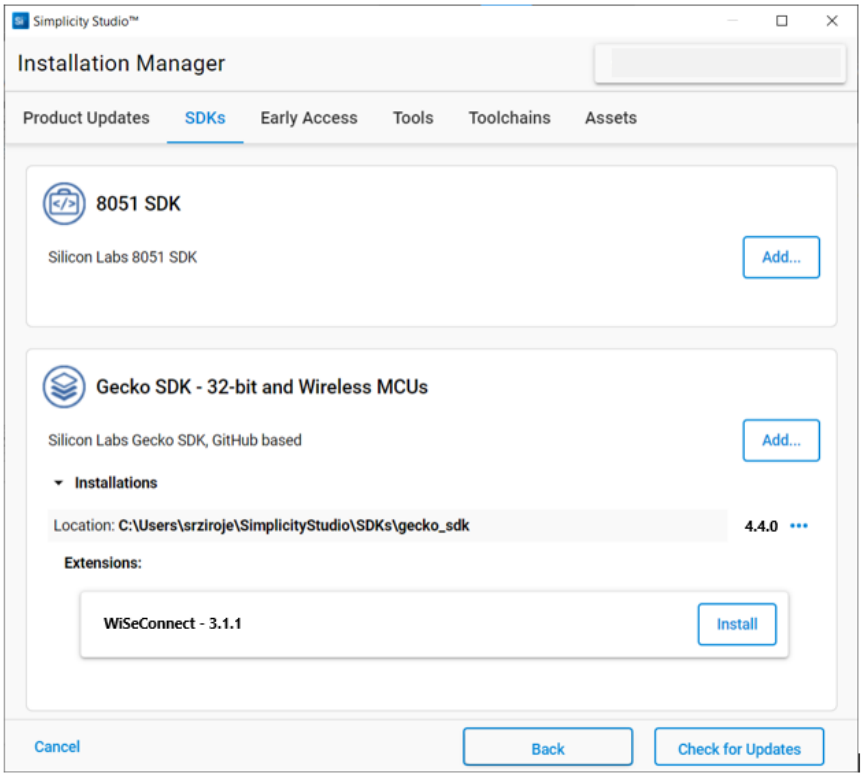
- Installation Manager (**recommended**)
- Manage SDKs Window (**hardware required**)

Install WiSeConnect 3 through the Installation Manager

1. Log in to Simplicity Studio if not already done.
2. In the Simplicity Studio home page, select **Install > Manage installed packages**.



3. Select the **SDKs** tab.
4. Next to the **WiSeConnect - 3.x.x** extension, click **Install**.



Install WiSeConnect 3 through the Manage SDKs Window

> **Note**: You must have the hardware available before using these steps to install the **WiSeConnect 3** extension.

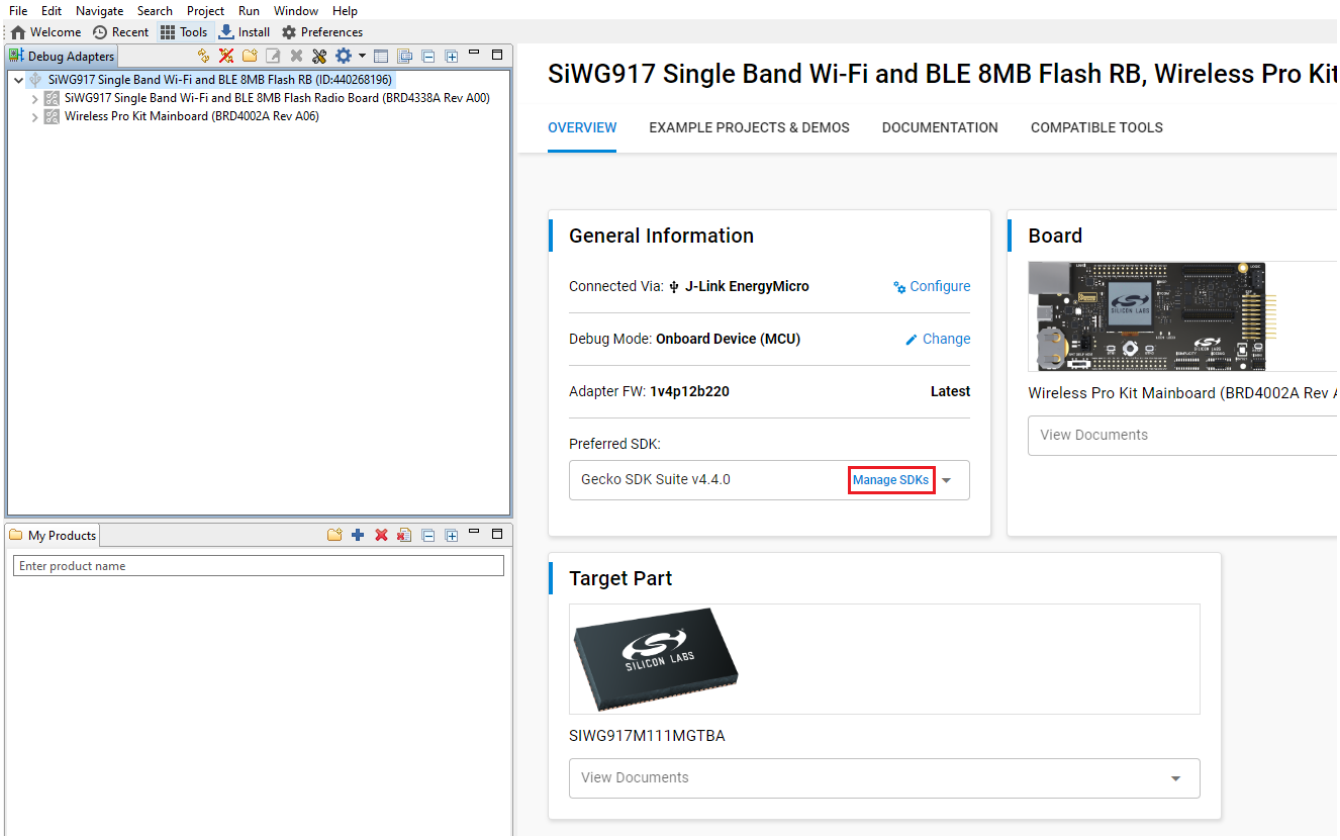
1. Download the WiSeConnect v3.x source code from the following URL after substituting **3.x.x** with the desired release version:

   ```
   https://github.com/SiliconLabs/wiseconnect/archive/refs/tags/v3.x.x.zip
   ```

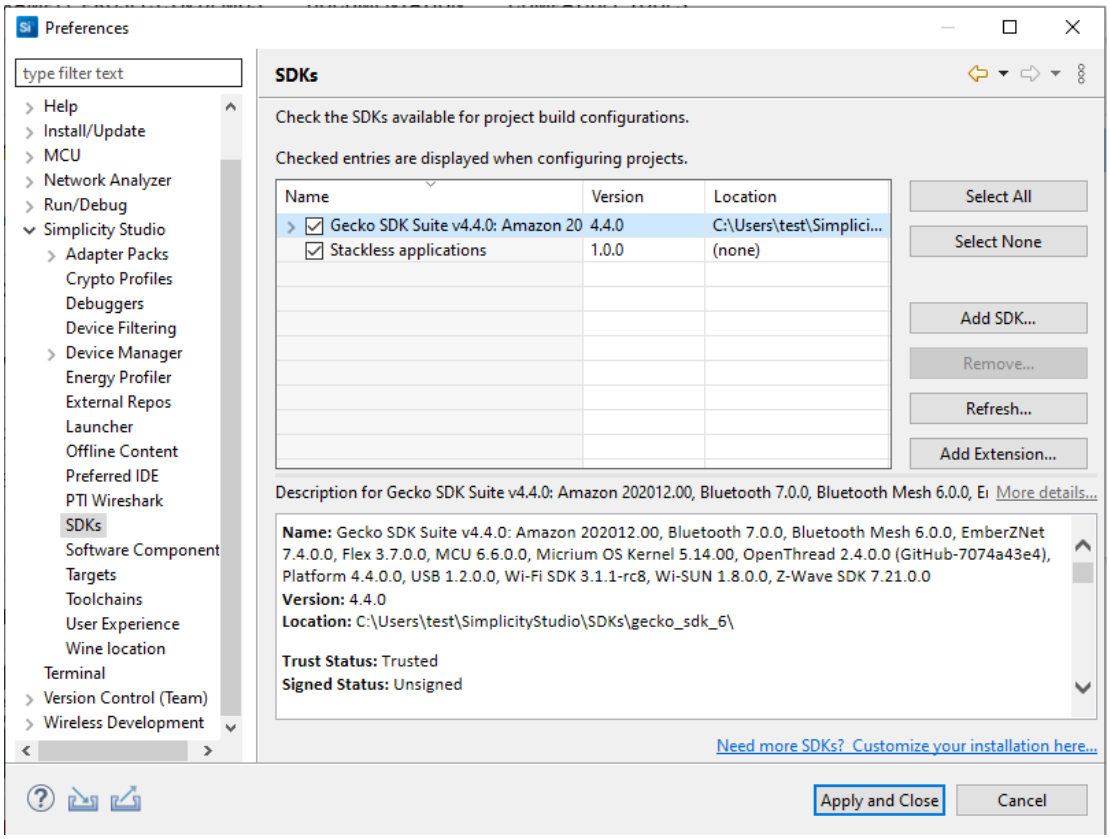
   - If you don't know your release version, go to the GitHub Releases Page and select the version to download.
2. Unzip the downloaded `wiseconnect-3.x.x.zip` file. It will be extracted into a folder structure similar to the following:

   ```
   wiseconnect-3.x.x
   |--- wiseconnect-3.x.x
   |------ <source code>
   ```

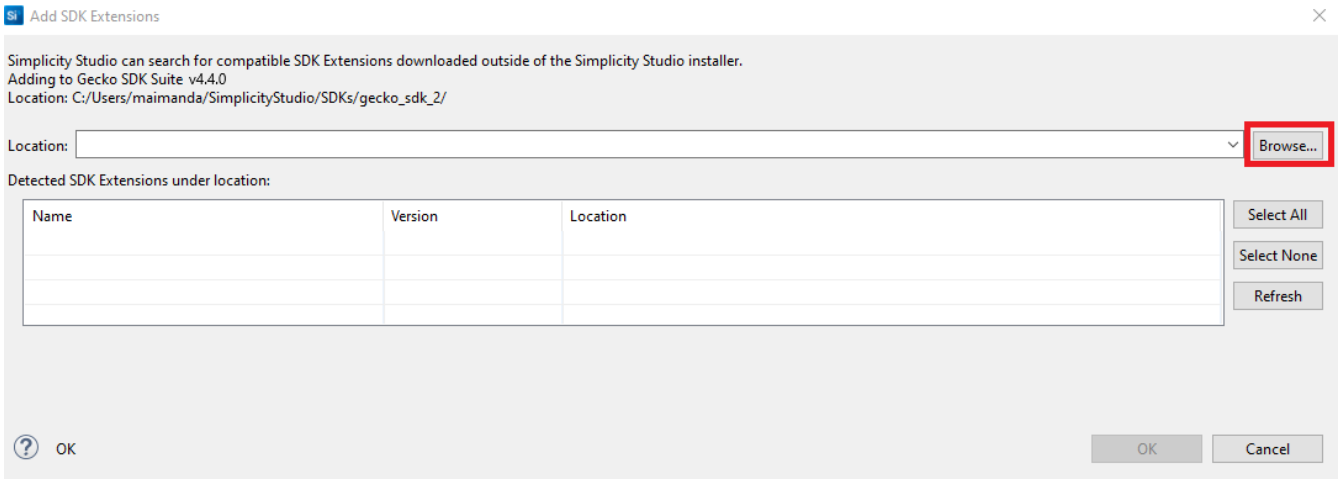
3. Launch **Simplicity Studio** and log in.
4. Connect the SiWx91x to your computer.
5. In the **Debug Adapters** pane, select your radio board.
6. In the **General Information** section, click **Manage SDKs**.



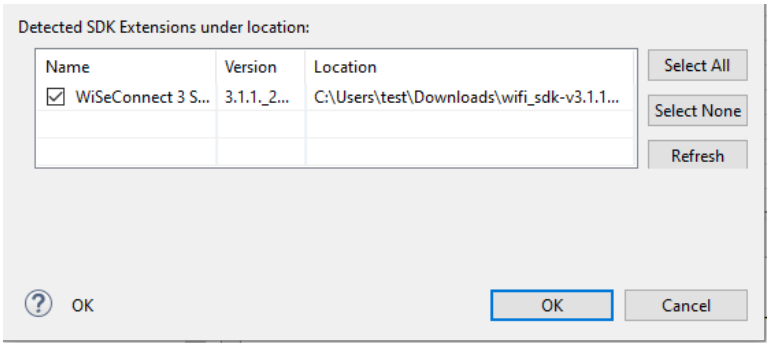
7. The **Preferences** window will be opened in the **SDKs** section.
8. Select **Gecko SDK Suite vx.x.x** and click **Add Extension**.



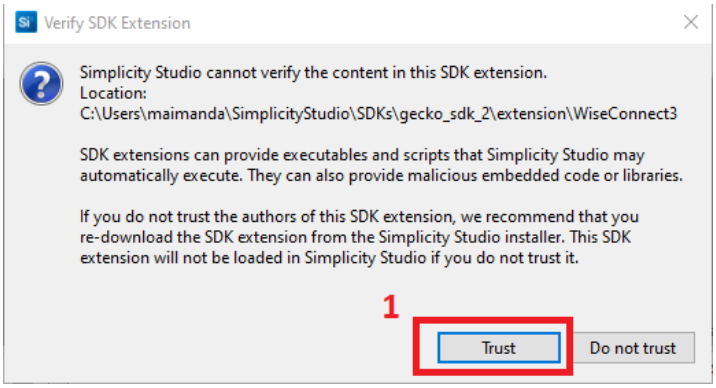
9. In the **Add SDK Extensions** window, click **Browse**.



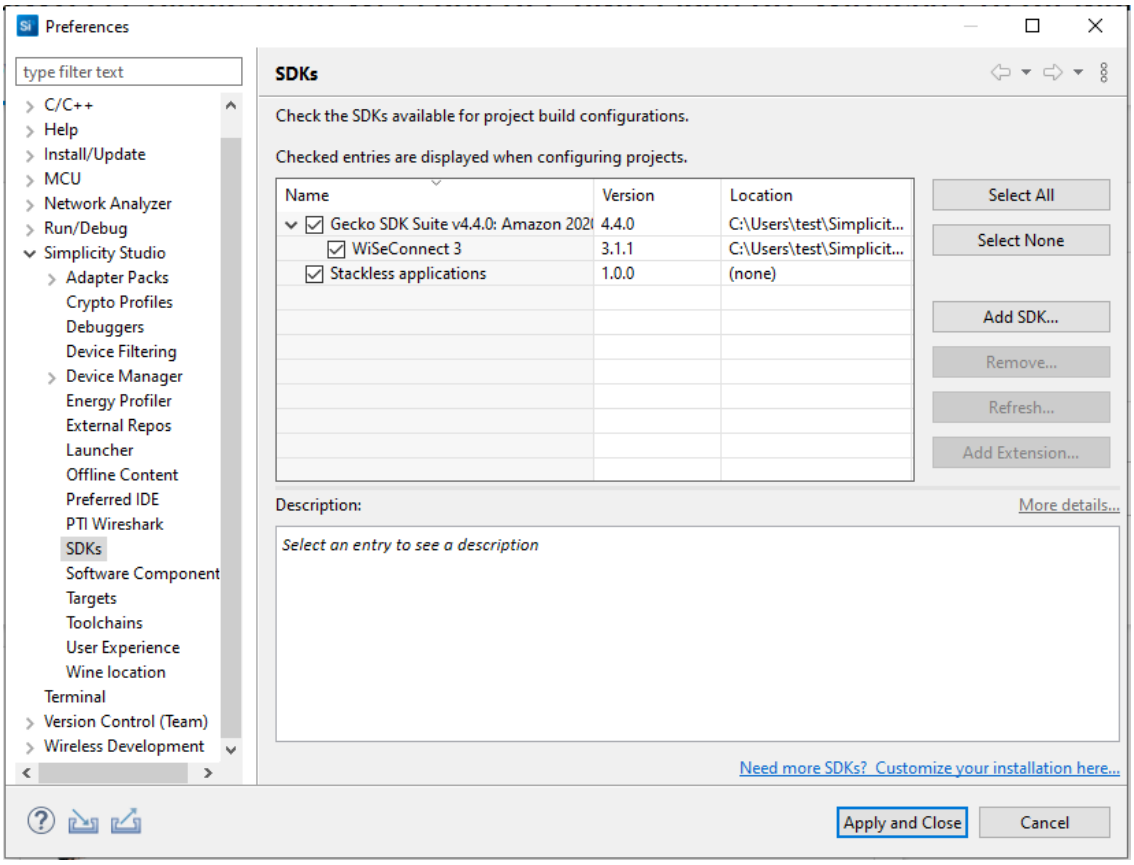
10. Locate and select the `wiseconnect-3.x.x` sub-folder extracted in step 2 above which contains the source code.
11. Studio will detect the **WiSeConnect 3 SDK** extension.
12. Select the detected extension and click **OK**.



13. If a **Verify SDK Extension** popup is displayed, click **Trust**.

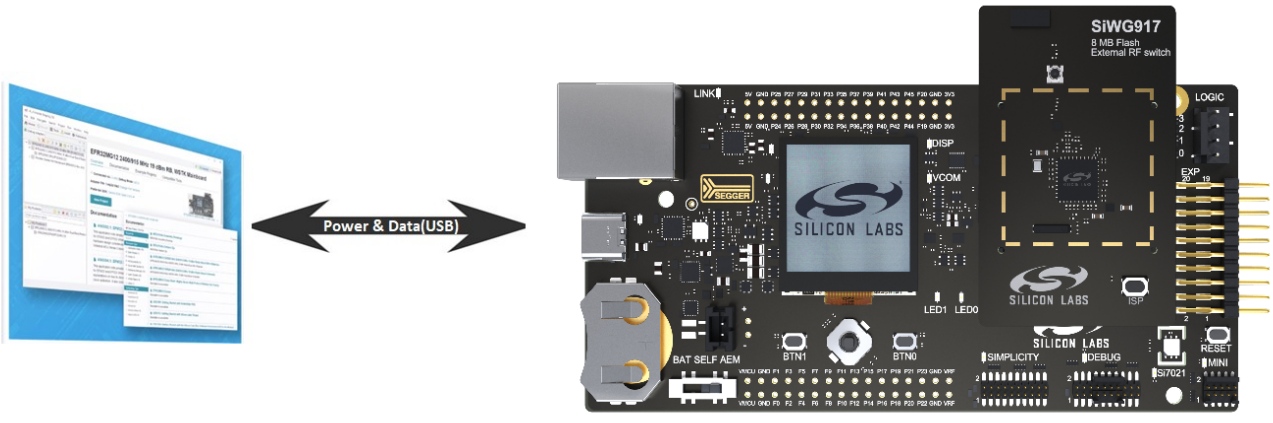


14. The selected **WiSeConnect 3** extension will be displayed.
15. Click **Apply and Close**.

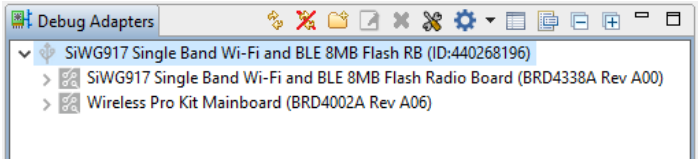


Connect SiWx91x to Computer

1. Connect the WPK board to your computer using a USB cable.



2. Simplicity Studio will detect and display your radio board.



Troubleshoot a Board Detection Failure

If Simplicity Studio does not detect your radio board, try the following:

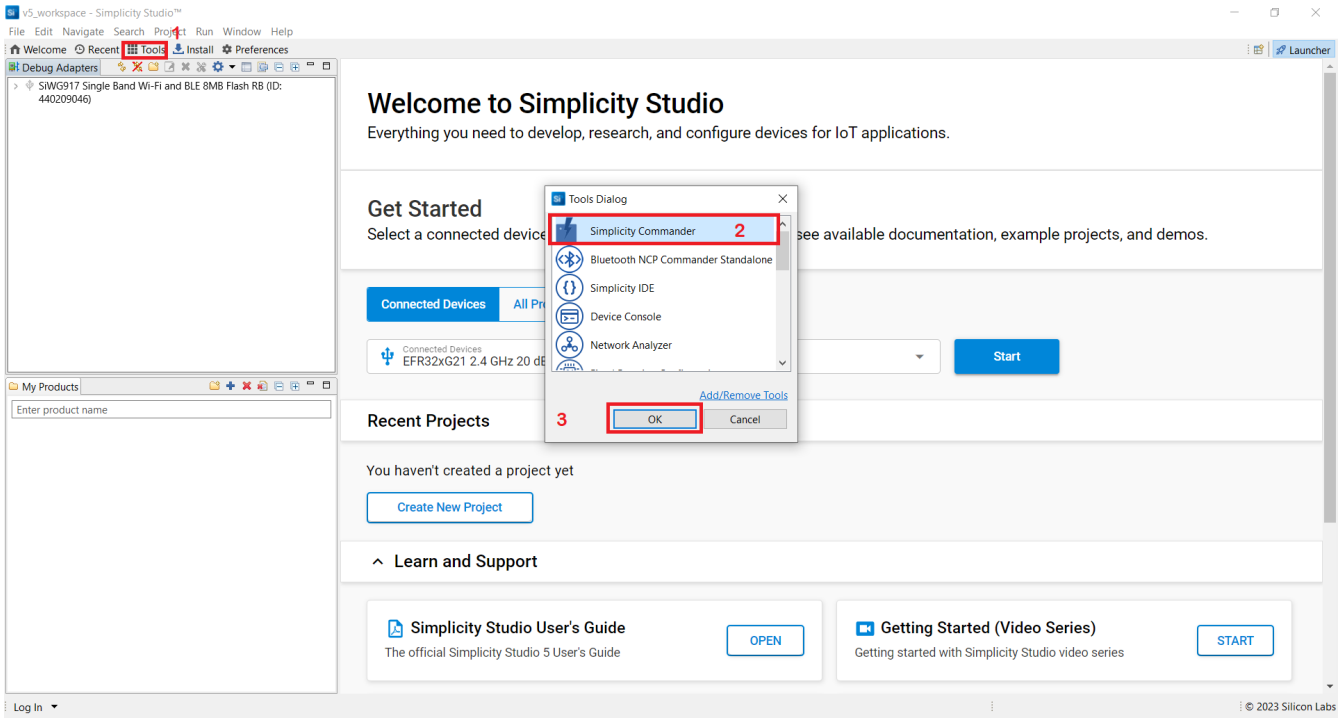
- In the **Debug Adapters** pane, Click the **Refresh** button (having an icon of two looping arrows).
- Press the **RESET** button on the WPK board.
- Power-cycle your device by disconnecting and reconnecting the USB cable.

Upgrade SiWx91x Connectivity Firmware

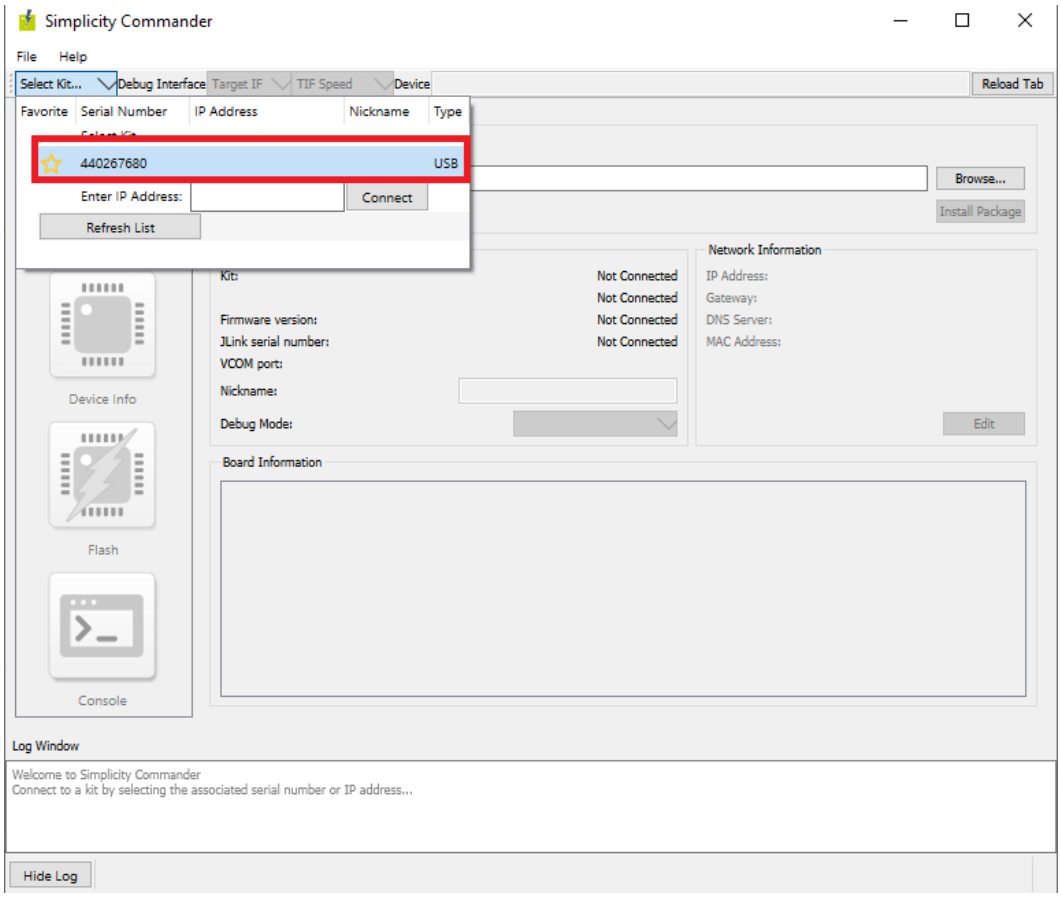
We recommend that you upgrade the SiWx917 connectivity firmware to the latest available version when:

- you first receive an SiWx917 Pro kit
- you first receive a radio board, or
- you upgrade to a new version of the WiSeConnect 3 extension

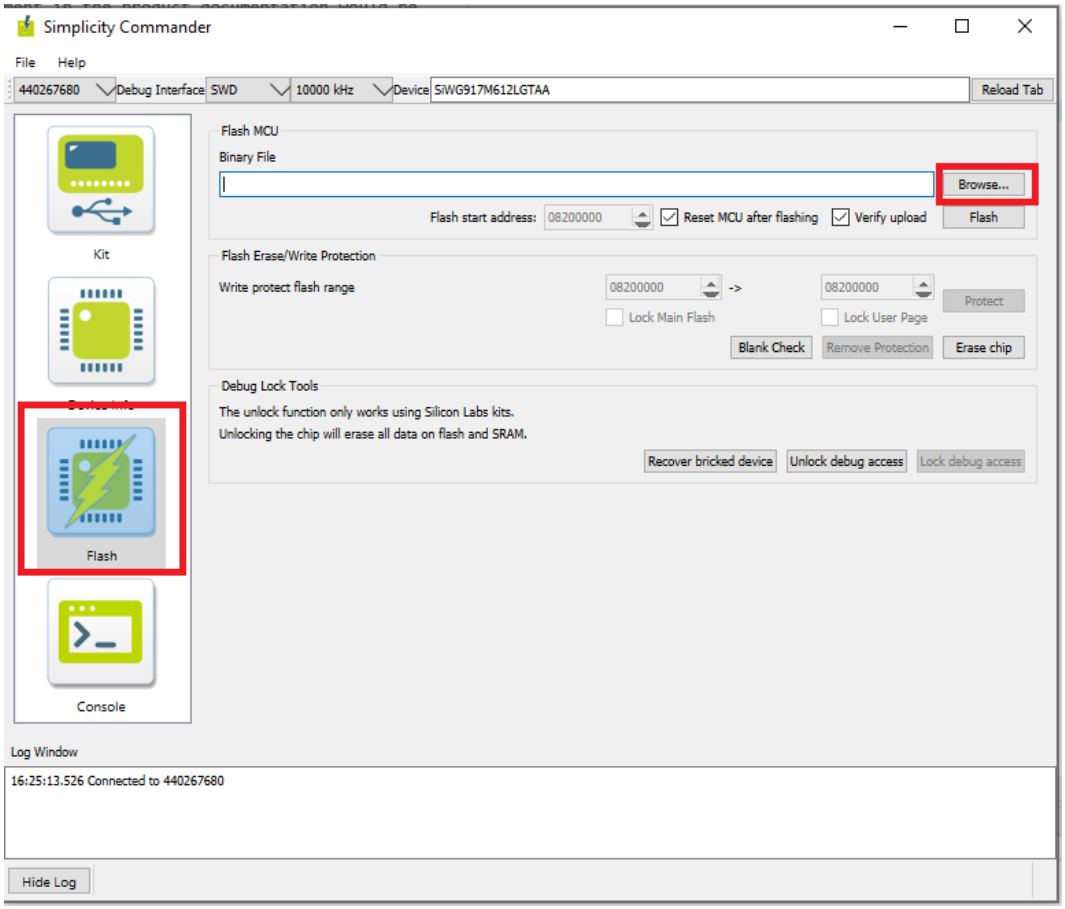
1. In the **Simplicity Studio** home page, click **Tools**.
2. In the **Tools** dialog, select **Simplicity Commander** and click **OK**.



3. In the **Simplicity Commander** window, click **Select Kit** and choose your radio board.

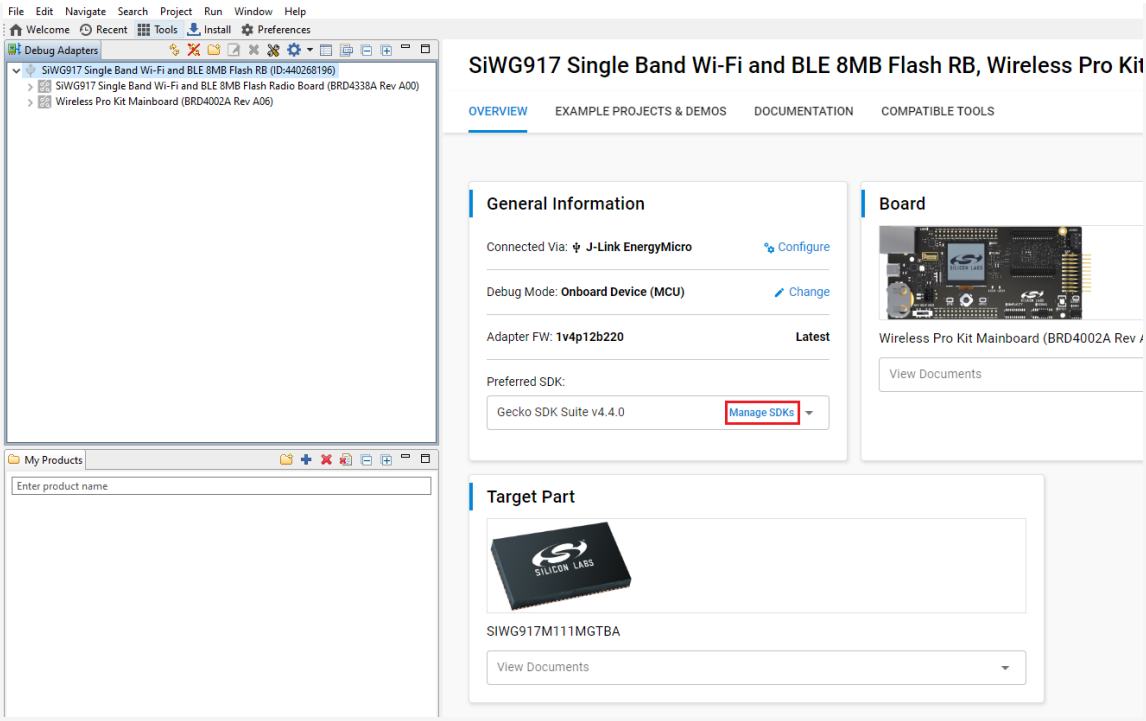


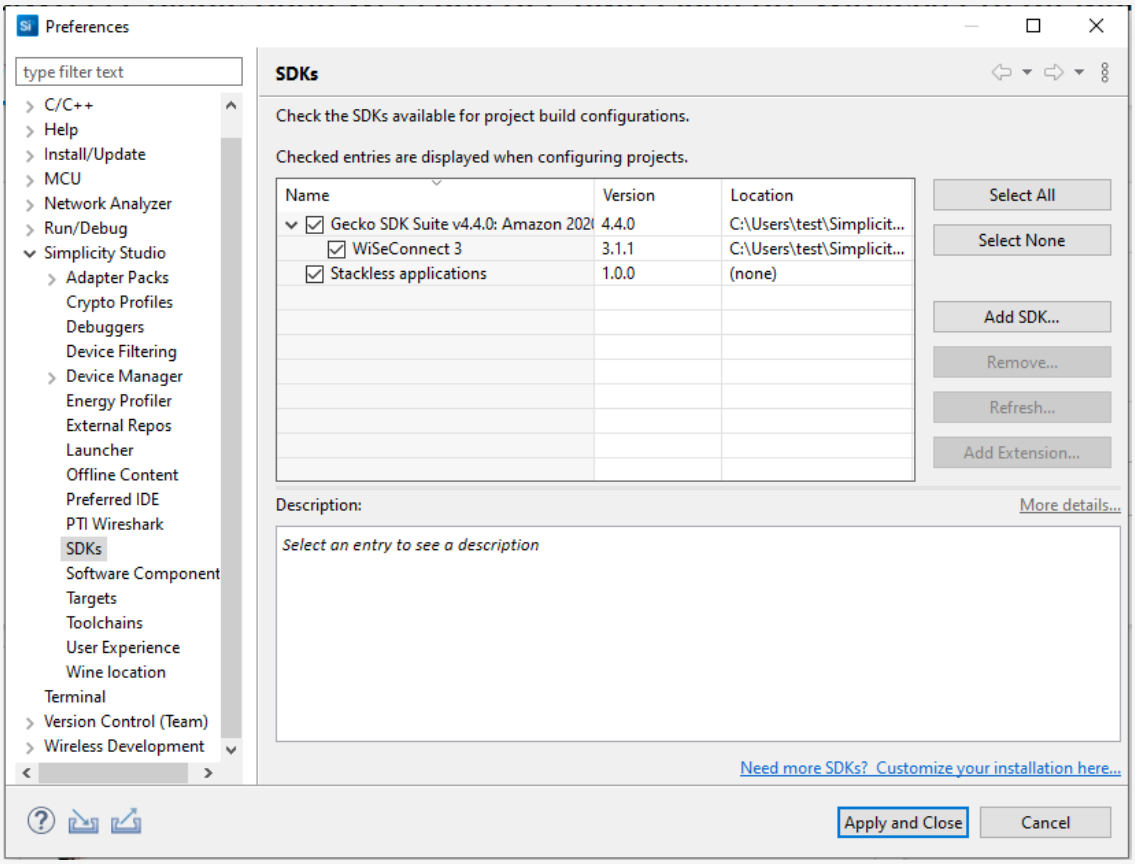
4. In the navigation pane, go to the **Flash** section.
5. Click **Browse** next to the **Binary File** field.



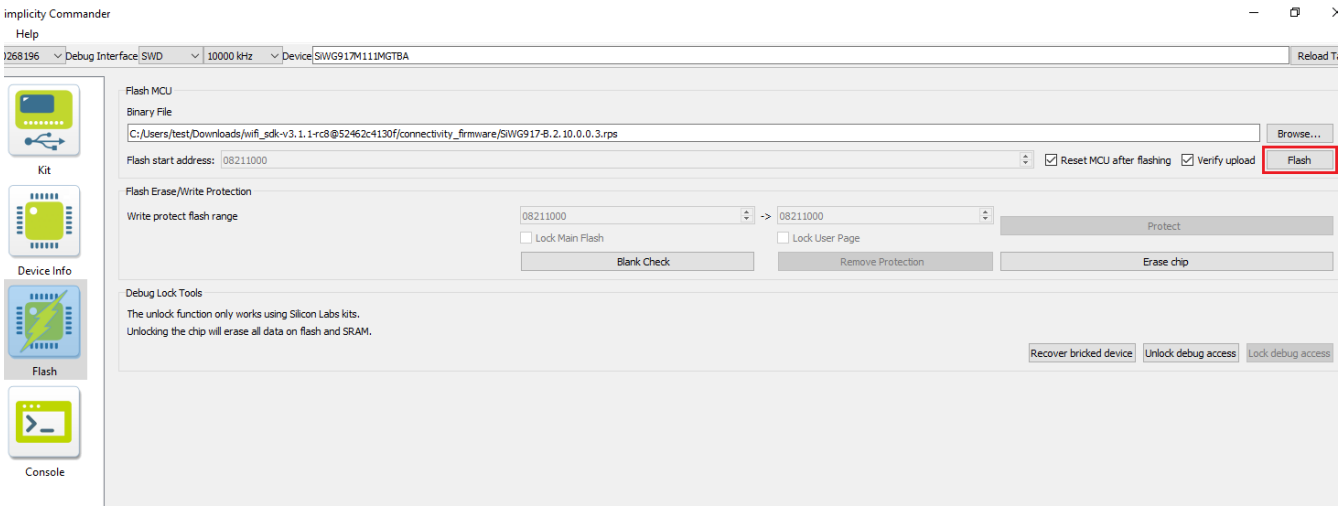
6. Locate and select the firmware file to flash from within the `connectivity_firmware` sub-folder of the **WiSeConnect 3** extension path.

> Note:
>
> - If you don't see any files in the sub-folder, select **All files** in the file filter field in the Browse dialog.
> - The **WiSeConnect 3** extension path is the one where the extension was downloaded during installation. If you're not sure what the path is, you may refer to the location in the **Preferences > SDKs** page shown on clicking **Manage SDKs**.





7. Click **Flash**.



8. The firmware will be flashed and the **Log Window** will display the message: "Flashing completed successfully!"

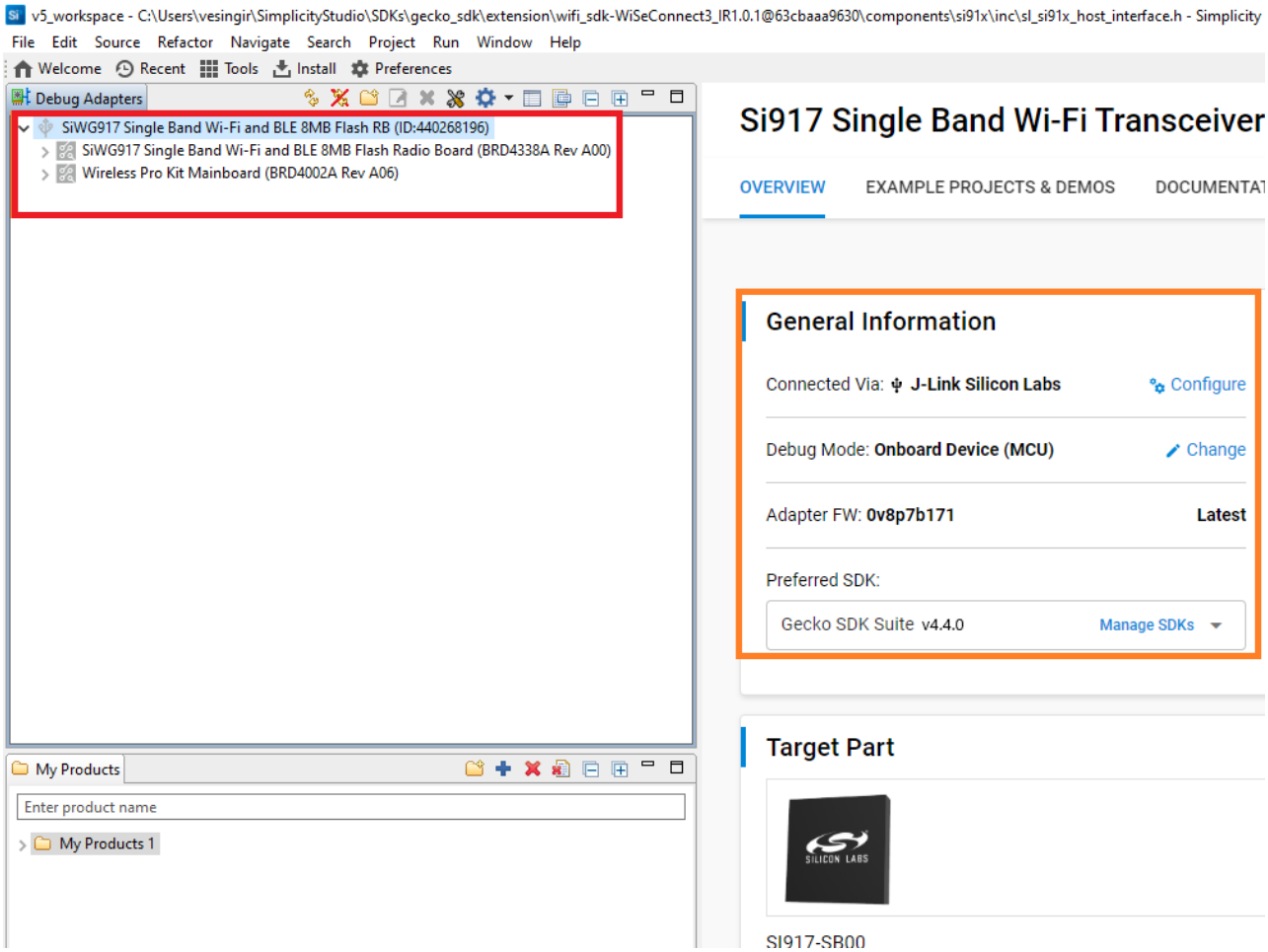
Troubleshoot Firmware Update Failure

If the firmware update fails, try the following:

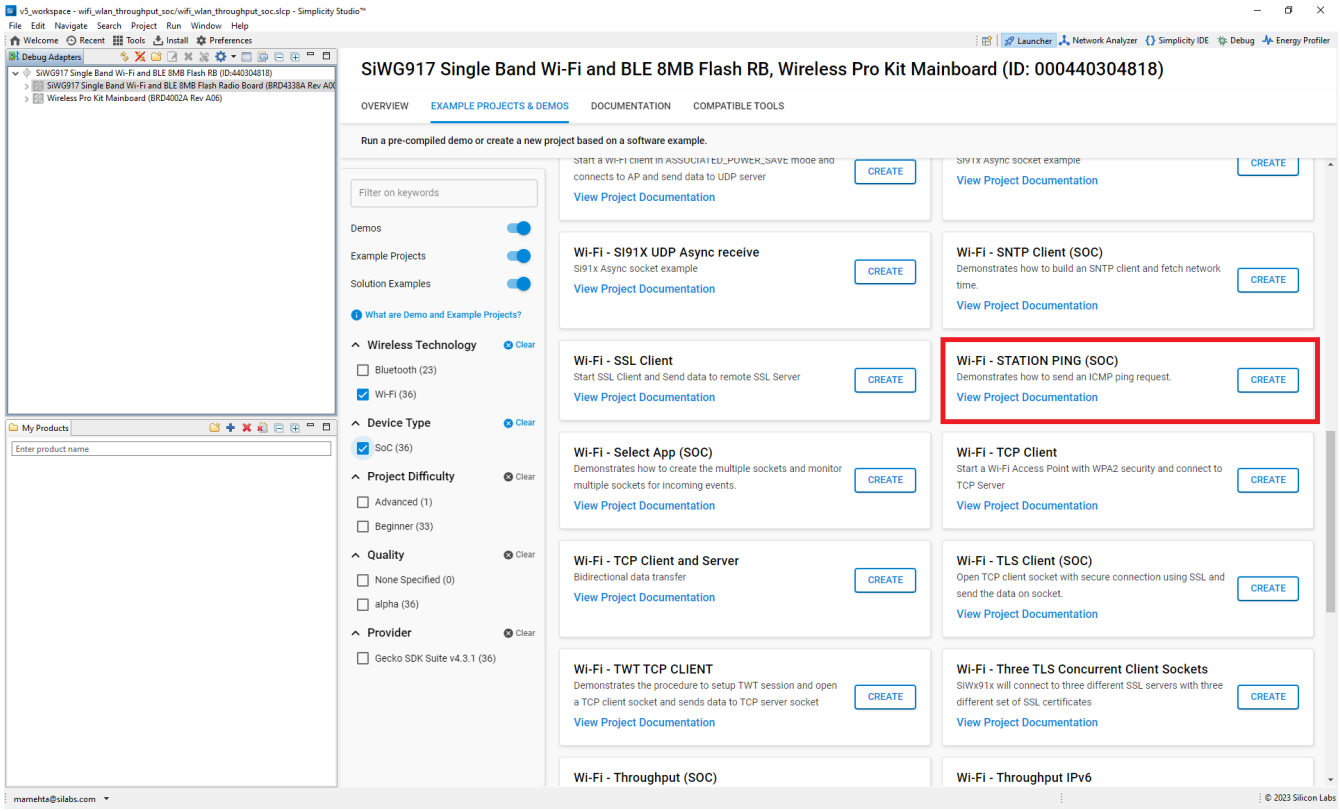
- Toggle the power switch towards **AEM** (Advanced Energy Monitoring) on the WPK board
- Try the following steps:
  - Press and hold the **ISP** button on the radio board.
  - Press and release the **RESET** button on the WPK board.
  - Release the **ISP** button on the radio board.
  - In the **Flash** section in step 5 above, click **Erase chip**.
  - The flash will be erased.
  - Press the **RESET** button on the WPK again.
  - Retry the firmware upgrade.
- In case studio failed to detect your board, see the Troubleshoot a Board Detection Failure section.

Create Project

1. Log in to Simplicity Studio and connect the SiWx91x to your computer.
2. Go to the **Debug Adapters** section.
3. Select your radio board from the displayed list.
4. The **Launcher** page will display the selected radio board's details.
5. Select the **OVERVIEW** tab.
6. Verify the following in the **General Information** section:
   - The **Debug Mode** is **Onboard Device (MCU)**.
   - The **Preferred SDK** is the version you selected earlier.

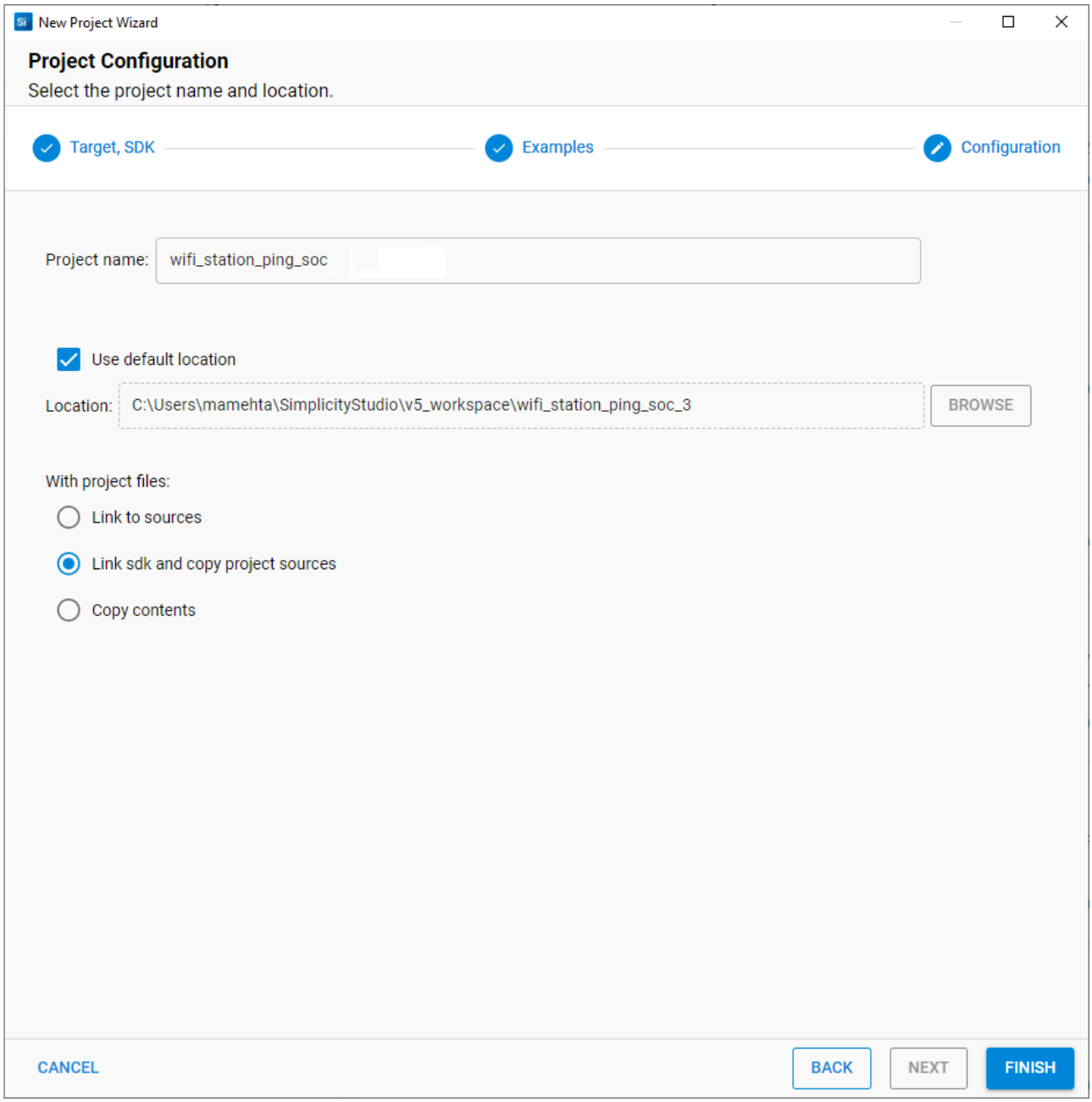


7. Select the **EXAMPLE PROJECTS AND DEMOS** tab.
8. Locate the example you want and click **CREATE**.

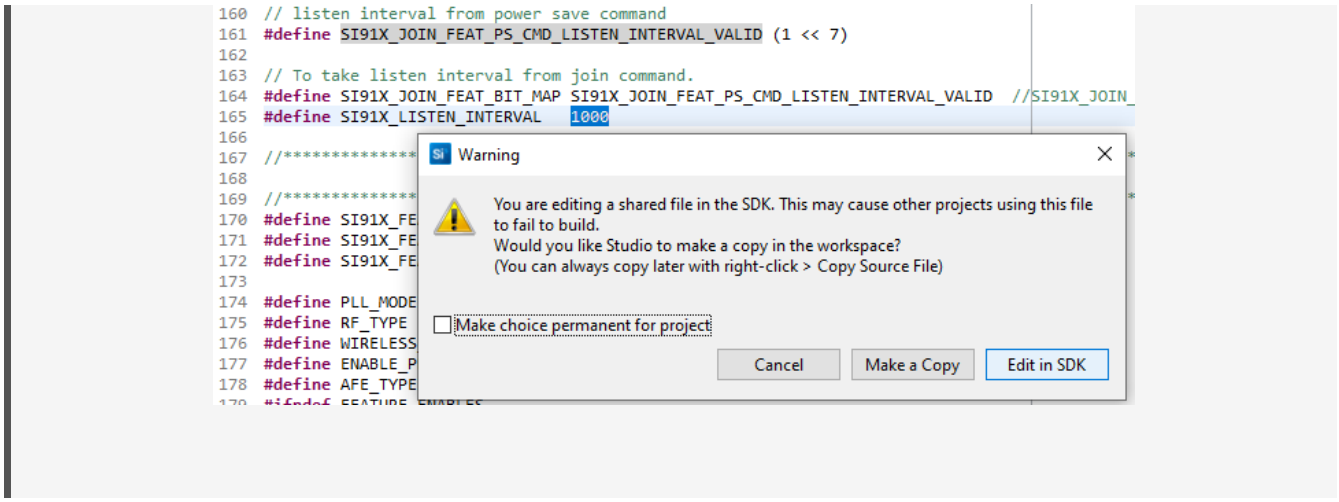


9. In the **New Project Wizard** window, click **FINISH**.

> **Note**: The **Copy contents** option is not currently supported.



> **Note**: The **Make a Copy** option is not currently supported while editing a shared SDK file, when prompted to either make a copy of the file or edit it in the SDK itself.



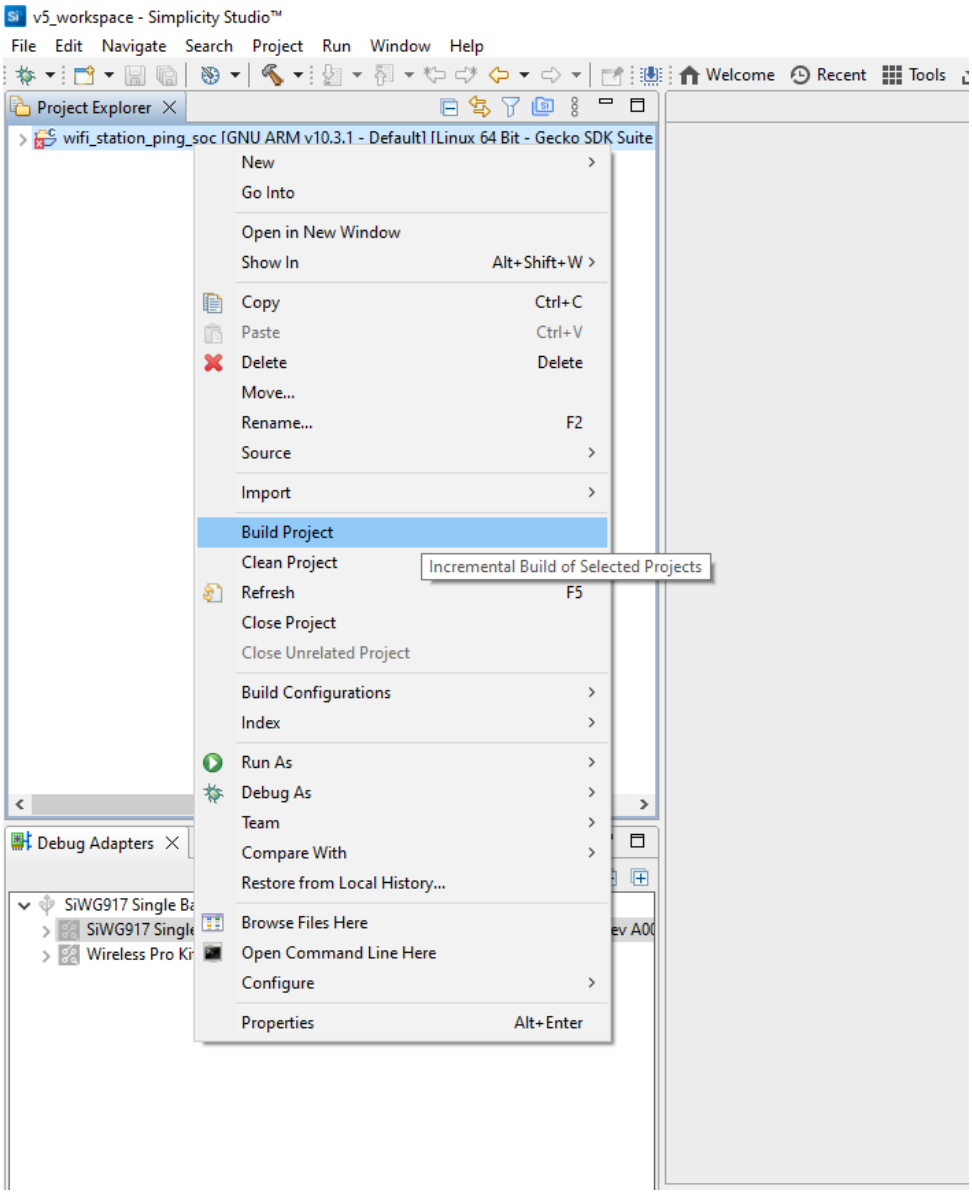
Configure an Application

Configure the settings for your example. For **Wi-Fi - STATION PING (SOC)** (the recommended example for this guide) or for other examples, see the Application Build Environment section in the **README** page for configuration instructions.

You may use the Component Editor to configure the components in your example.

Build an Application

1. Launch Simplicity Studio and log in.
2. In the **Project Explorer** pane, right-click the project name and select **Build Project**.
    - You may also click the **Build** button with a hammer icon on the Simplicity IDE perspective toolbar.



Flash an Application

> **Note**: The SiWx91x SoC comes pre-flashed with a bootloader. A bootloader image does not need to be flashed separately.

There are two alternative methods to flash an application to the application processor of the SiWx91x device:

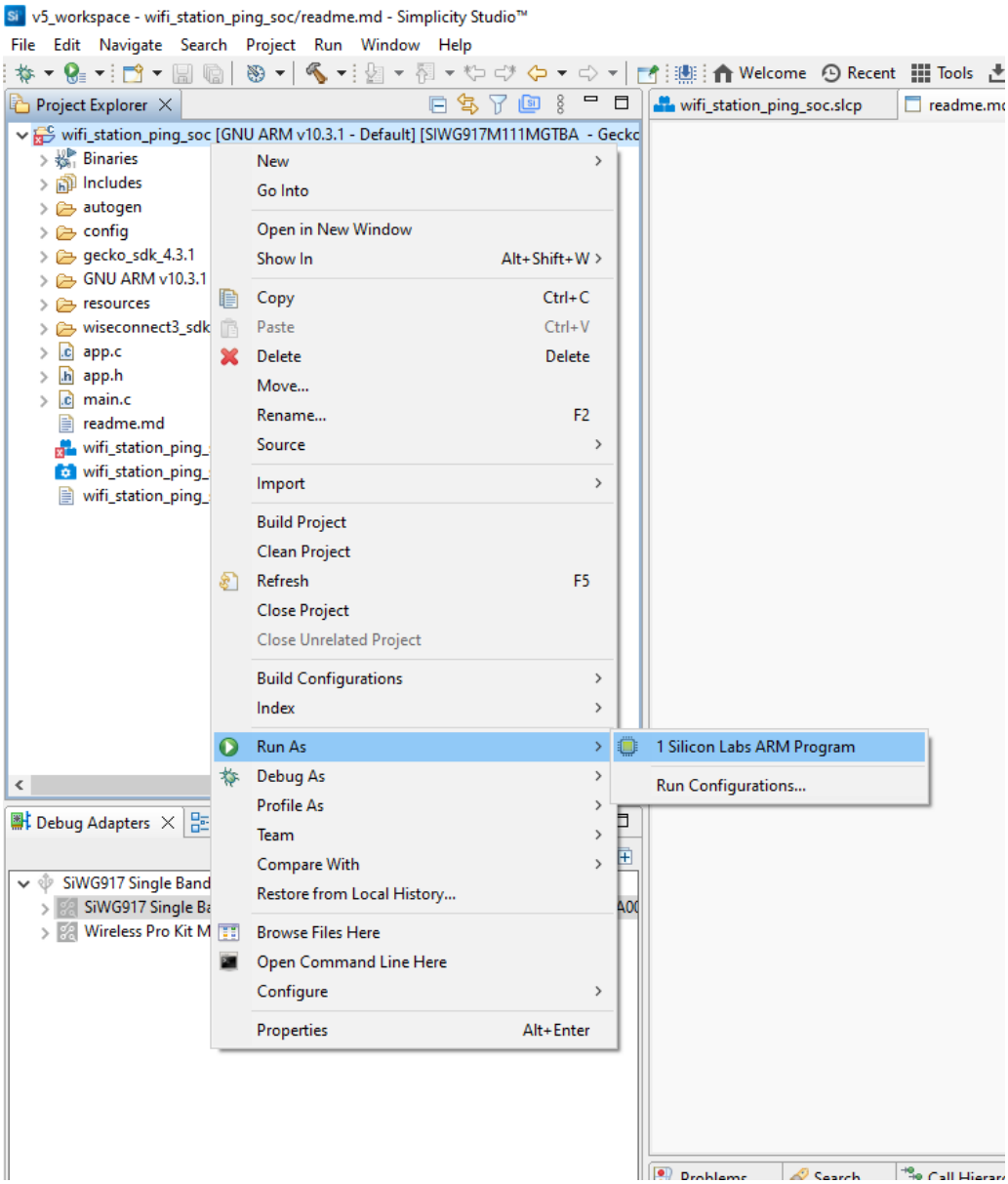
- Flash an application Built Using Simplicity Studio
- Flash a Binary that is Already Available

Once you flash the **Wi-Fi - STATION PING (SOC)** example (the recommended example for this guide), you may refer to the Test the Application section of its **README** page to explore its output. The other sections of the **README** like the Purpose/Scope section and **Overview** section (not present in some **README**'s) provide more information about the example.

See the Examples page to explore all available examples and view their **README** pages.

Flash an Application Built Using Simplicity Studio

1. Build the application as described in the Build an Application section.
2. In the **Project Explorer** pane, right-click on your project name and select **Run As > 1 Silicon Labs ARM Program**.

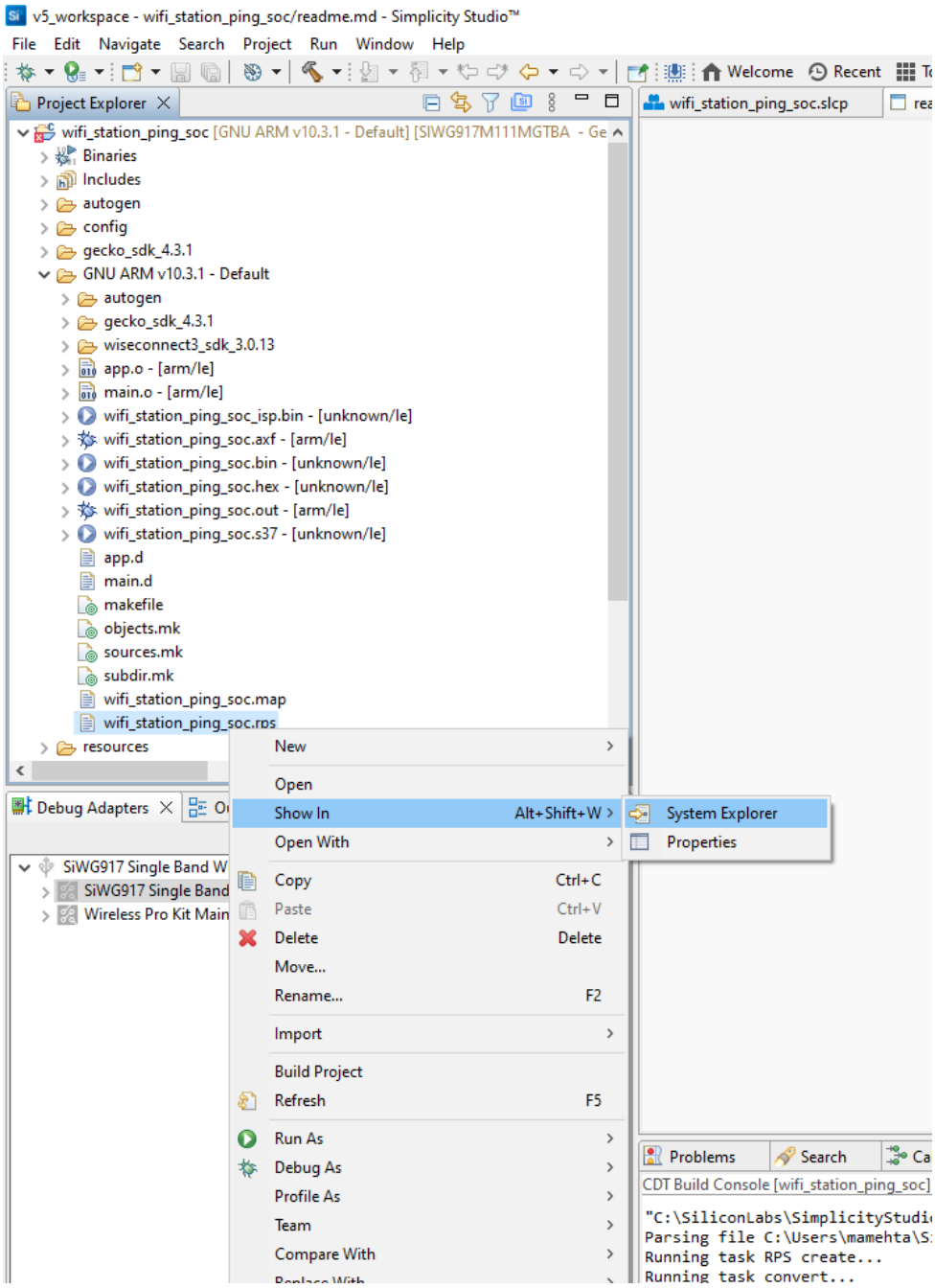


3. The application binary will be flashed on the radio board and the application will start running.
4. View the standard output or enter input data as needed. See the Console Input and Output section.

> **Note**: See the troubleshooting section in case the application fails to flash.

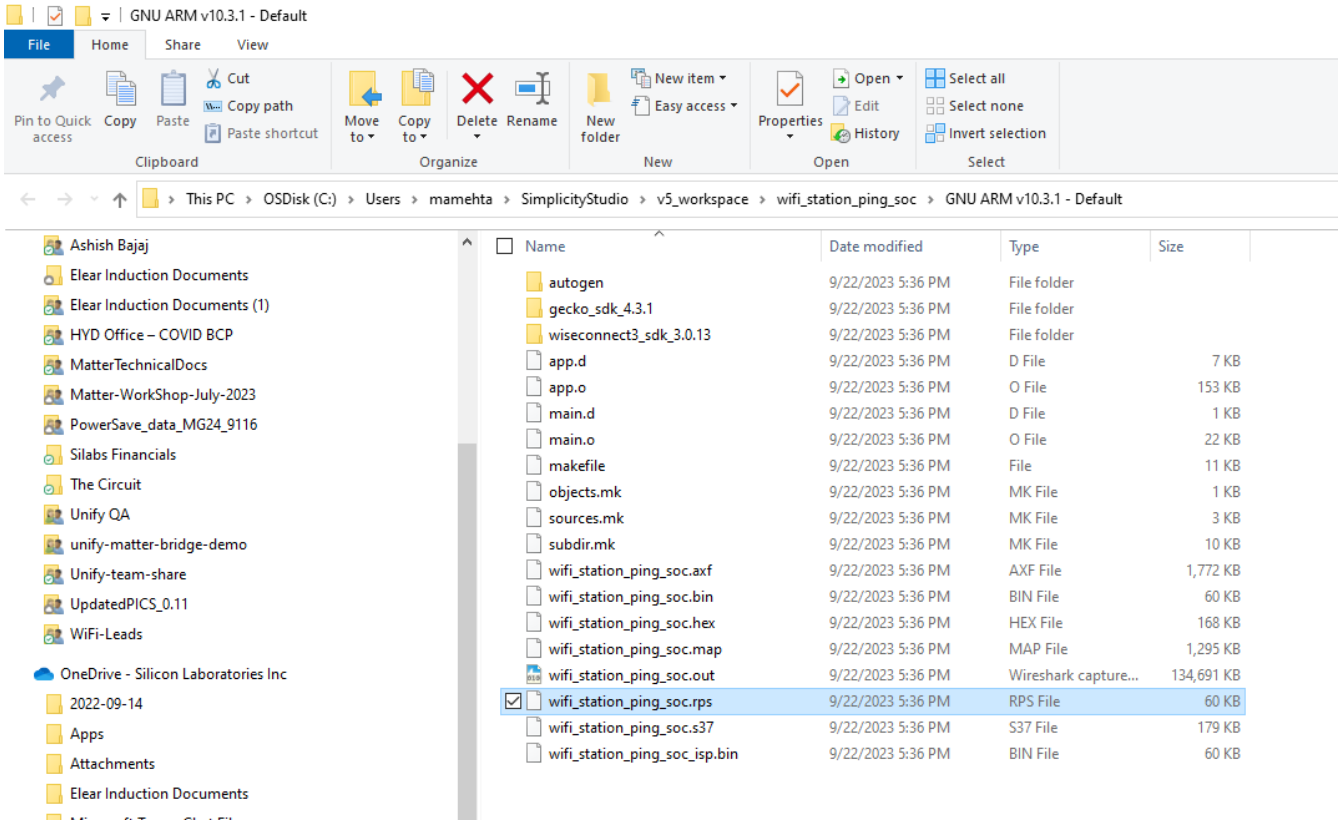
Flash an Application Binary

1. If the binary was built as described in the Build an Application section, a file with a `.rps` extension is generated. This file will be available under **GNU ARM vXX.x.x - Default** in the users workspace. To see its location in your computer's file system, right-click on the `.rps` file and select **Show In > System Explorer**.
   - Alternatively, you may have obtained a `.rps` file through another means such as someone else building and providing the binary to you.



2. The instructions to flash the application binary are *almost* the same as those for flashing an SiWx91x firmware file, except for the point noted below:
   - Instead of choosing the SiWx91x firmware file, select the `.rps` you obtained in step 1 above.

> **Note**: If you don't see `.rps` files in the sub-folder, select **All files** in the file filter field in the Browse dialog.



Troubleshoot an Application Flash Failure

The application may fail to flash for one of the following reasons:

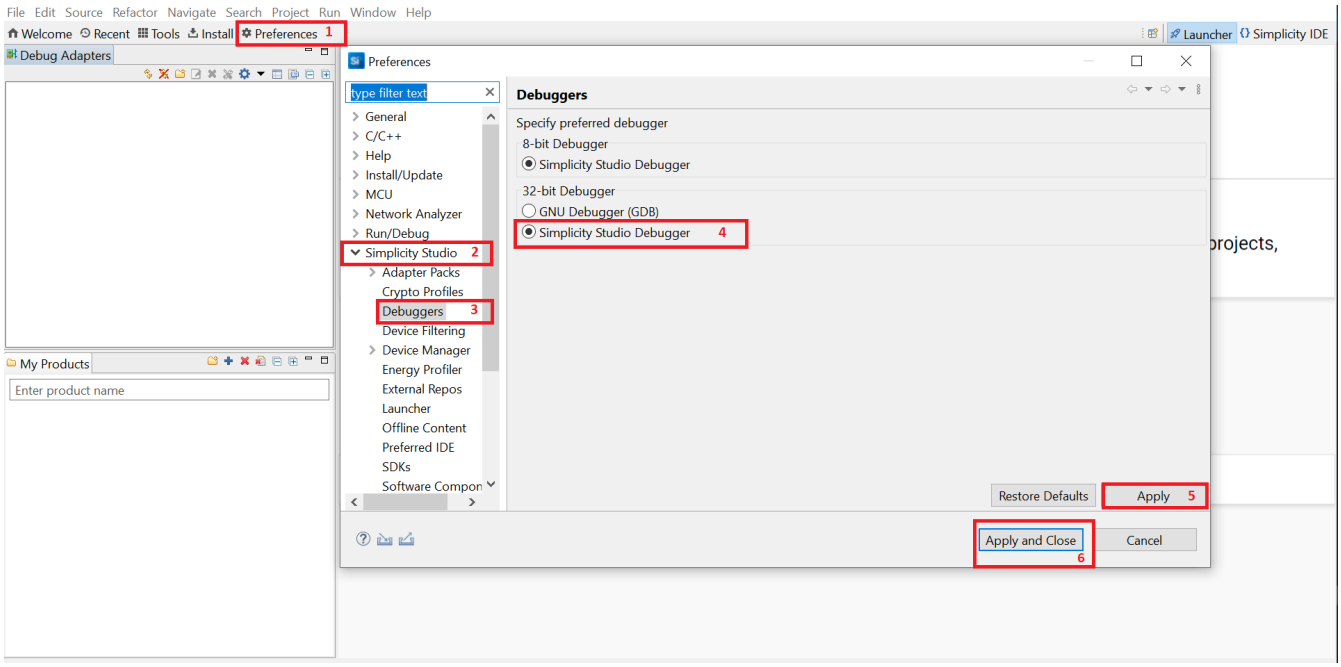
- Error code **102** is displayed in the logs, indicating that **ISP** mode is enabled. Try the following steps:
  - Press and hold the **ISP** button on the radio board.
  - Press and release the **RESET** button on the WPK board.
  - Release the **ISP** button on the radio board.
  - Retry flashing the application.
- "Could not connect debugger. Could not connect to target device" is displayed in the logs, indicating that the application processor is in a low power state with no flash access. Try the same steps as those described above for error **102**.
- Unknown reason - try re-launching Simplicity Studio.
- For additional tips, see the Troubleshoot Firmware Update Failure section. Follow the suggested steps and then, instead of retrying the connectivity firmware update, retry flashing the application.

Configure the Debugger

> **Note**: Application debugging currently does not work. This is a known issue and has two alternative workarounds:
>
> - Configure the Simplicity Studio Debugger (recommended)
> - Configure the GDB Debugger

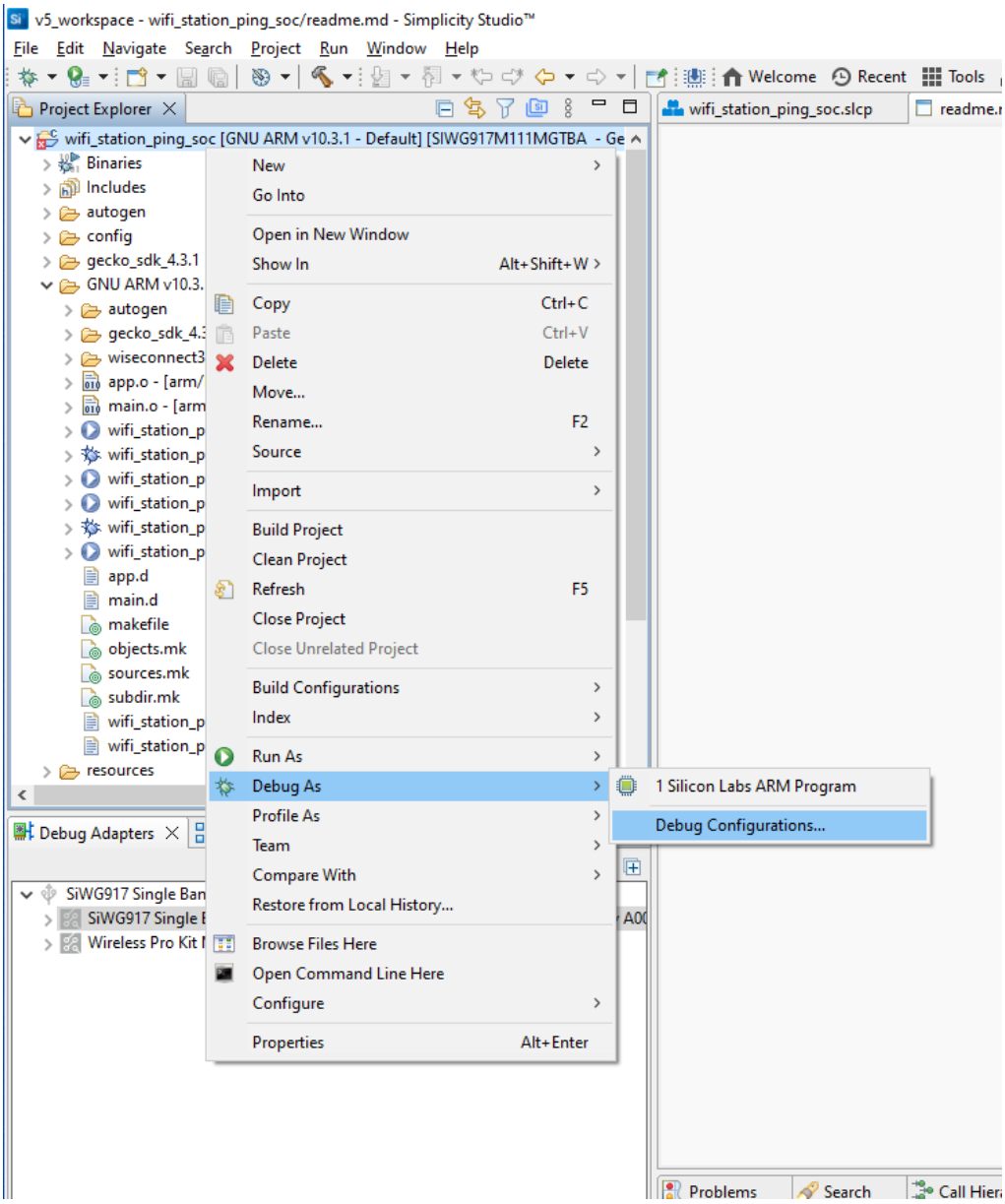
Configure the Simplicity Studio Debugger

1. Open Simplicity Studio.
2. Click **Preferences**.
3. In the **Preferences** window, select **Simplicity Studio > Debuggers**.
4. Select **Simplicity Studio Debugger** and click **Apply**.

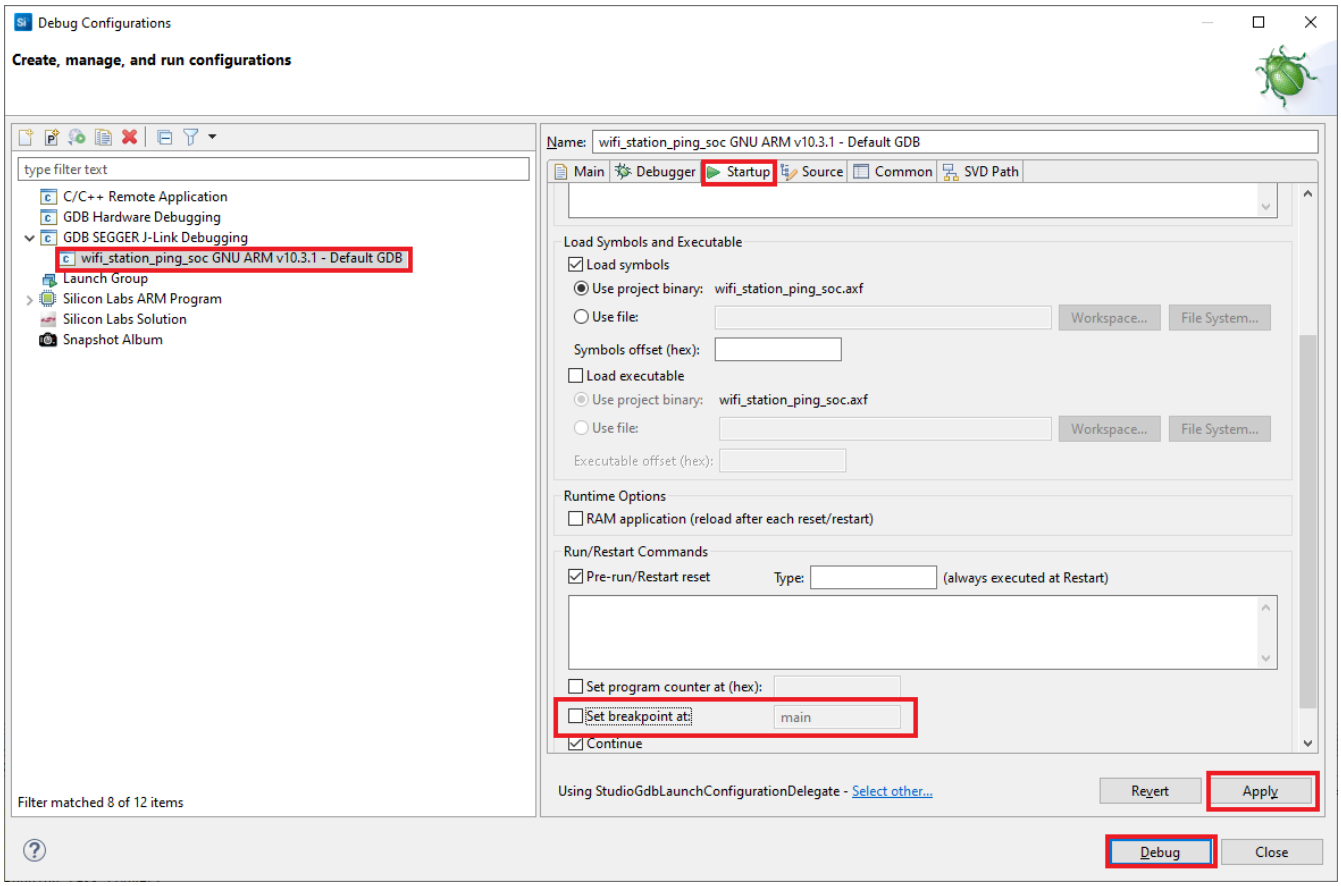


Configure the GDB Debugger

1. In the **Project Explorer** pane, right-click on your project name and select **Debug As > 1 Silicon Labs ARM Program**.
2. Studio will switch to debug mode and fail to halt execution at `main()` as described in the debugging section.
3. Close the **Debug** pane.
4. In the **Project Explorer** pane, right-click on your project name and select **Debug As > Debug Configurations**.

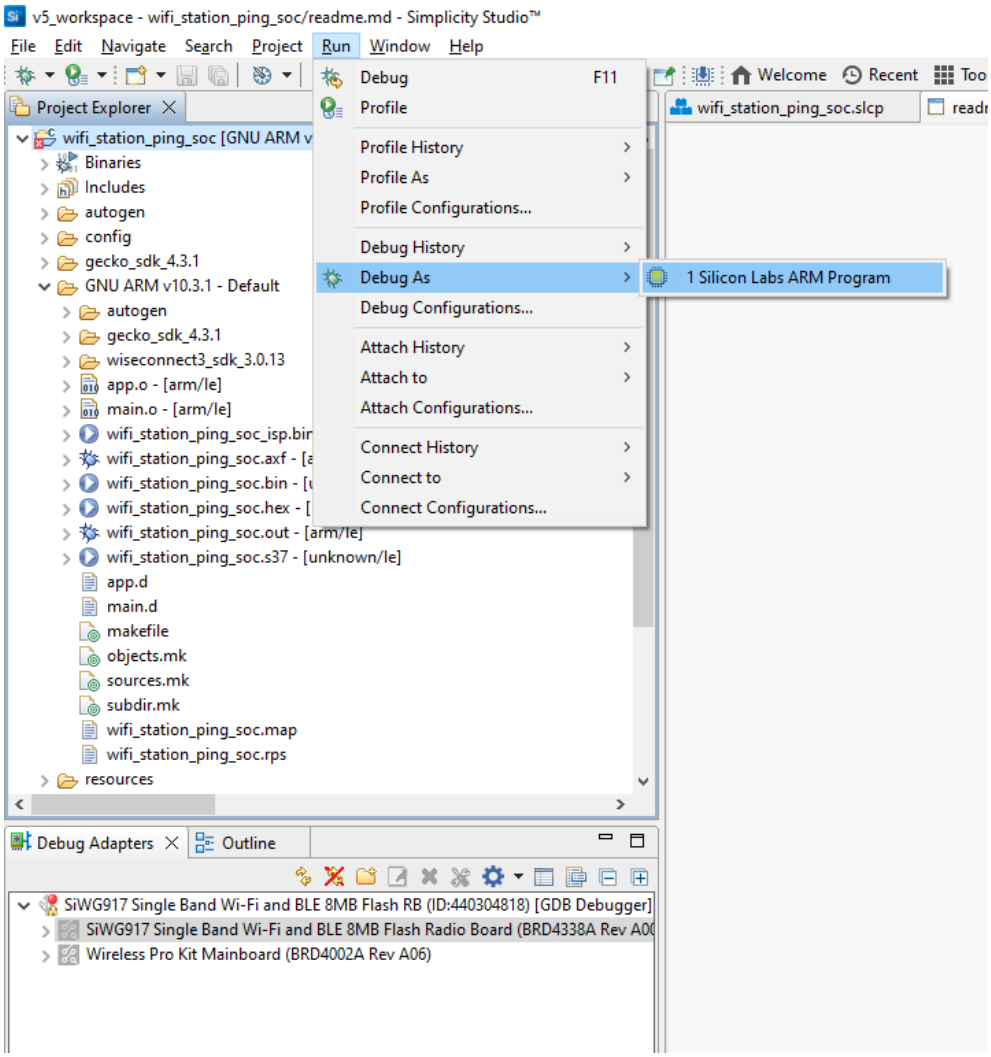


5. Under **GDB SEGGER J-Link Debugging**, select your project name.
6. Select the **Startup** tab.
7. Unselect the **Set breakpoint at** field.
8. Click **Apply** and click **Debug**.

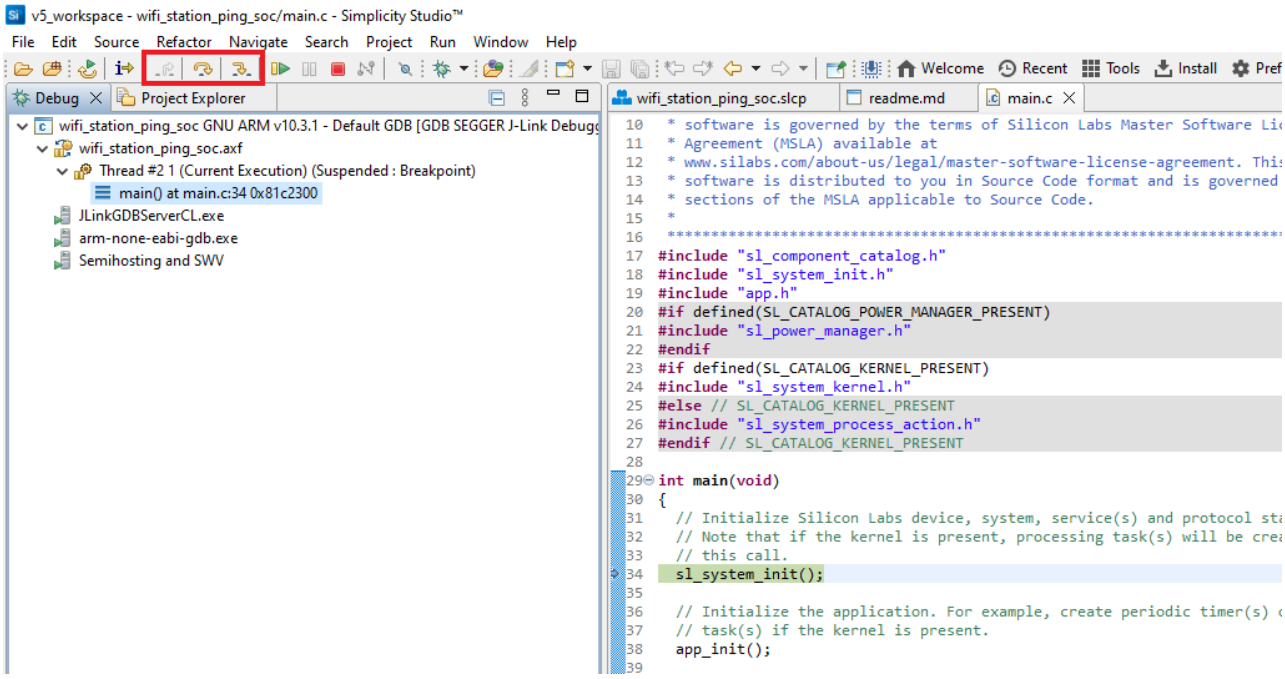


Debug an Application

1. In the **Project Explorer** pane, select your project name.
2. From the menu, select **Run > Debug As > 1 Silicon Labs ARM Program**.



3. Studio will switch to debug mode and halt execution at the `main()` function in your application.
4. Add a break point in the desired location of the code and click the **Resume** button (having an icon with a rectangular bar and play button).
5. Execution will halt at the break point.
6. Use the following debug functions to direct the execution of the code:
   - **Step In** button (having an icon with a arrow pointing between two dots).
   - **Step Over** button (having an icon with an arrow going over a dot).
   - **Step Out** button (having an icon with an arrow pointing out from between two dots).

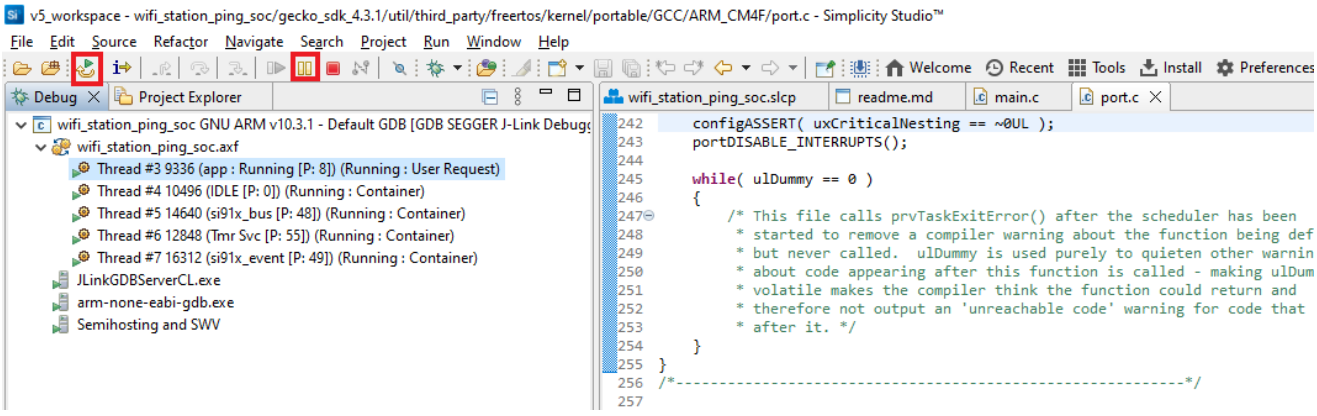


7. View the standard output or enter input data as needed. See the Console Input and Output section.

Troubleshoot an Application Debug Failure

The application may fail to enter the debug mode due to one of the following reasons:

- Studio failed to halt execution at the `main()` function. Perform the following steps before trying again:
  - Click the **Suspend** button (having a pause button icon).
  - Click the **Reset** button (having an icon of a play button with an arrow underneath).

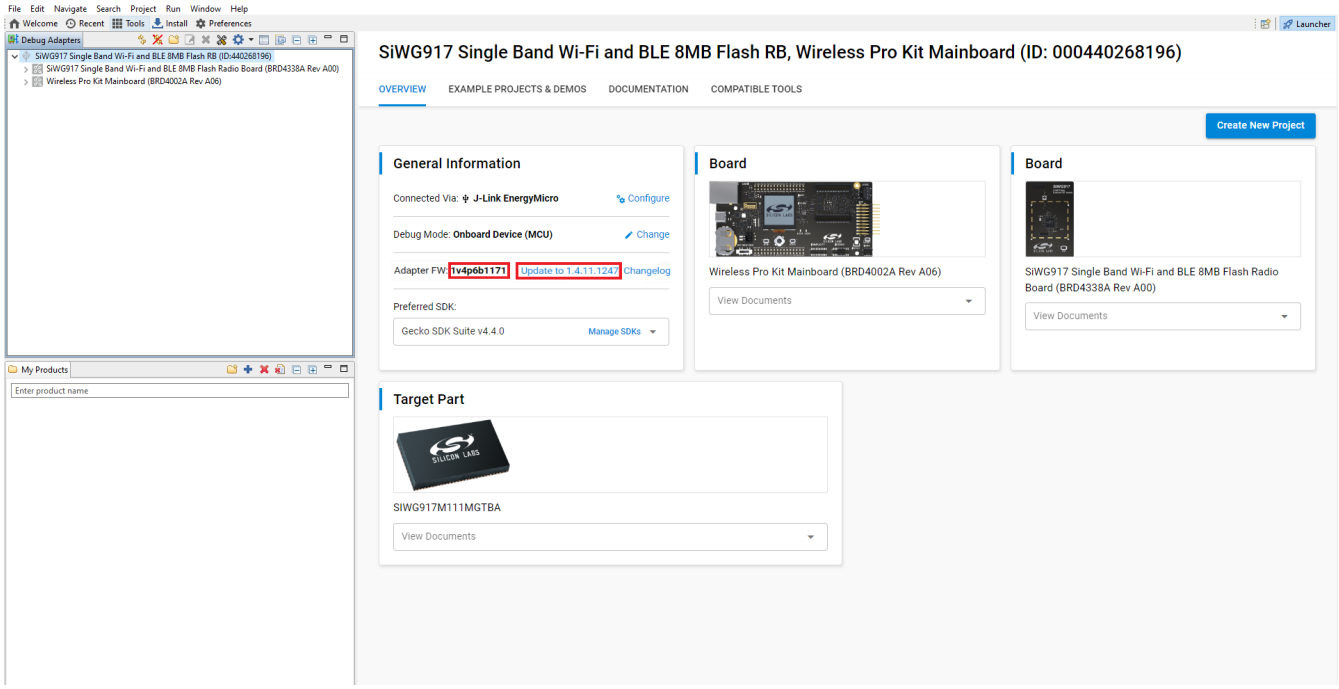


  - Retry debugging the application.
- For additional tips, see the Troubleshoot an Application Flash Failure section. Follow the suggested steps and then, instead of retrying the flashing of the application, retry debugging the application.

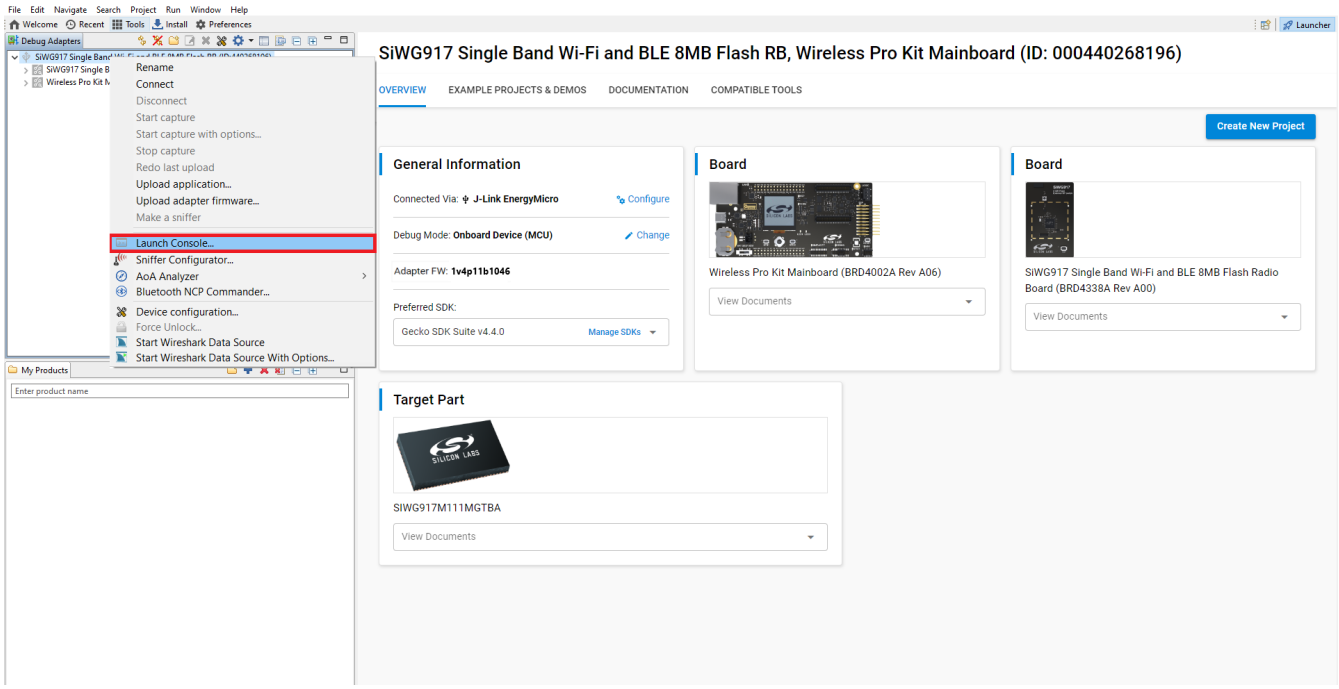
Console Input and Output

1. Connect your WPK board to your computer.
2. Open Simplicity Studio.
3. In the **Debug Adapters** pane, select your WPK board.

4. The **Adapter FW** field shows your WPK board's current firmware version, similar to *1vnpxxbyyy*, where **n** is the major version, **xx** is the patch version number and **yyy** is the build number.
5. Click **Update to 1.4.xx.yyyy** if the version is before **1v4p10b215**, or in other words:
    - Major version = 4 and one of the following is true:
        - patch < 10 or
        - patch = 10 and build < 215



6. The firmware will be upgraded on your WPK board.
7. In the **Debug Adapters** pane, right-click on your radio board and click **Launch Console**.

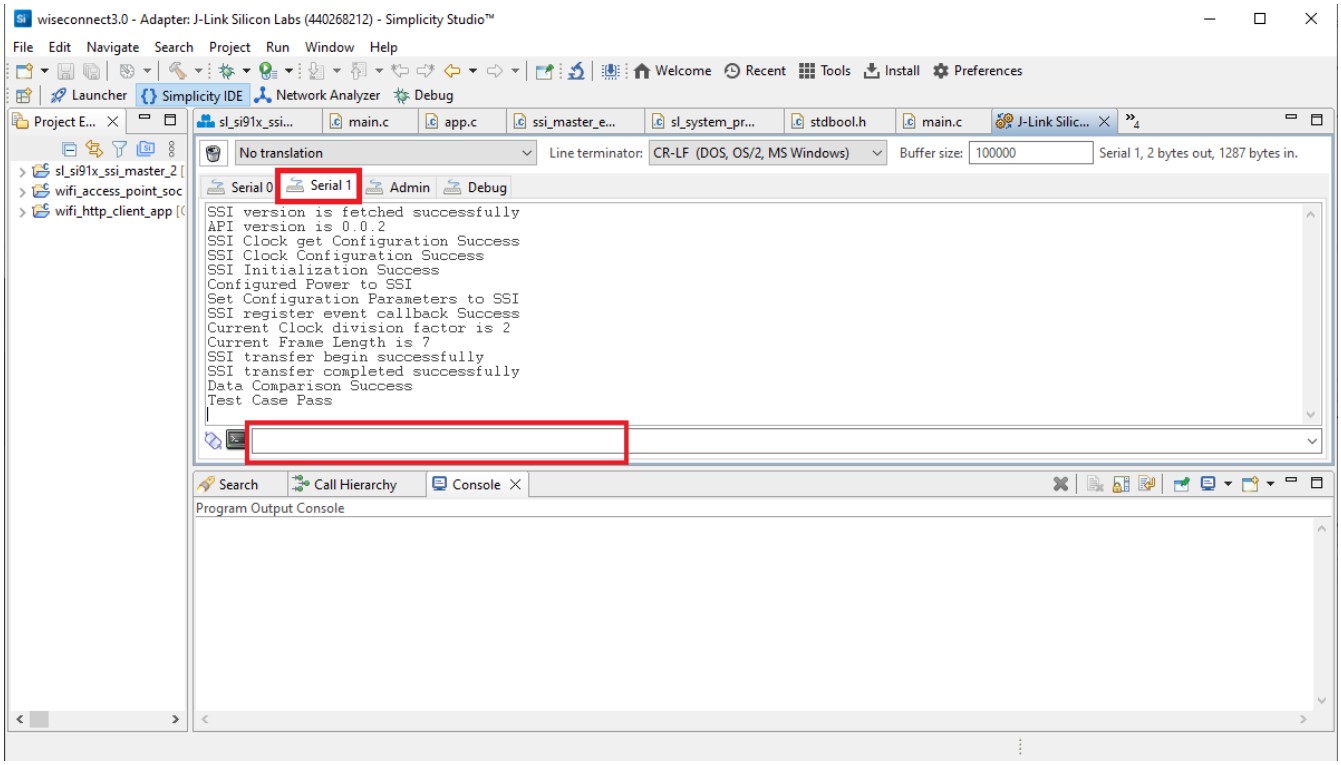


8. Select the **Admin** tab and press **Enter** to see the `WPK>` prompt.

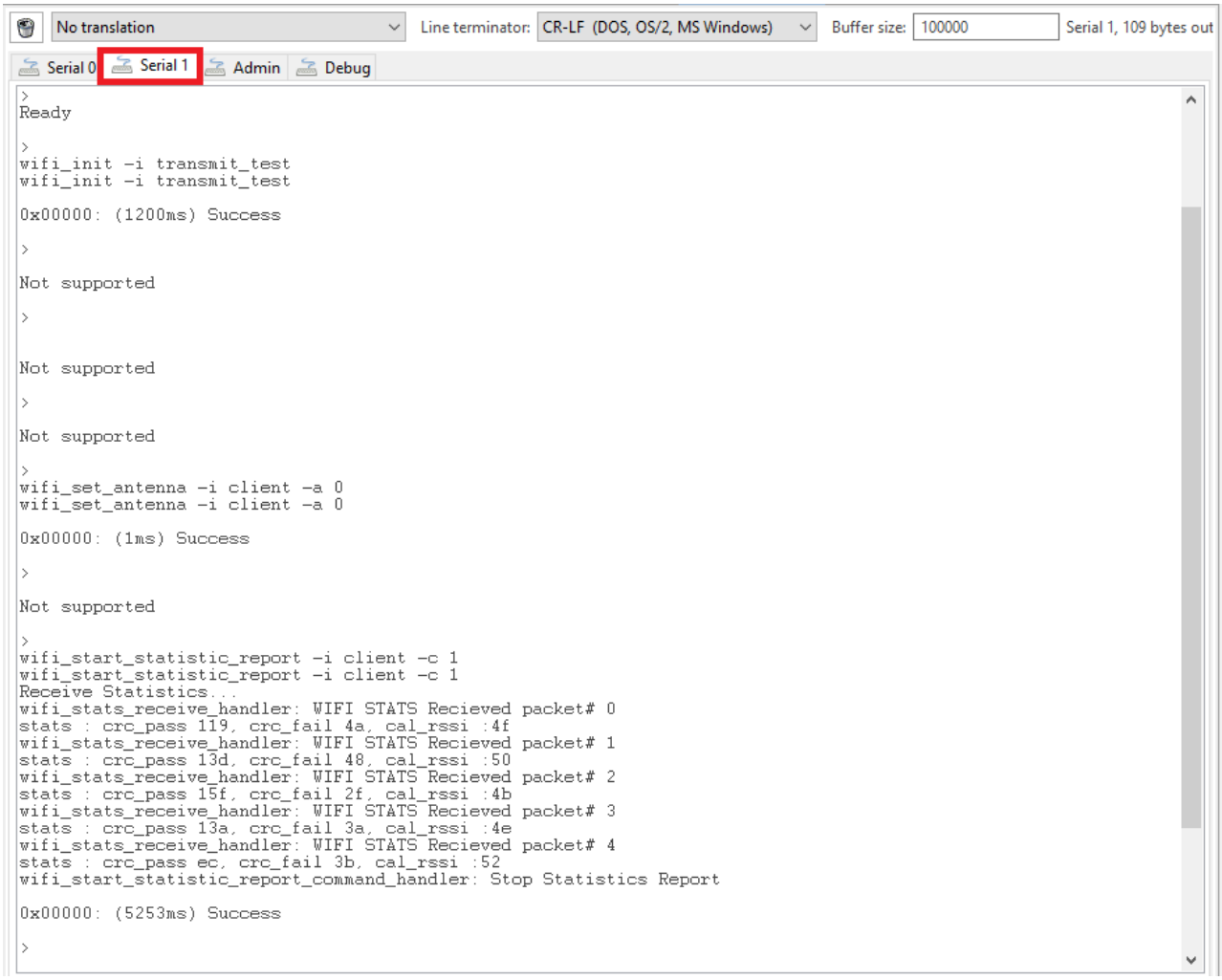
9. Enter the following command to set the baud rate:

```
pti config 0 vuart 115200
```

10. Select the **Serial 1** tab.
11. Place the cursor inside the text input field and hit **Enter**.
12. Console output will start getting displayed in the **Serial 1** tab.



13. Console input can be entered and sent to the device.



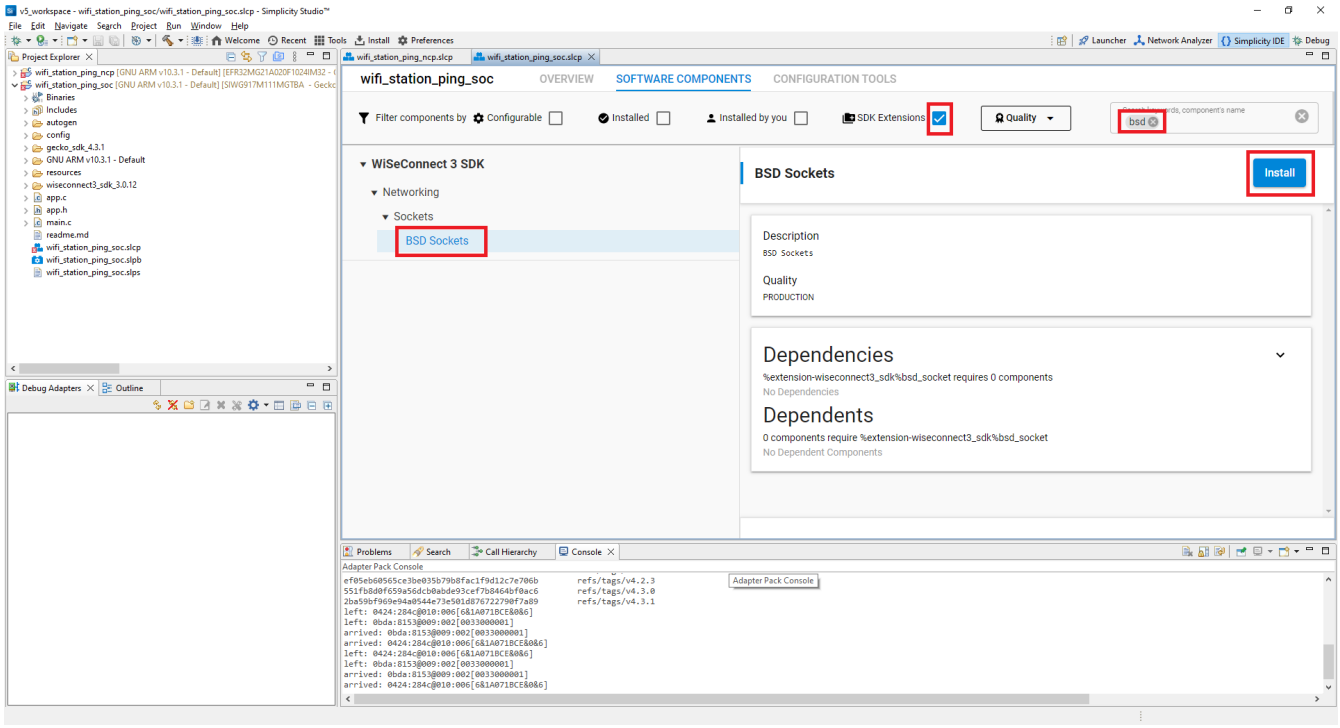
Customize Application Components

Simplicity Studio allows you to add or remove functional components in your application, such as BSD Sockets.

> **Note**: For information about the functional components available with WiSeConnect SDK v3.x, see the Application Components section.

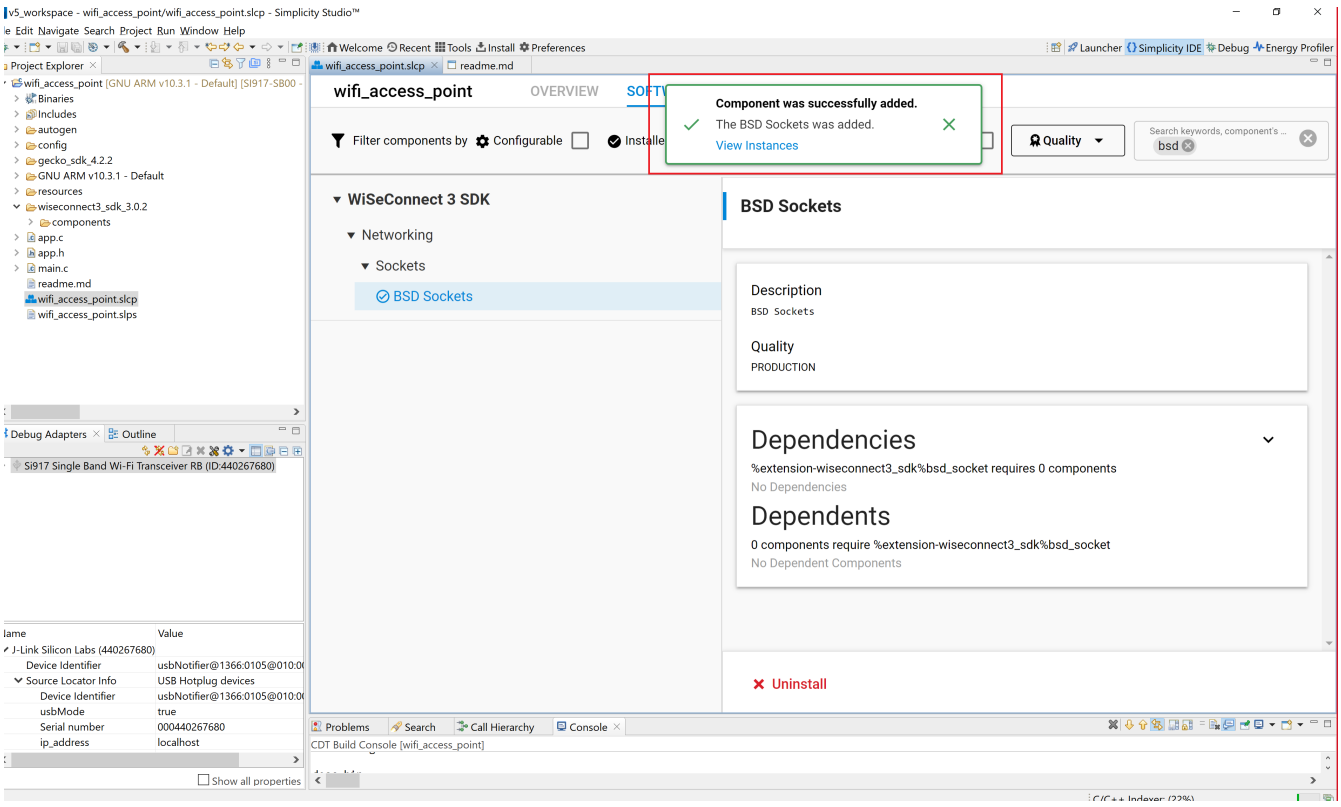
Add a Component

1. In the **Project Explorer** pane, double-click the **project_name.slcp** file.
2. Select the **SOFTWARE COMPONENTS** tab.
3. Select the **SDK Extensions** filter.
4. Browse or search for and select the component that you want to install.
5. Click **Install**.



> **Note**: Image is for illustration only. Component details shown may be outdated.

6. Studio will add the component and display a success message.

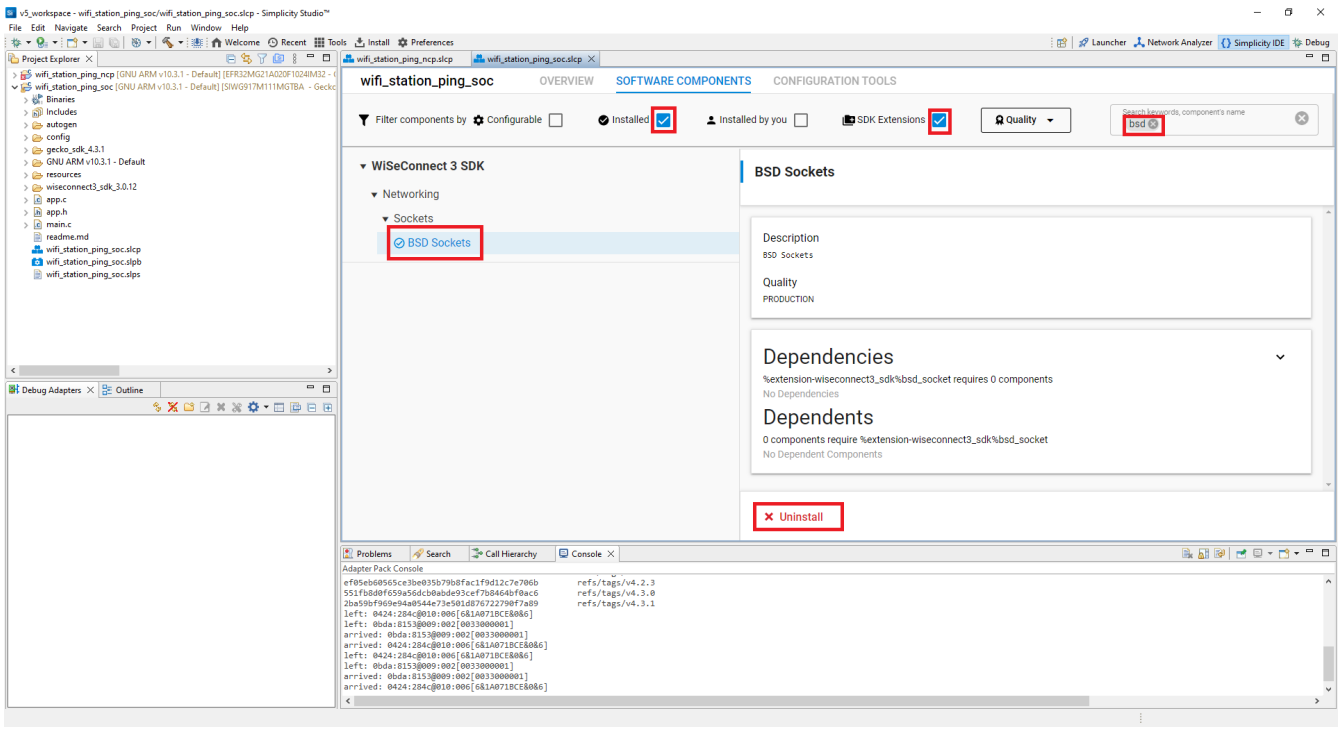


> **Note**: Image is for illustration only. Component details shown may be outdated.

> **Note**: You may use the Component Editor to configure a component after adding it or to configure other components in your example.

Remove a Component

1. View the list of **WiSeConnect 3** extension components by following steps 1-3 of the previous section.
2. Select the **Installed** filter to view the components you have installed.
3. Browse or search for and select the component you want to remove.
4. Select the component and click **Uninstall**.



> **Note**: Image is for illustration only. Component details shown may be outdated.

Overview

Preprocessor Build Settings

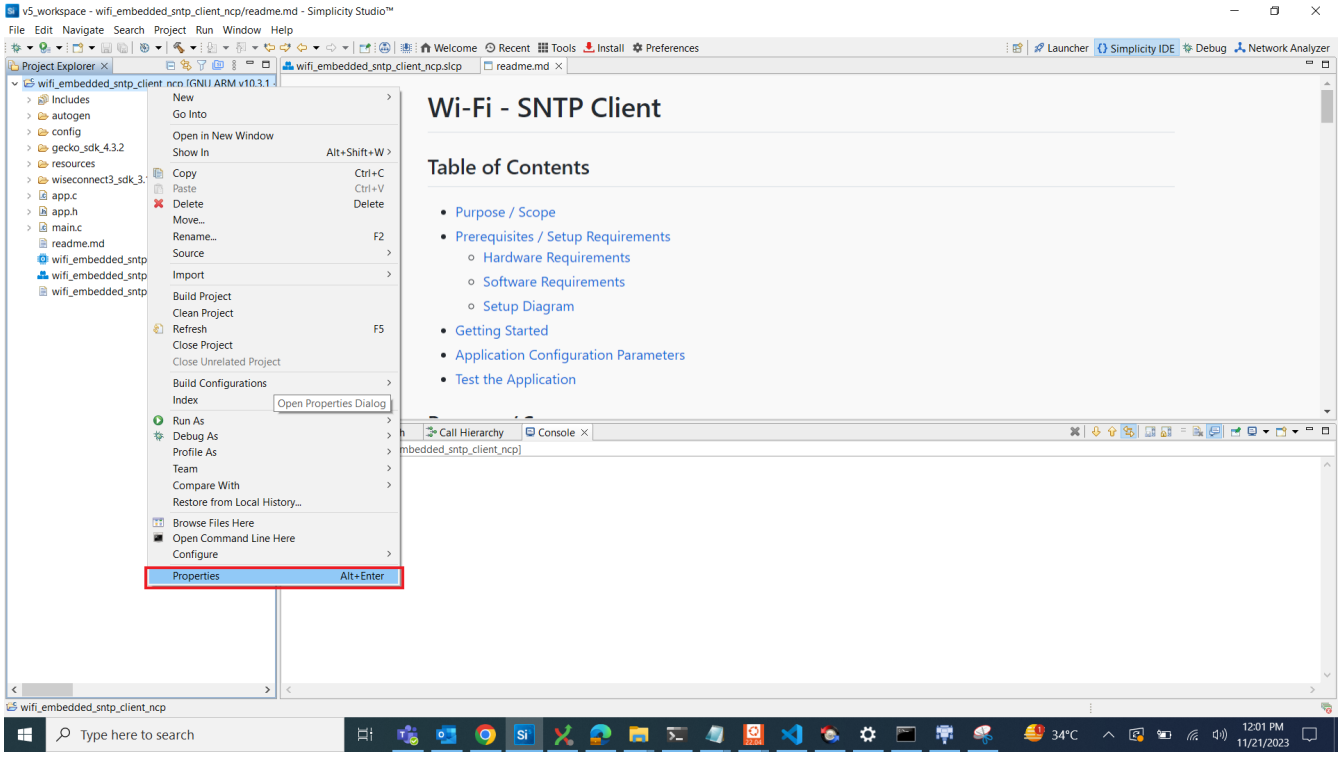
This section covers the proprocessor macros that are available as build settings with the WiSeConnect™ SDK v3.x in Simplicity Studio for the SiWx91x™ host in System-on-chip (SoC) mode and the EFR32™ host in Network Co-processor (NCP) mode.

Configure Preprocessor Build Settings

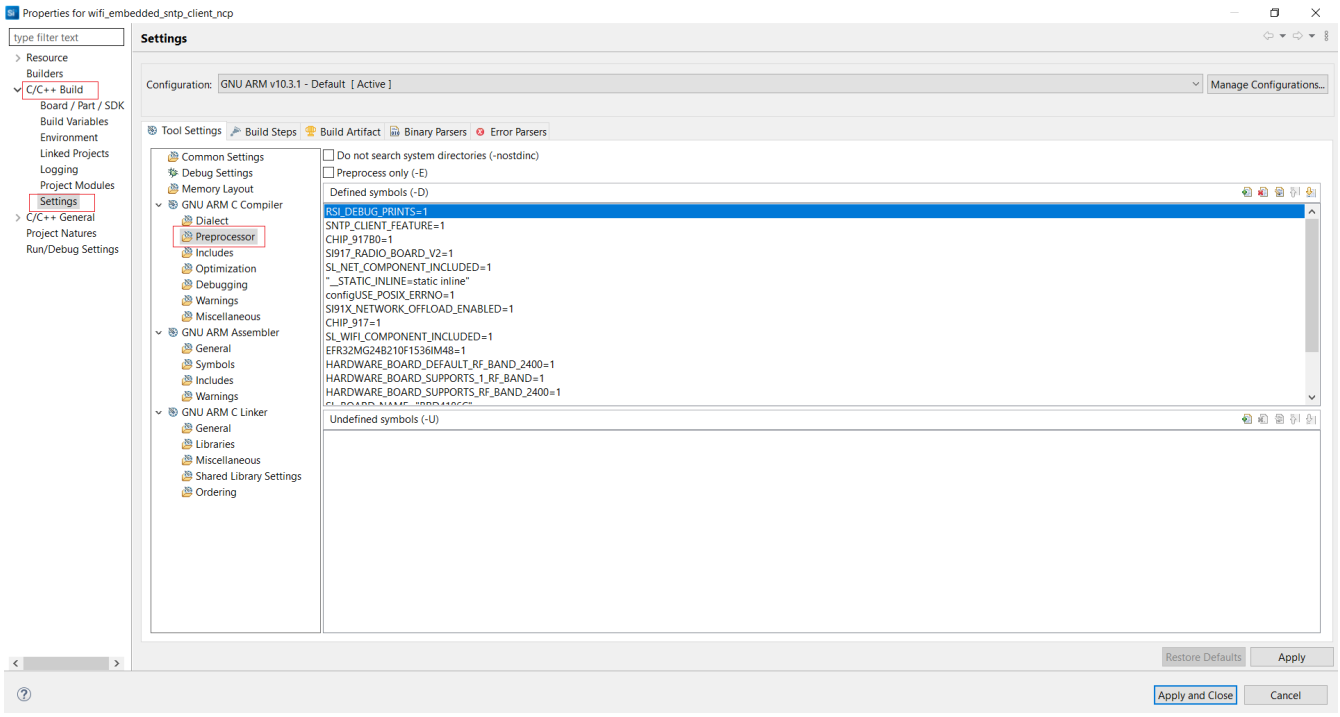
Configure Preprocessor Build Settings

This section describes how to configure the preprocessor build settings available with the WiSeConnect™ SDK v3.x with the SiWx91x™ and EFR32™ hosts.

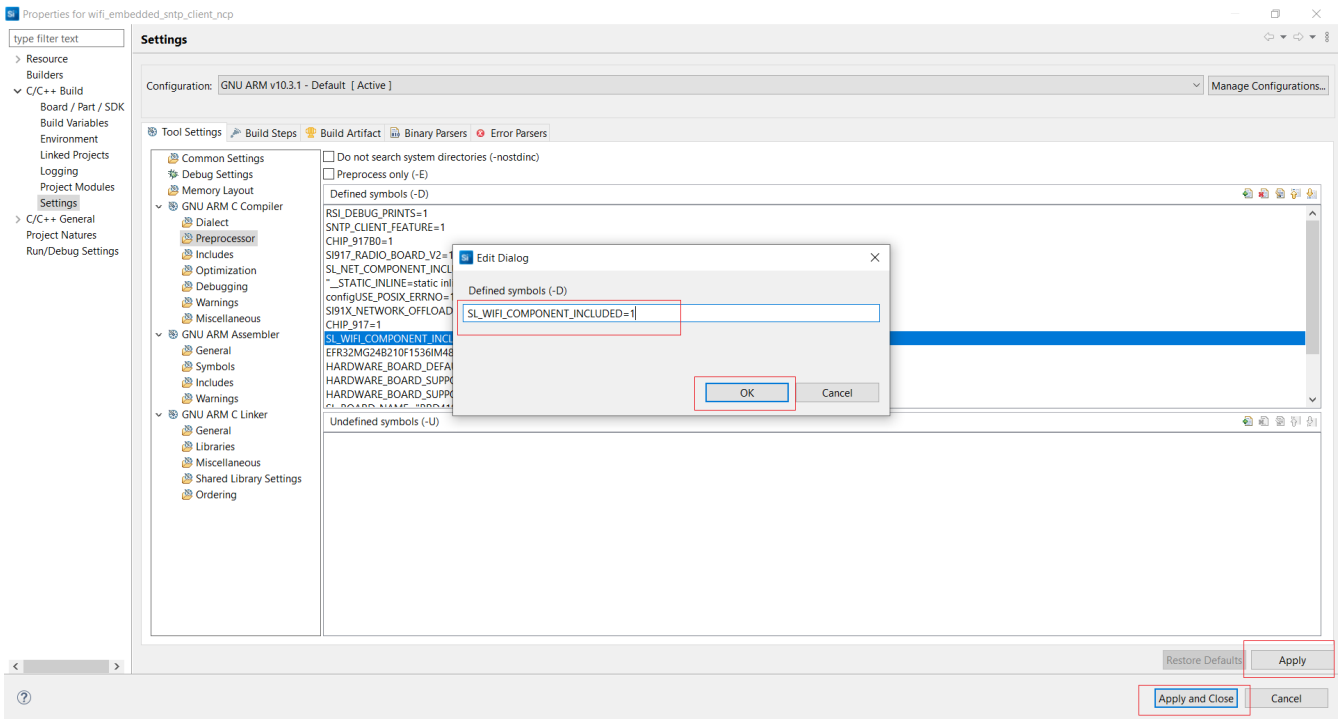
1. Open Simplicity Studio and log in.
2. If you haven't already created a project, create it by following the instructions in the relevant guide:
   - For the SiWx91x host, see the Developing for SiWx91x Host guide.
   - For EFR32 hosts, see the Getting Started with EFR32 Host in NCP Mode guide.
3. Click on the **Simplicity IDE** tab.
4. In the **Project Explorer** pane, right-click on your project name and select **Properties**.



5. In the **Settings** window, select **C/C++ Build > Settings > GNU ARM C Compiler > Preprocessor**.
6. The list of preprocessor macros and their values will be displayed.



7. Double-click on the preprocessor macro you wish to edit.
8. A dialog will open for editing the value.
9. Enter the new value and click **OK**.
10. The dialog will close.
11. Click **Apply** and **Apply and Close**.



List of Preprocessor Build Settings

List of Preprocessor Build Settings

This section provides the list of preprocessor macros that can be edited in the build settings and are available with the WiSeConnect™ SDK v3.x with the SiWx91x™ and EFR32™ hosts.

Some macros shown in the preprocessor build settings window are not listed below if they cannot be modified by the user.

Macros are listed in alphabetical order.

Macro Name and Default Value	Valid Values	Description
SL_SI91X_MCU_ALARM_BASED_WAKEUP=1	1 to enable, 0 to disable	Enabling or disabling the alarm-based wake up method. Only used when the **Si91x Cortex-M4 Powersave** component is enabled.
SL_SI91X_MCU_BUTTON_BASED_WAKEUP=1	1 to enable, 0 to disable	Enabling or disabling the button-based wake up method. Only used when the **Si91x Cortex-M4 Powersave** component is enabled.
SL_SI91X_MCU_WIRELESS_BASED_WAKEUP=1	1 to enable, 0 to disable	Enabling or disabling the wireless-based wake up method. Only used when the **Si91x Cortex-M4 Powersave** component is enabled.
SL_SI91X_PRINT_DBG_LOG=1	1 to enable, 0 to disable	Enabling or disabling `LOG_PRINT` support in Bluetooth Low Energy (BLE) mode and Wi-Fi + BLE mode
SLI_SI91X_CONFIG_WIFI6_PARAMS=1	1 to enable, 0 to disable	Enabling or disabling the Wi-Fi 6 protocol. Only used when the **Wi-Fi** component is enabled.
SLI_SI91X_EMBEDDED_MQTT_CLIENT=1	1 to enable, 0 to disable	Enabling or disabling the Message Queue Telemetry Transport (MQTT) client functionality of the SiWx91x chipset. Only used when the **MQTT Client** component is enabled.
SLI_SI91X_ENABLE_IPV6=1	1 to enable, 0 to disable	Enabling or disabling Internet Protocol version 6 (IPv6) support.
SLI_SI91X_INTERNAL_HTTP_CLIENT=1	1 to enable, 0 to disable	Enabling or disabling the Hyper-Text Transfer Protocol (HTTP) client functionality of the SiWx91x chipset with asynchronous event support. Only used when the **HTTP Client** component is enabled.

Macro Name and Default Value	Valid Values	Description
SLI_SI91X_INTERNAL_SNTP_CLIENT=1	1 to enable, 0 to disable	Enabling or disabling the Simple Network Time Protocol (SNTP) client functionality of the SiWx91x chipset. Only used when the **SNTP Client** component is enabled.
SLI_SI91X_LWIP_HOSTED_NETWORK_STACK=1	1 to enable, 0 to disable	Enabling or disabling the lightweight Internet Protocol (LwIP) network stack on the external microconroller unit (MCU) host in network co-processor (NCP) mode.
SLI_SI91X_LWIP_NETWORK_STACK=1	1 to enable, 0 to disable	Enabling or disabling the lightweight Internet Protocol (LwIP) network stack on the application processor in system-on-chip (SoC) mode.
SLI_SI91X_MCU_ENABLE_PSRAM_FEATURE=1	1 to enable, 0 to disable	Enabling or disabling the sleep and wakeup functionality of the pseudo-static random-access memory (PSRAM) of the SiWx91x chipset. This option is only available on radio boards that include PSRAM.
SLI_SI91X_MCU_FW_UPGRADE_OTA_DUAL_FLASH=1	1 to enable, 0 to disable	Enabling or disabling the firmware update of the application processor on dual flash radio boards using the OTAF protocol. This option is only available on dual flash boards.
SLI_SI91X_MCU_INTR_BASED_RX_ON_UART=1	1 to enable, 0 to disable	Enabling or disabling the receiving of data over the Universal Asynchronous Receiver-Transmitter (UART) interface using interrupts.
SLI_SI91X_SOCKETS=1	1 to enable, 0 to disable	Enabling or disabling socket programming support. Only used when the **Internal Network Stack** component is enabled.

Overview

Migration Guide

This is a guide for updating an existing application using the WiSeConnect™ SDK v2.x to a v3.x application.

This guide describes the changes made in the WiSeConnect SDK v3.x to simplify the API and make it modular.

It also provides a migration example, and the steps for updating a v2.x application to a v3.x application.

Overview

Significant Changes between WiSeConnect™ SDK v2.x and v3.x

This section describes the significant changes made in WiSeConnect™ SDK v3.x, which impact the way applications are written and determine the steps that are followed to update an existing v2.x application to a v3.x application.

This is not an exhaustive list of all the changes between v2.x and v3.x. The sections later in this migration guide provide the list of those changes in detail.

At a high level, WiSeConnect SDK v3.x differs from v2.x in the following aspects:

- Simplifying the way configuration settings are updated.
- Following Silicon Labs' standards for application programming interface (API) signatures and behavior.
- Providing industry-standard APIs for socket programming, following the popular standards for Internet-of-Things (IoT) sockets and Berkeley Software Distribution (BSD) sockets.
- Facilitating the passing of application context data in API calls and receiving that data back in callbacks.
- Using the same API signature to invoke either synchronous or asynchronous processing in the application.

Functionality Breaks

Functionality Breaks between SDK v2.x and v3.x

The following are the features in WiSeConnect™ SDK v2.x that are not supported in v3.x:

- Synchronous scanning of Wi-Fi Service Set Identifiers (SSIDs) - the SDK supports asynchronous scanning.
- Bare Metal applications - applications must run on the Real-Time Operating System (RTOS).
- File Transfer Protocol (FTP)
- Compiling applications with STM35™ and RT595™ processors as external hosts.

Build System

Build System

This section compares the various ways to build applications in WiSeConnect™ SDK v2.x and 3.x.

Build Method	Mode	Host	2.x Support	3.x Support
Simplicity Studio	SoC	SiWx91x™	Yes	Yes
Simplicity Studio	NCP	EFx32™	Yes	Yes
Keil	NCP	STM32™	Yes	No
IAR	NCP	RT595™	Yes	No

Folder Structure

Folder Structure

This section compares the folders in the WiSeConnect™ SDK v2.x and v3.x GitHub repositories.

2.x Folder	Description	3.x Folder	Description
		components	WiSeConnect 3.x source code and Simplicity Studio component definitions
connectivity_firmware	Network processor firmware images	connectivity_firmware	Same as v2.x
docs	All v2.x documentation including release notes, getting started, API reference, and so on.	docs	Release notes, software reference manual, and app notes for v3.x
examples	Application examples	examples	Same as v2.x
platforms	Configuration settings and Simplicity Studio component definitions for platforms (EFR™, EFM™, etc.) and modes (NCP, SoC, etc.)		
resources	See sub-folders below	resources	See sub-folders below
|-- certificates	Certificates	|-- certificates	Same as v2.x
|-- component	Simplicity Studio component definition for certificates		
		|-- defaults	Default configuration header files
		|-- lwip_defaults	Default configurations for the Lightweight IP (LwIP) networking layers used in hosted mode
		|-- other	Miscellaneous declarations and configuration settings
|-- scripts	Python test scripts for examples.	|-- scripts	Same as v2.x
sapi	WiSeConnect 2.x source code and Simplicity Studio component definitions		
third_party	Third-party software such as Amazon web services internet-of-things (AWS IoT) SDK, Azure SDK, and others	third_party	Same as v2.x but the third-party software included is different
utilities	Common utility functions for internet protocol version 6 (IPv6), logging, certificate handling, and others	utilities	Same as v2.x

Event Handling

Event Handling

WiSeConnect™ SDK v3.x utilizes **two threads** for handling events:

- The `bus_thread` sends commands to the SiWx91x™ network processor and processes the responses sent back by the network processor.
- The `event_thread` performs asynchronous processing for commands and events.

WiSeConnect SDK v2.x utilizes **a single thread** (`wireless_driver_task`) to send commands, process responses, and perform asynchronous processing.

Queue Management

Queue Management

WiSeConnect™ SDK v3.x has a total of **12 queues**:

- 5 for sending commands to the network processor, such as a command to scan for access points (APs).
- 5 for processing command responses from the network processor.
- 2 for asynchronous processing such as the handling of asynchronous command responses.

WiSeConnect SDK v2.x has **6 queues**:

- 1 for sending commands.
- 5 that are shared for the purpose of processing command responses, as well as asynchronous processing.

Asynchronous API Invocation

Asynchronous API Invocation

WiSeConnect™ SDK v3.x has common API functions for both synchronous and asynchronous processing, while v2.x has different functions for synchronous and asynchronous processing.

As an example, the difference between access point connection APIs is illustrated below:

- In v2.x, the `rsi_wlan_connect_async()` function is invoked to connect to a Wi-Fi AP asynchronously, and `rsi_wlan_connect()` to connect synchronously.

```
status = rsi_wlan_connect_async(ssid, sec_type, secret_key, callback_ptr);   // Connect asynchronously
status = rsi_wlan_connect((int8_t *)SSID, SECURITY_TYPE, PSK);               // Connect synchronously
```

- In v3.x, the sl_wifi_connect() function is invoked with a `timeout_ms` parameter value of 0 to connect to a Wi-Fi access point (AP) asynchronously, and a value other than 0 to connect synchronously.

```
status = sl_wifi_connect(SL_WIFI_CLIENT_INTERFACE, &profile.config, 0);      // Connect asynchronously
status = sl_wifi_connect(SL_WIFI_CLIENT_INTERFACE, &profile.config, 15000);  // Connect synchronously
```

Error Handling

Error Handling

The APIs in WiSeConnect™ SDK v3.x return an `sl_status_t` value that contains one of the standard Silicon Labs error code values.

The APIs in WiSeConnect SDK v2.x return an `int32_t` value that contains one of the error codes defined for the 91x family of Wi-Fi chipsets.

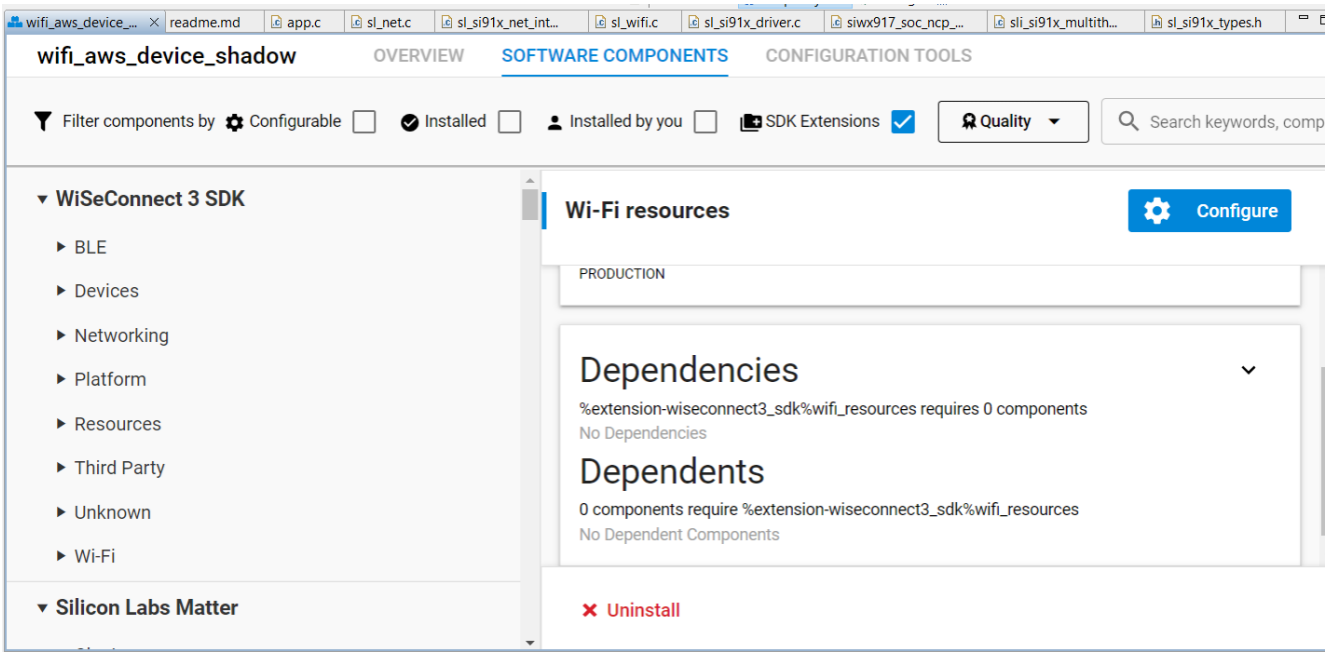
Application Components

Application Components

WiSeConnect™ SDK v3.x has revamped the component organization such that almost none of the v2.x components remain in v3.x.

The following sections provide a list of v3.x components.

The component category path is listed for each component, specifying how to navigate to the component in the **SOFTWARE COMPONENTS** tab of Simplicity Studio. Components are listed in alphabetical order of the component category path.



Common

Component Name	Component Category Path	Description
WiSeConnect3 Common Library	Common	Commonly used components
WiSeConnect3 Resources	Resources	Default values and certificates
Unity test framework	Test > Framework	Unity test framework

Wireless

Component Name	Component Category Path	Description
BLE	Device > Si91x > Wireless	API for Bluetooth Low Energy (BLE) functionality

Component Name	Component Category Path	Description
Si91x Wireless Subsystem	Device > Si91x > Wireless	API to manage the wireless subsystem in the SiWx91x™ chipset
Basic Buffer Manager	Device > Si91x > Wireless > Buffer Manager	SiWx91x memory buffer manager for malloc-based memory management
Internal Network Stack	Device > Si91x > Wireless > Network Stack	API for running the networking layers on the SiWx91x chipset
LwIP Network Stack (Hosted)	Device > Si91x > Wireless > Network Stack	Lightweight Internet Protocol (LwIP) third party library-based API for running networking layers on a host MCU
Wi-Fi	Protocol	Application programming interface (API) for Wi-Fi functionality

Network Management

Component Name	Component Category Path	Description
Network Manager	Service > Network Management	API to manage network interfaces and their behaviour
Basic Network Configuration Manager	Service > Network Management > Configuration	Basic reference implementation for managing network credentials and profiles
NVM3 Network Configuration Manager	Service > Network Management > Configuration	API for managing network credentials and profiles in non-volatile memory with key-based storage

Sockets

Component Name	Component Category Path	Description
BSD Socket API	Common	Berkeley Software Distribution (BSD) standard API for socket programming
Si91x Asynchronous Socket	Device > Si91x > Wireless > Socket	API for SiWx91x asynchronous socket programming
Si91x Socket	Device > Si91x > Wireless > Socket	API for SiWx91x socket programming
BSD Socket	Service	API for BSD-style communication over the Transport Control Protocol (TCP) and Internet Protocol (IP) networking layers
IoT Socket	Third Party	ARM Internet-of-Things (IoT) standard API for socket programming

SiWx91x MCU

The following SiWx91x MCU (application processor) components are available:

- Common MCU Components
- Peripherals
- Drivers
- Services

Common MCU components

Component Name	Component Category Path	Description
Board Configuration Header Files	N/A (no category)	Radio board configuration header files
brd4338a config	Board > Configuration	Configuration files for radio board BRD4338A
brd4342a config	Board > Configuration	Configuration files for radio board BRD4342A

Component Name	Component Category Path	Description
EFx32 - Si91x Connection Configurator	Board > Configuration	Configuration for connections between SiWx91x and EFx32 host MCU
Si91x SoC Board Configurations	Board > Configuration	Hardware configurations for SiWx91x SoC boards
BRD4338A	Board > Radio Board	Radio Board support for BRD4338A
BRD4342A	Board > Radio Board	Radio Board support for BRD4342A
Default Configuration	Device > Si91x > Memory Configuration	Default memory configuration
MCU Advanced Features and Wireless Basic Features	Device > Si91x > Memory Configuration	Configuration for random-access memory (RAM) split option 3 between processors - 256KB for network processor and 448KB for application processor.
MCU Basic Features and Wireless Advanced Features	Device > Si91x > Memory Configuration	Configuration for default RAM split (option 1) between processors - 512KB for network processor and 192KB for application processor.
MCU Medium Features and Wireless Medium Features	Device > Si91x > Memory Configuration	Configuration for RAM split option 2 between processors - 448KB for network processor and 256KB for application processor.
Si91x MCU Subsystem	Device > Si91x > MCU	Initializing the SiWx91x MCU (application processor) subsystem on startup
Debug Unit	Device > Si91x > MCU > Common	Debug functionality for the SiWx91x MCU
System Calls	Device > Si91x > MCU > Common	System call support for the SiWx91x MCU
Core	Device > Si91x > MCU > Core	Core SiWx91x MCU functionality
CMSIS Core	Device > Si91x > MCU > Core	Common Microcontroller Software Interface Standard (CMSIS) core functionality
FreeRTOS Configuration	Device > Si91x > MCU > Core	Default configuration for FreeRTOS (operating system) instance on the SiWx91x MCU
NVIC Interrupt Priorities Configuration	Device > Si91x > MCU > Core	Configuration of nested vector interrupt controller (NVIC) priorities
RAM Execution	Device > Si91x > MCU > Core	Enabling certain pre-determined components to execute from random-access memory (RAM)
RSI User Configuration	Device > Si91x > MCU > Core	User configuration for the SiWx91x MCU
UDMA Linker Configuration	Device > Si91x > MCU > Core	Linker configuration for unified direct memory access (UDMA)
ulp_mode_execution	Device > Si91x > MCU > Core	Enabling ultra low power (ULP) mode
Debug	Device > Si91x > MCU > Debug	Debug functionality for the SiWx91x application processor
SI917-SB00	Device > Si91x > MCU > Device Part > SiWG917	CMSIS device part headers for SI917-SB00

Component Name	Component Category Path	Description
SIWG917M100MGTBA	Device > Si91x > MCU > Device Part > SiWG917	CMSIS device part headers for SIWG917M100MGTBA
SIWG917M100XNTBA	Device > Si91x > MCU > Device Part > SiWG917	CMSIS device part headers for SIWG917M100XNTBA
SIWG917M110LGTBA	Device > Si91x > MCU > Device Part > SiWG917	CMSIS device part headers for SIWG917M110LGTBA
SIWG917M111MGTBA	Device > Si91x > MCU > Device Part > SiWG917	CMSIS device part headers for SIWG917M111MGTBA
SIWG917M111XGTBA	Device > Si91x > MCU > Device Part > SiWG917	CMSIS device part headers for SIWG917M111XGTBA
SIWG917M121XGTBA	Device > Si91x > MCU > Device Part > SiWG917	CMSIS device part headers for SIWG917M121XGTBA
SIWG917M141XGTBA	Device > Si91x > MCU > Device Part > SiWG917	CMSIS device part headers for SIWG917M141XGTBA
8MB Flash (External)	Device > Si91x > MCU > Flash	Adding 8MB external flash to the SiWx91x SoC
8MB PSRAM (External)	Device > Si91x > MCU > Flash	Adding 8MB external pseudo-static random-access memory (PSRAM) to the SiWx91x SoC
No External Flash	Device > Si91x > MCU > Flash	Excluding the external flash from the SiWx91x SoC
No External PSRAM	Device > Si91x > MCU > Flash	Excluding the external PSRAM from the SiWx91x SoC
HAL	Device > Si91x > MCU > HAL	Hardware abstraction layer (HAL) for the SiWx91x MCU
Soft Reset	Device > Si91x > MCU > HAL	Soft reset functionality for the SiWx91x MCU
SL errno	Device > Si91x > MCU > Service	Enabling thread-safe error numbers
GCC Toolchain Support	Device > Si91x > MCU > Toolchain	Support for the Gnu's Not UNIX (GNU) Compiler Collection (GCC) toolchain
SIWG917Y111MGAB	Platform > Device > Si91x > MCU > Family > SIWG917Y	CMSIS device part headers for SIWG917Y111MGAB
SIWG917Y111MGNB	Platform > Device > Si91x > MCU > Family > SIWG917Y	CMSIS device part headers for SIWG917Y111MGNB

Peripherals

Component Name	Component Category Path	Description
ADC	Device > Si91x > MCU > Peripheral	API for the analog-to-digital-converter (ADC) peripheral
ADC Instance	Device > Si91x > MCU > Peripheral	ADC peripheral instance
Calendar	Device > Si91x > MCU > Peripheral	API for the calendar peripheral
Config Timer	Device > Si91x > MCU > Peripheral	API for the config timer peripheral
DMA	Device > Si91x > MCU > Peripheral	API for the direct memory access (DMA) peripheral
Efuse	Device > Si91x > MCU > Peripheral	API for the e-fuse peripheral
GPIO	Device > Si91x > MCU > Peripheral	API for the general-purpose input/output (GPIO) ports

Component Name	Component Category Path	Description
GSPI	Device > Si91x > MCU > Peripheral	API for the Generic Serial Peripheral Interface (GSPI) peripheral
I2C	Device > Si91x > MCU > Peripheral	API for the Inter-Integrated Circuit (I2C) peripheral
I2C Instance	Device > Si91x > MCU > Peripheral	I2C peripheral instance
I2S	Device > Si91x > MCU > Peripheral	API for the Inter-Integrated Circuit Sound (I2S) peripheral
IO Stream: USART	Device > Si91x > MCU > Peripheral	API for input-output (I/O) stream over a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) interface
PWM	Device > Si91x > MCU > Peripheral	API for the Pulse width modulation (PWM) peripheral
PWM Instance	Device > Si91x > MCU > Peripheral	PWM peripheral instance
SIO	Device > Si91x > MCU > Peripheral	API for the Serial Input Output (SIO) peripheral
SSI	Device > Si91x > MCU > Peripheral	API for the Synchronous Serial Interface (SSI) peripheral
SysRTC	Device > Si91x > MCU > Peripheral	API for the system real-time clock (SysRTC) peripheral
ULP Timer	Device > Si91x > MCU > Peripheral	API for the ultra low power (ULP) timer peripheral
ULP Timer Instance	Device > Si91x > MCU > Peripheral	ULP timer peripheral instance
USART	Device > Si91x > MCU > Peripheral	API for the USART peripheral
WDT	Device > Si91x > MCU > Peripheral	API for the watchdog timer peripheral
SL GPIO Peripheral	Device > Si91x > MCU > Peripheral > Register-Level API	API for GPIO ports
SL I2C Peripheral	Device > Si91x > MCU > Peripheral > Register-Level API	API for the Inter-Integrated Circuit (I2C) peripheral
SL SDIO Secondary Peripheral	Device > Si91x > MCU > Peripheral > Register-Level API	API for Secure Digital Input Output (SDIO) peripheral as secondary
PSRAM Core	Device > Si91x > MCU > Peripheral > PSRAM Driver	PSRAM driver core functionality
APS1604M-SQR PSRAM Device	Device > Si91x > MCU > Peripheral > PSRAM Driver > PSRAM Device	AP Memory APS1604M-SQR PSRAM device related configurations
APS6404L_SQH PSRAM Device	Device > Si91x > MCU > Peripheral > PSRAM Driver > PSRAM Device	AP Memory APS6404L-SQH PSRAM device related configurations
APS6404L_SQRH PSRAM Device	Device > Si91x > MCU > Peripheral > PSRAM Driver > PSRAM Device	AP Memory APS6404L-SQRH PSRAM device related configurations
BSS Segment in PSRAM	Device > Si91x > MCU > Peripheral > PSRAM Driver > PSRAM Linker Configurations	Adding the Block Starting Symbol (BSS) segment in pseudo-static random-access memory (PSRAM)
Data Segment in PSRAM	Device > Si91x > MCU > Peripheral > PSRAM Driver > PSRAM Linker Configurations	Adding the data segment in PSRAM
Heap Segment in PSRAM	Device > Si91x > MCU > Peripheral > PSRAM Driver > PSRAM Linker Configurations	Adding the heap segment in PSRAM

Component Name	Component Category Path	Description
Stack Segment in PSRAM	Device > Si91x > MCU > Peripheral > PSRAM Driver > PSRAM Linker Configurations	Adding the stack segment in PSRAM
Text Segment in PSRAM	Device > Si91x > MCU > Peripheral > PSRAM Driver > PSRAM Linker Configurations	Adding the text segment in PSRAM
SDIO Secondary	Device > Si91x > MCU > Peripheral > SDIO Slave Driver	API for SDIO secondary driver
SDIO Slave Driver	Device > Si91x > MCU > Peripheral > SDIO Slave Driver	SDIO slave driver
ADC	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP ADC peripheral
BOD	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP BOD peripheral
Calendar	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP Calendar
Comparator	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP Comparator
CTS	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP CTS peripheral
DAC	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP digital-to-analog converter (DAC)
DMA	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP DMA peripheral
GPIO	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP GPIO ports
Low-Power Debug	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	Debug functionality in low power mode
I2C	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP I2C peripheral
I2S	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP I2S peripheral
IR	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP IR peripheral
SSI	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP SSI peripheral
SysRTC	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP SysRTC peripheral
UART	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP UART peripheral
ULP Timer	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP timer peripheral
User Files	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP user files execution from random-access memory (RAM)
Watchdog Timer	Device > Si91x > MCU > Service > Power Manager > ULP Peripheral	ULP watchdog timer

Drivers

Component Name	Component Category Path	Description
Button	Device > Si91x > MCU > Hardware	API for the button driver
Joystick	Device > Si91x > MCU > Hardware	API for the joystick peripheral driver
LED	Device > Si91x > MCU > Hardware	API for the light-emitting diode (LED) peripheral driver
Memory LCD SPI driver	Device > Si91x > MCU > Hardware	API for the memory liquid crystal display (LCD) peripheral driver over the Serial Peripheral Interface (SPI)
Si70xx Humidity and Temperature Sensor	Device > Si91x > MCU > Hardware	API for the Si70XX humidity and temperature sensor driver

Services

Component Name	Component Category Path	Description
Sensor Hub	Device > Si91x > MCU > Service	API for the sensor hub on the SiWx91x
NVM3 Common Flash	Device > Si91x > MCU > Service	Non-volatile memory (NVM3) API for SiWx91x common flash chip variants
NVM3 Dual Flash	Device > Si91x > MCU > Service	NVM3 API for SiWx91x dual flash chip variants
NVM3 for Si91x	Device > Si91x > MCU > Service	Common features for NVM3 APIs
IO Stream	Device > Si91x > MCU > Service	Data transfer over physical communication interfaces
Sleep Timer for Si91x	Device > Si91x > MCU > Service	Sleep timers
Sleep Timer SysRTC HAL	Device > Si91x > MCU > Service	Sleep timers using the system real-time clock (SysRTC)
Sleep Timer ULP Timer HAL	Device > Si91x > MCU > Service	Sleep timers using the ultra low power (ULP) timer
Power Manager	Device > Si91x > MCU > Service > Power Manager	API for SiWx91x power management

Cryptography

Component Name	Component Category Path	Description
AES	Device > Si91x > Wireless > Crypto	API for encyption using the Advanced Encryption Standard (AES) method
ATTESTATION	Device > Si91x > Wireless > Crypto	API for performing attestation of the SiWx91x using cryptographic methods
CCM	Device > Si91x > Wireless > Crypto	API for encryption using the counter with cipher block chaining message authentication code (CCM) method
CHACHAPOLY	Device > Si91x > Wireless > Crypto	API for encryption using the ChaChaPoly method
Crypto - Common	Device > Si91x > Wireless > Crypto	API for cryptographic functionality
Crypto - Utility	Device > Si91x > Wireless > Crypto	API for cryptographic utilities

Component Name	Component Category Path	Description
ECDH	Device > Si91x > Wireless > Crypto	API for encryption using the elliptic-curve Diffie-Hellman (ECDH) method
GCM	Device > Si91x > Wireless > Crypto	API for the Galois/Counter Mode (GCM) of operation
HMAC	Device > Si91x > Wireless > Crypto	API for encryption using the hash-based message authentication code (HMAC) method
PSA WRAP	Device > Si91x > Wireless > Crypto	API for Platform Security Architecture (PSA) standard cryptography using key wrap algorithms
SHA	Device > Si91x > Wireless > Crypto	API for encryption using the secure hash algorithm (SHA) method
TRNG	Device > Si91x > Wireless > Crypto	API for the True Random Number Generator (TRNG) algorithm
WRAP	Device > Si91x > Wireless > Crypto	API for encryption using key wrap algorithms
PSA AEAD	Device > Si91x > Wireless > PSA Crypto	API for PSA standard cryptography with the authenticated encryption with associated data (AEAD) method
PSA AES	Device > Si91x > Wireless > PSA Crypto	API for PSA standard cryptography with the AES method
PSA Crypto - Utility	Device > Si91x > Wireless > PSA Crypto	API for PSA standard cryptographic utilities
PSA ECDH	Device > Si91x > Wireless > PSA Crypto	API for PSA standard cryptography with the ECDH method
PSA HMAC	Device > Si91x > Wireless > PSA Crypto	API for PSA standard cryptography with the HMAC method
PSA SHA	Device > Si91x > Wireless > PSA Crypto	API for PSA standard cryptography with the SHA method
PSA TRNG	Device > Si91x > Wireless > PSA Crypto	API for the PSA standard TRNG algorithm

Network Protocols

Component Name	Component Category Path	Description
SNTP Client	Service	API for Simple Network Time Protocol (SNTP) client functionality

Application Protocols

Component Name	Component Category Path	Description
AWS IoT Device SDK	Third Party	API for accessing the Amazon Web Services (AWS) Internet-of-Things (IoT) cloud service
Built-In Command Database	Service > Console	Built-in console commands for WiSeConnect SDK v3.x features
Console	Service > Console	Console command-line interpreter (CLI) functionality
Console Variables	Service > Console	Functionality for reading and writing console variables
HTTP Client	Service	API for Hyper-Text Transfer Protocol (HTTP) client functionality
MQTT Client	Service	API for Message Queue Telemetry Transport (MQTT) client functionality

Bare Metal and OS Support

Bare Metal and OS Support

WiSeConnect SDK v3.x supports only FreeRTOS applications, while the v2.x SDK supports both Bare Metal and FreeRTOS applications.

Configuration

Configuration

WiSeConnect™ SDK v3.x passes all configurations as function arguments, while SDK v2.x passes some configurations as function arguments and sets the remaining as macros in header files.

Migration Example

Migration Example

This section provides an example of rewriting a WiSeConnect SDK v2.x Wi-Fi client (station mode) application with the v3.x API.

Some code, such as that for event handlers, is omitted for brevity.

In the example below:

- In the `main()` function, we replaced the v2.x `rsi_task_create()` API call with the FreeRTOS `osThreadNew()` API call, to create a new FreeRTOS task to run the application.
- In the initialization function, we replaced the v2.x `rsi_task_create()` and `rsi_wireless_init()` API calls with the v3.x sl_net_init() API call, to initialize the WiSeConnect SDK and SiWx917 device.
- In the initialization function, we replaced the v2.x `rsi_wlan_scan()` , `rsi_wlan_set()` , `rsi_wlan_connect()` , and `rsi_config_ipaddress()` API calls with the v3.x sl_net_set_profile(), sl_net_set_credential(), and sl_net_up() API calls, to scan for a wireless access point, connect to it, and configure an IP address.
- In v3.x, no task needs to be created by the application to process events, as compared with v2.x where an `rsi_wireless_driver_task` must be created. The v3.x SDK automatically creates a `bus_thread` and `event_thread` to manage events.

v2.x API Code	v3.x API Code
`int32_t application_start() {`	`static void application_start(void *argument) {`
`rsi_task_create((rsi_task_function_t) rsi_wireless_driver_task, (uint8_t) "driver_task", RSI_DRIVER_TASK_STACK_SIZE, NULL, RSI_DRIVER_TASK_PRIORITY), &driver_task_handle);`	
`int32_t status = rsi_wireless_init(0, 0);`	`sl_status_t status = sl_net_init(SL_NET_DEFAULT_WIFI_CLIENT_INTERFACE, NULL, NULL, NULL);`
`status = rsi_wlan_scan((int8_t *) SSID, (uint8_t) CHANNEL_NO, NUIL, 0);`	
`uint8_t join_bssid[6] = AP_BSSID;`	
`status = rsi_wlan_set(CMD_TYPE, join_bssid, 6);`	`sl_net_set_profile(SL_NET_WIFI_CLIENT_INTERFACE, SL_NET_DEFAULT_WIFI_CLIENT_PROFILE_ID, &wifi_client_profile);`
`status = rsi_wlan_connect((int8_t *) SSID, SECURITY_TYPE, PSK);`	`sl_net_set_credential(SL_NET_DEFAULT_WIFI_CLIENT_CREDENTIAL_ID, SL_NET_WIFI_PSK, &(wifi_client_credential.data), sizeof(wifi_client_credential.data));`
`status = rsi_config_ipaddress(RSI_IP_VERSION_4, RSI_STATIC, (uint8_t *) &ip_addr, (uint8_t *) &network_mask, (uint8_t *) &gateway, NULL, 0, 0);`	`status = sl_net_up(SL_NET_DEFAULT_WIFI_CLIENT_INTERFACE, SL_NET_DEAULT_WIFI_CLIENT_PROFILE_ID);`
`while (1) { rsi_wireless_driver_task(); }`	`while(1) { osDelay(osWaitForever); }`
`}`	`}`
`int main() {`	`void main(const void *unused) {`
	`sl_system_init();`

v2.x API Code	v3.x API Code
rsi_task_handle_t wlan_task_handle = NULL;	UNUSED_PARAMETER(unused);
rsi_task_create((rsi_task_function_t) application_start, (uint8_t *) "wlan_task", RSI_WLAN_TASK_STACK_SIZE, NULL, RSI_WLAN_TASK_PRIORITY, &wlan_task_handle);	osThreadNew((osThreadFunc_t) application_start, NULL,, &thread_attributes);
rsi_os_start_scheduler();	sl_system_kernel_start();
}	}

Overview

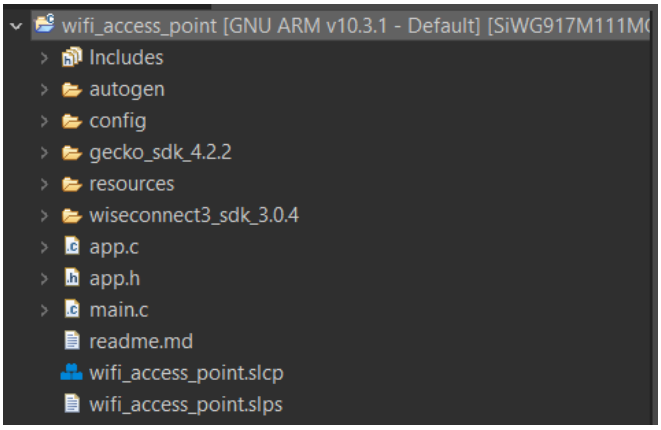
Steps for Migrating an SDK v2.x Application to v3.x

This section provides the steps to follow to convert your existing application written with WiSeConnect™ SDK v2.x into a 3.x application.

Create Studio Project

Create Studio Project

1. Follow the appropriate getting started guide to:
    - install Simplicity Studio,
    - install the WiSeConnect 3 extension, and
    - create a Studio project.



Update Init Code

Update Init Code

1. In the newly created v3.x Studio project, add your new initialization code in the `app.c` file in the `application_start()` function.
2. The `application_start()` function should include a call to the sl_net_init() function.
3. Device initialization parameters are set using a structure of type sl_si91x_boot_configuration_t, which is a parameter of the sl_net_init() function.
4. You may either choose one of the available default configurations or create a custom device configuration.

Update Configuration

Update Configuration

In your existing application, replace each WiSeConnect™ SDK v2.x configuration parameter with the equivalent v3.x configuration parameter. Refer to the table below for the mapping of v2.x to v3.x configuration parameters.

v2.x configuration parameters are listed in alphabetical order of parameter name.

V2.x Configuration	v3.x Configuration	Notes
uint16_t **beacon_interval** parameter in rsi_wlan_ap_start()	uint16_t sl_wifi_ap_configuration_t :: beacon_interval	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t rsi_opermode_t :: ble_ext_feature_bit_map[4]	uint32_t sl_si91x_boot_configuration_t :: ble_ext_feature_bit_map	
uint8_t rsi_opermode_t :: ble_feature_bit_map[4]	uint32_t sl_si91x_boot_configuration_t :: ble_feature_bit_map	
uint16_t **coex_mode** parameter in rsi_wireless_init()	uint16_t sl_si91x_boot_configuration_t :: coex_mode	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t **channel** parameter in rsi_wlan_ap_start()	sl_wifi_channel_t sl_wifi_ap_configuration_t :: channel	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t rsi_opermode_t :: config_feature_bit_map[4]	uint32_t sl_si91x_boot_configuration_t :: config_feature_bit_map	
uint8_t **dtim_period** parameter in rsi_wlan_ap_start()	uint16_t sl_wifi_ap_configuration_t :: dtim_beacon_count	API parameter in v2.x API is mapped to struct member in v3.x.
rsi_encryption_mode_t **encryption_mode** parameter in rsi_wlan_ap_start()	sl_wifi_encryption_t sl_wifi_ap_configuration_t :: encryption	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t twt_user_params_t :: implicit_twt	uint8_t sl_wifi_twt_request_t :: implicit_twt	
uint8_t * **ip_addr** , uint8_t * **mask** , uint8_t * **gw** parameters in rsi_config_ipaddress()	union { sl_net_ipv4_setting_t v4; sl_net_ipv6_setting_t v6; } sl_net_ip_configuration_t :: ip	All three v2.x function parameters are set in v3.x using a struct member.
uint8_t **mode** parameter in rsi_config_ipaddress()	sl_ip_management_t sl_net_ip_configuration_t :: mode	API parameter in v2.x API is mapped to struct member in v3.x.
uint16_t **oper_mode** parameter in rsi_wireless_init()	uint16_t sl_si91x_boot_configuration_t :: oper_mode	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t rsi_eap_credentials_t :: password[128]	uint8_t sl_wifi_eap_credential_t :: password[SL_EAP_PASSWORD_LENGTH]	
uint8_t twt_user_params_t :: restrict_tx_outside_tsp	uint8_t sl_wifi_twt_request_t :: restrict_tx_outside_tsp	
uint8_t twt_user_params_t :: req_type	uint8_t sl_wifi_twt_request_t :: req_type	

V2.x Configuration	v3.x Configuration	Notes
RSI_CUSTOM_FEATURE_BIT_MAP	uint32_t sl_si91x_boot_configuration_t :: custom_feature_bit_map	Macro in v2.x API is mapped to struct member in v3.x.
RSI_EXT_CUSTOM_FEATURE_BIT_MAP	uint32_t sl_si91x_boot_configuration_t :: ext_custom_feature_bit_map	Macro in v2.x API is mapped to struct member in v3.x.
RSI_EXT_TCPIP_FEATURE_BITMAP	uint32_t sl_si91x_boot_configuration_t :: ext_tcp_ip_feature_bit_map	Macro in v2.x API is mapped to struct member in v3.x.
RSI_FEATURE_BIT_MAP	uint32_t sl_si91x_boot_configuration_t :: feature_bit_map	Macro in v2.x API is mapped to struct member in v3.x.
RSI_TCP_IP_FEATURE_BIT_MAP	uint32_t sl_si91x_boot_configuration_t :: tcp_ip_feature_bit_map	Macro in v2.x API is mapped to struct member in v3.x.
rsi_security_mode_t **sec_type** parameter in rsi_wla_connect()	sl_wifi_security_t sl_wifi_client_configuration_t :: security	API parameter in v2.x API is mapped to struct member in v3.x.
int8_t * **ssid** parameter in rsi_wlan_connect()	sl_wifi_ssid_t sl_wifi_client_configuration_t :: ssid	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t twt_user_params_t :: triggered_twt	uint8_t sl_wifi_twt_request_t :: triggered_twt	
uint8_t twt_user_params_t :: twt_channel	uint8_t sl_wifi_twt_request_t :: twt_channel	
uint8_t **twt_enable** parameter in rsi_wlan_twt_config()	uint8_t sl_wifi_twt_request_t :: twt_enable	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t **twt_flow_id** parameter in rsi_wlan_twt_config()	uint8_t sl_wifi_twt_request_t :: twt_flow_id	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t twt_user_params_t :: twt_retry_limit	uint8_t sl_wifi_twt_request_t :: twt_retry_limit	
uint8_t twt_user_params_t :: twt_retry_interval	uint8_t sl_wifi_twt_request_t :: twt_retry_interval	
uint8_t twt_user_params_t :: twt_protection	uint8_t sl_wifi_twt_request_t :: twt_protection	
uint8_t twt_user_params_t :: un_announced_twt	uint8_t sl_wifi_twt_request_t :: un_announced_twt	
uint8_t rsi_eap_credentials_t :: username[64]	uint8_t sl_wifi_eap_credential_t :: username[SL_EAP_USER_NAME_LENGTH]	
rsi_ip_version_t **version** parameter in rsi_config_ipaddress()	sl_ip_address_type_t sl_net_ip_configuration_t :: type	API parameter in v2.x API is mapped to struct member in v3.x.
uint8_t twt_user_params_t :: wake_duration	uint8_t sl_wifi_twt_request_t :: wake_duration	
uint8_t twt_user_params_t :: wake_duration_tol	uint8_t sl_wifi_twt_request_t :: wake_duration_tol	
uint8_t twt_user_params_t :: wake_duration_unit	uint8_t sl_wifi_twt_request_t :: wake_duration_unit	

V2.x Configuration	v3.x Configuration	Notes
uint8_t twt_user_params_t :: wake_int_exp	uint8_t sl_wifi_twt_request_t :: wake_int_exp	
uint8_t twt_user_params_t :: wake_int_exp_tol	uint8_t sl_wifi_twt_request_t :: wake_int_exp_tol	
uint16_t twt_user_params_t :: wake_int_mantissa	uint16_t sl_wifi_twt_request_t :: wake_int_mantissa	
uint16_t twt_user_params_t :: wake_int_mantissa_tol	uint16_t sl_wifi_twt_request_t :: wake_int_mantissa_tol	

Update API Calls

Update API Calls

In your existing application, replace each WiSeConnect™ SDK v2.x API call with the equivalent v3.x API call(s). Refer to the tables in the sections below for the mapping of v2.x to v3.x APIs.

v2.x APIs are listed in alphabetical order of API name.

If your v2.x API is not mentioned below, it is not supported in the v3.x API.

BLE APIs

BLE APIs are identical in WiSeConnect SDK v2.x and v3.x.

Common APIs

v2.x API	v3.x API	Notes
int32_t rsi_assert (void)	sl_status_t sl_si91x_assert (void)	
int32_t rsi_cmd_m4_ta_secure_handshake (uint8_t sub_cmd_type, uint8_t input_len, uint8_t *input_data, uint8_t output_len, uint8_t *output_data)	sl_status_t sl_si91x_m4_ta_secure_handshake (uint8_t sub_cmd_type, uint8_t input_len, uint8_t *input_data, uint8_t output_len, uint8_t *output_data)	
int32_t rsi_device_init (uint8_t select_option)	sl_status_t sl_wifi_init (const sl_wifi_device_configuration_t *configuration, sl_wifi_event_handler_t event_handler)	
int32_t rsi_device_deinit (void)	sl_status_t sl_wifi_deinit (void)	
int32_t rsi_driver_deinit (void)	sl_status_t sl_si91x_driver_deinit (void)	
int32_t rsi_get_fw_version (uint8_t *response, uint16_t length)	sl_status_t sl_wifi_get_firmware_version (sl_wifi_version_string_t *version)	
int32_t rsi_send_feature_frame_dyn (uint32_t feature_enables)	sl_wifi_init()	See signature above.
int32_t rsi_send_feature_frame (void)	sl_status_t sl_si91x_driver_init (const sl_wifi_device_configuration_t *config, sl_wifi_event_handler_t event_handler)	The sl_si91x_driver_init() function sends the feature frames.
int32_t rsi_wireless_antenna (uint8_t type, uint8_t gain_2g, uint8_t gain_5g)	sl_status_t sl_wifi_get_antenna() (sl_wifi_interface_t interface, sl_wifi_antenna_t *antenna)	No support for gain_2g or gain_5g parameters.
int32_t rsi_wireless_deinit (void)	sl_wifi_deinit()	See signature above.

v2.x API	v3.x API	Notes
void rsi_wireless_driver_task (void)	void sl_system_kernel_start (void)	sl_system_kernel_start() is a Gecko Platform API.
int32_t rsi_wireless_init (uint16_t opermode, uint16_t coex_mode)	sl_wifi_init()	See signature above.

Wi-Fi APIs

v2.x API	v3.x API	Notes
int32_t rsi_config_ipaddress (rsi_ip_version_t version, uint8_t mode, uint8_t *ip_addr, uint8_t *mask, uint8_t *gw, uint8_t *ipconfig_rsp, uint16_t length, uint8_t vap_id)	sl_status_t sl_si91x_configure_ip_address (sl_net_ip_configuration_t *address)	
int32_t rsi_driver_process_recv_data (rsi_pkt_t *pkt)	None	The v3.x SDK processes this internally in the bus_thread .
int16_t rsi_nwk_register_callbacks (uint32_t callback_id, void (*callback_handler_ptr) (uint8_t command_type, uint32_t status, const uint8_t *buffer, const uint32_t length))	sl_status_t sl_si91x_register_callback (sl_net_event_t event, sl_net_event_handler_t function)	Every service has APIs to register callbacks.
void rsi_sort_scan_results_array_based_on_rssi (struct wpa_scan_results_arr *scan_results_array)	sl_status_t sl_wifi_wait_for_scan_results (sl_wifi_scan_result_t **scan_result_array, uint32_t max_scan_result_count)	
int32_t rsi_wlan_ap_start (int8_t *ssid, uint8_t channel, rsi_security_mode_t security_type, rsi_encryption_mode_t encryption_mode, uint8_t *password, uint16_t beacon_interval, uint8_t dtim_period)	sl_status_t sl_wifi_start_ap (sl_wifi_interface_t interface, const sl_wifi_ap_configuration_t *configuration)	
int32_t rsi_wlan_bgscan_profile (uint8_t cmd, rsi_rsp_scan_t *result, uint32_t length)	sl_status_t sl_wifi_start_scan (sl_wifi_scan_result_t **scan_result_array, uint32_t max_scan_result_count)	
int32_t rsi_wlan_connect (int8_t *ssid, rsi_security_mode_t sec_type, void *secret_key)	sl_status_t sl_wifi_connect (sl_wifi_interface_t interface, const sl_wifi_client_configuration_t *access_point, uint32_t timeout_ms)	Currently not supported: RSI_WLAN_REQ_WMM_PS, RSI_WLAN_REQ_WPS_METHOD, RSI_WLAN_REQ_SET_WEP_KEYS, RSI_WLAN_REQ_REJOIN_PARAMS
int32_t rsi_wlan_connect_async (int8_t *ssid, rsi_security_mode_t sec_type, void *secret_key, void (*join_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length))	sl_wifi_connect()	See signature above.

v2.x API	v3.x API	Notes
int32_t **rsi_wlan_disconnect** (void)	sl_status_t **sl_wifi_disconnect** (sl_wifi_interface_t interface)	
int32_t **rsi_wlan_disconnect_stations** (uint8_t *mac_address)	sl_status_t **sl_wifi_disconnect_ap_client** (sl_wifi_interface_t interface, const sl_mac_address_t *mac, sl_wifi_deauth_reason_t reason)	
int32_t **rsi_wlan_filter_broadcast** (uint16_t beacon_drop_threshold, uint8_t filter_bcast_in_tim, uint8_t filter_bcast_tim_till_next_cmd)	sl_status_t **sl_wifi_filter_broadcast** (uint16_t beacon_drop_threshold, uint8_t filter_bcast_in_tim, uint8_t filter_bcast_tim_till_next_cmd)	
int32_t **rsi_wlan_get** (rsi_wlan_query_cmd_t cmd_type, uint8_t *response, uint16_t length)	**sl_wifi_get_mac_address()**, **sl_wifi_get_ip_address()**, **sl_wifi_get_signal_strength()**, **sl_wifi_get_ap_client_info()**, getsockname(), **sl_wifi_get_statistics()**, **sl_wifi_get_channel()**, **sl_wifi_get_firmware_version()**	Currently not supported: RSI_WLAN_REQ_CONNECTION_STATUS, RSI_WLAN_REQ_GET_CFG, RSI_WLAN_REQ_GET_STATS
int32_t **rsi_wlan_ping_async** (uint8_t flags, uint8_t *ip_address, uint16_t size, void (*wlan_ping_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length))	sl_status_t **sl_si91x_send_ping** (sl_ip_address_t ip_address, uint16_t ping_packet_size)	
int32_t **rsi_wlan_pmk_generate** (int8_t type, int8_t *psk, int8_t *ssid, uint8_t *pmk, uint16_t length)	sl_status_t **sl_wifi_get_pairwise_master_key** (sl_wifi_interface_t interface, const uint8_t type, const sl_wifi_ssid_t *ssid, const char *pre_shared_key, uint8_t *pairwise_master_key)	
int32_t **rsi_wlan_power_save_with_listen_interval** (uint8_t psp_mode, uint8_t psp_type, uint16_t listen_interval)	sl_status_t **sl_wifi_set_listen_interval** (sl_wifi_interface_t interface, sl_wifi_listen_interval_t listen_interval)	
int32_t **rsi_wlan_power_save_profile** (uint8_t psp_mode, uint8_t psp_type)	sl_status_t **sl_wifi_set_performance_profile** (const sl_wifi_performance_profile_t *profile)	
int32_t **rsi_wlan_radio_init** (void)	sl_status_t **sl_wifi_init** (const sl_wifi_device_configuration_t *configuration, sl_wifi_event_handler_t event_handler)	RSI_TIMEOUT_SUPPORT not supported.

v2.x API	v3.x API	Notes
int32_t rsi_wlan_receive_stats_start (uint16_t channel)	sl_status_t sl_wifi_start_statistic_report (sl_wifi_interface_t interface, sl_wifi_channel_t channel)	
int32_t rsi_wlan_receive_stats_stop (void)	sl_status_t sl_wifi_stop_statistic_report (sl_wifi_interface_t interface)	
uint16_t rsi_wlan_register_callbacks (uint32_t callback_id, void (*callback_handler_ptr)(uint16_t status, uint8_t *buffer, const uint32_t length))	sl_status_t sl_wifi_set_callback (sl_wifi_event_group_t group, sl_wifi_callback_function_t function, void *optional_arg)	Every service has APIs to register callbacks.
int32_t rsi_wlan_scan (int8_t *ssid, uint8_t chno, rsi_rsp_scan_t *result, uint32_t length)	sl_wifi_start_scan()	See signature above.
int32_t rsi_wlan_scan_async (int8_t *ssid, uint8_t chno, void (*scan_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length))	sl_wifi_start_scan()	See signature above.
int32_t rsi_wlan_send_data (uint8_t *buffer, uint32_t length)	sl_status_t sl_wifi_send_raw_data_frame (sl_wifi_interface_t interface, const void *data, uint16_t data_length)	
int32_t rsi_wlan_set_certificate_index (uint8_t certificate_type, uint8_t cert_inx, uint8_t *buffer, uint32_t certificate_length)	sl_status_t sl_wifi_set_certificate_with_index (uint8_t certificate_type, uint8_t certificate_index, uint8_t *buffer, uint32_t certificate_length)	
int32_t rsi_wlan_set (rsi_wlan_set_cmd_t cmd_type, uint8_t *request, uint16_t length)	sl_status_t sl_wifi_set_mac_address (sl_wifi_interface_t interface, const sl_mac_address_t *mac)	
int32_t rsi_wlan_wfd_connect (int8_t *device_name, void (*join_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length))	sl_status_t sl_wifi_p2p_connect (sl_wifi_interface_t interface, const sl_wifi_p2p_configuration_t *configuration)	
int32_t rsi_wlan_wfd_start_discovery (uint16_t go_intent, int8_t *device_name, uint16_t channel, int8_t *ssid_post_fix, uint8_t *psk, void (*wlan_wfd_discovery_notify_handler)(uint16_t status, uint8_t *buffer, const uint32_t length), void (*wlan_wfd_connection_request_notify_handler)(uint16_t status, uint8_t *buffer, const uint32_t length))	sl_status_t sl_wifi_start_p2p_discovery (sl_wifi_interface_t interface, const sl_wifi_p2p_configuration_t *configuration, sl_wifi_credential_id_t credential_id)	
int32_t rsi_wlan_wps_generate_pin (uint8_t *wps_pin, uint16_t length)	sl_status_t sl_wifi_generate_wps_pin (sl_wifi_wps_pin_t *response)	
int32_t rsi_wlan_wps_push_button_event (int8_t *ssid)	sl_status_t sl_wifi_start_wps (sl_wifi_interface_t interface, sl_wifi_wps_mode_t mode, const sl_wifi_wps_pin_t *optional_wps_pin)	

Network Management APIs

v2.x API	v3.x API	Notes
int32_t **rsi_accept** (int32_t sockID, struct rsi_sockaddr *ClientAddress, int32_t *addressLength)	accept()	BSD-standard API.
int32_t **rsi_bind** (int32_t sockID, struct rsi_sockaddr *localAddress, int32_t addressLength)	bind()	BSD-standard API.
void **rsi_clear_sockets** (int32_t sockID)	sl_status_t **sl_si91x_socket_deinit** (void)	
int32_t **rsi_connect** (int32_t sockID, struct rsi_sockaddr *remoteAddress, int32_t addressLength)	connect()	BSD-standard API.
int32_t **rsi_dns_req** (uint8_t ip_version, uint8_t *url_name, uint8_t *primary_server_address, uint8_t *secondary_server_address, rsi_rsp_dns_query_t *dns_query_resp, uint16_t length)	sl_status_t **sl_dns_host_get_by_name** (const char *host_name, const uint32_t timeout, const sl_net_dns_resolution_ip_type_t dns_resolution_ip, sl_ip_address_t *ip_address)	Also see the next row.
Same as above.	struct hostent **gethostbyname** (const char *name)	
int32_t **rsi_emb_mqtt_client_init** (int8_t *server_ip, uint32_t server_port, uint32_t client_port, uint16_t flags, uint16_t keep_alive_interval, int8_t *clientid, int8_t *username, int8_t *password)	sl_status_t **sl_mqtt_client_init** (sl_mqtt_client_t *client, sl_mqtt_client_event_handler event_handler)	
int32_t **rsi_emb_mqtt_connect** (uint8_t mqtt_flags, int8_t *will_topic, uint16_t will_message_len, int8_t *will_message)	sl_status_t **sl_mqtt_client_connect** (sl_mqtt_client_t *client, const sl_mqtt_broker_t *broker, const sl_mqtt_client_last_will_message_t *last_will_message, const sl_mqtt_client_configuration_t *configuration, uint32_t timeout)	
int32_t **rsi_emb_mqtt_destroy** (void)	sl_status_t **sl_mqtt_client_deinit** (sl_mqtt_client_t *client)	
int32_t **rsi_emb_mqtt_disconnect** (void)	sl_status_t **sl_mqtt_client_disconnect** (sl_mqtt_client_t *client, uint32_t timeout)	
int32_t **rsi_emb_mqtt_publish** (int8_t *topic, rsi_mqtt_pubmsg_t *publish_msg)	sl_status_t **sl_mqtt_client_publish** (sl_mqtt_client_t *client, const sl_mqtt_client_message_t *message, uint32_t timeout, void *context)	
int32_t **rsi_emb_mqtt_subscribe** (uint8_t qos, int8_t *topic)	sl_status_t **sl_mqtt_client_subscribe** (sl_mqtt_client_t *client, const uint8_t *topic, uint16_t topic_length, sl_mqtt_qos_t qos_level, uint32_t timeout, sl_mqtt_client_message_received message_handler, void *context)	
int32_t **rsi_emb_mqtt_unsubscribe** (int8_t *topic)	sl_status_t **sl_mqtt_client_unsubscribe** (sl_mqtt_client_t *client, const uint8_t *topic, uint16_t length, uint32_t timeout, void *context)	
int32_t **rsi_get_socket_descriptor** (uint8_t *src_port, uint8_t *dst_port, uint8_t *ip_addr, uint16_t ip_version, uint16_t sockid)	getsockname()	BSD-standard API.

v2.x API	v3.x API	Notes
int32_t rsi_get_socket_id (uint32_t src_port, uint32_t dst_port)	int get_socket (int socket_id)	
int rsi_getsockopt (int32_t sockID, int level, int option_name, const void *option_value, rsi_socklen_t option_len)	getsockopt()	BSD-standard API.
int32_t rsi_http_client_abort (void)	sl_status_t sl_http_client_deinit (sl_http_client_t *client)	
int32_t rsi_http_client_async (uint8_t type, uint8_t flags, uint8_t *ip_address, uint16_t port, uint8_t *resource, uint8_t *host_name, uint8_t *extended_header, uint8_t *user_name, uint8_t *password, uint8_t *post_data, uint32_t post_data_length, #if RSI_HTTP_STATUS_INDICATION_EN void (*callback)(uint16_t status, const uint8_t *buffer, const uint16_t length, const uint32_t moredata, uint16_t status_code) #else void (*callback)(uint16_t status, const uint8_t *buffer, const uint16_t length, const uint32_t moredata) #endif)	sl_status_t sl_http_client_send_request (const sl_http_client_t *client, const sl_http_client_request_t *request)	
int32_t rsi_http_client_get_async (uint8_t flags, uint8_t *ip_address, uint16_t port, uint8_t *resource, uint8_t *host_name, uint8_t *extended_header, uint8_t *user_name, uint8_t *password, #if RSI_HTTP_STATUS_INDICATION_EN void (*http_client_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length, const uint32_t moredata, uint16_t status_code) #else void (*http_client_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length, const uint32_t moredata) #endif)	sl_http_client_send_request()	See signature above.

v2.x API	v3.x API	Notes
int32_t rsi_http_client_post_async (uint8_t flags, uint8_t *ip_address, uint16_t port, uint8_t *resource, uint8_t *host_name, uint8_t *extended_header, uint8_t *user_name, uint8_t *password, uint8_t *post_data, uint32_t post_data_length, #if RSI_HTTP_STATUS_INDICATION_EN void (*http_client_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length, const uint32_t moredata, uint16_t status_code) #else void (*http_client_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length, const uint32_t moredata) #endif)	sl_http_client_send_request()	See signature above.
int32_t rsi_http_client_post_data (uint8_t *file_content, uint16_t current_chunk_length, #if RSI_HTTP_STATUS_INDICATION_EN void (*rsi_http_post_data_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length, const uint32_t moredata, uint16_t status_code) #else void (*rsi_http_post_data_response_handler)(uint16_t status, const uint8_t *buffer, const uint16_t length, const uint32_t moredata) #endif)	sl_http_client_send_request()	See signature above.
int32_t rsi_http_client_put_create (void)	sl_http_client_send_request()	See signature above.
int32_t rsi_http_client_put_delete (void)	sl_http_client_send_request()	See signature above.
int32_t rsi_http_client_put_pkt (uint8_t *file_content, uint16_t current_chunk_length)	sl_http_client_send_request()	See signature above.
int32_t rsi_http_client_put_start (uint8_t flags, uint8_t *ip_address, uint32_t port_number, uint8_t *resource, uint8_t *host_name, uint8_t *extended_header, uint8_t *user_name, uint8_t *password, uint32_t content_length, void (*callback)(uint16_t status, uint8_t type, const uint8_t *buffer, uint16_t length, const uint8_t end_of_put_pkt))	sl_http_client_send_request()	See signature above.
int32_t rsi_listen (int32_t sockID, int32_t backlog)	listen()	BSD-standard API.
int32_t rsi_ota_firmware_upgradation (uint8_t flags, uint8_t *server_ip, uint32_t server_port, uint16_t chunk_number, uint16_t timeout, uint16_t tcp_retry_count, void (*ota_fw_up_response_handler)(uint16_t status, uint16_t chunk_number))	sl_status_t sl_si91x_ota_firmware_upgradation (sl_ip_address_t server_ip, uint16_t server_port, uint16_t chunk_number, uint16_t timeout, uint16_t tcp_retry_count, bool a synchronous)	
int32_t rsi_socket (int32_t protocolFamily, int32_t type, int32_t protocol)	socket()	BSD-standard API.

v2.x API	v3.x API	Notes
int32_t rsi_recv (int32_t sockID, void *rcvBuffer, int32_t bufferLength, int32_t flags)	recv()	BSD-standard API.
int32_t rsi_recvfrom (int32_t sockID, int8_t *buffer, int32_t buffersize, int32_t flags, struct rsi_sockaddr *fromAddr, int32_t *fromAddrLen)	recvfrom()	BSD-standard API.
int32_t rsi_select (int32_t nfds, rsi_fd_set *readfds, rsi_fd_set *writefds, rsi_fd_set *exceptfds, struct rsi_timeval *timeout, void (*callback)(rsi_fd_set *fd_read, rsi_fd_set *fd_write, rsi_fd_set *fd_except, int32_t status))	select()	BSD-standard API.
int rsi_setsockopt (int32_t sockID, int level, int option_name, const void *option_value, rsi_socklen_t option_len)	setsockopt()	BSD-standard API. tcp_max_retry, sni, retry_transmit, vap_id, ssl versions are not supported.
int32_t rsi_send (int32_t sockID, const int8_t *msg, int32_t msgLength, int32_t flags)	send()	BSD-standard API.
int32_t rsi_sendto (int32_t sockID, int8_t *msg, int32_t msgLength, int32_t flags, struct rsi_sockaddr *destAddr, int32_t destAddrLen)	sendto()	BSD-standard API.
int32_t rsi_shutdown (int32_t sockID, int32_t how)	close()	BSD-standard API. Port-based close functionality is not supported.
int32_t rsi_socket_bind (int32_t sockID, struct rsi_sockaddr *localAddress, int32_t addressLength)	bind()	BSD-standard API.
int32_t rsi_socket_create_async (int32_t sockID, int32_t type, int32_t backlog)	socket()	BSD-standard API.
int32_t rsi_socket_connect (int32_t sockID, struct rsi_sockaddr *remoteAddress, int32_t addressLength)	connect()	BSD-standard API.
int32_t rsi_socket_listen (int32_t sockID, int32_t backlog)	listen()	BSD-standard API.
int32_t rsi_socket_recvfrom (int32_t sockID, int8_t *buffer, int32_t buffersize, int32_t flags, struct rsi_sockaddr *fromAddr, int32_t *fromAddrLen)	recvfrom()	BSD-standard API. Large lengths are not supported.
int32_t rsi_socket_shutdown (int32_t sockID, int32_t how)	close()	BSD-standard API. Port-based close functionality is not supported.
int32_t rsi_wlan_set_certificate (uint8_t certificate_type, uint8_t *buffer, uint32_t certificate_length)	sl_status_t sl_wifi_set_certificate (uint8_t certificate_type, const uint8_t *buffer, uint32_t certificate_length)	

Firmware Update APIs

v2.x API	v3.x API
v2.x API	v3.x API
int32_t **rsi_fwup** (uint8_t type, uint8_t *content, uint16_t length)	sl_status_t **sl_si91x_fwup** (uint16_t type, uint8_t *content, uint16_t length)
int32_t **rsi_fwup_load** (uint8_t *content, uint16_t length)	sl_status_t **sl_si91x_fwup_load** (uint8_t *content, uint16_t length)
int32_t **rsi_fwup_start** (uint8_t *rps_header)	sl_status_t **sl_si91x_fwup_start** (uint8_t *rps_header)
int32_t **rsi_http_fw_update** (uint8_t flags, uint8_t *ip_address, uint16_t port, uint8_t *resource, uint8_t *host_name, uint8_t *extended_header, uint8_t *user_name, uint8_t *password, void (*http_otaf_response_handler)(uint16_t status, const uint8_t *buffer))	sl_status_t **sl_si91x_http_otaf** (uint8_t type, uint8_t flags, uint8_t *ip_address, uint16_t port, uint8_t *resource, uint8_t *host_name, uint8_t *extended_header, uint8_t *user_name, uint8_t *password, uint8_t *post_data, uint32_t post_data_length)

Update Callback Handlers

Update Callback Handlers

In your existing application, replace each WiSeConnect™ SDK v2.x callback handler with the equivalent v3.x handler. Refer to the table below for the mapping of v2.x to v3.x callback handlers.

v2.x callback handlers are listed in alphabetical order of parameter name.

v2.x Callback	v3.x Callback
void * callback_handler_ptr (uint16_t status, uint8_t *buffer, const uint32_t length)	sl_status_t * sl_net_event_handler_t (uint32_t event, sl_status_t status, void *data, uint32_t data_length)
callback_handler_ptr()	sl_status_t * sl_wifi_callback_function_t (sl_wifi_event_t event, void *data, uint32_t data_length, void *arg)
callback_handler_ptr()	sl_status_t * sl_wifi_stats_callback_t (sl_wifi_event_t event, void *data, uint32_t data_length, void *arg)
callback_handler_ptr()	sl_status_t * sl_wifi_twt_config_callback_t (sl_wifi_event_t event, sl_si91x_twt_response_t *data, uint32_t data_length, void *arg)
void * join_response_handler (uint16_t status, const uint8_t *buffer, const uint16_t length)	sl_status_t * sl_wifi_join_callback_t (sl_wifi_event_t event, char *data, uint32_t data_length, void *arg)
void * scan_response_handler (uint16_t status, const uint8_t *buffer, const uint16_t length)	sl_status_t * sl_wifi_scan_callback_t (sl_wifi_event_t event, sl_wifi_scan_result_t *data, uint32_t data_length, void *arg)

Update Events

Update Events

In your existing application, replace each WiSeConnect™ SDK v2.x event name with the equivalent v3.x event name. Refer to the table below for the mapping of v2.x to v3.x events.

v2.x events are listed in alphabetical order of event name.

v2.x Event	v3.x Event
RSI_STATIONS_CONNECT_NOTIFY_CB	SL_WIFI_CLIENT_CONNECTED
RSI_STATIONS_DISCONNECT_NOTIFY_CB	SL_WIFI_CLIENT_DISCONNECTED
RSI_WLAN_ASYNC_STATS	SL_WIFI_RECEIVE_STATS_RESPONSE_EVENT
RSI_WLAN_JOIN_RESPONSE_HANDLER	SL_WIFI_JOIN_EVENT
RSI_WLAN_RAW_DATA_RECEIVE_HANDLER	SL_WIFI_RX_PACKET_EVENT
RSI_WLAN_SCAN_RESPONSE_HANDLER	SL_WIFI_SCAN_RESULT_EVENT
RSI_WLAN_TWT_RESPONSE_CB	SL_WIFI_TWT_RESPONSE_EVENT

Update Enums

Update Enums

In your existing application, replace each WiSeConnect™ SDK v2.x enum value with the equivalent v3.x enum value. Refer to the table below for the mapping of v2.x to v3.x enums.

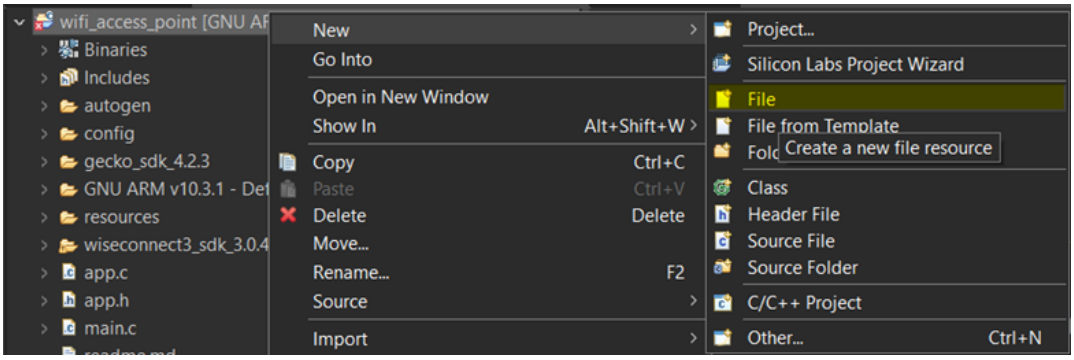
v2.x events are listed in alphabetical order of enum name.

v2.x Enum	v3.x Enum
RSI_WLAN_ACCESS_POINT_MODE	SL_SI91X_ACCESS_POINT_MODE
RSI_WLAN_CLIENT_MODE	SL_SI91X_CLIENT_MODE
RSI_WLAN_CONCURRENT_MODE	SL_SI91X_CONCURRENT_MODE
RSI_WLAN_ENTERPRISE_CLIENT_MODE	SL_SI91X_ENTERPRISE_CLIENT_MODE
RSI_WLAN_TRANSMIT_TEST_MODE	SL_SI91X_TRANSMIT_TEST_MODE

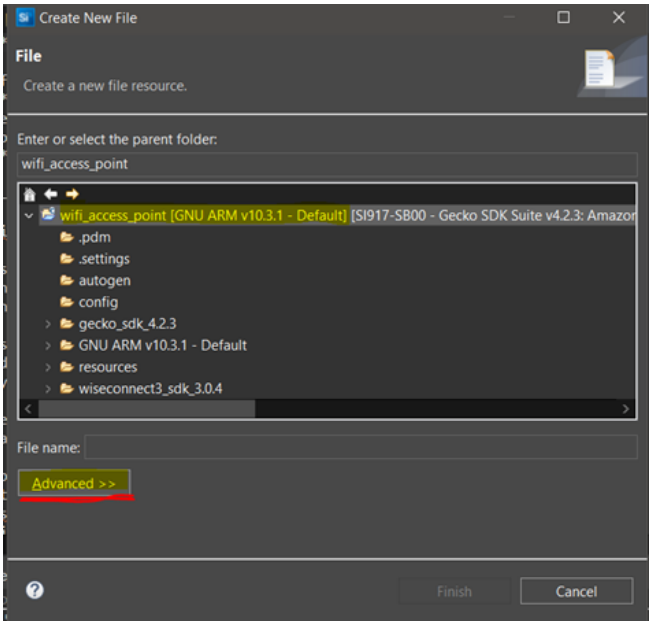
Import Updated Code into Studio

Import Updated Code into Studio

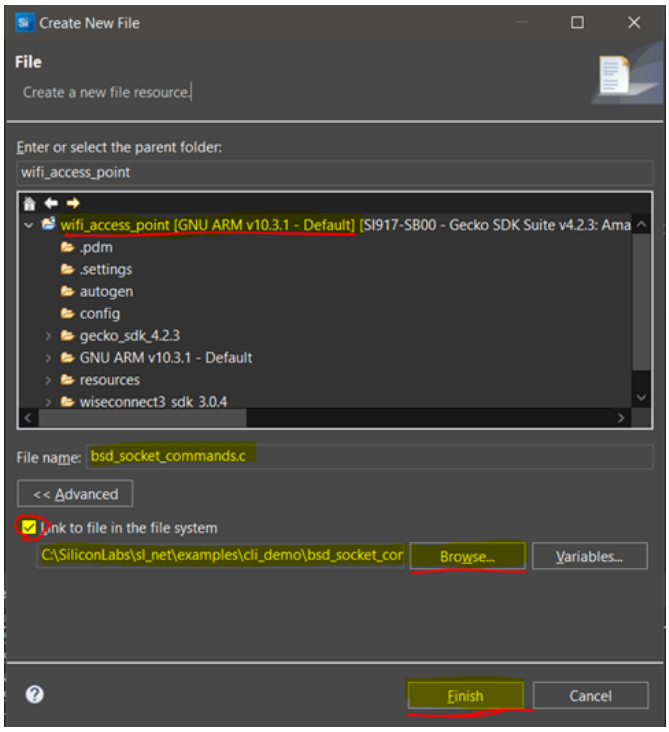
1. When you have updated your code for WiSeConnect™ SDK v3.x, either copy it into the existing `app.c` file in your Simplicity Studio project or organize it into multiple source files and add them to your Simplicity Studio project.
2. To add files to your Simplicty Studio project:
   - Open your project in Simplicity Studio.
   - Right-click on the project folder in the Project Explorer window, and select **New > File**.



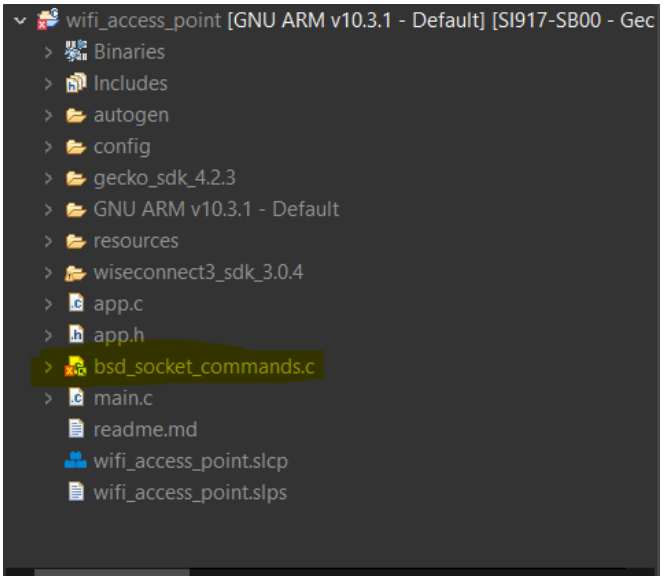
   - In the **Create New File** wizard, select the project folder in which you wish to add the file.
   - Click **Advanced>>**.



   - Select **Link to fie in the file system**
   - Click **Browse**
   - Locate and select the file you wish to add.
   - Click **Finish** to add the file to your project.



- The added file will appear in the project structure in your Project Explorer window.



Summary

API Reference Guide

This is a guide to the application programming interface (API) of the WiSeConnect™ SDK v3.x.

The following sections cover the various categories of APIs provided by the WiSeConnect SDK v3.x along with details of the functions, data types, constants, and callback frameworks within each category:

- **Wireless** APIs provide connectivity and networking features.
  - **BLE** APIs provide Bluetooth Low Energy (BLE) connectivity and protocol features.
  - **Wi-Fi** APIs provide Wi-Fi connectivity and protocol features, including client (station) mode, Access Point (AP) mode, Monitor mode, Wi-Fi Direct, Wi-Fi Protected Setup (WPS), and others.
- **Network Management** APIs provide Wi-Fi network interface and configuration management.
- **Socket** APIs provide various socket programming implementations including Berkeley Software Distribution (BSD) standard sockets, ARM Internet-of-Things (IoT) sockets, and others.
- **SiWx91x Device Management** APIs provide device management features including calibration, low-power settings, firmware upgrades, and others.
- **External Host Interface** APIs provide an interface between an external microcontroller unit (MCU) host and the SiWx91x chipset.
- **SiWx91x MCU** APIs provide access to the device platform including peripheral, driver, and service APIs.
  - **Peripheral** APIs provide access to device peripherals such as the e-fuse, clock, timers, general-purpose input-output (GPIO), and others.
  - **Driver** APIs provide access to peripherals that are external to the device such as buttons, joystick, sensors, and others.
  - **Service** APIs provide high level functions to access device peripherals without requiring low-level knowledge of such peripherals, such as power management, sensor hub, sleep timers, and others.
- **Crypto** APIs provide access to the cryptographic functions of the SiWx91x chipset.
- **Network Protocol** APIs provide implementations of networking protocols.
  - **Ping** APIs provide the ability to ping other Wi-Fi devices using the Internet Control Message Protocol (ICMP).
  - **SNTP** APIs provide Simple Network Time Protocol (SNTP) client features such as time and date synchronization.
- **Application Protocol** APIs provide implementations of application layer protocols.
  - **HTTP** APIs provide HyperText Transfer Protocol (HTTP) client features such as sending GET or POST requests to an HTTP server.
  - **MQTT** APIs provide Message Queue Telemetry Transport (MQTT) protocol client features such as subscribing or publishing to topics on an MQTT broker.
- The **Status Codes** section provides the list of status codes returned by APIs.

Overview

Overview

Introduction

This document describes the SL Wi-Fi API that exposes the low-level Wi-Fi functions.

The SL Wi-Fi API is the new standardized Silicon Labs Wi-Fi API that provides low-level access to Wi-Fi functionality although in most cases the application code doesn't typically interact with the Wi-Fi layer but rely on higher level API and components. Applications can leverage the high-level connectivity functionalities provided by SL network and IoT sockets that use and/or abstract the low-level Wi-Fi functions.

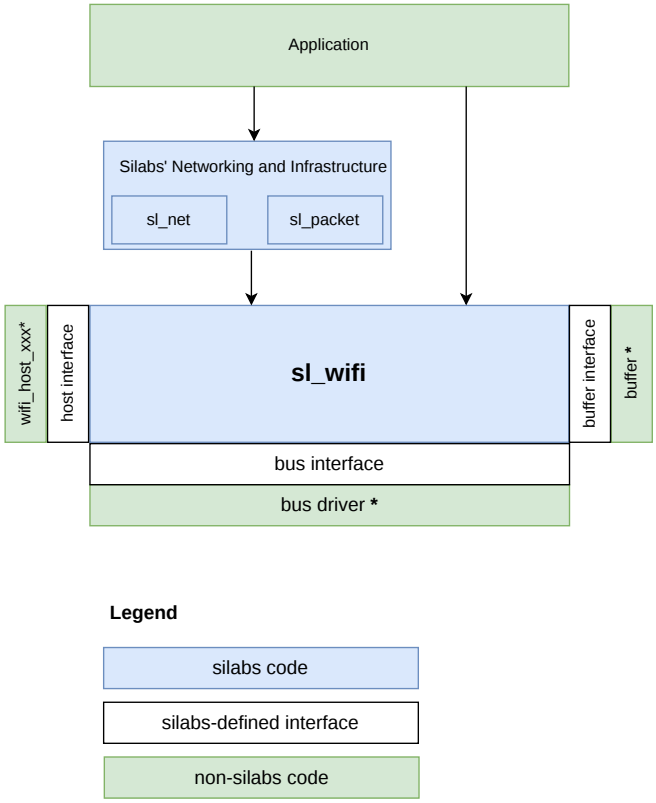
Other use cases such as calibration, tests, or certifications may need access to low-level Wi-Fi functions.

The SL Wi-Fi API is designed to be a minimal abstraction of key Wi-Fi concepts while also being extensible and allow for simplified customer frameworks. One example of a simplified customer framework is the Callback framework that implements the Wi-Fi event handler and exposes a callback mechanism to the customer.

API Structure

The SL Wi-Fi API must function on Non-Silicon Labs hardware and as such relies on two host interface abstractions:

- To interact with the host hardware platform.
- To interact with the host buffer management system (typically provided by the network stack).

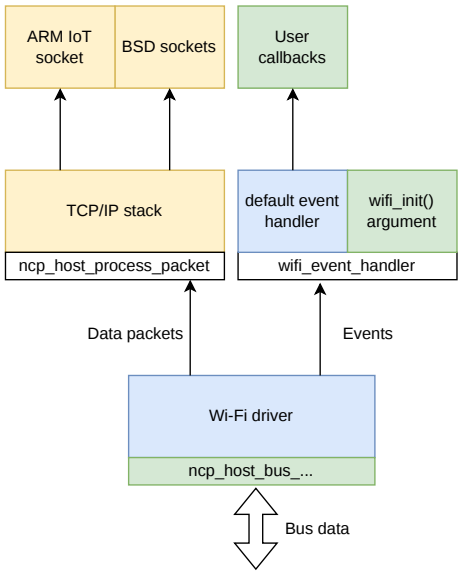


Data flow

Data from the Wi-Fi driver is of two types:

1. Data frames destined for the host network stack.
2. Event data.

All event data is passed through the Wi-Fi event handler that can be defined by the user. Simplified customer frameworks provide an implementation of this event handler and then expose an alternate interface such as a callback mechanism.



APIs

APIs

This section provides a reference to the Wi-Fi API including functions, data types, constants, and callback handlers.

Modules

Functions

Functions

This section provides a reference to Wi-Fi API functions:

- Common functions to initialize and configure Wi-Fi features.
- Radio functions to configure the Wi-Fi radio.
- Scanning functions to scan for Wi-Fi Access Points (APs) in the local area network (LAN).
- Client functions to implement a Wi-Fi client, otherwise known as a station (STA).
- Access Point functions to implement a Wi-FI Access Point (AP).
- Performance Management functions to configure Wi-Fi connectivity performance.
- Wi-Fi Protected Setup functions to implement Wi-Fi Protected Setup (WPS).
- Debugging functions for debugging the Wi-Fi APIs.

Modules

Common

Common

Functions

sl_status_t	sl_wifi_init(const sl_wifi_device_configuration_t *configuration, sl_wifi_device_context_t *device_context, sl_wifi_event_handler_t event_handler) Initialize the Wi-Fi device.
sl_status_t	sl_wifi_deinit(void) De-initialize the Wi-Fi device.
bool	sl_wifi_is_interface_up(sl_wifi_interface_t interface) Check if Wi-Fi interface is up.
sl_status_t	sl_wifi_get_firmware_version(sl_wifi_firmware_version_t *version) Return the firmware version running on the Wi-Fi device.
void	sl_wifi_set_default_interface(sl_wifi_interface_t interface) Set the default interface.
sl_wifi_interface_t	sl_wifi_get_default_interface(void) Get the default interface.
sl_status_t	sl_wifi_get_mac_address(sl_wifi_interface_t interface, sl_mac_address_t *mac) Get the Wi-Fi interface MAC address.
sl_status_t	sl_wifi_set_mac_address(sl_wifi_interface_t interface, const sl_mac_address_t *mac) Set the Wi-Fi interface MAC address.

Function Documentation

sl_wifi_init

> sl_status_t sl_wifi_init (const sl_wifi_device_configuration_t *configuration, sl_wifi_device_context_t *device_context, sl_wifi_event_handler_t event_handler)

Initialize the Wi-Fi device.

Parameters

[in]	configuration	sl_wifi_device_configuration_t object that contains Wi-Fi device configuration.
[in]	device_context	Reserved for future use.
[in]	event_handler	Wi-Fi event handler function of type sl_wifi_event_handler_t.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This function should be called before calling any other sl_wifi functions.

Definition at line 62 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_deinit

```
sl_status_t sl_wifi_deinit (void)
```

De-initialize the Wi-Fi device.

Parameters

N/A		

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `75` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_is_interface_up

```
bool sl_wifi_is_interface_up (sl_wifi_interface_t interface)
```

Check if Wi-Fi interface is up.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- 1. true: interface is up.
  2. false: interface is down.

Definition at line `89` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_firmware_version

```
sl_status_t sl_wifi_get_firmware_version (sl_wifi_firmware_version_t *version)
```

Return the firmware version running on the Wi-Fi device.

Parameters

[out]	version	sl_wifi_firmware_version_t object that contains the version string.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `102` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_set_default_interface

```
void sl_wifi_set_default_interface (sl_wifi_interface_t interface)
```

Set the default interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t

Used by API when SL_WIFI_DEFAULT_INTERFACE is provided.

Definition at line 111 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_get_default_interface

```
sl_wifi_interface_t sl_wifi_get_default_interface (void)
```

Get the default interface.

Parameters

N/A		

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_wifi_interface_t previously set by sl_wifi_set_default_interface

Definition at line 122 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_get_mac_address

```
sl_status_t sl_wifi_get_mac_address (sl_wifi_interface_t interface, sl_mac_address_t *mac)
```

Get the Wi-Fi interface MAC address.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	mac	sl_mac_address_t object that contains the MAC address of the interface.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 137 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_mac_address

```
sl_status_t sl_wifi_set_mac_address (sl_wifi_interface_t interface, const sl_mac_address_t *mac)
```

Set the Wi-Fi interface MAC address.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	mac	sl_mac_address_t object to store the MAC address.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This API is not supported by RS9116, Si917 when called directly due to firmware constraints. Alternatively, sl_wifi_init can be used to configure the MAC address. sl_wifi_init ensures the appropriate state of firmware and calls this API to set MAC address.

Definition at line 152 of file components/protocol/wifi/inc/sl_wifi.h

Radio

Radio

Functions

sl_status_t	sl_wifi_get_max_tx_power(sl_wifi_interface_t interface, sl_wifi_max_tx_power_t *max_tx_power) Get the maximum Wi-Fi transmit power.
sl_status_t	sl_wifi_set_max_tx_power(sl_wifi_interface_t interface, sl_wifi_max_tx_power_t max_tx_power) Set the maximum Wi-Fi transmit power.
sl_status_t	sl_wifi_set_antenna(sl_wifi_interface_t interface, sl_wifi_antenna_t antenna) Set the Wi-Fi antenna for an interface.
sl_status_t	sl_wifi_get_antenna(sl_wifi_interface_t interface, sl_wifi_antenna_t *antenna) Get the Wi-Fi antenna for an interface.
sl_status_t	sl_wifi_get_channel(sl_wifi_interface_t interface, sl_wifi_channel_t *channel) Get the current channel for the given Wi-Fi interface.
sl_status_t	sl_wifi_set_channel(sl_wifi_interface_t interface, sl_wifi_channel_t channel) Set the channel for the given Wi-Fi Access Point interface.
sl_status_t	sl_wifi_set_transmit_rate(sl_wifi_interface_t interface, sl_wifi_rate_protocol_t rate_protocol, sl_wifi_rate_t mask) Set the Wi-Fi transmit rate for the given 802.11 protocol on the specified Wi-Fi interface.
sl_status_t	sl_wifi_get_transmit_rate(sl_wifi_interface_t interface, sl_wifi_rate_protocol_t *rate_protocol, sl_wifi_rate_t *mask) Get the Wi-Fi transmit rate for the given 802.11 protocol on the specified Wi-Fi interface.
sl_status_t	sl_wifi_set_listen_interval(sl_wifi_interface_t interface, sl_wifi_listen_interval_t listen_interval) Set the Wi-Fi client interface listen interval.
sl_status_t	sl_wifi_get_listen_interval(sl_wifi_interface_t interface, sl_wifi_listen_interval_t *listen_interval) Get the Wi-Fi client listen interval.
sl_status_t	sl_wifi_update_gain_table(uint8_t band, uint8_t bandwidth, uint8_t *payload, uint16_t payload_len) Assign the user configurable channel gain values in different regions to the module from user.
sl_status_t	sl_wifi_set_11ax_config(uint8_t guard_interval) Configure the 11ax params.This is a blocking API.

Function Documentation

sl_wifi_get_max_tx_power

```
sl_status_t sl_wifi_get_max_tx_power (sl_wifi_interface_t interface, sl_wifi_max_tx_power_t *max_tx_power)
```

Get the maximum Wi-Fi transmit power.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t

[out]	max_tx_power	A variable that contains current maximum transmit power in dBm as identified by sl_wifi_max_tx_power_t.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This function gets the transmit power for a particular radio interface: SL_WIFI_2_4GHZ_INTERFACE or SL_WIFI_5GHZ_INTERFACE.

Definition at line 177 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_max_tx_power

```
sl_status_t sl_wifi_set_max_tx_power (sl_wifi_interface_t interface, sl_wifi_max_tx_power_t max_tx_power)
```

Set the maximum Wi-Fi transmit power.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	max_tx_power	Max transmission power to set in dBm as identified by sl_wifi_max_tx_power_t

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This function sets the transmit power for a particular radio interface: SL_WIFI_2_4GHZ_INTERFACE or SL_WIFI_5GHZ_INTERFACE. Eg: Setting transmit power for client interface at 2.4GHz will also set transmit power of the AP interface at 2.4GHz.

Definition at line 192 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_antenna

```
sl_status_t sl_wifi_set_antenna (sl_wifi_interface_t interface, sl_wifi_antenna_t antenna)
```

Set the Wi-Fi antenna for an interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	antenna	Antenna to select as identified by sl_wifi_antenna_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 207 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_get_antenna

> sl_status_t sl_wifi_get_antenna (sl_wifi_interface_t interface, sl_wifi_antenna_t *antenna)

Get the Wi-Fi antenna for an interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	antenna	sl_wifi_antenna_t object that contains current antenna selection.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `222` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_channel

> sl_status_t sl_wifi_get_channel (sl_wifi_interface_t interface, sl_wifi_channel_t *channel)

Get the current channel for the given Wi-Fi interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	channel	sl_wifi_channel_t object that contains current channel information.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `237` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_set_channel

> sl_status_t sl_wifi_set_channel (sl_wifi_interface_t interface, sl_wifi_channel_t channel)

Set the channel for the given Wi-Fi Access Point interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	channel	Channel as identified by sl_wifi_channel_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 252 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_transmit_rate

> sl_status_t sl_wifi_set_transmit_rate (sl_wifi_interface_t interface, sl_wifi_rate_protocol_t rate_protocol, sl_wifi_rate_t mask)

Set the Wi-Fi transmit rate for the given 802.11 protocol on the specified Wi-Fi interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	rate_protocol	802.11 protocol as identified by sl_wifi_rate_protocol_t
[in]	mask	Data rate as identified by sl_wifi_rate_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 269 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_get_transmit_rate

> sl_status_t sl_wifi_get_transmit_rate (sl_wifi_interface_t interface, sl_wifi_rate_protocol_t *rate_protocol, sl_wifi_rate_t *mask)

Get the Wi-Fi transmit rate for the given 802.11 protocol on the specified Wi-Fi interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	rate_protocol	802.11 protocol as identified by sl_wifi_rate_protocol_t
[out]	mask	Data rate as identified by sl_wifi_rate_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 288 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_listen_interval

> sl_status_t sl_wifi_set_listen_interval (sl_wifi_interface_t interface, sl_wifi_listen_interval_t listen_interval)

Set the Wi-Fi client interface listen interval.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	listen_interval	sl_wifi_listen_interval_t object

- Pre-conditions:

sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- By default listen interval is set 1000 millisecs. User can call this API to overwrite the value. Si91X implementation allows this API ONLY to be called before calling sl_wifi_connect(), sl_wifi_start_ap(), sl_wifi_start_wps()

Definition at line 308 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_get_listen_interval

> sl_status_t sl_wifi_get_listen_interval (sl_wifi_interface_t interface, sl_wifi_listen_interval_t *listen_interval)

Get the Wi-Fi client listen interval.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	listen_interval	sl_wifi_listen_interval_t object that will contain the current listen interval.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- By default, the listen interval is set to 1000 millisecs.

Definition at line 325 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_update_gain_table

> sl_status_t sl_wifi_update_gain_table (uint8_t band, uint8_t bandwidth, uint8_t *payload, uint16_t payload_len)

Assign the user configurable channel gain values in different regions to the module from user.

Parameters

[in]	band	1 - 2.4GHz, 2 - 5GHz
[in]	bandwidth	0 - 20 MHz, 1 - 40 MHz
[in]	payload	Pass channel gain values for different regions in a given array format.
[in]	payload_len	Max payload length (table size) in 2.4GHz is 128 bytes and in 5GHz is 64 bytes.

- Pre-conditions:
  - This method is used for overwriting default gain tables that are present in firmware.
- Pre-conditions:
  - Customer can load all the three gain tables (i.e., 2.4GHz-20Mhz, 5GHz-20Mhz, 5GHz-40Mhz) one the after other by changing band and bandwidth values.
- Pre-conditions:
  - This is a blocking API.
- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- 1. This frame must be used only by customers who have done FCC/ETSI/TELEC/KCC certification with their own antenna. Silicon Labs is not liable for inappropriate usage of this frame that may result in violation of FCC/ETSI/TELEC/KCC or any certifications.
  2. Internally, firmware maintains two tables: Worldwide table & Region-based table. Worldwide table is populated by the firmware with max power values that the chip can transmit and meet target specs like EVM. Region-based table has a default gain value set.
  3. When certifying with user antenna, the Region has to be set to Worldwide and sweep the power from 0 to 21 dBm. Arrive at a max power level that will pass certifications, especially band-edge.
  4. The FCC/ETSI/TELEC/KCC max power level should be loaded in an end-to-end mode via WLAN User Gain table. This has to be called done for every boot-up as this information is not saved inside the flash. Region-based user gain table sent by the application is copied onto the Region-based table. SoC uses this table in FCC/ETSI/TELEC/KCC to limit the power and to not violate the allowed limits.
  5. For Worldwide region, the firmware uses the Worldwide table for Tx. For other regions (FCC/ETSI/TELEC/KCC), the firmware uses the min value out of the Worldwide & Region-based table for Tx. Also, there will be part to part variation across the chips. Offsets that are estimated during the flow of manufacture will be applied as correction factor during normal mode of operation.
  6. In a 2.4 Ghz band, 40 Mhz is not supported.

Definition at line `362` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_set_11ax_config

```
sl_status_t sl_wifi_set_11ax_config (uint8_t guard_interval)
```

Configure the 11ax params.This is a blocking API.

Parameters

[in]	guard_interval	Period of time delta between two packets in wireless transmission. Valid values : 0 - 3 (0 = 8us, 1 = 16us, 2 = 32us, 3 = 64us).

- Pre-conditions:
  - This API should be called before sl_wifi_connect

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `375` of file `components/protocol/wifi/inc/sl_wifi.h`

Scanning

Functions

sl_status_t	sl_wifi_start_scan(sl_wifi_interface_t interface, const sl_wifi_ssid_t *optional_ssid, const sl_wifi_scan_configuration_t *configuration) Start scanning for Wi-Fi networks.
sl_status_t	sl_wifi_stop_scan(sl_wifi_interface_t interface) Stop Wi-Fi scan.
sl_status_t	sl_wifi_set_advanced_scan_configuration(const sl_wifi_advanced_scan_configuration_t *configuration) Set advanced scan configuration.
sl_status_t	sl_wifi_get_advanced_scan_configuration(sl_wifi_advanced_scan_configuration_t *configuration) Get advanced scan configuration.
sl_status_t	sl_wifi_wait_for_scan_results(sl_wifi_scan_result_t **scan_result_array, uint32_t max_scan_result_count) Wait for current scan to complete and stores the results in the provided array.

Function Documentation

sl_wifi_start_scan

> sl_status_t sl_wifi_start_scan (sl_wifi_interface_t interface, const sl_wifi_ssid_t *optional_ssid, const sl_wifi_scan_configuration_t *configuration)

Start scanning for Wi-Fi networks.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	optional_ssid	Optional SSID of type sl_wifi_ssid_t can be used to scan for a particular Wi-Fi network
[in]	configuration	sl_wifi_scan_configuration_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- For 911x, Advanced scan results are not populated to user. Default Active Channel time is 100 milliseconds. If the user needs to modify the time, sl_wifi_set_advanced_scan_configuration can be called. If the scan_type is not ADV_SCAN then, the time is for foreground scan. Otherwise it is used for back ground scanning.

Definition at line `402` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_stop_scan

sl_status_t sl_wifi_stop_scan (sl_wifi_interface_t interface)

Stop Wi-Fi scan.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- For 911x, sl_wifi_stop_scan is ONLY supported for advanced scan

Definition at line 419 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_advanced_scan_configuration

sl_status_t sl_wifi_set_advanced_scan_configuration (const sl_wifi_advanced_scan_configuration_t *configuration)

Set advanced scan configuration.

Parameters

[in]	configuration	Set advanced scan configuration as identified by sl_wifi_advanced_scan_configuration_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 432 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_get_advanced_scan_configuration

sl_status_t sl_wifi_get_advanced_scan_configuration (sl_wifi_advanced_scan_configuration_t *configuration)

Get advanced scan configuration.

Parameters

[out]	configuration	sl_wifi_advanced_scan_configuration_t object that will contain the current advanced scan configuration.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 445 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_wait_for_scan_results

```
sl_status_t sl_wifi_wait_for_scan_results (sl_wifi_scan_result_t **scan_result_array, uint32_t max_scan_result_count)
```

Wait for current scan to complete and stores the results in the provided array.

Parameters

[in]	scan_result_array	Array of sl_wifi_scan_result_t objects to store the scan results.
[in]	max_scan_result_count	The maximum number of scan result objects that can fit in the scan result array.

- Pre-conditions:
  - This function also returns when the scan result array is full.
- Pre-conditions:
  - Once the scan result array is full, any further scan results will be lost.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This API is not supported in the current release.

Definition at line `465` of file `components/protocol/wifi/inc/sl_wifi.h`

Client

Client

Functions

sl_status_t sl_wifi_connect(sl_wifi_interface_t interface, const sl_wifi_client_configuration_t *access_point, uint32_t timeout_ms)
Connect to the given Wi-Fi AP.

sl_status_t sl_wifi_disconnect(sl_wifi_interface_t interface)
Disconnect the Wi-Fi client interface.

sl_status_t sl_wifi_get_signal_strength(sl_wifi_interface_t interface, int32_t *rssi)
Get current Wi-Fi client's signal strength (RSSI).

sl_status_t sl_wifi_set_roam_configuration(sl_wifi_interface_t interface, sl_wifi_roam_configuration_t *roam_configuration)
Set the Wi-Fi roaming configuration.

sl_status_t sl_wifi_get_roam_configuration(sl_wifi_interface_t interface, sl_wifi_roam_configuration_t *roam_configuration)
Get the Wi-Fi roaming configuration.

sl_status_t sl_wifi_test_client_configuration(sl_wifi_interface_t interface, const sl_wifi_client_configuration_t *ap, uint32_t timeout_ms)
Verify the Wi-Fi client configuration is valid and available.

sl_status_t sl_wifi_set_certificate(uint8_t certificate_type, const uint8_t *buffer, uint32_t certificate_length)
Load the certificate into the device.

sl_status_t sl_wifi_set_certificate_with_index(uint8_t certificate_type, uint8_t certificate_index, uint8_t *buffer, uint32_t certificate_length)
Load the certificate into the device.

sl_status_t sl_wifi_set_advanced_client_configuration(sl_wifi_interface_t interface, const sl_wifi_advanced_client_configuration_t *configuration)
Set the advanced configuration options of a client interface.

sl_status_t sl_wifi_send_raw_data_frame(sl_wifi_interface_t interface, const void *data, uint16_t data_length)
Send raw data frame.

sl_status_t sl_wifi_enable_target_wake_time(sl_wifi_twt_request_t *twt_req)
Configure TWT parameters.

sl_status_t sl_wifi_disable_target_wake_time(sl_wifi_twt_request_t *twt_req)
Configure TWT parameters.

sl_status_t sl_wifi_target_wake_time_auto_selection(sl_wifi_twt_selection_t *twt_selection_req)
Calculates and configures TWT parameters based on the given inputs.

sl_status_t sl_wifi_reschedule_twt(uint8_t flow_id, sl_wifi_reschedule_twt_action_t twt_action, uint64_t suspend_duration)
Suspends the TWT agreement corresponding to given flow id and resumes it when suspend duration expires.

sl_status_t sl_wifi_filter_broadcast(uint16_t beacon_drop_threshold, uint8_t filter_bcast_in_tim, uint8_t filter_bcast_tim_till_next_cmd)
Send Filter Broadcast Request frame.

sl_status_t	sl_wifi_get_pairwise_master_key(sl_wifi_interface_t interface, const uint8_t type, const sl_wifi_ssid_t *ssid, const char *pre_shared_key, uint8_t *pairwise_master_key) Generate PMK if PSK and SSID are provided.

Function Documentation

sl_wifi_connect

```
sl_status_t sl_wifi_connect (sl_wifi_interface_t interface, const sl_wifi_client_configuration_t *access_point, uint32_t timeout_ms)
```

Connect to the given Wi-Fi AP.

Parameters

[in]	interface	Wi-Fi client interface as identified by sl_wifi_interface_t
[in]	access_point	sl_wifi_client_configuration_t object that contains the details of Access Point.
[in]	timeout_ms	Timeout value in milliseconds. The function will abort and return when the timeout timer expires. A value of 0 indicates an asynchronous action. TBD: asynchronous action page.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- If channel, band, and BSSID are provided, this API will attempt to connect without scanning. If security_type is SL_WIFI_WPA3 then SL_SI91X_JOIN_FEAT_MFP_CAPABLE_REQUIRED join feature is enabled internally by SDK. If security_type is SL_WIFI_WPA3_TRANSITION then SL_SI91X_JOIN_FEAT_MFP_CAPABLE_REQUIRED join feature is disabled and SL_SI91X_JOIN_FEAT_MFP_CAPABLE_ONLY join feature is enabled internally by SDK." Default Active Channel time is 100 milliseconds. If the user needs to modify the time, sl_wifi_set_advanced_scan_configuration can be called. Default Auth Association timeout is 300 milliseconds. If the user needs to modify the time, sl_wifi_set_advanced_client_configuration can be called. Default Keep Alive timeout is 30 milliseconds. If the user needs to modify the time, sl_wifi_set_advanced_client_configuration can be called.

Definition at line `497` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_disconnect

```
sl_status_t sl_wifi_disconnect (sl_wifi_interface_t interface)
```

Disconnect the Wi-Fi client interface.

Parameters

[in]	interface	Wi-Fi client interface as identified by sl_wifi_interface_t

- Pre-conditions:
  - sl_wifi_connect should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `512` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_signal_strength

> sl_status_t sl_wifi_get_signal_strength (sl_wifi_interface_t interface, int32_t *rssi)

Get current Wi-Fi client's signal strength (RSSI).

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	rssi	signal strength (RSSI) in dBm.

- Pre-conditions:
  - sl_wifi_connect should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `527` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_set_roam_configuration

> sl_status_t sl_wifi_set_roam_configuration (sl_wifi_interface_t interface, sl_wifi_roam_configuration_t *roam_configuration)

Set the Wi-Fi roaming configuration.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	roam_configuration	sl_wifi_roam_configuration_t object to store Wi-Fi roaming configuration.

- Pre-conditions:
  - sl_wifi_set_advanced_scan_configuration should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- For si91x chips, following ranges are valid: trigger_level: [-10, -100] , trigger_level_change: [0, 90]

Definition at line `546` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_roam_configuration

> sl_status_t sl_wifi_get_roam_configuration (sl_wifi_interface_t interface, sl_wifi_roam_configuration_t *roam_configuration)

Get the Wi-Fi roaming configuration.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	roam_configuration	sl_wifi_roam_configuration_t object that will contain the current roam configuration.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This API is not yet implemented.

Definition at line 561 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_test_client_configuration

> sl_status_t sl_wifi_test_client_configuration (sl_wifi_interface_t interface, const sl_wifi_client_configuration_t *ap, uint32_t timeout_ms)

Verify the Wi-Fi client configuration is valid and available.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	ap	sl_wifi_client_configuration_t object that contains the details of Access Point.
[in]	timeout_ms	Timeout value in milliseconds. The function will abort and return when the timeout timer expires. A value of 0 indicates an asynchronous action.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 580 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_certificate

> sl_status_t sl_wifi_set_certificate (uint8_t certificate_type, const uint8_t *buffer, uint32_t certificate_length)

Load the certificate into the device.

Parameters

[in]	certificate_type	Certificate type being set
[in]	buffer	Pointer to buffer containing the certificate.
[in]	certificate_length	Length of certificate buffer data.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 599 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_certificate_with_index

> sl_status_t sl_wifi_set_certificate_with_index (uint8_t certificate_type, uint8_t certificate_index, uint8_t *buffer, uint32_t certificate_length)

Load the certificate into the device.

Parameters

[in]	certificate_type	Certificate type being set.
[in]	certificate_index	Certificate to be loaded in specified index.
[in]	buffer	Pointer to buffer containing the certificate.
[in]	certificate_length	Length of certificate buffer data.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 618 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_set_advanced_client_configuration

> sl_status_t sl_wifi_set_advanced_client_configuration (sl_wifi_interface_t interface, const sl_wifi_advanced_client_configuration_t *configuration)

Set the advanced configuration options of a client interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	configuration	Wi-Fi client advanced configuration. See sl_wifi_advanced_client_configuration_t

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 632 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_send_raw_data_frame

> sl_status_t sl_wifi_send_raw_data_frame (sl_wifi_interface_t interface, const void *data, uint16_t data_length)

Send raw data frame.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	data	Data buffer.
[in]	data_length	length of the data.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 650 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_enable_target_wake_time

> sl_status_t sl_wifi_enable_target_wake_time (sl_wifi_twt_request_t *twt_req)

Configure TWT parameters.

Parameters

[in]	twt_req	Configurable TWT parameters specified in sl_wifi_twt_request_t.

Enables a TWT session. This is blocking API.

- Pre-conditions:
  - sl_wifi_connect should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `663` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_disable_target_wake_time

> sl_status_t sl_wifi_disable_target_wake_time (sl_wifi_twt_request_t *twt_req)

Configure TWT parameters.

Parameters

[in]	twt_req	Configurable TWT parameters specified in sl_wifi_twt_request_t.

Disables a TWT session. This is blocking API.

- Pre-conditions:
  - sl_wifi_enable_target_wake_time should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `676` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_target_wake_time_auto_selection

> sl_status_t sl_wifi_target_wake_time_auto_selection (sl_wifi_twt_selection_t *twt_selection_req)

Calculates and configures TWT parameters based on the given inputs.

Parameters

[in]	twt_selection_req	sl_wifi_twt_selection_t object containing configurable TWT selection parameters.

Enables or disables a TWT session. This is blocking API.

- Pre-conditions:
  - sl_wifi_connect should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 689 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_reschedule_twt

> sl_status_t sl_wifi_reschedule_twt (uint8_t flow_id, sl_wifi_reschedule_twt_action_t twt_action, uint64_t suspend_duration)

Suspends the TWT agreement corresponding to given flow id and resumes it when suspend duration expires.

Parameters

[in]	flow_id	Flow id of the twt agreement.
[in]	twt_action	sl_wifi_reschedule_twt_action_t specifying different actions that can be taken in relation to rescheduling TWT.
[in]	suspend_duration	Time interval until which twt agreement is suspended, value taken in milliseconds.

This API performs following actions on TWT agreement: SL_WIFI_SUSPEND_INDEFINITELY, SL_WIFI_RESUME_IMMEDIATELY, SL_WIFI_SUSPEND_FOR_DURATION. **Note**

- The reschedule TWT actions are valid till the end of current TWT agreement. If the TWT agreement is terminated (TWT tear down or wlan disconnection), these actions are not retained. To reapply these actions upon new TWT agreement, the user must re-issue the command.
- Pre-conditions:
  - sl_wifi_connect should be called before this API.

The table below outlines the valid values for TWT actions and their corresponding suspend durations:

twt_action	Valid values for suspend duration
SL_WIFI_SUSPEND_INDEFINITELY	0
SL_WIFI_RESUME_IMMEDIATELY	0
SL_WIFI_SUSPEND_FOR_DURATION	1 to 86400000

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 717 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_filter_broadcast

> sl_status_t sl_wifi_filter_broadcast (uint16_t beacon_drop_threshold, uint8_t filter_bcast_in_tim, uint8_t filter_bcast_tim_till_next_cmd)

Send Filter Broadcast Request frame.

Parameters

[in]	beacon_drop_threshold	The amount of time that FW waits to receive full beacon. Default value is 5000ms.
[in]	filter_bcast_in_tim	If this bit is set, then from the next dtim any broadcast data pending bit in TIM indicated will be ignored valid values: 0 - 1.
[in]	filter_bcast_tim_till_next_cmd	0 - filter_bcast_in_tim is valid till disconnect of the STA. 1 - filter_bcast_in_tim is valid till next update by giving the same command.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 737 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_get_pairwise_master_key

> sl_status_t sl_wifi_get_pairwise_master_key (sl_wifi_interface_t interface, const uint8_t type, const sl_wifi_ssid_t *ssid, const char *pre_shared_key, uint8_t *pairwise_master_key)

Generate PMK if PSK and SSID are provided.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	type	Possible values of this field are 1, 2, and 3, but we only pass 3 for generation of PMK.
[in]	ssid	SSID of type sl_wifi_ssid_t has the SSID of the access point
[in]	pre_shared_key	Expected parameters are pre-shared key(PSK) of the access point
[in]	pairwise_master_key	PMK array

This is a blocking API.

- Pre-conditions:
  - This API should be called after sl_wifi_init and called before sl_wifi_connect.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 760 of file components/protocol/wifi/inc/sl_wifi.h

Access Point

Access Point

Functions

sl_status_t sl_wifi_start_ap(sl_wifi_interface_t interface, const sl_wifi_ap_configuration_t *configuration)
Start a Wi-Fi access point (AP) interface.

sl_status_t sl_wifi_set_ap_configuration(sl_wifi_interface_t interface, const sl_wifi_ap_configuration_t *configuration)
Set the configuration of a running Wi-Fi access point (AP).

sl_status_t sl_wifi_get_ap_configuration(sl_wifi_interface_t interface, sl_wifi_ap_configuration_t *configuration)
Get the configuration of a Wi-Fi AP interface.

sl_status_t sl_wifi_set_advanced_ap_configuration(sl_wifi_interface_t interface, const sl_wifi_advanced_ap_configuration_t *configuration)
Set the advanced configuration options of a running Wi-Fi access point (AP).

sl_status_t sl_wifi_get_advanced_ap_configuration(sl_wifi_interface_t interface, const sl_wifi_advanced_ap_configuration_t *configuration)
Get the advanced configuration options of a running Wi-Fi access point interface.

sl_status_t sl_wifi_stop_ap(sl_wifi_interface_t interface)
Stop Wi-Fi access point.

sl_status_t sl_wifi_disconnect_ap_client(sl_wifi_interface_t interface, const sl_mac_address_t *mac, sl_wifi_deauth_reason_t reason)
De-authenticate Wi-Fi client with the given MAC address.

sl_status_t sl_wifi_get_ap_client_info(sl_wifi_interface_t interface, sl_wifi_client_info_response_t *client_info)
Return the Wi-Fi client information of all clients connected to the AP.

sl_status_t sl_wifi_get_ap_client_list(sl_wifi_interface_t interface, uint16_t client_list_count, sl_mac_address_t *client_list)
Return a list of Wi-Fi clients connected to the Wi-Fi access point.

sl_status_t sl_wifi_get_ap_client_count(sl_wifi_interface_t interface, uint32_t *client_count)
Provide the number of Wi-Fi clients connected to the Wi-Fi access point.

Function Documentation

sl_wifi_start_ap

> sl_status_t sl_wifi_start_ap (sl_wifi_interface_t interface, const sl_wifi_ap_configuration_t *configuration)

Start a Wi-Fi access point (AP) interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	configuration	Wi-Fi AP configuration. See sl_wifi_ap_configuration_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- For AP mode with WPA3 security, only SAE-H2E method is supported. SAE Hunting and pecking method is not supported. TKIP encryption mode is not supported. Encryption mode is automatically configured to RSI_CCMP. PMKSA is not supported in WPA3 AP mode.

Definition at line `790` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_set_ap_configuration

> sl_status_t sl_wifi_set_ap_configuration (sl_wifi_interface_t interface, const sl_wifi_ap_configuration_t *configuration)

Set the configuration of a running Wi-Fi access point (AP).

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	configuration	Wi-Fi AP configuration. See sl_wifi_ap_configuration_t

If the new configuration modifies vital settings such as SSID or security, the AP will be stopped and restarted automatically.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This API is not yet implemented.

Definition at line `805` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_ap_configuration

> sl_status_t sl_wifi_get_ap_configuration (sl_wifi_interface_t interface, sl_wifi_ap_configuration_t *configuration)

Get the configuration of a Wi-Fi AP interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	configuration	sl_wifi_ap_configuration_t object that contains the AP configuration.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `821` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_set_advanced_ap_configuration

> sl_status_t sl_wifi_set_advanced_ap_configuration (sl_wifi_interface_t interface, const sl_wifi_advanced_ap_configuration_t *configuration)

Set the advanced configuration options of a running Wi-Fi access point (AP).

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	configuration	Wi-Fi AP advanced configuration. See sl_wifi_advanced_ap_configuration_t

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This API is not yet implemented.

Definition at line `835` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_advanced_ap_configuration

> sl_status_t sl_wifi_get_advanced_ap_configuration (sl_wifi_interface_t interface, const sl_wifi_advanced_ap_configuration_t *configuration)

Get the advanced configuration options of a running Wi-Fi access point interface.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	configuration	sl_wifi_advanced_ap_configuration_t object that will contain the AP advanced configuration.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This API is not yet implemented.

Definition at line `850` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_stop_ap

> sl_status_t sl_wifi_stop_ap (sl_wifi_interface_t interface)

Stop Wi-Fi access point.

Parameters

[in]	interface	Wi-Fi Access Point interface as identified by sl_wifi_interface_t

- Pre-conditions:
  - sl_wifi_start_ap should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `864` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_disconnect_ap_client

```
sl_status_t sl_wifi_disconnect_ap_client (sl_wifi_interface_t interface, const sl_mac_address_t *mac,
sl_wifi_deauth_reason_t reason)
```

De-authenticate Wi-Fi client with the given MAC address.

Parameters

[in]	interface	Wi-Fi Access Point interface as identified by sl_wifi_interface_t
[in]	mac	Wi-Fi client's MAC address of type sl_mac_address_t
[in]	reason	Reason for de-authentication as specified in sl_wifi_deauth_reason_t

- Pre-conditions:
  - sl_wifi_start_ap should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- Client interfaces are not supported.

Definition at line `883` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_ap_client_info

```
sl_status_t sl_wifi_get_ap_client_info (sl_wifi_interface_t interface, sl_wifi_client_info_response_t *client_info)
```

Return the Wi-Fi client information of all clients connected to the AP.

Parameters

[in]	interface	Wi-Fi Access Point interface as identified by sl_wifi_interface_t
[out]	client_info	sl_wifi_client_info_response_t object to store the client info.

- Pre-conditions:
  - sl_wifi_start_ap should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- Client interfaces are not supported.

Definition at line `902` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_ap_client_list

```
sl_status_t sl_wifi_get_ap_client_list (sl_wifi_interface_t interface, uint16_t client_list_count, sl_mac_address_t *client_list)
```

Return a list of Wi-Fi clients connected to the Wi-Fi access point.

Parameters

[in]	interface	Wi-Fi Access Point interface as identified by sl_wifi_interface_t
[in]	client_list_count	The number of sl_mac_address_t objects the client_list can store.

[out]	client_list	A pointer to an array of client_list_count number of sl_mac_address_t objects where the client list will be copied to.

- Pre-conditions:
  - sl_wifi_start_ap should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- Client interfaces are not supported.

Definition at line 921 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_get_ap_client_count

```
sl_status_t sl_wifi_get_ap_client_count (sl_wifi_interface_t interface, uint32_t *client_count)
```

Provide the number of Wi-Fi clients connected to the Wi-Fi access point.

Parameters

[in]	interface	Wi-Fi Access Point interface as identified by sl_wifi_interface_t
[out]	client_count	A uint32_t pointer that will store the number of associated clients.

- Pre-conditions:
  - sl_wifi_start_ap should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- Client interfaces are not supported.

Definition at line 940 of file components/protocol/wifi/inc/sl_wifi.h

Performance Management

Performance Management

Functions

sl_status_t	sl_wifi_set_performance_profile(const sl_wifi_performance_profile_t *profile) Set Wi-Fi performance profile.
sl_status_t	sl_wifi_get_performance_profile(sl_wifi_performance_profile_t *profile) Get Wi-Fi performance profile.

Function Documentation

sl_wifi_set_performance_profile

```
sl_status_t sl_wifi_set_performance_profile (const sl_wifi_performance_profile_t *profile)
```

Set Wi-Fi performance profile.

Parameters

[in]	profile	Wi-Fi performance profile as indicated by sl_wifi_performance_profile_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- For SI91x chips Enhanced MAX PSP is supported when profile is set to ASSOCIATED_POWER_SAVE_LOW_LATENCY and SL_SI91X_ENABLE_ENHANCED_MAX_PSP bit is enabled in config feature bitmap

Definition at line `962` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_get_performance_profile

```
sl_status_t sl_wifi_get_performance_profile (sl_wifi_performance_profile_t *profile)
```

Get Wi-Fi performance profile.

Parameters

[out]	profile	Wi-Fi performance profile as indicated by sl_wifi_performance_profile_t

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 975 of file components/protocol/wifi/inc/sl_wifi.h

Wi-Fi Protected Setup

Wi-Fi Protected Setup

Functions

sl_status_t	sl_wifi_generate_wps_pin(sl_wifi_wps_pin_t *response) Generate Wi-Fi Protected Setup (WPS) pin.
sl_status_t	sl_wifi_start_wps(sl_wifi_interface_t interface, sl_wifi_wps_mode_t mode, const sl_wifi_wps_pin_t *optional_wps_pin) Start Wi-Fi Protected Setup (WPS).
sl_status_t	sl_wifi_stop_wps(sl_wifi_interface_t interface) Stop current running Wi-Fi Protected Setup (WPS).

Function Documentation

sl_wifi_generate_wps_pin

```
sl_status_t sl_wifi_generate_wps_pin (sl_wifi_wps_pin_t *response)
```

Generate Wi-Fi Protected Setup (WPS) pin.

Parameters

[out]	response	sl_wifi_wps_pin_t object that will contain the WPS pin.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 1056 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_start_wps

```
sl_status_t sl_wifi_start_wps (sl_wifi_interface_t interface, sl_wifi_wps_mode_t mode, const sl_wifi_wps_pin_t *optional_wps_pin)
```

Start Wi-Fi Protected Setup (WPS).

Parameters

[in]	interface	Wi-Fi Access Point interface as identified by sl_wifi_interface_t
[in]	mode	WPS mode as identified by sl_wifi_wps_mode_t
[in]	optional_wps_pin	WPS pin object sl_wifi_wps_pin_t when SL_WIFI_WPS_PIN_MODE is used.

- Pre-conditions:
  - sl_wifi_start_ap should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- Client interfaces are not supported.

Definition at line 1075 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_stop_wps

```
sl_status_t sl_wifi_stop_wps (sl_wifi_interface_t interface)
```

Stop current running Wi-Fi Protected Setup (WPS).

Parameters

[in]	interface	Wi-Fi Access Point interface as identified by sl_wifi_interface_t

- Pre-conditions:
  - sl_wifi_start_wps should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- Client interfaces are not supported.

Definition at line 1092 of file components/protocol/wifi/inc/sl_wifi.h

Debugging

Debugging

Functions

sl_status_t	sl_wifi_get_statistics(sl_wifi_interface_t interface, sl_wifi_statistics_t *statistics) Return Wi-Fi operational statistics.
sl_status_t	sl_wifi_start_statistic_report(sl_wifi_interface_t interface, sl_wifi_channel_t channel) Start collecting statistical data.
sl_status_t	sl_wifi_stop_statistic_report(sl_wifi_interface_t interface) Stop collecting statistical data.

Function Documentation

sl_wifi_get_statistics

```
sl_status_t sl_wifi_get_statistics (sl_wifi_interface_t interface, sl_wifi_statistics_t *statistics)
```

Return Wi-Fi operational statistics.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[out]	statistics	sl_wifi_statistics_t object that contains Wi-Fi statistics.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `1114` of file `components/protocol/wifi/inc/sl_wifi.h`

sl_wifi_start_statistic_report

```
sl_status_t sl_wifi_start_statistic_report (sl_wifi_interface_t interface, sl_wifi_channel_t channel)
```

Start collecting statistical data.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t
[in]	channel	Provides the statistics report on the channel specified by sl_wifi_channel_t.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 1129 of file components/protocol/wifi/inc/sl_wifi.h

sl_wifi_stop_statistic_report

```
sl_status_t sl_wifi_stop_statistic_report (sl_wifi_interface_t interface)
```

Stop collecting statistical data.

Parameters

[in]	interface	Wi-Fi interface as identified by sl_wifi_interface_t

- Pre-conditions:
  - sl_wifi_start_statistic_report should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 1142 of file components/protocol/wifi/inc/sl_wifi.h

Types

Types

This section provides a reference to Wi-Fi API data types.

Modules

sl_wifi_client_info_response_t

sl_wifi_max_tx_power_t

Typedefs

typedef sl_status_t(*	sl_wifi_event_handler_t)(sl_wifi_event_t event, sl_wifi_buffer_t *buffer) Generic callback for Wi-Fi event.
typedef uint32_t	sl_wifi_credential_id_t Wi-Fi credential handle.

Typedef Documentation

sl_wifi_event_handler_t

```
sl_wifi_event_handler_t )(sl_wifi_event_t event, sl_wifi_buffer_t *buffer)
```

Generic callback for Wi-Fi event.

Parameters

N/A	event	Wi-Fi event of type sl_wifi_event_t
N/A	buffer	Wi-Fi buffer containing information related to the event of type sl_wifi_buffer_t

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- In case of event failure, SL_WIFI_FAIL_EVENT_STATUS_INDICATION bit is set in event, data will be of type sl_status_t and data_length can be ignored

Definition at line 38 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_credential_id_t

```
typedef uint32_t sl_wifi_credential_id_t
```

Wi-Fi credential handle.

Definition at line 41 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_buffer_t

Wi-Fi buffer.

Public Attributes

sl_slist_node_t	node
uint32_t	length
uint8_t	data

Public Attribute Documentation

node

```
sl_slist_node_t sl_wifi_buffer_t::node
```

Definition at line 28 of file components/protocol/wifi/inc/sl_wifi_host_interface.h

length

```
uint32_t sl_wifi_buffer_t::length
```

Definition at line 29 of file components/protocol/wifi/inc/sl_wifi_host_interface.h

data

```
uint8_t sl_wifi_buffer_t::data[]
```

Definition at line 30 of file components/protocol/wifi/inc/sl_wifi_host_interface.h

sl_wifi_channel_t

Wi-Fi channel.

Public Attributes

uint16_t	channel Channel number.
sl_wifi_band_t	band Wi-Fi Radio Band.
sl_wifi_bandwidth_t	bandwidth Channel bandwidth.

Public Attribute Documentation

channel

```
uint16_t sl_wifi_channel_t::channel
```

Channel number.

Definition at line 45 of file components/protocol/wifi/inc/sl_wifi_types.h

band

```
sl_wifi_band_t sl_wifi_channel_t::band
```

Wi-Fi Radio Band.

Definition at line 46 of file components/protocol/wifi/inc/sl_wifi_types.h

bandwidth

```
sl_wifi_bandwidth_t sl_wifi_channel_t::bandwidth
```

Channel bandwidth.

Definition at line 47 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_ssid_t

SSID data structure.

Public Attributes

uint8_t	value SSID value.
uint8_t	length Length of SSID.

Public Attribute Documentation

value

```
uint8_t sl_wifi_ssid_t::value[32]
```

SSID value.

Definition at line 52 of file components/protocol/wifi/inc/sl_wifi_types.h

length

```
uint8_t sl_wifi_ssid_t::length
```

Length of SSID.

Definition at line 53 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_roam_configuration_t

Roam configuration structure.

Public Attributes

int32_t	trigger_level RSSI level to trigger roam algorithm, Setting the value to SL_WIFI_NEVER_ROAM will disable roaming configuration.
uint32_t	trigger_level_change RSSI level delta change to trigger roam algorithm.

Public Attribute Documentation

trigger_level

```
int32_t sl_wifi_roam_configuration_t::trigger_level
```

RSSI level to trigger roam algorithm, Setting the value to SL_WIFI_NEVER_ROAM will disable roaming configuration.

Definition at line 59 of file components/protocol/wifi/inc/sl_wifi_types.h

trigger_level_change

```
uint32_t sl_wifi_roam_configuration_t::trigger_level_change
```

RSSI level delta change to trigger roam algorithm.

Definition at line 60 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_firmware_version_t

Wi-Fi firmware version information.

Public Attributes

uint8_t	chip_id Chip ID.
uint8_t	rom_id ROM ID.
uint8_t	major Major version number.
uint8_t	minor Minor version number.
uint8_t	security_version Security enabled or disabled.
uint8_t	patch_num Patch number.
uint8_t	customer_id Customer ID.
uint16_t	build_num Build number.

Public Attribute Documentation

chip_id

```
uint8_t sl_wifi_firmware_version_t::chip_id
```

Chip ID.

Definition at line `65` of file `components/protocol/wifi/inc/sl_wifi_types.h`

rom_id

```
uint8_t sl_wifi_firmware_version_t::rom_id
```

ROM ID.

Definition at line `66` of file `components/protocol/wifi/inc/sl_wifi_types.h`

major

```
uint8_t sl_wifi_firmware_version_t::major
```

Major version number.

Definition at line 67 of file components/protocol/wifi/inc/sl_wifi_types.h

minor

```
uint8_t sl_wifi_firmware_version_t::minor
```

Minor version number.

Definition at line 68 of file components/protocol/wifi/inc/sl_wifi_types.h

security_version

```
uint8_t sl_wifi_firmware_version_t::security_version
```

Security enabled or disabled.

Definition at line 69 of file components/protocol/wifi/inc/sl_wifi_types.h

patch_num

```
uint8_t sl_wifi_firmware_version_t::patch_num
```

Patch number.

Definition at line 70 of file components/protocol/wifi/inc/sl_wifi_types.h

customer_id

```
uint8_t sl_wifi_firmware_version_t::customer_id
```

Customer ID.

Definition at line 71 of file components/protocol/wifi/inc/sl_wifi_types.h

build_num

```
uint16_t sl_wifi_firmware_version_t::build_num
```

Build number.

Definition at line 72 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_scan_result_t

Wi-Fi scan result.

Public Attributes

uint32_t	scan_count Number of available scan results.
uint32_t	reserved Reserved.
uint8_t	rf_channel Channel number of the AP.
uint8_t	security_mode Security mode of the AP.
uint8_t	rssi_val RSSI value of the AP.
uint8_t	network_type AP network type.
uint8_t	ssid SSID of the AP.
uint8_t	bssid BSSID of the AP.
uint8_t	reserved Reserved bytes for future use.
struct sl_wifi_scan_result_t::@0	scan_info Array of scan result data.

Public Attribute Documentation

scan_count

```
uint32_t sl_wifi_scan_result_t::scan_count
```

Number of available scan results.

Definition at line 77 of file components/protocol/wifi/inc/sl_wifi_types.h

reserved

```
uint32_t sl_wifi_scan_result_t::reserved
```

Reserved.

Definition at line 78 of file components/protocol/wifi/inc/sl_wifi_types.h

rf_channel

```
uint8_t sl_wifi_scan_result_t::rf_channel
```

Channel number of the AP.

Definition at line 80 of file components/protocol/wifi/inc/sl_wifi_types.h

security_mode

```
uint8_t sl_wifi_scan_result_t::security_mode
```

Security mode of the AP.

Definition at line 81 of file components/protocol/wifi/inc/sl_wifi_types.h

rssi_val

```
uint8_t sl_wifi_scan_result_t::rssi_val
```

RSSI value of the AP.

Definition at line 82 of file components/protocol/wifi/inc/sl_wifi_types.h

network_type

```
uint8_t sl_wifi_scan_result_t::network_type
```

AP network type.

Definition at line 83 of file components/protocol/wifi/inc/sl_wifi_types.h

ssid

```
uint8_t sl_wifi_scan_result_t::ssid[34]
```

SSID of the AP.

Definition at line 84 of file components/protocol/wifi/inc/sl_wifi_types.h

bssid

```
uint8_t sl_wifi_scan_result_t::bssid[6]
```

BSSID of the AP.

Definition at line 85 of file components/protocol/wifi/inc/sl_wifi_types.h

reserved

```
uint8_t sl_wifi_scan_result_t::reserved[2]
```

Reserved bytes for future use.

Definition at line 86 of file components/protocol/wifi/inc/sl_wifi_types.h

scan_info

```
struct sl_wifi_scan_result_t::@0 sl_wifi_scan_result_t::scan_info[]
```

Array of scan result data.

Definition at line 87 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_scan_configuration_t

Wi-Fi scan configuration.

Note

- channel_bitmap_2g4 uses the lower 14 bits to represent channels from 1 - 14 where channel 1 = (1 << 0), channel 2 = (1 << 1), etc

Public Attributes

sl_wifi_scan_type_t	type Scan type to be configured.
uint32_t	flags Flags for the scan configuration.
uint32_t	periodic_scan_interval Duration in milliseconds between periodic scans.
uint16_t	channel_bitmap_2g4 Bitmap of selected 2.4GHz channels.
uint32_t	channel_bitmap_5g Bitmap of selected 5GHz channels.

Public Attribute Documentation

type

```
sl_wifi_scan_type_t sl_wifi_scan_configuration_t::type
```

Scan type to be configured.

Definition at line `94` of file `components/protocol/wifi/inc/sl_wifi_types.h`

flags

```
uint32_t sl_wifi_scan_configuration_t::flags
```

Flags for the scan configuration.

Definition at line `95` of file `components/protocol/wifi/inc/sl_wifi_types.h`

periodic_scan_interval

```
uint32_t sl_wifi_scan_configuration_t::periodic_scan_interval
```

Duration in milliseconds between periodic scans.

Definition at line 96 of file components/protocol/wifi/inc/sl_wifi_types.h

channel_bitmap_2g4

```
uint16_t sl_wifi_scan_configuration_t::channel_bitmap_2g4
```

Bitmap of selected 2.4GHz channels.

Definition at line 97 of file components/protocol/wifi/inc/sl_wifi_types.h

channel_bitmap_5g

```
uint32_t sl_wifi_scan_configuration_t::channel_bitmap_5g[8]
```

Bitmap of selected 5GHz channels.

Definition at line 98 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_advanced_scan_configuration_t

Wi-Fi advanced scan configuration options.

Public Attributes

int32_t	trigger_level RSSI level to trigger advanced scan.
uint32_t	trigger_level_change RSSI level to trigger advanced scan.
uint16_t	active_channel_time Time spent on each channel when performing active scan (milliseconds).
uint16_t	passive_channel_time Time spent on each channel when performing passive scan (milliseconds).
uint8_t	enable_instant_scan Flag to start advanced scan immediately.
uint8_t	enable_multi_probe Flag to indicate to send multiple probes to AP. If set. a probe request would be sent to all access points in addition to connected SSID.

Public Attribute Documentation

trigger_level

```
int32_t sl_wifi_advanced_scan_configuration_t::trigger_level
```

RSSI level to trigger advanced scan.

Definition at line 103 of file components/protocol/wifi/inc/sl_wifi_types.h

trigger_level_change

```
uint32_t sl_wifi_advanced_scan_configuration_t::trigger_level_change
```

RSSI level to trigger advanced scan.

Definition at line 104 of file components/protocol/wifi/inc/sl_wifi_types.h

active_channel_time

```
uint16_t sl_wifi_advanced_scan_configuration_t::active_channel_time
```

Time spent on each channel when performing active scan (milliseconds).

Definition at line 105 of file components/protocol/wifi/inc/sl_wifi_types.h

passive_channel_time

> uint16_t sl_wifi_advanced_scan_configuration_t::passive_channel_time

Time spent on each channel when performing passive scan (milliseconds).

Definition at line 106 of file components/protocol/wifi/inc/sl_wifi_types.h

enable_instant_scan

> uint8_t sl_wifi_advanced_scan_configuration_t::enable_instant_scan

Flag to start advanced scan immediately.

Definition at line 107 of file components/protocol/wifi/inc/sl_wifi_types.h

enable_multi_probe

> uint8_t sl_wifi_advanced_scan_configuration_t::enable_multi_probe

Flag to indicate to send multiple probes to AP. If set. a probe request would be sent to all access points in addition to connected SSID.

Definition at line 109 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_ap_configuration_t

Wi-Fi Access Point configuration.

Public Attributes

sl_wifi_ssid_t	ssid SSID.
sl_wifi_security_t	security Security mode.
sl_wifi_encryption_t	encryption Encryption mode.
sl_wifi_channel_t	channel Channel.
sl_wifi_rate_protocol_t	rate_protocol Rate protocol.
sl_wifi_ap_flag_t	options Optional flags for AP configuration.
sl_wifi_credential_id_t	credential_id ID of secure credentials.
uint8_t	keepalive_type Keep alive type of the access point. One of the values from sl_si91x_ap_keepalive_type_t
uint16_t	beacon_interval Beacon interval of the access point in milliseconds.
uint32_t	client_idle_timeout Duration in milliseconds to kick idle client.
uint16_t	dtim_beacon_count How many beacons per DTIM.
uint8_t	maximum_clients Maximum number of associated clients.

Public Attribute Documentation

ssid

```
sl_wifi_ssid_t sl_wifi_ap_configuration_t::ssid
```

SSID.

Definition at line 114 of file components/protocol/wifi/inc/sl_wifi_types.h

security

sl_wifi_security_t sl_wifi_ap_configuration_t::security

Security mode.

Definition at line 115 of file components/protocol/wifi/inc/sl_wifi_types.h

encryption

sl_wifi_encryption_t sl_wifi_ap_configuration_t::encryption

Encryption mode.

Definition at line 116 of file components/protocol/wifi/inc/sl_wifi_types.h

channel

sl_wifi_channel_t sl_wifi_ap_configuration_t::channel

Channel.

Definition at line 117 of file components/protocol/wifi/inc/sl_wifi_types.h

rate_protocol

sl_wifi_rate_protocol_t sl_wifi_ap_configuration_t::rate_protocol

Rate protocol.

Definition at line 118 of file components/protocol/wifi/inc/sl_wifi_types.h

options

sl_wifi_ap_flag_t sl_wifi_ap_configuration_t::options

Optional flags for AP configuration.

Definition at line 119 of file components/protocol/wifi/inc/sl_wifi_types.h

credential_id

sl_wifi_credential_id_t sl_wifi_ap_configuration_t::credential_id

ID of secure credentials.

Definition at line 120 of file components/protocol/wifi/inc/sl_wifi_types.h

keepalive_type

> uint8_t sl_wifi_ap_configuration_t::keepalive_type

Keep alive type of the access point. One of the values from sl_si91x_ap_keepalive_type_t

Definition at line 122 of file components/protocol/wifi/inc/sl_wifi_types.h

beacon_interval

> uint16_t sl_wifi_ap_configuration_t::beacon_interval

Beacon interval of the access point in milliseconds.

Definition at line 123 of file components/protocol/wifi/inc/sl_wifi_types.h

client_idle_timeout

> uint32_t sl_wifi_ap_configuration_t::client_idle_timeout

Duration in milliseconds to kick idle client.

Definition at line 124 of file components/protocol/wifi/inc/sl_wifi_types.h

dtim_beacon_count

> uint16_t sl_wifi_ap_configuration_t::dtim_beacon_count

How many beacons per DTIM.

Definition at line 125 of file components/protocol/wifi/inc/sl_wifi_types.h

maximum_clients

> uint8_t sl_wifi_ap_configuration_t::maximum_clients

Maximum number of associated clients.

Definition at line 126 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_advanced_ap_configuration_t

Wi-Fi Access Point advanced configuration.

Public Attributes

uint8_t	csa_announcement_delay In beacon periods.
uint32_t	tbd Advanced configuration option to be added.

Public Attribute Documentation

csa_announcement_delay

```
uint8_t sl_wifi_advanced_ap_configuration_t::csa_announcement_delay
```

In beacon periods.

Definition at line 131 of file components/protocol/wifi/inc/sl_wifi_types.h

tbd

```
uint32_t sl_wifi_advanced_ap_configuration_t::tbd
```

Advanced configuration option to be added.

Definition at line 132 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_client_configuration_t

Wi-Fi Client interface configuration.

Public Attributes

sl_wifi_ssid_t	ssid SSID.
sl_wifi_channel_t	channel Channel.
sl_mac_address_t	bssid BSSID.
sl_wifi_bss_type_t	bss_type BSS type.
sl_wifi_security_t	security Security mode.
sl_wifi_encryption_t	encryption Encryption mode.
sl_wifi_client_flag_t	client_options Optional flags for client configuration.
sl_wifi_credential_id_t	credential_id ID of secure credentials.

Public Attribute Documentation

ssid

```
sl_wifi_ssid_t sl_wifi_client_configuration_t::ssid
```

SSID.

Definition at line 137 of file components/protocol/wifi/inc/sl_wifi_types.h

channel

```
sl_wifi_channel_t sl_wifi_client_configuration_t::channel
```

Channel.

Definition at line 138 of file components/protocol/wifi/inc/sl_wifi_types.h

bssid

sl_mac_address_t sl_wifi_client_configuration_t::bssid

BSSID.

Definition at line 139 of file components/protocol/wifi/inc/sl_wifi_types.h

bss_type

sl_wifi_bss_type_t sl_wifi_client_configuration_t::bss_type

BSS type.

Definition at line 140 of file components/protocol/wifi/inc/sl_wifi_types.h

security

sl_wifi_security_t sl_wifi_client_configuration_t::security

Security mode.

Definition at line 141 of file components/protocol/wifi/inc/sl_wifi_types.h

encryption

sl_wifi_encryption_t sl_wifi_client_configuration_t::encryption

Encryption mode.

Definition at line 142 of file components/protocol/wifi/inc/sl_wifi_types.h

client_options

sl_wifi_client_flag_t sl_wifi_client_configuration_t::client_options

Optional flags for client configuration.

Definition at line 143 of file components/protocol/wifi/inc/sl_wifi_types.h

credential_id

sl_wifi_credential_id_t sl_wifi_client_configuration_t::credential_id

ID of secure credentials.

Definition at line 144 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_advanced_client_configuration_t

Wi-Fi Client interface advance configuration.

Public Attributes

uint32_t **eap_flags**
EAP Flags.

uint32_t **max_retry_attempts**
Maximum number of retries before indicating join failure.

uint32_t **scan_interval**
Scan interval between each retry.

uint32_t **beacon_missed_count**
Number of missed beacons that will trigger rejoin.

uint32_t **first_time_retry_enable**
Retry enable or disable for first time joining.

Public Attribute Documentation

eap_flags

```
uint32_t sl_wifi_advanced_client_configuration_t::eap_flags
```

EAP Flags.

Note

- BIT[0] of OKC is used to enable or disable opportunistic key caching (OKC), –0 – disable –1 – enable – When this is enabled, module will use cached PMKID to get MSK(Master Session Key) which is need for generating PMK which is needed for 4-way handshake.
- BIT[1] of OKC is used to enable or disable CA certification for PEAP connection. –0 – CA certificate is not required. –1 – CA certificate is required.
- BIT[2-12] of OKC argument are used for Cipher list selection for EAP connection. All possible ciphers are listed below.

BIT position	Cipher selected
2	DHE-RSA-AES256-SHA256
3	DHE-RSA-AES128-SHA256
4	DHE-RSA-AES256-SHA
5	DHE-RSA-AES128-SHA
6	AES256-SHA256
7	AES128-SHA256
8	AES256-SHA
9	AES128-SHA
10	RC4-SHA
11	DES-CBC3-SHA
12	RC4-MD5

BIT[13-31] of OKC argument is reserved.

- When user sets BIT[1] and does not provide the CA certificate for PEAP connection then error is thrown. If user provides invalid CA certificate then also error is thrown. User can set either one or multiple bits from BIT[2-12] to provide the cipher's list. When user does not provide any value in OKC's BIT[2-12] then by default all the ciphers are selected.

Definition at line 170 of file components/protocol/wifi/inc/sl_wifi_types.h

max_retry_attempts

> uint32_t sl_wifi_advanced_client_configuration_t::max_retry_attempts

Maximum number of retries before indicating join failure.

Definition at line 171 of file components/protocol/wifi/inc/sl_wifi_types.h

scan_interval

> uint32_t sl_wifi_advanced_client_configuration_t::scan_interval

Scan interval between each retry.

Definition at line 172 of file components/protocol/wifi/inc/sl_wifi_types.h

beacon_missed_count

> uint32_t sl_wifi_advanced_client_configuration_t::beacon_missed_count

Number of missed beacons that will trigger rejoin.

Definition at line 173 of file components/protocol/wifi/inc/sl_wifi_types.h

first_time_retry_enable

> uint32_t sl_wifi_advanced_client_configuration_t::first_time_retry_enable

Retry enable or disable for first time joining.

Definition at line 174 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_psk_credential_t

Wi-Fi PSK security credentials.

Public Attributes

uint8_t	value PSK buffer.

Public Attribute Documentation

value

```
uint8_t sl_wifi_psk_credential_t::value[SL_WIFI_MAX_PSK_LENGTH]
```

PSK buffer.

Definition at line 179 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_pmk_credential_t

Wi-Fi PMK security credentials.

Public Attributes

uint8_t	value PMK buffer.

Public Attribute Documentation

value

```
uint8_t sl_wifi_pmk_credential_t::value[SL_WIFI_MAX_PMK_LENGTH]
```

PMK buffer.

Definition at line 184 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_wep_credential_t

Wi-Fi WEP security credentials.

Public Attributes

uint8_t	index Index.
uint8_t	key WEP Keys.

Public Attribute Documentation

index

```
uint8_t sl_wifi_wep_credential_t::index[2]
```

Index.

Definition at line 189 of file components/protocol/wifi/inc/sl_wifi_types.h

key

```
uint8_t sl_wifi_wep_credential_t::key[SL_WIFI_WEP_KEY_COUNT][SL_WIFI_WEP_KEY_LENGTH]
```

WEP Keys.

Definition at line 190 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_eap_credential_t

Wi-Fi Enterprise security credentials.

Public Attributes

uint8_t	username Enterprise User Name.
uint8_t	password Enterprise password.
uint8_t	certificate_key Certificate password.
uint32_t	certificate_id Certificate Id for Enterprise authentication.

Public Attribute Documentation

username

```
uint8_t sl_wifi_eap_credential_t::username[SL_WIFI_EAP_USER_NAME_LENGTH]
```

Enterprise User Name.

Definition at line 195 of file components/protocol/wifi/inc/sl_wifi_types.h

password

```
uint8_t sl_wifi_eap_credential_t::password[SL_WIFI_EAP_PASSWORD_LENGTH]
```

Enterprise password.

Definition at line 196 of file components/protocol/wifi/inc/sl_wifi_types.h

certificate_key

```
uint8_t sl_wifi_eap_credential_t::certificate_key[SL_WIFI_EAP_CERTIFICATE_KEY_LENGTH]
```

Certificate password.

Definition at line 197 of file components/protocol/wifi/inc/sl_wifi_types.h

certificate_id

> uint32_t sl_wifi_eap_credential_t::certificate_id

Certificate Id for Enterprise authentication.

Definition at line 198 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_credential_t

Wi-Fi security credentials.

Public Attributes

sl_wifi_credential_type_t	type Credential type.
sl_wifi_psk_credential_t	psk WiFi Personal credentials.
sl_wifi_pmk_credential_t	pmk WiFi PMK credentials.
sl_wifi_wep_credential_t	wep WEP keys.
sl_wifi_eap_credential_t	eap Enterprise client credentials.
union sl_wifi_credential_t::@1	@2 WiFi Credential structure.

Public Attribute Documentation

type

```
sl_wifi_credential_type_t sl_wifi_credential_t::type
```

Credential type.

Definition at line 207 of file components/protocol/wifi/inc/sl_wifi_types.h

psk

```
sl_wifi_psk_credential_t sl_wifi_credential_t::psk
```

WiFi Personal credentials.

Definition at line 209 of file components/protocol/wifi/inc/sl_wifi_types.h

pmk

```
sl_wifi_pmk_credential_t sl_wifi_credential_t::pmk
```

WiFi PMK credentials.

Definition at line 210 of file components/protocol/wifi/inc/sl_wifi_types.h

wep

> sl_wifi_wep_credential_t sl_wifi_credential_t::wep

WEP keys.

Definition at line 211 of file components/protocol/wifi/inc/sl_wifi_types.h

eap

> sl_wifi_eap_credential_t sl_wifi_credential_t::eap

Enterprise client credentials.

Definition at line 212 of file components/protocol/wifi/inc/sl_wifi_types.h

@2

> union sl_wifi_credential_t::@1 sl_wifi_credential_t::@2

WiFi Credential structure.

Definition at line 213 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_twt_request_t

TWT request structure to configure a session.

Public Attributes

uint8_t **wake_duration**
Nominal minimum wake duration. Range : 0 - 255.

uint8_t **wake_duration_tol**
Tolerance allowed for wake duration in case of suggest TWT. Received TWT wake duration from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Range : 0 - 255.

uint8_t **wake_int_exp**
Wake interval exponent to the base 2. Range : 0 - 31.

uint8_t **wake_int_exp_tol**
Tolerance allowed for wake_int_exp in case of suggest TWT request. Received TWT wake interval exponent from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Range : 0 - 31.

uint16_t **wake_int_mantissa**
Wake interval mantissa. Range : 0 - 65535.

uint16_t **wake_int_mantissa_tol**
Tolerance allowed for wake_int_mantissa in case of suggest TWT. Received TWT wake interval mantissa from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Range : 0 - 65535.

uint8_t **implicit_twt**
If enabled (1), the TWT requesting STA calculates the next TWT by adding a fixed value to the current TWT value. Explicit TWT is currently not allowed.

uint8_t **un_announced_twt**
If enabled (1), the TWT requesting STA does not announce its wake up to AP through PS-POLLs or UAPSD Trigger frames. Values : 0 or 1.

uint8_t **triggered_twt**
If enabled(1), at least one trigger frame is included in the TWT Service Period(TSP). Values : 0 or 1.

uint8_t **negotiation_type**
Negotiation type : 0 - Individual TWT; 1 - Broadcast TWT.

uint8_t **twt_channel**
Currently this configuration is not supported. Range : 0 - 7.

uint8_t **twt_protection**
If enabled (1), TSP is protected. This is negotiable with AP. Currently this is not supported. Values : 0 or 1.

uint8_t **twt_flow_id**
TWT session flow id. 0 - 7 valid. 0xFF to disable all active TWT sessions.

uint8_t **restrict_tx_outside_tsp**
1 - Any Tx outside the TSP is restricted. 0 - TX can happen outside the TSP also.

uint8_t **twt_retry_limit**
TWT retry limit. Range : 0 - 15.

uint8_t twt_retry_interval
TWT retry interval in seconds between two twt requests. Range : 5 - 255.

uint8_t req_type
TWT request type. 0 - Request TWT; 1 - Suggest TWT; 2 - Demand TWT.

uint8_t twt_enable
TWT enable. 0 - TWT session teardown; 1 - TWT session setup.

uint8_t wake_duration_unit
Wake duration unit. 0 - 256 microseconds ; 1 - 1024 microseconds.

Public Attribute Documentation

wake_duration

```
uint8_t sl_wifi_twt_request_t::wake_duration
```

Nominal minimum wake duration. Range : 0 - 255.

Definition at line 218 of file components/protocol/wifi/inc/sl_wifi_types.h

wake_duration_tol

```
uint8_t sl_wifi_twt_request_t::wake_duration_tol
```

Tolerance allowed for wake duration in case of suggest TWT. Received TWT wake duration from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Range : 0 - 255.

Definition at line 220 of file components/protocol/wifi/inc/sl_wifi_types.h

wake_int_exp

```
uint8_t sl_wifi_twt_request_t::wake_int_exp
```

Wake interval exponent to the base 2. Range : 0 - 31.

Definition at line 221 of file components/protocol/wifi/inc/sl_wifi_types.h

wake_int_exp_tol

```
uint8_t sl_wifi_twt_request_t::wake_int_exp_tol
```

Tolerance allowed for wake_int_exp in case of suggest TWT request. Received TWT wake interval exponent from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Range : 0 - 31.

Definition at line 223 of file components/protocol/wifi/inc/sl_wifi_types.h

wake_int_mantissa

uint16_t sl_wifi_twt_request_t::wake_int_mantissa

Wake interval mantissa. Range : 0 - 65535.

Definition at line 224 of file components/protocol/wifi/inc/sl_wifi_types.h

wake_int_mantissa_tol

uint16_t sl_wifi_twt_request_t::wake_int_mantissa_tol

Tolerance allowed for wake_int_mantissa in case of suggest TWT. Received TWT wake interval mantissa from AP will be validated against tolerance limits and decided if TWT config received is in acceptable range. Range : 0 - 65535.

Definition at line 226 of file components/protocol/wifi/inc/sl_wifi_types.h

implicit_twt

uint8_t sl_wifi_twt_request_t::implicit_twt

If enabled (1), the TWT requesting STA calculates the next TWT by adding a fixed value to the current TWT value. Explicit TWT is currently not allowed.

Definition at line 228 of file components/protocol/wifi/inc/sl_wifi_types.h

un_announced_twt

uint8_t sl_wifi_twt_request_t::un_announced_twt

If enabled (1), the TWT requesting STA does not announce its wake up to AP through PS-POLLs or UAPSD Trigger frames. Values : 0 or 1.

Definition at line 230 of file components/protocol/wifi/inc/sl_wifi_types.h

triggered_twt

uint8_t sl_wifi_twt_request_t::triggered_twt

If enabled(1), at least one trigger frame is included in the TWT Service Period(TSP). Values : 0 or 1.

Definition at line 232 of file components/protocol/wifi/inc/sl_wifi_types.h

negotiation_type

uint8_t sl_wifi_twt_request_t::negotiation_type

Negotiation type : 0 - Individual TWT; 1 - Broadcast TWT.

Definition at line 233 of file components/protocol/wifi/inc/sl_wifi_types.h

twt_channel

```
uint8_t sl_wifi_twt_request_t::twt_channel
```

Currently this configuration is not supported. Range : 0 - 7.

Definition at line 234 of file components/protocol/wifi/inc/sl_wifi_types.h

twt_protection

```
uint8_t sl_wifi_twt_request_t::twt_protection
```

If enabled (1), TSP is protected. This is negotiable with AP. Currently this is not supported. Values : 0 or 1.

Definition at line 236 of file components/protocol/wifi/inc/sl_wifi_types.h

twt_flow_id

```
uint8_t sl_wifi_twt_request_t::twt_flow_id
```

TWT session flow id. 0 - 7 valid. 0xFF to disable all active TWT sessions.

Definition at line 237 of file components/protocol/wifi/inc/sl_wifi_types.h

restrict_tx_outside_tsp

```
uint8_t sl_wifi_twt_request_t::restrict_tx_outside_tsp
```

1 - Any Tx outside the TSP is restricted. 0 - TX can happen outside the TSP also.

Definition at line 239 of file components/protocol/wifi/inc/sl_wifi_types.h

twt_retry_limit

```
uint8_t sl_wifi_twt_request_t::twt_retry_limit
```

TWT retry limit. Range : 0 - 15.

Definition at line 240 of file components/protocol/wifi/inc/sl_wifi_types.h

twt_retry_interval

```
uint8_t sl_wifi_twt_request_t::twt_retry_interval
```

TWT retry interval in seconds between two twt requests. Range : 5 - 255.

Definition at line 241 of file components/protocol/wifi/inc/sl_wifi_types.h

req_type

```
uint8_t sl_wifi_twt_request_t::req_type
```

TWT request type. 0 - Request TWT; 1 - Suggest TWT; 2 - Demand TWT.

Definition at line 242 of file components/protocol/wifi/inc/sl_wifi_types.h

twt_enable

```
uint8_t sl_wifi_twt_request_t::twt_enable
```

TWT enable. 0 - TWT session teardown; 1 - TWT session setup.

Definition at line 243 of file components/protocol/wifi/inc/sl_wifi_types.h

wake_duration_unit

```
uint8_t sl_wifi_twt_request_t::wake_duration_unit
```

Wake duration unit. 0 - 256 microseconds ; 1 - 1024 microseconds.

Definition at line 244 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_twt_selection_t

TWT request structure to auto select a session.

Public Attributes

uint8_t	twt_enable TWT enable. 0 - TWT session teardown; 1 - TWT session setup.
uint16_t	average_tx_throughput The expected average Tx throughput in Kbps. The value configured should be between 0 and half of device average throughput.
uint32_t	tx_latency The allowed latency, in milliseconds, within which the given Tx operation is expected to be completed. If 0 is configured, maximum allowed Tx latency is same as rx_latency. Otherwise, valid values are in the range of [200ms - 6hrs].
uint32_t	rx_latency The maximum latency for receiving buffered packets from the AP. The device wakes up at least once for a TWT service period within the configured rx_latency if there are any pending packets from the AP. If set to 0, the default latency of 2 seconds is used. Valid range is between 2 seconds to 6 hours.
uint16_t	device_average_throughput Refers to the average Tx throughput that the device is capable of achieving.
uint8_t	estimated_extra_wake_duration_percent The percentage by which wake duration is supposed to be overestimated to compensate for bss congestion. Valid input range is 0 - 50%.
uint8_t	twt_tolerable_deviation The allowed deviation percentage of wake duration TWT response. Valid input range is 0 - 50%.
uint32_t	default_wake_interval_ms Default minimum wake interval. Recommended Range: 512 to 1024ms.
uint32_t	default_minimum_wake_duration_ms Default minimum wake interval. Recommended Range: 8 - 16ms.
uint8_t	beacon_wake_up_count_after_sp The number of beacons after the service period completion for which the module wakes up and listens for any pending RX.

Public Attribute Documentation

twt_enable

```
uint8_t sl_wifi_twt_selection_t::twt_enable
```

TWT enable. 0 - TWT session teardown; 1 - TWT session setup.

Definition at line `249` of file `components/protocol/wifi/inc/sl_wifi_types.h`

average_tx_throughput

```
uint16_t sl_wifi_twt_selection_t::average_tx_throughput
```

The expected average Tx throughput in Kbps. The value configured should be between 0 and half of device average throughput.

Definition at line 251 of file components/protocol/wifi/inc/sl_wifi_types.h

tx_latency

```
uint32_t sl_wifi_twt_selection_t::tx_latency
```

The allowed latency, in milliseconds, within which the given Tx operation is expected to be completed. If 0 is configured, maximum allowed Tx latency is same as rx_latency. Otherwise, valid values are in the range of [200ms - 6hrs].

Definition at line 253 of file components/protocol/wifi/inc/sl_wifi_types.h

rx_latency

```
uint32_t sl_wifi_twt_selection_t::rx_latency
```

The maximum latency for receiving buffered packets from the AP. The device wakes up at least once for a TWT service period within the configured rx_latency if there are any pending packets from the AP. If set to 0, the default latency of 2 seconds is used. Valid range is between 2 seconds to 6 hours.

Definition at line 255 of file components/protocol/wifi/inc/sl_wifi_types.h

device_average_throughput

```
uint16_t sl_wifi_twt_selection_t::device_average_throughput
```

Refers to the average Tx throughput that the device is capable of achieving.

Definition at line 256 of file components/protocol/wifi/inc/sl_wifi_types.h

estimated_extra_wake_duration_percent

```
uint8_t sl_wifi_twt_selection_t::estimated_extra_wake_duration_percent
```

The percentage by which wake duration is supposed to be overestimated to compensate for bss congestion. Valid input range is 0 - 50%.

Definition at line 258 of file components/protocol/wifi/inc/sl_wifi_types.h

twt_tolerable_deviation

```
uint8_t sl_wifi_twt_selection_t::twt_tolerable_deviation
```

The allowed deviation percentage of wake duration TWT response. Valid input range is 0 - 50%.

Definition at line 260 of file components/protocol/wifi/inc/sl_wifi_types.h

default_wake_interval_ms

```
uint32_t sl_wifi_twt_selection_t::default_wake_interval_ms
```

Default minimum wake interval. Recommended Range: 512 to 1024ms.

Definition at line 261 of file components/protocol/wifi/inc/sl_wifi_types.h

default_minimum_wake_duration_ms

```
uint32_t sl_wifi_twt_selection_t::default_minimum_wake_duration_ms
```

Default minimum wake interval. Recommended Range: 8 - 16ms.

Definition at line 262 of file components/protocol/wifi/inc/sl_wifi_types.h

beacon_wake_up_count_after_sp

```
uint8_t sl_wifi_twt_selection_t::beacon_wake_up_count_after_sp
```

The number of beacons after the service period completion for which the module wakes up and listens for any pending RX.

Definition at line 264 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_reschedule_twt_config_t

TWT reschedule structure.

Public Attributes

uint8_t	flow_id TWT session flow id.
sl_wifi_reschedule_twt_action_t	twt_action Suspension time for TWT.
uint16_t	reserved1 Reserved.
uint8_t	reserved2 Reserved.
uint64_t	suspend_duration Duration to suspend respective TWT session.

Public Attribute Documentation

flow_id

```
uint8_t sl_wifi_reschedule_twt_config_t::flow_id
```

TWT session flow id.

Definition at line `269` of file `components/protocol/wifi/inc/sl_wifi_types.h`

twt_action

```
sl_wifi_reschedule_twt_action_t sl_wifi_reschedule_twt_config_t::twt_action
```

Suspension time for TWT.

Definition at line `270` of file `components/protocol/wifi/inc/sl_wifi_types.h`

reserved1

```
uint16_t sl_wifi_reschedule_twt_config_t::reserved1
```

Reserved.

Definition at line `271` of file `components/protocol/wifi/inc/sl_wifi_types.h`

reserved2

> uint8_t sl_wifi_reschedule_twt_config_t::reserved2

Reserved.

Definition at line 272 of file components/protocol/wifi/inc/sl_wifi_types.h

suspend_duration

> uint64_t sl_wifi_reschedule_twt_config_t::suspend_duration

Duration to suspend respective TWT session.

Definition at line 273 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_status_t

Wi-Fi device status.

Public Attributes

uint8_t	client_active WiFi Client active.
uint8_t	ap_active WiFi Access point active.
uint8_t	monitor_mode_active WiFi promiscuous mode active.
uint8_t	wfd_go_active Reserved Status bit.
uint8_t	wfd_client_active Reserved Status bit.
uint8_t	scan_active Scan in Progress.
uint8_t	_reserved Reserved Status bit.
uint8_t	_reserved2 Reserved Status bit.

Public Attribute Documentation

client_active

```
uint8_t sl_wifi_status_t::client_active
```

WiFi Client active.

Definition at line 278 of file components/protocol/wifi/inc/sl_wifi_types.h

ap_active

```
uint8_t sl_wifi_status_t::ap_active
```

WiFi Access point active.

Definition at line 279 of file components/protocol/wifi/inc/sl_wifi_types.h

monitor_mode_active

uint8_t sl_wifi_status_t::monitor_mode_active

WiFi promiscuous mode active.

Definition at line 280 of file components/protocol/wifi/inc/sl_wifi_types.h

wfd_go_active

uint8_t sl_wifi_status_t::wfd_go_active

Reserved Status bit.

Definition at line 281 of file components/protocol/wifi/inc/sl_wifi_types.h

wfd_client_active

uint8_t sl_wifi_status_t::wfd_client_active

Reserved Status bit.

Definition at line 282 of file components/protocol/wifi/inc/sl_wifi_types.h

scan_active

uint8_t sl_wifi_status_t::scan_active

Scan in Progress.

Definition at line 283 of file components/protocol/wifi/inc/sl_wifi_types.h

_reserved

uint8_t sl_wifi_status_t::_reserved

Reserved Status bit.

Definition at line 284 of file components/protocol/wifi/inc/sl_wifi_types.h

_reserved2

uint8_t sl_wifi_status_t::_reserved2

Reserved Status bit.

Definition at line 285 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_statistics_t

Wi-Fi interface statistics.

Public Attributes

uint32_t	beacon_lost_count Number of missed beacons.
uint32_t	beacon_rx_count Number of received beacons.
uint32_t	mcast_rx_count Multicast packets received.
uint32_t	mcast_tx_count Multicast packets transmitted.
uint32_t	ucast_rx_count Unicast packets received.
uint32_t	ucast_tx_count Unicast packets transmitted.
uint32_t	overrun_count Number of packets dropped either at ingress or egress, due to lack of buffer memory to retain all packets.

Public Attribute Documentation

beacon_lost_count

```
uint32_t sl_wifi_statistics_t::beacon_lost_count
```

Number of missed beacons.

Definition at line `290` of file `components/protocol/wifi/inc/sl_wifi_types.h`

beacon_rx_count

```
uint32_t sl_wifi_statistics_t::beacon_rx_count
```

Number of received beacons.

Definition at line `291` of file `components/protocol/wifi/inc/sl_wifi_types.h`

mcast_rx_count

```
uint32_t sl_wifi_statistics_t::mcast_rx_count
```

Multicast packets received.

Definition at line 292 of file components/protocol/wifi/inc/sl_wifi_types.h

mcast_tx_count

```
uint32_t sl_wifi_statistics_t::mcast_tx_count
```

Multicast packets transmitted.

Definition at line 293 of file components/protocol/wifi/inc/sl_wifi_types.h

ucast_rx_count

```
uint32_t sl_wifi_statistics_t::ucast_rx_count
```

Unicast packets received.

Definition at line 294 of file components/protocol/wifi/inc/sl_wifi_types.h

ucast_tx_count

```
uint32_t sl_wifi_statistics_t::ucast_tx_count
```

Unicast packets transmitted.

Definition at line 295 of file components/protocol/wifi/inc/sl_wifi_types.h

overrun_count

```
uint32_t sl_wifi_statistics_t::overrun_count
```

Number of packets dropped either at ingress or egress, due to lack of buffer memory to retain all packets.

Definition at line 297 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_p2p_configuration_t

Wi-Fi Direct (P2P) configuration.

Public Attributes

uint16_t	group_owner_intent Group owner intent.
const char *	device_name Device name.
sl_wifi_channel_t	channel Wi-Fi channel.
char	ssid_suffix SSID suffix.

Public Attribute Documentation

group_owner_intent

```
uint16_t sl_wifi_p2p_configuration_t::group_owner_intent
```

Group owner intent.

Definition at line `302` of file `components/protocol/wifi/inc/sl_wifi_types.h`

device_name

```
const char* sl_wifi_p2p_configuration_t::device_name
```

Device name.

Definition at line `303` of file `components/protocol/wifi/inc/sl_wifi_types.h`

channel

```
sl_wifi_channel_t sl_wifi_p2p_configuration_t::channel
```

Wi-Fi channel.

Definition at line `304` of file `components/protocol/wifi/inc/sl_wifi_types.h`

ssid_suffix

> char sl_wifi_p2p_configuration_t::ssid_suffix[6]

SSID suffix.

Definition at line 305 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_event_data_t

Wi-Fi event data.

Public Attributes

sl_wifi_scan_result_t	scan_results Scan Result structure.
uint32_t	join_status Join status.

Public Attribute Documentation

scan_results

```
sl_wifi_scan_result_t sl_wifi_event_data_t::scan_results
```

Scan Result structure.

Definition at line 310 of file components/protocol/wifi/inc/sl_wifi_types.h

join_status

```
uint32_t sl_wifi_event_data_t::join_status
```

Join status.

Definition at line 311 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_wps_pin_t

Wi-Fi WPS PIN object that is an 8 digit number.

Public Attributes

char	digits Array to store digits of WPS Pin.

Public Attribute Documentation

digits

```
char sl_wifi_wps_pin_t::digits[8]
```

Array to store digits of WPS Pin.

Definition at line 316 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_listen_interval_t

Wi-Fi listen interval.

Public Attributes

uint32_t	listen_interval Wi-Fi Listen interval in millisecs.

Public Attribute Documentation

listen_interval

> uint32_t sl_wifi_listen_interval_t::listen_interval

Wi-Fi Listen interval in millisecs.

Definition at line 321 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_client_info_t

Wi-Fi client info.

Public Attributes

sl_mac_address_t	mac_adddress MAC Address of the client.
sl_ip_address_t	ip_address IP address of client.

Public Attribute Documentation

mac_adddress

```
sl_mac_address_t sl_wifi_client_info_t::mac_adddress
```

MAC Address of the client.

Definition at line 326 of file components/protocol/wifi/inc/sl_wifi_types.h

ip_address

```
sl_ip_address_t sl_wifi_client_info_t::ip_address
```

IP address of client.

Definition at line 327 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_client_info_response_t

Wi-Fi client info response.

Public Attributes

uint8_t	client_count Client count.
sl_wifi_client_info_t	client_info Array of client info.

Public Attribute Documentation

client_count

```
uint8_t sl_wifi_client_info_response_t::client_count
```

Client count.

Definition at line 332 of file components/protocol/wifi/inc/sl_wifi_types.h

client_info

```
sl_wifi_client_info_t sl_wifi_client_info_response_t::client_info[SL_WIFI_MAX_CLIENT_COUNT]
```

Array of client info.

Definition at line 333 of file components/protocol/wifi/inc/sl_wifi_types.h

sl_wifi_max_tx_power_t

Wi-Fi max transmit power.

Public Attributes

uint8_t join_tx_power

Public Attribute Documentation

join_tx_power

```
uint8_t sl_wifi_max_tx_power_t::join_tx_power
```

Definition at line 338 of file components/protocol/wifi/inc/sl_wifi_types.h

Constants

Constants

This section provides a reference to Wi-Fi API constants.

Enumerations

enum sl_wifi_security_t {

SL_WIFI_OPEN = 0
SL_WIFI_WPA = 1
SL_WIFI_WPA2 = 2
SL_WIFI_WEP = 3
SL_WIFI_WPA_ENTERPRISE = 4
SL_WIFI_WPA2_ENTERPRISE = 5
SL_WIFI_WPA_WPA2_MIXED = 6
SL_WIFI_WPA3 = 7
SL_WIFI_WPA3_TRANSITION = 8
SL_WIFI_SECURITY_UNKNOWN = 0xFFFF

}

Wi-Fi security.

enum sl_wifi_encryption_t {

SL_WIFI_NO_ENCRYPTION
SL_WIFI_WEP_ENCRYPTION
SL_WIFI_TKIP_ENCRYPTION
SL_WIFI_CCMP_ENCRYPTION
SL_WIFI_EAP_TLS_ENCRYPTION
SL_WIFI_EAP_TTLS_ENCRYPTION
SL_WIFI_EAP_FAST_ENCRYPTION
SL_WIFI_PEAP_MSCHAPV2_ENCRYPTION

}

Wi-Fi encryption method.

enum sl_wifi_credential_type_t {

SL_WIFI_PSK_CREDENTIAL = 0
SL_WIFI_PMK_CREDENTIAL
SL_WIFI_WEP_CREDENTIAL
SL_WIFI_EAP_CREDENTIAL

}

Wi-Fi Credential.

enum sl_wifi_antenna_t {

SL_WIFI_ANTENNA_1
SL_WIFI_ANTENNA_2
SL_WIFI_ANTENNA_AUTO
SL_WIFI_ANTENNA_EXTERNAL
SL_WIFI_ANTENNA_INTERNAL

}

Wi-Fi antenna selections.

enum sl_wifi_interface_index_t {

SL_WIFI_CLIENT_2_4GHZ_INTERFACE_INDEX = 0
SL_WIFI_AP_2_4GHZ_INTERFACE_INDEX
SL_WIFI_CLIENT_5GHZ_INTERFACE_INDEX
SL_WIFI_AP_5GHZ_INTERFACE_INDEX
SL_WIFI_MAX_INTERFACE_INDEX

}

Wi-Fi interface index enumeration.

enum sl_wifi_interface_t {

SL_WIFI_INVALID_INTERFACE = 0
SL_WIFI_CLIENT_INTERFACE = (1 << 0)
SL_WIFI_AP_INTERFACE = (1 << 1)
SL_WIFI_2_4GHZ_INTERFACE = (1 << 2)
SL_WIFI_5GHZ_INTERFACE = (1 << 3)
SL_WIFI_CLIENT_2_4GHZ_INTERFACE = SL_WIFI_CLIENT_INTERFACE | SL_WIFI_2_4GHZ_INTERFACE
SL_WIFI_AP_2_4GHZ_INTERFACE = SL_WIFI_AP_INTERFACE | SL_WIFI_2_4GHZ_INTERFACE
SL_WIFI_CLIENT_5GHZ_INTERFACE = SL_WIFI_CLIENT_INTERFACE | SL_WIFI_5GHZ_INTERFACE
SL_WIFI_AP_5GHZ_INTERFACE = SL_WIFI_AP_INTERFACE | SL_WIFI_5GHZ_INTERFACE
SL_WIFI_ALL_INTERFACES = SL_WIFI_CLIENT_INTERFACE | SL_WIFI_AP_INTERFACE |
SL_WIFI_2_4GHZ_INTERFACE | SL_WIFI_5GHZ_INTERFACE

}

Wi-Fi interface enumeration.

enum sl_wifi_deauth_reason_t {

SL_WIFI_DEAUTH
SL_WIFI_DEAUTH_UNSPECIFIED

}

Enumeration of de-authentication reasons from an access point.

enum sl_wifi_regulatory_region_t {

SL_WIFI_REGION_AUSTRALIA
SL_WIFI_REGION_FRANCE
SL_WIFI_REGION_EUROPEAN_UNION
SL_WIFI_REGION_JAPAN
SL_WIFI_REGION_UNITED_STATES

}

W-Fi regulatory region.

enum sl_wifi_rate_protocol_t {

SL_WIFI_RATE_PROTOCOL_B_ONLY
SL_WIFI_RATE_PROTOCOL_G_ONLY
SL_WIFI_RATE_PROTOCOL_N_ONLY
SL_WIFI_RATE_PROTOCOL_AC_ONLY
SL_WIFI_RATE_PROTOCOL_AX_ONLY
SL_WIFI_RATE_PROTOCOL_AUTO

}

Wi-Fi rate protocols.

enum sl_wifi_scan_type_t {

SL_WIFI_SCAN_TYPE_ACTIVE = 0x00
SL_WIFI_SCAN_TYPE_PASSIVE = 0x01
SL_WIFI_SCAN_TYPE_PROHIBITED_CHANNELS = 0x04
SL_WIFI_SCAN_TYPE_ADV_SCAN = 0X08

}

Wi-Fi scan types.

enum sl_wifi_rate_t {

SL_WIFI_AUTO_RATE = 0
SL_WIFI_RATE_11B_1
SL_WIFI_RATE_11B_MIN = SL_WIFI_RATE_11B_1
SL_WIFI_RATE_11B_2
SL_WIFI_RATE_11B_5_5
SL_WIFI_RATE_11B_11
SL_WIFI_RATE_11B_MAX = SL_WIFI_RATE_11B_11
SL_WIFI_RATE_11G_6
SL_WIFI_RATE_11G_MIN = SL_WIFI_RATE_11G_6
SL_WIFI_RATE_11G_9
SL_WIFI_RATE_11G_12
SL_WIFI_RATE_11G_18
SL_WIFI_RATE_11G_24
SL_WIFI_RATE_11G_36
SL_WIFI_RATE_11G_48
SL_WIFI_RATE_11G_54
SL_WIFI_RATE_11G_MAX = SL_WIFI_RATE_11G_54
SL_WIFI_RATE_11N_MCS0
SL_WIFI_RATE_11N_MIN = SL_WIFI_RATE_11N_MCS0
SL_WIFI_RATE_11N_MCS1
SL_WIFI_RATE_11N_MCS2
SL_WIFI_RATE_11N_MCS3
SL_WIFI_RATE_11N_MCS4
SL_WIFI_RATE_11N_MCS5
SL_WIFI_RATE_11N_MCS6
SL_WIFI_RATE_11N_MCS7
SL_WIFI_RATE_11N_MAX = SL_WIFI_RATE_11N_MCS7
SL_WIFI_RATE_11AX_MCS0
SL_WIFI_RATE_11AX_MIN = SL_WIFI_RATE_11AX_MCS0
SL_WIFI_RATE_11AX_MCS1
SL_WIFI_RATE_11AX_MCS2
SL_WIFI_RATE_11AX_MCS3
SL_WIFI_RATE_11AX_MCS4
SL_WIFI_RATE_11AX_MCS5
SL_WIFI_RATE_11AX_MCS6
SL_WIFI_RATE_11AX_MCS7
SL_WIFI_RATE_11AX_MAX = SL_WIFI_RATE_11AX_MCS7
SL_WIFI_RATE_INVALID = 0xFF

}

Wi-Fi transfer rates.

enum sl_wifi_bss_type_t {

SL_WIFI_BSS_TYPE_INFRASTRUCTURE = 0
SL_WIFI_BSS_TYPE_ADHOC = 1
SL_WIFI_BSS_TYPE_ANY = 2
SL_WIFI_BSS_TYPE_UNKNOWN = 0xFF

}

Wi-Fi BSS type.

enum sl_wifi_band_t {

SL_WIFI_AUTO_BAND = 0
SL_WIFI_BAND_900MHZ = 1
SL_WIFI_BAND_2_4GHZ = 2
SL_WIFI_BAND_5GHZ = 3
SL_WIFI_BAND_6GHZ = 4
SL_WIFI_BAND_60GHZ = 5

}

Wi-Fi radio band.

enum sl_wifi_bandwidth_t {

SL_WIFI_AUTO_BANDWIDTH = 0
SL_WIFI_BANDWIDTH_10MHz = 0
SL_WIFI_BANDWIDTH_20MHz = 1
SL_WIFI_BANDWIDTH_40MHz = 2
SL_WIFI_BANDWIDTH_80MHz = 3
SL_WIFI_BANDWIDTH_160MHz = 4

}

enum sl_wifi_client_flag_t {

SL_WIFI_NO_JOIN_OPTION = 0
SL_WIFI_JOIN_WITH_NO_CSA = (1 << 0)
SL_WIFI_JOIN_WITH_SCAN = (1 << 1)

}

Option flags for client interfaces.

enum sl_wifi_ap_flag_t {

SL_WIFI_HIDDEN_SSID = (1 << 0)

}

Option flags for AP interfaces.

enum sl_wifi_listen_interval_time_unit_t {

SL_WIFI_LISTEN_INTERVAL_TIME_UNIT_BEACON
SL_WIFI_LISTEN_INTERVAL_TIME_UNIT_DTIM

}

Listen interval time units.

enum sl_wifi_wps_mode_t {

SL_WIFI_WPS_PIN_MODE
SL_WIFI_WPS_PUSH_BUTTON_MODE

}

Wi-Fi WPS mode.

enum sl_wifi_event_group_t {

SL_WIFI_SCAN_RESULT_EVENTS
SL_WIFI_JOIN_EVENTS
SL_WIFI_RX_PACKET_EVENTS
SL_WIFI_COMMAND_RESPONSE_EVENTS
SL_WIFI_STATS_RESPONSE_EVENTS
SL_WIFI_HTTP_OTA_FW_UPDATE_EVENTS
SL_WIFI_NETWORK_DOWN_EVENTS
SL_WIFI_NETWORK_UP_EVENTS
SL_WIFI_CLIENT_CONNECTED_EVENTS
SL_WIFI_TWT_RESPONSE_EVENTS
SL_WIFI_CLIENT_DISCONNECTED_EVENTS
SL_WIFI_EVENT_GROUP_COUNT
SL_WIFI_EVENT_FAIL_INDICATION_EVENTS = (1 << 31)

}

Wi-Fi event group.

enum sl_wifi_event_t {

SL_WIFI_SCAN_RESULT_EVENT = SL_WIFI_SCAN_RESULT_EVENTS
SL_WIFI_JOIN_EVENT = SL_WIFI_JOIN_EVENTS
SL_WIFI_RX_PACKET_EVENT = SL_WIFI_RX_PACKET_EVENTS
SL_WIFI_COMMAND_RESPONSE_EVENT = SL_WIFI_COMMAND_RESPONSE_EVENTS
SL_WIFI_STATS_RESPONSE_EVENT = SL_WIFI_STATS_RESPONSE_EVENTS
SL_WIFI_HTTP_OTA_FW_UPDATE_EVENT = SL_WIFI_HTTP_OTA_FW_UPDATE_EVENTS
SL_WIFI_NETWORK_DOWN_EVENT = SL_WIFI_NETWORK_DOWN_EVENTS
SL_WIFI_NETWORK_UP_EVENT = SL_WIFI_NETWORK_UP_EVENTS
SL_WIFI_CLIENT_CONNECTED_EVENT = SL_WIFI_CLIENT_CONNECTED_EVENTS
SL_WIFI_TWT_RESPONSE_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS
SL_WIFI_CLIENT_DISCONNECTED_EVENT = SL_WIFI_CLIENT_DISCONNECTED_EVENTS
SL_WIFI_TWT_UNSOLICITED_SESSION_SUCCESS_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (1 << 16)
SL_WIFI_TWT_AP_REJECTED_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (4 << 16)
SL_WIFI_TWT_OUT_OF_TOLERANCE_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (5 << 16)
SL_WIFI_TWT_RESPONSE_NOT_MATCHED_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (6 << 16)
SL_WIFI_TWT_UNSUPPORTED_RESPONSE_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (10 << 16)
SL_WIFI_TWT_TEARDOWN_SUCCESS_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (11 << 16)
SL_WIFI_TWT_AP_TEARDOWN_SUCCESS_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (12 << 16)
SL_WIFI_TWT_FAIL_MAX_RETRIES_REACHED_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (15 << 16)
SL_WIFI_TWT_INACTIVE_DUE_TO_ROAMING_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (16 << 16)
SL_WIFI_TWT_INACTIVE_DUE_TO_DISCONNECT_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (17 << 16)
SL_WIFI_TWT_INACTIVE_NO_AP_SUPPORT_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (18 << 16)
SL_WIFI_RESCHEDULE_TWT_SUCCESS_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (19 << 16)
SL_WIFI_TWT_INFO_FRAME_EXCHANGE_FAILED_EVENT = SL_WIFI_TWT_RESPONSE_EVENTS | (20 << 16)
SL_WIFI_TWT_EVENTS_END = SL_WIFI_TWT_RESPONSE_EVENTS | (21 << 16)
SL_WIFI_STATS_EVENT = SL_WIFI_STATS_RESPONSE_EVENTS | (1 << 16)
SL_WIFI_STATS_AYSNC_EVENT = SL_WIFI_STATS_RESPONSE_EVENTS | (2 << 16)
SL_WIFI_STATS_ADVANCE_EVENT = SL_WIFI_STATS_RESPONSE_EVENTS | (3 << 16)
SL_WIFI_STATS_TEST_MODE_EVENT = SL_WIFI_STATS_RESPONSE_EVENTS | (4 << 16)
SL_WIFI_STATS_MODULE_STATE_EVENT = SL_WIFI_STATS_RESPONSE_EVENTS | (5 << 16)
SL_WIFI_EVENT_FAIL_INDICATION = (1 << 31)
SL_WIFI_INVALID_EVENT = 0xFFFFFFFF

}

Wi-Fi events.

enum sl_wifi_reschedule_twt_action_t {

SL_WIFI_SUSPEND_INDEFINITELY
SL_WIFI_SUSPEND_FOR_DURATION
SL_WIFI_RESUME_IMMEDIATELY

}

Enumeration defining actions related to Target Wake Time (TWT).

enum sl_wifi_data_rate_t {

SL_WIFI_DATA_RATE_1 = 0
SL_WIFI_DATA_RATE_2 = 2
SL_WIFI_DATA_RATE_5_5 = 4
SL_WIFI_DATA_RATE_11 = 6
SL_WIFI_DATA_RATE_6 = 139
SL_WIFI_DATA_RATE_9 = 143
SL_WIFI_DATA_RATE_12 = 138
SL_WIFI_DATA_RATE_18 = 142
SL_WIFI_DATA_RATE_24 = 137
SL_WIFI_DATA_RATE_36 = 141
SL_WIFI_DATA_RATE_48 = 136
SL_WIFI_DATA_RATE_54 = 140
SL_WIFI_DATA_RATE_MCS0 = 256
SL_WIFI_DATA_RATE_MCS1 = 257
SL_WIFI_DATA_RATE_MCS2 = 258
SL_WIFI_DATA_RATE_MCS3 = 259
SL_WIFI_DATA_RATE_MCS4 = 260
SL_WIFI_DATA_RATE_MCS5 = 261
SL_WIFI_DATA_RATE_MCS6 = 262
SL_WIFI_DATA_RATE_MCS7 = 263
SL_WIFI_DATA_RATE_MCS7_SG = 775

}

Macros

#define SL_WIFI_MAX_SCANNED_AP 11
Max number of Access points that can be scanned.

#define SL_WIFI_MAX_CLIENT_COUNT 16
Max number of stations when module is running in access point mode.

#define SL_WIFI_MAX_PSK_LENGTH 32
Max Length of Wi-Fi PSK credential.

#define SL_WIFI_MAX_PMK_LENGTH 64
Max Length of Wi-Fi PMK credential.

#define SL_WIFI_WEP_KEY_LENGTH 32
Max length of Key in WEP security.

#define SL_WIFI_WEP_KEY_COUNT 4
Max number of keys for WEP security.

#define SL_WIFI_EAP_USER_NAME_LENGTH 64
Max Length of User Name in enterprise security.

#define SL_WIFI_EAP_PASSWORD_LENGTH 128
Max Length of password in enterprise security.

#define SL_WIFI_EAP_CERTIFICATE_KEY_LENGTH 80
Max Length of certificate key in enterprise security.

#define SL_WIFI_SELECT_INTERNAL_ANTENNA 0
Select Internal Antenna for Wi-Fi.

#define SL_WIFI_SELECT_EXTERNAL_ANTENNA 1
Select External Antenna for Wi-Fi.

#define SL_WIFI_DEFAULT_INTERFACE sl_wifi_get_default_interface()
Default Wi-Fi interface macro.

#define SL_WIFI_NEVER_ROAM 0×7FFFFFFF
Max Wi-Fi roaming trigger interval.

#define SL_WIFI_AUTO_CHANNEL 0
Auto detect channel.

#define SL_WIFI_ARGS_CHECK_NULL_POINTER (ptr)
API input checks.

#define SL_WIFI_ARGS_CHECK_INVALID_INTERFACE (interface)
Interface input checks.

Enumeration Documentation

sl_wifi_security_t

> sl_wifi_security_t

Wi-Fi security.

Note

- WPA3 Transition mode not currently supported while running as an Access Point.

Enumerator	
SL_WIFI_OPEN	Wi-Fi Open security type.
SL_WIFI_WPA	Wi-Fi WPA security type.
SL_WIFI_WPA2	Wi-Fi WPA2 security type.
SL_WIFI_WEP	Wi-Fi WEP security type.
SL_WIFI_WPA_ENTERPRISE	Wi-Fi WPA Enterprise security type.
SL_WIFI_WPA2_ENTERPRISE	Wi-Fi WPA2 Enterprise security type.
SL_WIFI_WPA_WPA2_MIXED	Wi-Fi WPA/WPA2 Mixed security type.
SL_WIFI_WPA3	Wi-Fi WPA3 security type.
SL_WIFI_WPA3_TRANSITION	Wi-Fi WPA3 Transition security type (not currently supported in AP mode)
SL_WIFI_SECURITY_UNKNOWN	Wi-Fi Unknown Security type.

Definition at line 58 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_encryption_t

> sl_wifi_encryption_t

Wi-Fi encryption method.

Note

- Some encryption types not currently supported in station (STA) mode.

Enumerator	
SL_WIFI_NO_ENCRYPTION	Wi-Fi with No Encryption (not currently supported in STA mode)
SL_WIFI_WEP_ENCRYPTION	Wi-Fi with WEP Encryption (not currently supported in STA mode)
SL_WIFI_TKIP_ENCRYPTION	Wi-Fi with TKIP Encryption (not currently supported in STA mode)
SL_WIFI_CCMP_ENCRYPTION	Wi-Fi with CCMP Encryption (not currently supported in STA mode)

SL_WIFI_EAP_TLS_ENCRYPTION	Wi-Fi with Enterprise TLS Encryption.
SL_WIFI_EAP_TTLS_ENCRYPTION	Wi-Fi with Enterprise TTLS Encryption.
SL_WIFI_EAP_FAST_ENCRYPTION	Wi-Fi with Enterprise FAST Encryption.
SL_WIFI_PEAP_MSCHAPV2_ENCRYPTION	Wi-Fi with Enterprise PEAP Encryption.

Definition at line 74 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_credential_type_t

> sl_wifi_credential_type_t

Wi-Fi Credential.

Enumerator	
SL_WIFI_PSK_CREDENTIAL	Wi-Fi Personal Credential.
SL_WIFI_PMK_CREDENTIAL	Wi-Fi Pairwise master key.
SL_WIFI_WEP_CREDENTIAL	Wi-Fi WEP Credential.
SL_WIFI_EAP_CREDENTIAL	Wi-Fi Enterprise client Credential.

Definition at line 86 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_antenna_t

> sl_wifi_antenna_t

Wi-Fi antenna selections.

Note

- Only internal antenna currently supported.

Enumerator	
SL_WIFI_ANTENNA_1	Wi-Fi Radio Antenna 1 (not currently supported)
SL_WIFI_ANTENNA_2	Wi-Fi Radio Antenna 2 (not currently supported)
SL_WIFI_ANTENNA_AUTO	Wi-Fi Radio Antenna Auto Selection (not currently supported)
SL_WIFI_ANTENNA_EXTERNAL	Wi-Fi Radio External Antenna (not currently supported)
SL_WIFI_ANTENNA_INTERNAL	Wi-Fi Radio Internal Antenna.

Definition at line 95 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_interface_index_t

> sl_wifi_interface_index_t

Wi-Fi interface index enumeration.

Note

- 5 GHz interfaces not currently supported.

Enumerator	

SL_WIFI_CLIENT_2_4GHZ_INTERFACE_INDEX	Wi-Fi client on 2.4GHz interface.
SL_WIFI_AP_2_4GHZ_INTERFACE_INDEX	Wi-Fi access point on 2.4GHz interface.
SL_WIFI_CLIENT_5GHZ_INTERFACE_INDEX	Wi-Fi client on 5GHz interface (not currently supported)
SL_WIFI_AP_5GHZ_INTERFACE_INDEX	Wi-Fi access point on 5GHz interface (not currently supported)
SL_WIFI_MAX_INTERFACE_INDEX	Used for internally by SDK.

Definition at line 104 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_interface_t

> sl_wifi_interface_t

Wi-Fi interface enumeration.

	Enumerator
SL_WIFI_INVALID_INTERFACE	Invalid interface.
SL_WIFI_CLIENT_INTERFACE	Wi-Fi client interface.
SL_WIFI_AP_INTERFACE	Wi-Fi access point interface.
SL_WIFI_2_4GHZ_INTERFACE	2.4GHz radio interface
SL_WIFI_5GHZ_INTERFACE	5GHz radio interface
SL_WIFI_CLIENT_2_4GHZ_INTERFACE	Wi-Fi client interface on 2.4GHz radio.
SL_WIFI_AP_2_4GHZ_INTERFACE	Wi-Fi access point interface on 2.4GHz radio.
SL_WIFI_CLIENT_5GHZ_INTERFACE	Wi-Fi client interface on 5GHz radio.
SL_WIFI_AP_5GHZ_INTERFACE	Wi-Fi access point interface on 5GHz radio.
SL_WIFI_ALL_INTERFACES	All available Wi-Fi interfaces.

Definition at line 113 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_deauth_reason_t

> sl_wifi_deauth_reason_t

Enumeration of de-authentication reasons from an access point.

	Enumerator
SL_WIFI_DEAUTH	De-Authentication from radius server.
SL_WIFI_DEAUTH_UNSPECIFIED	Unspecified de-authentication reason.

Definition at line 136 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_regulatory_region_t

> sl_wifi_regulatory_region_t

W-Fi regulatory region.

Note

- Australia and France regions not currently supported.

Enumerator	
SL_WIFI_REGION_AUSTRALIA	Wi-Fi Region Australia (not currently supported)
SL_WIFI_REGION_FRANCE	Wi-Fi Region France (not currently supported)
SL_WIFI_REGION_EUROPEAN_UNION	Wi-Fi Region European Union.
SL_WIFI_REGION_JAPAN	Wi-Fi Region Japan.
SL_WIFI_REGION_UNITED_STATES	Wi-Fi Region United States.

Definition at line 143 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_rate_protocol_t

> sl_wifi_rate_protocol_t

Wi-Fi rate protocols.

Note

- Recommended value for default behavior is SL_WIFI_RATE_PROTOCOL_AUTO
- 802.11ac not currently supported.

Enumerator	
SL_WIFI_RATE_PROTOCOL_B_ONLY	802.11b rates only (rates go here)
SL_WIFI_RATE_PROTOCOL_G_ONLY	802.11g rates only (rates go here)
SL_WIFI_RATE_PROTOCOL_N_ONLY	802.11n rates only (rates go here)
SL_WIFI_RATE_PROTOCOL_AC_ONLY	802.11ac rates only (rates go here) (not currently supported)
SL_WIFI_RATE_PROTOCOL_AX_ONLY	802.11ax rates only (rates go here)
SL_WIFI_RATE_PROTOCOL_AUTO	Automatic rate selection.

Definition at line 156 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_scan_type_t

> sl_wifi_scan_type_t

Wi-Fi scan types.

Enumerator	
SL_WIFI_SCAN_TYPE_ACTIVE	Active scan. Transmit probe requests and listen for responses.
SL_WIFI_SCAN_TYPE_PASSIVE	Passive scan. No active transmissions, listen for AP beacons and probe responses.
SL_WIFI_SCAN_TYPE_PROHIBITED_CHANNELS	Scan channels prohibited by regulatory region.
SL_WIFI_SCAN_TYPE_ADV_SCAN	Advance scanning of Access Points, when module is in connected state.

Definition at line 166 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_rate_t

> sl_wifi_rate_t

Wi-Fi transfer rates.

Enumerator	
SL_WIFI_AUTO_RATE	Wi-Fi Auto transfer rate.
SL_WIFI_RATE_11B_1	Wi-Fi 1 Mbps transfer rate for 802.11b.
SL_WIFI_RATE_11B_MIN	Wi-Fi Minimum transfer rate for 802.11b.
SL_WIFI_RATE_11B_2	Wi-Fi 2 Mbps transfer rate for 802.11b.
SL_WIFI_RATE_11B_5_5	Wi-Fi 5.5 Mbps transfer rate for 802.11b.
SL_WIFI_RATE_11B_11	Wi-Fi 11 Mbps transfer rate for 802.11b.
SL_WIFI_RATE_11B_MAX	Wi-Fi Maximum transfer rate for 802.11b.
SL_WIFI_RATE_11G_6	Wi-Fi 6 Mbps transfer rate for 802.11g.
SL_WIFI_RATE_11G_MIN	Wi-Fi Minimum transfer rate for 802.11g.
SL_WIFI_RATE_11G_9	Wi-Fi 9 Mbps transfer rate for 802.11g.
SL_WIFI_RATE_11G_12	Wi-Fi 12 Mbps transfer rate for 802.11g.
SL_WIFI_RATE_11G_18	Wi-Fi 18 Mbps transfer rate for 802.11g.
SL_WIFI_RATE_11G_24	Wi-Fi 24 Mbps transfer rate for 802.11g.
SL_WIFI_RATE_11G_36	Wi-Fi 36 Mbps transfer rate for 802.11g.
SL_WIFI_RATE_11G_48	Wi-Fi 48 Mbps transfer rate for 802.11g.
SL_WIFI_RATE_11G_54	Wi-Fi 54 Mbps transfer rate for 802.11g.
SL_WIFI_RATE_11G_MAX	Wi-Fi Maximum transfer rate for 802.11g.
SL_WIFI_RATE_11N_MCS0	Wi-Fi MCS index 0 transfer rate for 802.11n.
SL_WIFI_RATE_11N_MIN	Wi-Fi Minimum transfer rate for 802.11n.
SL_WIFI_RATE_11N_MCS1	Wi-Fi MCS index 1 transfer rate for 802.11n.
SL_WIFI_RATE_11N_MCS2	Wi-Fi MCS index 2 transfer rate for 802.11n.
SL_WIFI_RATE_11N_MCS3	Wi-Fi MCS index 3 transfer rate for 802.11n.
SL_WIFI_RATE_11N_MCS4	Wi-Fi MCS index 4 transfer rate for 802.11n.
SL_WIFI_RATE_11N_MCS5	Wi-Fi MCS index 5 transfer rate for 802.11n.
SL_WIFI_RATE_11N_MCS6	Wi-Fi MCS index 6 transfer rate for 802.11n.
SL_WIFI_RATE_11N_MCS7	Wi-Fi MCS index 7 transfer rate for 802.11n.
SL_WIFI_RATE_11N_MAX	Wi-Fi Maximum transfer rate for 802.11n.
SL_WIFI_RATE_11AX_MCS0	Wi-Fi MCS index 0 transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MIN	Wi-Fi Minimum transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MCS1	Wi-Fi MCS index 1 transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MCS2	Wi-Fi MCS index 2 transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MCS3	Wi-Fi MCS index 3 transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MCS4	Wi-Fi MCS index 4 transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MCS5	Wi-Fi MCS index 5 transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MCS6	Wi-Fi MCS index 6 transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MCS7	Wi-Fi MCS index 7 transfer rate for 802.11ax.
SL_WIFI_RATE_11AX_MAX	Wi-Fi Maximum transfer rate for 802.11ax.
SL_WIFI_RATE_INVALID	Wi-Fi Invalid transfer rate.

Definition at line 175 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_bss_type_t

sl_wifi_bss_type_t

Wi-Fi BSS type.

Enumerator	
SL_WIFI_BSS_TYPE_INFRASTRUCTURE	Wi-Fi BSS Type Infrastructure.
SL_WIFI_BSS_TYPE_ADHOC	Wi-Fi BSS Type ADHOC.
SL_WIFI_BSS_TYPE_ANY	Wi-Fi BSS Type ANY.
SL_WIFI_BSS_TYPE_UNKNOWN	Wi-Fi BSS Type Unknown.

Definition at line 222 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_band_t

sl_wifi_band_t

Wi-Fi radio band.

Note

- Only 2.4 GHz currently supported.

Enumerator	
SL_WIFI_AUTO_BAND	Wi-Fi Band Auto.
SL_WIFI_BAND_900MHZ	Wi-Fi Band 900Mhz (not currently supported)
SL_WIFI_BAND_2_4GHZ	Wi-Fi Band 2.4Ghz.
SL_WIFI_BAND_5GHZ	Wi-Fi Band 5Ghz (not currently supported)
SL_WIFI_BAND_6GHZ	Wi-Fi Band 6Ghz (not currently supported)
SL_WIFI_BAND_60GHZ	Wi-Fi Band 60Ghz (not currently supported)

Definition at line 231 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_bandwidth_t

sl_wifi_bandwidth_t

Note

- Only 20 MHz currently supported.

Enumerator	
SL_WIFI_AUTO_BANDWIDTH	Wi-Fi Bandwidth Auto.
SL_WIFI_BANDWIDTH_10MHz	Wi-Fi Bandwidth 10Mhz (not currently supported)
SL_WIFI_BANDWIDTH_20MHz	Wi-Fi Bandwidth 20Mhz.
SL_WIFI_BANDWIDTH_40MHz	Wi-Fi Bandwidth 40Mhz (not currently supported)
SL_WIFI_BANDWIDTH_80MHz	Wi-Fi Bandwidth 80Mhz (not currently supported)
SL_WIFI_BANDWIDTH_160MHz	Wi-Fi Bandwidth 160Mhz (not currently supported)

Definition at line 241 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_client_flag_t

> sl_wifi_client_flag_t

Option flags for client interfaces.

Enumerator	
SL_WIFI_NO_JOIN_OPTION	Wi-Fi Client Join with no flags.
SL_WIFI_JOIN_WITH_NO_CSA	Wi-Fi Client Join with no CSA.
SL_WIFI_JOIN_WITH_SCAN	Wi-Fi Client Join with Scan.

Definition at line 251 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_ap_flag_t

> sl_wifi_ap_flag_t

Option flags for AP interfaces.

Enumerator	
SL_WIFI_HIDDEN_SSID	Hide SSID of the AP.

Definition at line 258 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_listen_interval_time_unit_t

> sl_wifi_listen_interval_time_unit_t

Listen interval time units.

Enumerator	
SL_WIFI_LISTEN_INTERVAL_TIME_UNIT_BEACON	Time units specified in beacon periods.
SL_WIFI_LISTEN_INTERVAL_TIME_UNIT_DTIM	Time units specified in DTIM periods.

Definition at line 263 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_wps_mode_t

> sl_wifi_wps_mode_t

Wi-Fi WPS mode.

Enumerator	
SL_WIFI_WPS_PIN_MODE	WPS pin mode.
SL_WIFI_WPS_PUSH_BUTTON_MODE	WPS push button mode.

Definition at line 269 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_event_group_t

> sl_wifi_event_group_t

Wi-Fi event group.

Enumerator	
SL_WIFI_SCAN_RESULT_EVENTS	Event group for Wi-Fi scan results.
SL_WIFI_JOIN_EVENTS	Event group for Wi-Fi join status.
SL_WIFI_RX_PACKET_EVENTS	Event group for Wi-Fi received packet. This feature is not supported in current release.
SL_WIFI_COMMAND_RESPONSE_EVENTS	Event group for Wi-Fi command response. This feature is not supported in current release.
SL_WIFI_STATS_RESPONSE_EVENTS	Event group for Wi-Fi statistics response.
SL_WIFI_HTTP_OTA_FW_UPDATE_EVENTS	Event group for Wi-Fi OTA firmware update status via HTTP.
SL_WIFI_NETWORK_DOWN_EVENTS	Event group for Wi-Fi network down. This feature is not supported in current release.
SL_WIFI_NETWORK_UP_EVENTS	Event group for Wi-Fi network up. This feature is not supported in current release.
SL_WIFI_CLIENT_CONNECTED_EVENTS	Event group for Wi-Fi client connected status.
SL_WIFI_TWT_RESPONSE_EVENTS	Event group for Wi-Fi TWT response.
SL_WIFI_CLIENT_DISCONNECTED_EVENTS	Event group for Wi-Fi client disconnection status.
SL_WIFI_EVENT_GROUP_COUNT	Event group for Wi-Fi maximum default group count. Used internally by SDK.
SL_WIFI_EVENT_FAIL_INDICATION_EVENTS	Event group for Wi-Fi fail indication.

Definition at line 275 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_event_t

```
sl_wifi_event_t
```

Wi-Fi events.

Note

- Each event group has a matching event.
- Each event group may be a source of multiple different events.

Enumerator	
SL_WIFI_SCAN_RESULT_EVENT	Event for Wi-Fi scan result. Data would be of type of sl_wifi_scan_result_t.
SL_WIFI_JOIN_EVENT	Event for Wi-Fi join status. Data would be of type string.
SL_WIFI_RX_PACKET_EVENT	Event for Wi-Fi received packet. This feature is not supported in current release.
SL_WIFI_COMMAND_RESPONSE_EVENT	Event for Wi-Fi command response. This feature is not supported in current release.
SL_WIFI_STATS_RESPONSE_EVENT	Event for Wi-Fi statistics response. Data would be NULL.
SL_WIFI_HTTP_OTA_FW_UPDATE_EVENT	Event for Wi-Fi OTA firmware update status via HTTP. Data would be NULL.
SL_WIFI_NETWORK_DOWN_EVENT	Event for Wi-Fi network down. This feature is not supported in current release.
SL_WIFI_NETWORK_UP_EVENT	Event for Wi-Fi network up. This feature is not supported in current release.

SL_WIFI_CLIENT_CONNECTED_EVENT	Event for Wi-Fi client connected status. Data would be of type sl_mac_address_t.
SL_WIFI_TWT_RESPONSE_EVENT	Event for Wi-Fi TWT response. Data would be NULL.
SL_WIFI_CLIENT_DISCONNECTED_EVENT	Event for Wi-Fi client disconnection status. Data would of type sl_mac_address_t.
SL_WIFI_TWT_UNSOLICITED_SESSION_SUCCESS_EVENT	Event for TWT unsolicited session success. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_AP_REJECTED_EVENT	Event for TWT AP rejection. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_OUT_OF_TOLERANCE_EVENT	Event for TWT out of tolerance. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_RESPONSE_NOT_MATCHED_EVENT	Event for TWT response not matched. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_UNSUPPORTED_RESPONSE_EVENT	Event for TWT unsupported response. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_TEARDOWN_SUCCESS_EVENT	Event for TWT teardown success. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_AP_TEARDOWN_SUCCESS_EVENT	Event for TWT AP teardown success. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_FAIL_MAX_RETRIES_REACHED_EVENT	Event for TWT maximum retries reached. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_INACTIVE_DUE_TO_ROAMING_EVENT	Event for TWT inactive due to roaming. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_INACTIVE_DUE_TO_DISCONNECT_EVENT	Event for TWT inactive due to disconnect. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_INACTIVE_NO_AP_SUPPORT_EVENT	Event for TWT inactive due to no AP support. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_RESCHEDULE_TWT_SUCCESS_EVENT	Event for TWT suspend resume success. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_INFO_FRAME_EXCHANGE_FAILED_EVENT	Event for TWT info frame exchange failure. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_TWT_EVENTS_END	Event for TWT event end. Data would be of type sl_si91x_twt_response_t.
SL_WIFI_STATS_EVENT	Event for Wi-Fi statistics. This feature is not supported in current release.
SL_WIFI_STATS_AYSNC_EVENT	Event for Wi-Fi asynchronous statistics. Data would be of type sl_si91x_async_stats_response_t
SL_WIFI_STATS_ADVANCE_EVENT	Event for Wi-Fi advance statistics. Data would be of type sl_si91x_advance_stats_response_t
SL_WIFI_STATS_TEST_MODE_EVENT	Event for Wi-Fi test mode statistics. This feature is not supported in current release.
SL_WIFI_STATS_MODULE_STATE_EVENT	Event for Wi-Fi module state statistics. Data would be of type sl_si91x_module_state_stats_response_t
SL_WIFI_EVENT_FAIL_INDICATION	Event for Wi-Fi event failure indication.
SL_WIFI_INVALID_EVENT	Invalid Wi-Fi event. Data would be NULL.

Definition at line 294 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_reschedule_twt_action_t

sl_wifi_reschedule_twt_action_t

Enumeration defining actions related to Target Wake Time (TWT).

Enumerator	
SL_WIFI_SUSPEND_INDEFINITELY	Indicates the suspension of TWT for an indefinite period, effectively disabling TWT functionality until explicitly resumed.
SL_WIFI_SUSPEND_FOR_DURATION	Specifies that TWT should be suspended for a specified duration of time, after which it can automatically resume.
SL_WIFI_RESUME_IMMEDIATELY	Signifies an immediate resumption of TWT, allowing devices to continue adhering to TWT schedules.

Definition at line 385 of file components/protocol/wifi/inc/sl_wifi_constants.h

sl_wifi_data_rate_t

sl_wifi_data_rate_t

Enumerator	
SL_WIFI_DATA_RATE_1	Wi-Fi 1 Mbps transfer rate.
SL_WIFI_DATA_RATE_2	Wi-Fi 2 Mbps transfer rate.
SL_WIFI_DATA_RATE_5_5	Wi-Fi 5.5 Mbps transfer rate.
SL_WIFI_DATA_RATE_11	Wi-Fi 11 Mbps transfer rate.
SL_WIFI_DATA_RATE_6	Wi-Fi 6 Mbps transfer rate.
SL_WIFI_DATA_RATE_9	Wi-Fi 9 Mbps transfer rate.
SL_WIFI_DATA_RATE_12	Wi-Fi 12 Mbps transfer rate.
SL_WIFI_DATA_RATE_18	Wi-Fi 18 Mbps transfer rate.
SL_WIFI_DATA_RATE_24	Wi-Fi 24 Mbps transfer rate.
SL_WIFI_DATA_RATE_36	Wi-Fi 36 Mbps transfer rate.
SL_WIFI_DATA_RATE_48	Wif-Fi 48 Mbps transfer rate.
SL_WIFI_DATA_RATE_54	Wi-Fi 54 Mbps transfer rate.
SL_WIFI_DATA_RATE_MCS0	Wi-Fi MCS index 0 transfer rate.
SL_WIFI_DATA_RATE_MCS1	Wi-Fi MCS index 1 transfer rate.
SL_WIFI_DATA_RATE_MCS2	Wi-Fi MCS index 2 transfer rate.
SL_WIFI_DATA_RATE_MCS3	Wi-Fi MCS index 3 transfer rate.
SL_WIFI_DATA_RATE_MCS4	Wi-Fi MCS index 4 transfer rate.
SL_WIFI_DATA_RATE_MCS5	Wi-Fi MCS index 5 transfer rate.
SL_WIFI_DATA_RATE_MCS6	Wi-Fi MCS index 6 transfer rate.
SL_WIFI_DATA_RATE_MCS7	Wi-Fi MCS index 7 transfer rate.
SL_WIFI_DATA_RATE_MCS7_SG	

Definition at line 391 of file components/protocol/wifi/inc/sl_wifi_constants.h

Macro Definition Documentation

SL_WIFI_MAX_SCANNED_AP

```
#define SL_WIFI_MAX_SCANNED_AP
```

Value:

```
11
```

Max number of Access points that can be scanned.

Definition at line 18 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_MAX_CLIENT_COUNT

```
#define SL_WIFI_MAX_CLIENT_COUNT
```

Value:

```
16
```

Max number of stations when module is running in access point mode.

Definition at line 21 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_MAX_PSK_LENGTH

```
#define SL_WIFI_MAX_PSK_LENGTH
```

Value:

```
32
```

Max Length of Wi-Fi PSK credential.

Definition at line 24 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_MAX_PMK_LENGTH

```
#define SL_WIFI_MAX_PMK_LENGTH
```

Value:

```
64
```

Max Length of Wi-Fi PMK credential.

Definition at line 27 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_WEP_KEY_LENGTH

```
#define SL_WIFI_WEP_KEY_LENGTH
```

Value:

```
32
```

Max length of Key in WEP security.

Definition at line 30 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_WEP_KEY_COUNT

```
#define SL_WIFI_WEP_KEY_COUNT
```

Value:

```
4
```

Max number of keys for WEP security.

Definition at line 33 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_EAP_USER_NAME_LENGTH

```
#define SL_WIFI_EAP_USER_NAME_LENGTH
```

Value:

```
64
```

Max Length of User Name in enterprise security.

Definition at line 36 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_EAP_PASSWORD_LENGTH

```
#define SL_WIFI_EAP_PASSWORD_LENGTH
```

Value:

```
128
```

Max Length of password in enterprise security.

Definition at line 39 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_EAP_CERTIFICATE_KEY_LENGTH

```
#define SL_WIFI_EAP_CERTIFICATE_KEY_LENGTH
```

Value:

```
80
```

Max Length of certificate key in enterprise security.

Definition at line 42 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_SELECT_INTERNAL_ANTENNA

```
#define SL_WIFI_SELECT_INTERNAL_ANTENNA
```

Value:

```
0
```

Select Internal Antenna for Wi-Fi.

Definition at line 45 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_SELECT_EXTERNAL_ANTENNA

```
#define SL_WIFI_SELECT_EXTERNAL_ANTENNA
```

Value:

```
1
```

Select External Antenna for Wi-Fi.

Definition at line 48 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_DEFAULT_INTERFACE

```
#define SL_WIFI_DEFAULT_INTERFACE
```

Value:

```
sl_wifi_get_default_interface()
```

Default Wi-Fi interface macro.

Definition at line 51 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_NEVER_ROAM

```
#define SL_WIFI_NEVER_ROAM
```

Value:

```
0x7FFFFFFF
```

Max Wi-Fi roaming trigger interval.

Definition at line 54 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_AUTO_CHANNEL

```
#define SL_WIFI_AUTO_CHANNEL
```

Value:

```
0
```

Auto detect channel.

Definition at line 416 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_ARGS_CHECK_NULL_POINTER

```
#define SL_WIFI_ARGS_CHECK_NULL_POINTER
```

Value:

```
{                              \
  if (ptr == NULL) {           \
    return SL_STATUS_NULL_POINTER;  \
  }                            \
}
```

API input checks.

Definition at line 419 of file components/protocol/wifi/inc/sl_wifi_constants.h

SL_WIFI_ARGS_CHECK_INVALID_INTERFACE

```
#define SL_WIFI_ARGS_CHECK_INVALID_INTERFACE
```

Value:

```
{                                                    \
  if (!((interface == SL_WIFI_CLIENT_INTERFACE) || (interface == SL_WIFI_CLIENT_INTERFACE)    \
      || (interface == SL_WIFI_AP_INTERFACE) || (interface == SL_WIFI_2_4GHZ_INTERFACE))) { \
    return SL_STATUS_WIFI_UNKNOWN_INTERFACE;                                  \
  }                                                  \
}
```

Interface input checks.

Definition at line 427 of file components/protocol/wifi/inc/sl_wifi_constants.h

Callback Framework

Callback Framework

This section provides a reference to Wi-Fi API callback handling functions.

Typedefs

typedef sl_status_t(* sl_wifi_callback_function_t)(sl_wifi_event_t event, void *data, uint32_t data_length, void *arg)
Generic callback for Wi-Fi group event of type sl_wifi_event_group_t.

typedef sl_status_t(* sl_wifi_scan_callback_t)(sl_wifi_event_t event, sl_wifi_scan_result_t *data, uint32_t data_length, void *optional_arg)
Callback for SL_WIFI_SCAN_RESULT_EVENTS group event.

typedef sl_status_t(* sl_wifi_stats_callback_t)(sl_wifi_event_t event, void *data, uint32_t data_length, void *optional_arg)
Callback for SL_WIFI_STATS_RESPONSE_EVENTS group events.

typedef sl_status_t(* sl_wifi_join_callback_t)(sl_wifi_event_t event, char *data, uint32_t data_length, void *arg)
Callback for SL_WIFI_JOIN_EVENTS group events.

typedef sl_status_t(* sl_wifi_twt_config_callback_t)(sl_wifi_event_t event, sl_si91x_twt_response_t *data, uint32_t data_length, void *arg)
Callback for SL_WIFI_TWT_RESPONSE_EVENTS group events.

Functions

sl_status_t sl_wifi_set_callback(sl_wifi_event_group_t group, sl_wifi_callback_function_t function, void *optional_arg)
Register a callback for selected event group.

sl_status_t sl_wifi_default_event_handler(sl_wifi_event_t event, sl_wifi_buffer_t *buffer)
Default Wi-Fi event handler to passed to sl_wifi_init.

sl_status_t sl_wifi_set_scan_callback(sl_wifi_scan_callback_t function, void *optional_arg)
Register callback for SL_WIFI_SCAN_RESULT_EVENTS group event from sl_wifi_event_group_t.

sl_status_t sl_wifi_set_join_callback(sl_wifi_join_callback_t function, void *optional_arg)
Register callback for SL_WIFI_JOIN_EVENTS group event from sl_wifi_event_group_t.

sl_status_t sl_wifi_set_twt_config_callback(sl_wifi_twt_config_callback_t function, void *optional_arg)
Register callback for SL_WIFI_TWT_RESPONSE_EVENTS group event from sl_wifi_event_group_t.

sl_status_t sl_wifi_set_stats_callback(sl_wifi_stats_callback_t function, void *optional_arg)
Register callback for SL_WIFI_STATS_RESPONSE_EVENTS group event from sl_wifi_event_group_t.

Macros

#define SL_WIFI_CHECK_IF_EVENT_FAILED (event)
Generic macro for callback functions to check if the event has Failed.

Typedef Documentation

sl_wifi_callback_function_t

```
sl_wifi_callback_function_t )(sl_wifi_event_t event, void *data, uint32_t data_length, void *arg)
```

Generic callback for Wi-Fi group event of type sl_wifi_event_group_t.

Parameters

N/A	event	Wi-Fi event of type sl_wifi_event_t
N/A	data	Data received
N/A	data_length	Data length
N/A	optional_arg	Optional user provided argument passed in sl_wifi_set_callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- In case of event failure, SL_WIFI_FAIL_EVENT_STATUS_INDICATION bit is set in event, data will be of type sl_status_t and data_length can be ignored

Definition at line `47` of file `components/protocol/wifi/inc/sl_wifi_callback_framework.h`

sl_wifi_scan_callback_t

```
sl_wifi_scan_callback_t )(sl_wifi_event_t event, sl_wifi_scan_result_t *data, uint32_t data_length, void *optional_arg)
```

Callback for SL_WIFI_SCAN_RESULT_EVENTS group event.

Parameters

N/A	event	Wi-Fi event of type sl_wifi_event_t
N/A	data	Scan results of type sl_wifi_scan_result_t
N/A	data_length	Data length
N/A	optional_arg	Optional user provided argument passed in sl_wifi_set_scan_callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- In case of event failure, SL_WIFI_FAIL_EVENT_STATUS_INDICATION bit is set in event, data will be of type sl_status_t and data_length can be ignored

Definition at line `65` of file `components/protocol/wifi/inc/sl_wifi_callback_framework.h`

sl_wifi_stats_callback_t

```
sl_wifi_stats_callback_t )(sl_wifi_event_t event, void *data, uint32_t data_length, void *optional_arg)
```

Callback for SL_WIFI_STATS_RESPONSE_EVENTS group events.

Parameters

N/A	event	Wi-Fi event of type sl_wifi_event_t Individual Wi-Fi events related to SL_WIFI_STATS_RESPONSE_EVENTS is as follows.

sl_wifi_event_t	DataType
SL_WIFI_STATS_EVENT	Not supported in current release
SL_WIFI_STATS_AYSNC_EVENT	sl_si91x_async_stats_response_t
SL_WIFI_STATS_ADVANCE_EVENT	sl_si91x_advance_stats_response_t
SL_WIFI_STATS_TEST_MODE_EVENT	Not supported in current release
SL_WIFI_STATS_MODULE_STATE_EVENT	sl_si91x_module_state_stats_response_t

N/A	data	Data received.
N/A	data_length	Data length
N/A	optional_arg	Optional user provided argument passed in sl_wifi_set_stats_callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- In case of event failure, SL_WIFI_FAIL_EVENT_STATUS_INDICATION bit is set in event, data will be of type sl_status_t and data_length can be ignored.

Definition at line 94 of file components/protocol/wifi/inc/sl_wifi_callback_framework.h

sl_wifi_join_callback_t

```
sl_wifi_join_callback_t )(sl_wifi_event_t event, char *data, uint32_t data_length, void *arg)
```

Callback for SL_WIFI_JOIN_EVENTS group events.

Parameters

N/A	event	Wi-Fi event of type sl_wifi_event_t
N/A	data	Data received in string.

Data received	Description
C	Module connection success
F	Module connection failed

N/A	data_length	Data length
N/A	optional_arg	Optional user provided argument passed in sl_wifi_set_join_callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- In case of event failure, SL_WIFI_FAIL_EVENT_STATUS_INDICATION bit is set in event, data will be of type sl_status_t and data_length can be ignored

Definition at line 119 of file components/protocol/wifi/inc/sl_wifi_callback_framework.h

sl_wifi_twt_config_callback_t

> sl_wifi_twt_config_callback_t)(sl_wifi_event_t event, sl_si91x_twt_response_t *data, uint32_t data_length, void *arg)

Callback for SL_WIFI_TWT_RESPONSE_EVENTS group events.

Parameters

N/A	event	Wi-Fi event of type sl_wifi_event_t Individual Wi-Fi events related to SL_WIFI_TWT_RESPONSE_EVENTS is as follows.

sl_wifi_event_t
SL_WIFI_TWT_UNSOLICITED_SESSION_SUCCESS_EVENT
SL_WIFI_TWT_AP_REJECTED_EVENT
SL_WIFI_TWT_OUT_OF_TOLERANCE_EVENT
SL_WIFI_TWT_RESPONSE_NOT_MATCHED_EVENT
SL_WIFI_TWT_UNSUPPORTED_RESPONSE_EVENT
SL_WIFI_TWT_TEARDOWN_SUCCESS_EVENT
SL_WIFI_TWT_AP_TEARDOWN_SUCCESS_EVENT
SL_WIFI_TWT_FAIL_MAX_RETRIES_REACHED_EVENT
SL_WIFI_TWT_INACTIVE_DUE_TO_ROAMING_EVENT
SL_WIFI_TWT_INACTIVE_DUE_TO_DISCONNECT_EVENT
SL_WIFI_TWT_INACTIVE_NO_AP_SUPPORT_EVENT
SL_WIFI_RESCHEDULE_TWT_SUCCESS_EVENT
SL_WIFI_TWT_INFO_FRAME_EXCHANGE_FAILED_EVENT

N/A	data	Data received of type sl_si91x_twt_response_t.
N/A	data_length	Data length
N/A	optional_arg	Optional user provided argument passed in sl_wifi_set_twt_config_callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- In case of event failure, SL_WIFI_FAIL_EVENT_STATUS_INDICATION bit is set in event, data will be of type sl_status_t and data_length can be ignored

Definition at line `153` of file `components/protocol/wifi/inc/sl_wifi_callback_framework.h`

Function Documentation

sl_wifi_set_callback

> sl_status_t sl_wifi_set_callback (sl_wifi_event_group_t group, sl_wifi_callback_function_t function, void *optional_arg)

Register a callback for selected event group.

Parameters

[in]	group	group id of the event. See sl_wifi_event_group_t
[in]	function	Function pointer to callback of type sl_wifi_callback_function_t

[in]	optional_arg	Optional user provided argument. This will be passed back to callback handler of type sl_wifi_callback_function_t

All the individual Wi-Fi events related to specific group will be triggered via this group callback

- Pre-conditions:
  - Pre-conditions:
  - Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- Callbacks can be set only for sl_wifi_event_group_t, not for individual events (sl_wifi_event_t)

Definition at line 177 of file components/protocol/wifi/inc/sl_wifi_callback_framework.h

sl_wifi_default_event_handler

> sl_status_t sl_wifi_default_event_handler (sl_wifi_event_t event, sl_wifi_buffer_t *buffer)

Default Wi-Fi event handler to passed to sl_wifi_init.

Parameters

[in]	event	Wi-Fi event of type of sl_wifi_event_t
[in]	buffer	Buffer containing raw data from TA firmware

This function will dispatch Wi-Fi events and invokes respective Wi-Fi group event callback **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- Passing this event handler is optional. User can implement his own event dispatching framework.

Definition at line 191 of file components/protocol/wifi/inc/sl_wifi_callback_framework.h

sl_wifi_set_scan_callback

> static sl_status_t sl_wifi_set_scan_callback (sl_wifi_scan_callback_t function, void *optional_arg)

Register callback for SL_WIFI_SCAN_RESULT_EVENTS group event from sl_wifi_event_group_t.

Parameters

[in]	function	Optional user provided argument. This will be passed back to callback handler of type sl_wifi_scan_callback_t
N/A	optional_arg	

- Pre-conditions:
  - Pre-conditions:
  - Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

-

sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- All the individual Wi-Fi events related to this group will be triggered via this callback

Definition at line 208 of file components/protocol/wifi/inc/sl_wifi_callback_framework.h

sl_wifi_set_join_callback

```
static sl_status_t sl_wifi_set_join_callback (sl_wifi_join_callback_t function, void *optional_arg)
```

Register callback for SL_WIFI_JOIN_EVENTS group event from sl_wifi_event_group_t.

Parameters

[in]	function	Function pointer to callback. This will be passed back to callback handler of type sl_wifi_join_callback_t
[in]	optional_arg	Optional user provided argument. This will be passed back to callback handler.

- Pre-conditions:
  - Pre-conditions:
  - Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- All the individual Wi-Fi events related to this group will be triggered via this callback

Definition at line 230 of file components/protocol/wifi/inc/sl_wifi_callback_framework.h

sl_wifi_set_twt_config_callback

```
static sl_status_t sl_wifi_set_twt_config_callback (sl_wifi_twt_config_callback_t function, void *optional_arg)
```

Register callback for SL_WIFI_TWT_RESPONSE_EVENTS group event from sl_wifi_event_group_t.

Parameters

[in]	function	Function pointer to callback. This will be passed back to callback handler of type sl_wifi_twt_config_callback_t
[in]	optional_arg	Optional user provided argument. This will be passed back to callback handler.

- Pre-conditions:
  - Pre-conditions:
  - Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- All the individual Wi-Fi events related to this group will be triggered via this callback.

Definition at line 252 of file components/protocol/wifi/inc/sl_wifi_callback_framework.h

sl_wifi_set_stats_callback

```
static sl_status_t sl_wifi_set_stats_callback (sl_wifi_stats_callback_t function, void *optional_arg)
```

Register callback for SL_WIFI_STATS_RESPONSE_EVENTS group event from sl_wifi_event_group_t.

Parameters

[in]	function	Function pointer to callback. This will be passed back to callback handler of type sl_wifi_stats_callback_t
[in]	optional_arg	Optional user provided argument. This will be passed back to callback handler.

- Pre-conditions:
  - Pre-conditions:
  - Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- All the individual Wi-Fi events related to this group will be triggered via this callback.

Definition at line `274` of file `components/protocol/wifi/inc/sl_wifi_callback_framework.h`

Macro Definition Documentation

SL_WIFI_CHECK_IF_EVENT_FAILED

```
#define SL_WIFI_CHECK_IF_EVENT_FAILED
```

Value:

```
(event)
```

Generic macro for callback functions to check if the event has Failed.

Definition at line `29` of file `components/protocol/wifi/inc/sl_wifi_callback_framework.h`

Overview

Overview

Introduction

The v3.x Bluetooth stack is an advanced Bluetooth 5-compliant protocol stack implementing the Bluetooth Low Energy (BLE) standard. It supports multiple connections and concurrent central, peripheral, broadcaster, and observer roles. The v3.x Silicon Labs Bluetooth stack is meant for Silicon Labs SiWx91x SoCs modules.

BLE Features

- BLE Central, BLE Peripheral, BLE dual role support.
- Advertising, scanning, and connection interval support.
- 32-bit UUID support
- BLE secure connections
- Data length extensions
- BLE 2Mbps
- BLE long range: 125 Kbps, 500 Kbps
- BLE channel classification
- Supported profiles:
  - Generic Attribute Profile (GATT)
  - Generic Access Profile (GAP)
  - All GAP and GATT based Profiles & Services are Supported

APIs

APIs

This section provides a reference to the Bluetooth Low Energy (BLE) API including functions, data types, and events.

Modules

Functions

Functions

This section provides a reference to Bluetooth Low Energy (BLE) API functions:

- Common functions to initialize and configure BLE features.
- BLE functions providing BLE connectivity and protocol functions.

Modules

Common

BLE

Common

Common

Modules

rsi_bt_resp_get_bt_stack_version_s

rsi_bt_per_stats_s

Typedefs

typedef struct rsi_bt_resp_get_bt_stack_version_s — rsi_bt_resp_get_bt_stack_version_t

typedef struct rsi_bt_per_stats_s — rsi_bt_per_stats_t

Functions

int32_t rsi_bt_set_bd_addr(uint8_t *dev_addr)
Set the device BD address.

int32_t rsi_bt_set_local_name(uint8_t *local_name)
Set the given name to local device.

int32_t rsi_bt_cmd_update_gain_table_offset_or_max_pwr(uint8_t node_id, uint8_t payload_len, uint8_t *payload, uint8_t req_type)
Update gain table offset/max power.

int32_t rsi_bt_get_local_name(rsi_bt_resp_get_local_name_t *bt_resp_get_local_name)

int32_t rsi_bt_get_rssi(uint8_t *dev_addr, int8_t *resp)
Get the RSSI of the remote device.

int32_t rsi_bt_get_local_device_address(uint8_t *resp)
Get the local device address.

int32_t rsi_bt_get_bt_stack_version(rsi_bt_resp_get_bt_stack_version_t *bt_resp_get_bt_stack_version)
Get the BT stack version.

int32_t rsi_bt_init(void)
Initialize the BT device.

int32_t rsi_bt_deinit(void)
Deinitialize the BT device.

int32_t rsi_bt_set_antenna(uint8_t antenna_value)
Select either internal / external antenna on the chip.

int32_t rsi_bt_power_save_profile(uint8_t psp_mode, uint8_t psp_type)
Select the power save profile mode for BT / BLE.

int32_t rsi_bt_per_stats(uint8_t cmd_type, struct rsi_bt_per_stats_s *per_stats)
Request the local device for BT PER operation.

Typedef Documentation

rsi_bt_resp_get_bt_stack_version_t

```
typedef struct rsi_bt_resp_get_bt_stack_version_s rsi_bt_resp_get_bt_stack_version_t
```

Definition at line 278 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

rsi_bt_per_stats_t

```
typedef struct rsi_bt_per_stats_s rsi_bt_per_stats_t
```

Definition at line 342 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

Function Documentation

rsi_bt_set_bd_addr

```
int32_t rsi_bt_set_bd_addr (uint8_t *dev_addr)
```

Set the device BD address.

Parameters

[in]	dev_addr	- public address of the device to be set

This is a blocking API.

- Pre-conditions:
  - needs to be called immediately after device initialization.

Returns

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure
- -3 - Command is given in wrong state(i.e not immediate after opermode) **Note**
  - -This is a blocking API.Refer Error Codes section for above error codes error-codes .

Definition at line 77 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h

rsi_bt_set_local_name

```
int32_t rsi_bt_set_local_name (uint8_t *local_name)
```

Set the given name to local device.

Parameters

[in]	local_name	- Name to be set to the local device.

This is a blocking API.

- Pre-conditions:
  - Device should be initialized before calling this API.

Note

- For BLE alone Opermode : When the name of the local device is set to a value with length more than 16 bytes then error is returned with an error code 0x4E66.

Returns

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure **Note**
  - Refer Error Codes section for above error codes error-codes.

Definition at line `94` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h`

rsi_bt_cmd_update_gain_table_offset_or_max_pwr

```
int32_t rsi_bt_cmd_update_gain_table_offset_or_max_pwr (uint8_t node_id, uint8_t payload_len, uint8_t *payload, uint8_t req_type)
```

Update gain table offset/max power.

Parameters

[in]	node_id	- Node ID (0 - BLE, 1 - BT).
[in]	payload_len	- Length of the payload.
[in]	payload	- Payload containing table data of gain table offset/max power
[in]	req_type	- update gain table request type (0 - max power update, 1 - offset update)

This is blocking API.

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
- 0x4F01 - Invalid gain table payload length
- 0x4F02 - Invalid region.
- 0x4F03 - Invalid gain table offset request type
- 0x4F04 - Invalid node id.
  Note
  - Refer Error Codes section for above error codes error-codes.

Definition at line `120` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h`

rsi_bt_get_local_name

```
int32_t rsi_bt_get_local_name (rsi_bt_resp_get_local_name_t *bt_resp_get_local_name)
```

Parameters

N/A	bt_resp_get_local_name

Definition at line 138 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h

rsi_bt_get_rssi

```
int32_t rsi_bt_get_rssi (uint8_t *dev_addr, int8_t *resp)
```

Get the RSSI of the remote device.

Parameters

[in]	dev_addr	- Remote device address.
[out]	resp	- Parameter to hold the response of this API, RSSI is filled in this resp parameter.

This is a blocking API.

- Pre-conditions:
  - rsi_bt_connect() API need to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure **Note**
  - Refer Error Codes section for above error codes error-codes.

Definition at line 154 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h

rsi_bt_get_local_device_address

```
int32_t rsi_bt_get_local_device_address (uint8_t *resp)
```

Get the local device address.

Parameters

[out]	resp	- Parameter to hold the response of this API, local bd_addr is filled in this resp parameter.

This is a blocking API.

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure **Note**
  - Refer Error Codes section for above error codes error-codes.

Definition at line 169 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h

rsi_bt_get_bt_stack_version

```
int32_t rsi_bt_get_bt_stack_version (rsi_bt_resp_get_bt_stack_version_t *bt_resp_get_bt_stack_version)
```

Get the BT stack version.

Parameters

[out]	bt_resp_get_bt_stack_version	- Response buffer to hold the response of this API. Please refer rsi_bt_resp_get_bt_stack_version_s structure for more info

This is a blocking API. **Returns**

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure **Note**
  - Refer Error Codes section for above error codes error-codes.

Definition at line 182 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h

rsi_bt_init

```
int32_t rsi_bt_init (void)
```

Initialize the BT device.

Parameters

N/A		

This is a blocking API.

- Pre-conditions:
  - Device should be initialized before calling this API.
    If the device is in powersave, get back the device to ACTIVE MODE by using rsi_bt_power_save_profile()

Returns

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure **Note**
  - Refer Error Codes section for above error codes error-codes.

Definition at line 199 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h

rsi_bt_deinit

```
int32_t rsi_bt_deinit (void)
```

Deinitialize the BT device.

Parameters

N/A		

This is a blocking API.

- Pre-conditions:
  - Device should be initialized before this API.

Returns

- The following values are returned:

0 - Success

- Non-Zero Value - Failure **Note**
  - Refer Error Codes section for above error codes error-codes.

Definition at line `215` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h`

rsi_bt_set_antenna

```
int32_t rsi_bt_set_antenna (uint8_t antenna_value)
```

Select either internal / external antenna on the chip.

Parameters

[in]	antenna_value	- Parameter is used to select either internal or external antenna. Possible values: • 0x00 RSI_SEL_INTERNAL_ANTENNA • 0x01 RSI_SEL_EXTERNAL_ANTENNA

This is a blocking API.

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure **Note**
  - Refer Error Codes section for above error codes error-codes.

Definition at line `234` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h`

rsi_bt_power_save_profile

```
int32_t rsi_bt_power_save_profile (uint8_t psp_mode, uint8_t psp_type)
```

Select the power save profile mode for BT / BLE.

Parameters

[in]	psp_mode	Following psp_mode is defined. • 0 - RSI_ACTIVE. In this mode module is active and power save is disabled. • 1 - RSI_SLEEP_MODE_1. On mode. In this sleep mode, SoC will never turn off, therefore no • handshake is required before sending data to the module. BT/BLE does not support this mode. • 2 - RSI_SLEEP_MODE_2. Connected sleep mode. In this sleep mode, SoC will go to sleep based • on GPIO or Message, therefore handshake is required before sending data to the module. • 8 - RSI_SLEEP_MODE_8 :Deep sleep mode with RAM RETENTION. • 10- RSI_SLEEP_MODE_10 : Deep sleep mode without RAM RETENTION. • In this sleep mode, module will turn off the • SoC. Since SoC is turn off, therefore handshake is required before sending data to the module. •

[in]	psp_type	Following psp_type is defined. • 0 - RSI_MAX_PSP. This psp_type will be used for max power saving • 1 - Fast PSP •

This is a blocking API.

- Pre-conditions:
  - Device should be initialized before calling this API

Returns

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure
  Note
  - If the user wants to enable power save in CoEx mode (WLAN + BT LE) mode - It is mandatory to enable WLAN power save along with BT LE power save.
- **Note**
  - The device will enter into power save if and only if both protocol (WLAN, BLE) power save modes are enabled.
- **Note**
  - Refer Error Codes section for above error codes error-codes.
  - psp_type is only valid in psp_mode 2.
- BT/BLE does not support in RSI_SLEEP_MODE_1.
- BT/BLE supports only RSI_MAX_PSP mode. Remaining modes are not supported.

Definition at line `285` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h`

rsi_bt_per_stats

```
int32_t rsi_bt_per_stats (uint8_t cmd_type, struct rsi_bt_per_stats_s *per_stats)
```

Request the local device for BT PER operation.

Parameters

[in]	cmd_type	- Parameter to define the command id type for PER operation. • BT_PER_STATS_CMD_ID (0x08) - Command id enables PER statistics • BT_TRANSMIT_CMD_ID (0x15) - Command id enables PER transmit • BT_RECEIVE_CMD_ID (0x16) - Command id enables PER receive
[in]	rsi_bt_per_stats	- reference to the response structure. Please refer to rsi_bt_per_stats_s structure for more info.

- Pre-conditions:
  - Call rsi_bt_per_tx() or rsi_bt_per_rx() before calling this API.

Returns

- The following values are returned:
  - 0 - Success
- Non-Zero Value - Failure **Note**
  - Refer Error Codes section for common error codes error-codes .

Definition at line `307` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common_apis.h`

rsi_bt_resp_get_bt_stack_version_s

Public Attributes

int8_t	stack_version stack version variable

Public Attribute Documentation

stack_version

```
int8_t rsi_bt_resp_get_bt_stack_version_s::stack_version[10]
```

stack version variable

Definition at line 277 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

rsi_bt_per_stats_s

Public Attributes

uint16_t	crc_fail_cnt Packet count of CRC fails (Cyclic Redundancy Check (CRC))
uint16_t	crc_pass_cnt Packet count of CRC fails (Cyclic Redundancy Check (CRC))
uint16_t	tx_abort_cnt Packet count of aborted Tx.
uint16_t	rx_drop_cnt Packet count of dropped Rx.
uint16_t	rx_cca_idle_cnt Packet count of CCA Idle (Clear Channel Assessment (CCA))
uint16_t	rx_start_idle_cnt Packet count of Rx start.
uint16_t	rx_abrt_cnt Packet count of aborted Rx.
uint16_t	tx_dones Packet count of successful transmissions.
int8_t	rssi Received Signal Strength Indicator of the packet.
uint16_t	id_pkts_rcvd Packet count of ID packets received.
uint16_t	dummy Dummy array of length 5.

Public Attribute Documentation

crc_fail_cnt

```
uint16_t rsi_bt_per_stats_s::crc_fail_cnt
```

Packet count of CRC fails (Cyclic Redundancy Check (CRC))

Definition at line 321 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

crc_pass_cnt

```
uint16_t rsi_bt_per_stats_s::crc_pass_cnt
```

Packet count of CRC fails (Cyclic Redundancy Check (CRC))

Definition at line 323 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

tx_abort_cnt

```
uint16_t rsi_bt_per_stats_s::tx_abort_cnt
```

Packet count of aborted Tx.

Definition at line 325 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

rx_drop_cnt

```
uint16_t rsi_bt_per_stats_s::rx_drop_cnt
```

Packet count of dropped Rx.

Definition at line 327 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

rx_cca_idle_cnt

```
uint16_t rsi_bt_per_stats_s::rx_cca_idle_cnt
```

Packet count of CCA Idle (Clear Channel Assessment (CCA))

Definition at line 329 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

rx_start_idle_cnt

```
uint16_t rsi_bt_per_stats_s::rx_start_idle_cnt
```

Packet count of Rx start.

Definition at line 331 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

rx_abrt_cnt

```
uint16_t rsi_bt_per_stats_s::rx_abrt_cnt
```

Packet count of aborted Rx.

Definition at line 333 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

tx_dones

```
uint16_t rsi_bt_per_stats_s::tx_dones
```

Packet count of successful transmissions.

Definition at line 335 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

rssi

```
int8_t rsi_bt_per_stats_s::rssi
```

Received Signal Strength Indicator of the packet.

Definition at line 337 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

id_pkts_rcvd

```
uint16_t rsi_bt_per_stats_s::id_pkts_rcvd
```

Packet count of ID packets received.

Definition at line 339 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

dummy

```
uint16_t rsi_bt_per_stats_s::dummy[5]
```

Dummy array of length 5.

Definition at line 341 of file components/device/silabs/si91x/wireless/ble/inc/rsi_bt_common.h

BLE

BLE

BLE functions provide all the connectivity and protocol features of BLE:

- GAP functions to implement the Generic Access Profile (GAP) protocol.
- GATT functions to implement the Generic Attribute Profile (GATT) protocol.
- Test Mode functions to access Bluetooth Low Energy (BLE) test mode features.
- Register Callbacks section to refer to functions to register Bluetooth Low Energy (BLE) API callbacks.
- Callbacks Declarations for handling Bluetooth Low Energy (BLE) API callbacks.

Modules

GAP

Modules

rsi_ble_set_smp_pairing_capabilty_data

rsi_ble_resp_read_phy_s

rsi_ble_resp_read_max_data_length_s

Typedefs

typedef struct rsi_ble_set_smp_pairing_capabilty_data	rsi_ble_set_smp_pairing_capabilty_data_t
typedef struct rsi_ble_resp_read_phy_s	rsi_ble_resp_read_phy_t
typedef struct rsi_ble_resp_read_max_data_length_s	rsi_ble_read_max_data_length_t

Functions

int32_t	rsi_ble_set_random_address(void) Request the local device to set a random address.
int32_t	rsi_ble_set_random_address_with_value(uint8_t *random_addr) Request the local device to set a given random address.
int32_t	rsi_ble_start_advertising(void) Request the local device to start advertising.
int32_t	rsi_ble_start_advertising_with_values(void *rsi_ble_adv) Request the local device to start advertising with specified values.
int32_t	rsi_ble_encrypt(uint8_t *key, uint8_t *data, uint8_t *resp) Encrypt the plain text data fed by the user using the key provided.
int32_t	rsi_ble_stop_advertising(void) Stop advertising.
int32_t	rsi_ble_set_advertise_data(uint8_t *data, uint16_t data_len) Set the advertising data.
int32_t	rsi_ble_set_scan_response_data(uint8_t *data, uint16_t data_len) Request the local device to set the scan response data.

int32_t rsi_ble_start_scanning(void)
Start scanning.

int32_t rsi_ble_start_scanning_with_values(void *rsi_ble_scan_params)
Start scanning with values.

int32_t rsi_ble_stop_scanning(void)
Stop scanning.

int32_t rsi_ble_connect_with_params(uint8_t remote_dev_addr_type, int8_t *remote_dev_addr, uint16_t scan_interval, uint16_t scan_window, uint16_t conn_interval_max, uint16_t conn_interval_min, uint16_t conn_latency, uint16_t supervision_tout)
Connect to the remote BLE device with the user configured parameters.

int32_t rsi_ble_connect(uint8_t remote_dev_addr_type, int8_t *remote_dev_addr)
Connect to the remote BLE device.

int32_t rsi_ble_enhance_connect_with_params(void *ble_enhance_conn_params)
Connect to the remote BLE device with the user configured parameters.

int32_t rsi_ble_connect_cancel(int8_t *remote_dev_address)
Cancel the connection to the remote BLE device.

int32_t rsi_ble_disconnect(int8_t *remote_dev_address)
Disconnect with the remote BLE device.

int32_t rsi_ble_get_device_state(uint8_t *resp)
Get the local device state.

int32_t rsi_ble_set_smp_pairing_cap_data(rsi_ble_set_smp_pairing_capabilty_data_t *smp_pair_cap_data)
Set the SMP Pairing Capability of local device.

int32_t rsi_ble_set_local_irk_value(uint8_t *l_irk)
Set the IRK value to the local device.

int32_t rsi_ble_conn_param_resp(uint8_t *remote_dev_address, uint8_t status)
Give the response for the remote device connection parameter request.

int32_t rsi_ble_smp_pair_request(uint8_t *remote_dev_address, uint8_t io_capability, uint8_t mitm_req)
Request the SMP pairing process with the remote device.

int32_t rsi_ble_smp_pair_failed(uint8_t *remote_dev_address, uint8_t reason)
Send SMP pairing failure reason to the remote device.

int32_t rsi_ble_ltk_req_reply(uint8_t *remote_dev_address, uint8_t reply_type, uint8_t *ltk)
Send the local long term key of its associated local EDIV and local Rand.

int32_t rsi_ble_smp_pair_response(uint8_t *remote_dev_address, uint8_t io_capability, uint8_t mitm_req)
Send SMP pairing response during the process of pairing with the remote device.

int32_t rsi_ble_smp_passkey(uint8_t *remote_dev_address, uint32_t passkey)
Send SMP passkey during SMP pairing process with the remote device.

int32_t rsi_ble_get_le_ping_timeout(uint8_t *remote_dev_address, uint16_t *time_out)
Get the timeout value of the LE ping.

int32_t rsi_ble_set_le_ping_timeout(uint8_t *remote_dev_address, uint16_t time_out)
Set the timeout value of the LE ping.

int32_t rsi_ble_clear_acceptlist(void)
Clear all the BD address present in accept list.

int32_t rsi_ble_addto_acceptlist(int8_t *dev_address, uint8_t dev_addr_type)
Add BD address to accept list.

int32_t rsi_ble_deletefrom_acceptlist(int8_t *dev_address, uint8_t dev_addr_type)
Delete particular BD address from accept list.

int32_t rsi_ble_resolvlist(uint8_t process_type, uint8_t remote_dev_addr_type, uint8_t *remote_dev_address, uint8_t *peer_irk, uint8_t *local_irk)
resolvlist API used for multiple purpose based on the process type.

int32_t rsi_ble_get_resolving_list_size(uint8_t *resp)
Request to get resolving list size.

int32_t rsi_ble_set_addr_resolution_enable(uint8_t enable, uint16_t tout)
Request to enable address resolution, and to set resolvable private address timeout.

int32_t rsi_ble_set_privacy_mode(uint8_t remote_dev_addr_type, uint8_t *remote_dev_address, uint8_t privacy_mode)
Request to set privacy mode for particular device.

int32_t rsi_ble_readphy(int8_t *remote_dev_address, rsi_ble_resp_read_phy_t *resp)
Reads the TX and RX PHY rates of the Connection.

int32_t rsi_ble_setphy(int8_t *remote_dev_address, uint8_t tx_phy, uint8_t rx_phy, uint16_t coded_phy)
Set TX and RX PHY.

int32_t rsi_ble_conn_params_update(uint8_t *remote_dev_address, uint16_t min_int, uint16_t max_int, uint16_t latency, uint16_t timeout)
Requests the connection parameters change with the remote device.

int32_t rsi_ble_set_data_len(uint8_t *remote_dev_address, uint16_t tx_octets, uint16_t tx_time)
Sets the TX octets and the TX time of specified link (remote device connection).

int32_t rsi_ble_read_max_data_len(rsi_ble_read_max_data_length_t *blereaddatalen)
reads the max supported values of TX octets, TX time, RX octets and Rx time.

int32_t rsi_ble_accept_list_using_adv_data(uint8_t enable, uint8_t data_compare_index, uint8_t len_for_compare_data, uint8_t *payload)
Give vendor-specific command to set the acceptlist feature based on the advertisers advertising payload.

void BT_LE_ADPacketExtract(uint8_t *remote_name, uint8_t *pbuf, uint8_t buf_len)
Used to extract remote Bluetooth device name from the received advertising report.

int32_t rsi_ble_start_encryption(uint8_t *remote_dev_address, uint16_t ediv, uint8_t *rand, uint8_t *ltk)
Start the encryption process with the remote device.

int32_t rsi_ble_set_ble_tx_power(uint8_t role, uint8_t *remote_dev_address, int8_t tx_power)
Set the TX power value per GAP role.

int32_t rsi_ble_get_max_adv_data_len(uint8_t *resp)
Get maximum advertising data length.

int32_t rsi_ble_get_max_no_of_supp_adv_sets(uint8_t *resp)
Get maximum number of advertising sets.

int32_t rsi_ble_set_ae_set_random_address(uint8_t handle, uint8_t *rand_addr)
Update AE random address.

int32_t rsi_ble_set_ae_data(void *ble_ae_data)
Update AE advertiser data.

int32_t rsi_ble_set_ae_params(void *ble_ae_params, int8_t *sel_tx_pwr)
Update AE parameters.

int32_t rsi_ble_start_ae_advertising(void *adv_enable)
Enable or disable AE advertising.

int32_t rsi_ble_app_adv_set_clear_or_remove(uint8_t type, uint8_t handle)
Clear or remove an advertising set.

int32_t rsi_ble_app_set_periodic_ae_params(void *periodic_adv_params)
Update periodic AE parameters.

int32_t rsi_ble_app_set_periodic_ae_enable(uint8_t enable, uint8_t handle)
Enable or disable periodic advertising.

int32_t rsi_ble_ae_set_scan_params(void *ae_scan_params)
Update AE scan parameters.

int32_t rsi_ble_ae_set_scan_enable(void *ae_scan_enable)
Enable or disable legacy and extended scanning.

int32_t rsi_ble_ae_set_periodic_sync(uint8_t type, void *periodic_sync_data)
Synchronize periodic advertising with advertiser.

int32_t rsi_ble_ae_dev_to_periodic_list(void *dev_to_list)
Manage a device in the periodic advertiser list.

int32_t rsi_ble_ae_read_periodic_adv_list_size(uint8_t *resp)
Get periodic advertiser list size.

int32_t rsi_ble_extended_connect_with_params(void *ext_create_conn)
Establish ACL connection to advertiser.

int32_t rsi_ble_read_transmit_power(void *resp)
Get supported transmit power range.

Typedef Documentation

rsi_ble_set_smp_pairing_capabilty_data_t

```
typedef struct rsi_ble_set_smp_pairing_capabilty_data rsi_ble_set_smp_pairing_capabilty_data_t
```

Definition at line 950 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_read_phy_t

```
typedef struct rsi_ble_resp_read_phy_s rsi_ble_resp_read_phy_t
```

Definition at line 976 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_read_max_data_length_t

```
typedef struct rsi_ble_resp_read_max_data_length_s rsi_ble_read_max_data_length_t
```

Definition at line 1000 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Function Documentation

rsi_ble_set_random_address

```
int32_t rsi_ble_set_random_address (void)
```

Request the local device to set a random address.

Parameters

[in]		

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `1469` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_random_address_with_value

```
int32_t rsi_ble_set_random_address_with_value (uint8_t *random_addr)
```

Request the local device to set a given random address.

Parameters

[in]	random_addr	- random address of the device to be set

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `1488` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_start_advertising

```
int32_t rsi_ble_start_advertising (void)
```

Request the local device to start advertising.

Parameters

[in]		

This is a Blocking API

- A received event rsi_ble_on_enhance_connect_t/ rsi_ble_on_connect_t indicates remote device given ble connect command and got connected
  - Pre-conditions:
- Device should be initialized before calling this API. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command
- 0x4E0C - Command disallowed
- 0x4046 - Invalid Arguments **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `1514` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_start_advertising_with_values

```
int32_t rsi_ble_start_advertising_with_values (void *rsi_ble_adv)
```

Request the local device to start advertising with specified values.

Parameters

[in]	rsi_ble_adv	- This structure pointer holds the information of advertising values

This is a Blocking API

- A received event rsi_ble_on_enhance_connect_t/ rsi_ble_on_connect_t indicates remote device given ble connect command and got connected
  - Pre-conditions:
- Device should be initialized before calling this API.
- This variable is the pointer of the rsi_ble_req_adv_s structure **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command
- 0x4E0C - Command disallowed
- 0x4046 - Invalid Arguments **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `1541` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_encrypt

```
int32_t rsi_ble_encrypt (uint8_t *key, uint8_t *data, uint8_t *resp)
```

Encrypt the plain text data fed by the user using the key provided.

Parameters

[in]	key	- 16 Bytes key for Encryption of data.
[in]	data	- 16 Bytes of Data request to encrypt.
[out]	resp	- Encrypted data

- It uses the AES-128 bit block cypher a logo to generate encrypted data. Refer to Bluetooth Spec 5.0 for further details.
  - Pre-conditions:
- Device should be initialized before calling this API. This is a Blocking API **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `1564` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_stop_advertising

```
int32_t rsi_ble_stop_advertising (void)
```

Stop advertising.

Parameters

[in]		

This is a Blocking API

- Pre-conditions:
  - Call rsi_ble_start_advertising() before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command
  - 0x4E0C - Command disallowed
  - 0x4046 - Invalid Arguments

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `1587` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_advertise_data

```
int32_t rsi_ble_set_advertise_data (uint8_t *data, uint16_t data_len)
```

Set the advertising data.

Parameters

[in]	data	- Advertising data.
[in]	data_len	- Total length of advertising data.

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .
- 1. The maximum length of advertising data payload is 31 bytes.
  - 1. The basic format of advertising payload record contains length and data.
  -

Definition at line `1611` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_scan_response_data

> int32_t rsi_ble_set_scan_response_data (uint8_t *data, uint16_t data_len)

Request the local device to set the scan response data.

Parameters

[in]	data	- Data about to be sent
[in]	data_len	- Length of data, which is about to be sent

This is a Blocking API

- Pre-conditions:
- Device should be initialized before calling this API. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `1632` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_start_scanning

> int32_t rsi_ble_start_scanning (void)

Start scanning.

Parameters

[in]		

This is a Blocking API A received event rsi_ble_on_adv_report_event_t indicates advertise report of remote device received.

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command
  - 0x4E0C - Command disallowed
  - 0x4046 - Invalid Arguments

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 1656 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_start_scanning_with_values

```
int32_t rsi_ble_start_scanning_with_values (void *rsi_ble_scan_params)
```

Start scanning with values.

Parameters

[in]	rsi_ble_scan_params	- BLE scan parameters structure please refer rsi_ble_req_scan_s structure for more info

This is a Blocking API

- A received event rsi_ble_on_adv_report_event_t indicates advertise report of remote device received.
  - Pre-conditions:
- Device should be initialized before calling this API. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- 0x4E0C - Command disallowed
- 0x4046 - Invalid Arguments **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line 1678 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_stop_scanning

```
int32_t rsi_ble_stop_scanning (void)
```

Stop scanning.

Parameters

[in]		

This is a Blocking API

- Pre-conditions:
  - rsi_ble_start_scanning() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command 0x4E0C - Command disallowed

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 1698 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_connect_with_params

> int32_t rsi_ble_connect_with_params (uint8_t remote_dev_addr_type, int8_t *remote_dev_addr, uint16_t scan_interval, uint16_t scan_window, uint16_t conn_interval_max, uint16_t conn_interval_min, uint16_t conn_latency, uint16_t supervision_tout)

Connect to the remote BLE device with the user configured parameters.

Parameters

[in]	remote_dev_addr_type	- Supervision Timeout : N = 0xXXXX
[in]	remote_dev_addr	- LE Scan Interval : N=0xXXXX
N/A	scan_interval	
N/A	scan_window	
N/A	conn_interval_max	
N/A	conn_interval_min	
N/A	conn_latency	
N/A	supervision_tout	

This is a blocking API.

- A received event rsi_ble_on_enhance_connect_t / rsi_ble_on_connect_t indicates that the connection successful and
- a received event rsi_ble_on_disconnect_t indicates that connection failures have occurred. **Note**
  - If a connection can't be established, for example, the remote device has gone out of range, has entered into deep sleep, or is not advertising,
- the stack will try to connect forever. In this case, the application will not get an event related to the connection request.
- To recover from this situation, the application can implement a timeout and call rsi_ble_connect_cancel() to cancel the connection request.
- Subsequent calls of this command have to wait for the ongoing command to complete.
  - Pre-conditions:
- Device should be initialized before calling this API.
- 0 - Public Address
- 1 - Random Address
- It is defined as the time interval from when the Controller started its last LE scan until it
- begins the subsequent LE scan.
- Range: 0x0004 to 0x4000
- Time = N * 0.625 msec
- Time Range: 2.5 msec to 10 . 24 seconds
- Amount of time for the duration of the LE scan. LE_Scan_Window must be less than or equal to LE_Scan_Interval
- Range: 0x0004 to 0x4000
- Time = N * 0.625 msec
- Time Range: 2.5 msec to 10 . 24 seconds
- Minimum value for the connection event interval, which must
- be greater than or equal to Conn_Interval_Min.

Range: 0x0006 to 0x0C80
- Time = N * 1.25 msec
- Time Range: 7.5 msec to 4 seconds.
- 0x0000 - 0x0005 and 0x0C81 - 0xFFFF - Reserved for future use
- Minimum value for the connection event interval, which must be greater than or equal to Conn_Interval_Max.
- Range: 0x0006 to 0x0C80
- Time = N * 1.25 msec
- Time Range: 7.5 msec to 4 seconds.
- 0x0000 - 0x0005 and 0x0C81 - 0xFFFF - Reserved for future use
- Peripheral latency for the connection in number of connection events.
- Range: 0x0000 to 0x01F4
- Supervision timeout for the LE Link.
- Range: 0x000A to 0x0C80
- Time = N * 10 msec
- Time Range: 100 msec to 32 seconds
- 0x0000 - 0x0009 and 0x0C81 - 0xFFFF - Reserved for future use **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- 0x4E0C - Command disallowed
- 0x4046 - Invalid Arguments **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line 1803 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_connect

```
int32_t rsi_ble_connect (uint8_t remote_dev_addr_type, int8_t *remote_dev_addr)
```

Connect to the remote BLE device.

Parameters

[in]	remote_dev_addr_type	- This parameter describes the address type of the remote device
[in]	remote_dev_addr	- This parameter describes the device address of the remote device

This is a blocking API.

- A received event rsi_ble_on_enhance_connect_t/ rsi_ble_on_connect_t indicates that the connection successful and
- a received event rsi_ble_on_disconnect_t indicates that connection failures have occurred. **Note**
  - If a connection can't be established, for example, the remote device has gone out of range, has entered into deep sleep, or is not advertising,
- the stack will try to connect forever. In this case, the application will not get an event related to the connection request.
- To recover from this situation, the application can implement a timeout and call rsi_ble_connect_cancel() to cancel the connection request.
- Subsequent calls of this command have to wait for the ongoing command to complete.
  - Pre-conditions:
- Device should be initialized before calling this API. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- 0x4E0C - Command disallowed
- 0x4046 - Invalid Arguments **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line 1841 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_enhance_connect_with_params

int32_t rsi_ble_enhance_connect_with_params (void *ble_enhance_conn_params)

Connect to the remote BLE device with the user configured parameters.

Parameters

[in]	ble_enhance_conn_params	- BLE enhance connection parameter structure. See notes for the fields in this structure.

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command
  - 0x4E0C - Command disallowed
  - 0x4046 - Invalid Arguments
  -

Note

- Refer Error Codes section for above error codes error-codes.
- The following fields are included in the ble_enhance_conn_params parameter structure:
  - dev_addr_type - Address type of the device to connect.
    - 0 - Public Address
    - 1 - Random Address
  - dev_addr - Address of the device to connect.
  - filter_policy - Policy used to determine whether the filter accept list is used.
    - 0 - Filter accept list is not used to determine which advertiser to connect to.
    - 1 - Filter accept list is used to determine which advertiser to connect to.
  - own_addr_type - Own address type
  - le_scan_interval - The time interval from when the Controller started its last LE scan until it begins the subsequent LE scan.
    - Range: 0x0004 to 0x4000
    - Time = le_scan_interval * 0.625 msec
    - Time Range: 2.5 msec to 10 . 24 seconds
  - le_scan_window - Amount of time for the duration of the LE scan. This must be less than or equal to le_scan_interval.
    - Range: 0x0004 to 0x4000
    - Time = le_scan_window * 0.625 msec
    - Time Range: 2.5 msec to 10 . 24 seconds
  - conn_interval_min - Minimum value for the connection event interval. This must be greater than or equal to conn_interval_max.
    - Range: 0x0006 to 0x0C80
    - Time = conn_interval_min * 1.25 msec
    - Time Range: 7.5 msec to 4 seconds.
    - 0x0000 - 0x0005 and 0x0C81 - 0xFFFF - Reserved for future use
  - conn_interval_max - Maximum value for the connection event interval. This must be greater than or equal to conn_interval_min.
    - Range: 0x0006 to 0x0C80
    - Time = conn_interval_max * 1.25 msec
    - Time Range: 7.5 msec to 4 seconds.
    - 0x0000 - 0x0005 and 0x0C81 - 0xFFFF - Reserved for future use
  - conn_latency - Peripheral latency for the connection in number of connection events.
    - Range: 0x0000 to 0x01F4
  - supervision_tout - Supervision timeout for the LE Link.
    - Range: 0x000A to 0x0C80

Time = N * 10 msec
- Time Range: 100 msec to 32 seconds
- 0x0000 - 0x0009 and 0x0C81 - 0xFFFF - Reserved for future use

- min_ce_length - Minimum length of connection event recommended for this LE connection.
  - Range: 0x0000 to 0xFFFF
  - Time = N * 0.625 msec
- max_ce_length - Maximum length of connection event recommended for this LE connection.
  - Range: 0x0000 to 0xFFFF
  - Time = N * 0.625 msec

Definition at line 1905 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_connect_cancel

```
int32_t rsi_ble_connect_cancel (int8_t *remote_dev_address)
```

Cancel the connection to the remote BLE device.

Parameters

[in]	remote_dev_address	- This parameter describes the device address of the remote device

This is a blocking API.

- A received event rsi_ble_on_disconnect_t indicates disconnect complete.
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- 0x4E0C - Command disallowed
- 0x4046 - Invalid Arguments
- 0x4E02 - Unknown Connection Identifier
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line 1929 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_disconnect

```
int32_t rsi_ble_disconnect (int8_t *remote_dev_address)
```

Disconnect with the remote BLE device.

Parameters

[in]	remote_dev_address	- This parameter describes the device address of the remote device

This is a Blocking API

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E0C - Command disallowed

0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
-

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 1953 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_device_state

```
int32_t rsi_ble_get_device_state (uint8_t *resp)
```

Get the local device state.

Parameters

[out]	resp	- This is an output parameter which consists of local device state. • This is a 1-byte value. The possible states are described below • BIT(0) Advertising state • BIT(1) Scanning state • BIT(2) Initiating state • BIT(3) Connected state •

This is a Blocking API. The state value is filled in "resp".

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 1983 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_smp_pairing_cap_data

```
int32_t rsi_ble_set_smp_pairing_cap_data (rsi_ble_set_smp_pairing_capabilty_data_t *smp_pair_cap_data)
```

Set the SMP Pairing Capability of local device.

Parameters

[in]	smp_pair_cap_data	- This structure pointer holds the information of the SMP capability data values • please refer rsi_ble_set_smp_pairing_capabilty_data structure for more info

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2004` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_local_irk_value

> int32_t rsi_ble_set_local_irk_value (uint8_t *l_irk)

Set the IRK value to the local device.

Parameters

[in]	l_irk	- l_irk Pointer to local_irk

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2019` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_conn_param_resp

> int32_t rsi_ble_conn_param_resp (uint8_t *remote_dev_address, uint8_t status)

Give the response for the remote device connection parameter request.

Parameters

[in]	remote_dev_address	- remote device address
[in]	status	- accept or reject the connection parameters update request

This is a Blocking API

- A received event rsi_ble_on_conn_update_complete_t indicates connection update procedure is successful.
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API.

0 - ACCEPT,
- 1 - REJECT
- **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure 0x4E0C - Command disallowed
- 0x4046 - Invalid Arguments
- 0x4E02 - Unknown Connection Identifier **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `2048` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_smp_pair_request

```
int32_t rsi_ble_smp_pair_request (uint8_t *remote_dev_address, uint8_t io_capability, uint8_t mitm_req)
```

Request the SMP pairing process with the remote device.

Parameters

[in]	remote_dev_address	- MITM enable/disable
[in]	io_capability	- This is the device input output capability
N/A	mitm_req	

This is a Blocking API

- A received event rsi_ble_on_smp_request_t indicated remote device is given Security Request and need to respond back with rsi_ble_smp_pair_request
- A received event rsi_ble_on_smp_response_t indicated remote device is given SMP Pair Request and need to respond back with rsi_ble_smp_pair_response
- A received event rsi_ble_on_smp_failed_t indicated SMP procedure have failed
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API.
- 0x00 - Display Only
- 0x01 - Display Yes/No
- 0x02 - Keyboard Only
- 0x03 - No Input No Output
- 0 - Disable
- 1 - Enable **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command
- 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `2094` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_smp_pair_failed

```
int32_t rsi_ble_smp_pair_failed (uint8_t *remote_dev_address, uint8_t reason)
```

Send SMP pairing failure reason to the remote device.

Parameters

[in]	remote_dev_address	- This is the remote device address
[in]	reason	- This is the reason for SMP Pairing Failure • 0x05 - Pairing Not Supported • 0x08 - Unspecified Reason • 0x09 - Repeated Attempts •

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Definition at line 2120 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_ltk_req_reply

```
int32_t rsi_ble_ltk_req_reply (uint8_t *remote_dev_address, uint8_t reply_type, uint8_t *ltk)
```

Send the local long term key of its associated local EDIV and local Rand.

Parameters

[in]	remote_dev_address	- Long Term Key 16 bytes
[in]	reply_type	- 0 - Negative reply
N/A	ltk	

This is a Blocking API

- A received event rsi_ble_on_encrypt_started_t indicated encrypted event is received from module
- A received event rsi_ble_on_smp_failed_t indicated SMP procedure have failed
- BIT(0) - Positive Reply (Encryption Enabled)
- BIT(1) - Un authenticated LTK or STK-based Encryption Enabled
- BIT(2) - Authenticated LTK or STK-based Encryption Enabled
- BIT(3) - Authenticated LTK with LE Secure Connections based Encryption Enabled
- BIT(4) to BIT(6) - Reserved for Future use
- BIT(7) - LE Secure Connection Enabled
- **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command
- 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line 2164 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_smp_pair_response

```
int32_t rsi_ble_smp_pair_response (uint8_t *remote_dev_address, uint8_t io_capability, uint8_t mitm_req)
```

Send SMP pairing response during the process of pairing with the remote device.

Parameters

[in]	remote_dev_address	- MITM Request info
[in]	io_capability	- This is the device input output capability
N/A	mitm_req	

This is a Blocking API

- A received event rsi_ble_on_smp_passkey_t indicated Legacy SMP passkey is received and need to respond back with rsi_ble_smp_passkey()
- A received event rsi_ble_on_sc_passkey_t indicated BLE SC passkey is received and need to respond back with rsi_ble_smp_passkey()
- A received event rsi_ble_on_smp_passkey_display_t indicates SMP passkey display is received from the module
- A received event rsi_ble_on_smp_failed_t indicated SMP Failed event is received
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API.
- 0x00 - Display Only
- 0x01 - Display Yes/No
- 0x02 - Keyboard Only
- 0x03 - No Input No Output
- 0 - Disable
- 1 - Enable **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command
- 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line 2212 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_smp_passkey

```
int32_t rsi_ble_smp_passkey (uint8_t *remote_dev_address, uint32_t passkey)
```

Send SMP passkey during SMP pairing process with the remote device.

Parameters

[in]	remote_dev_address	- This is the remote device address
[in]	passkey	- This is the key required in pairing process

This is a Blocking API

- A received event rsi_ble_on_encrypt_started_t indicated encrypted event is received from module
- A received event rsi_ble_on_le_security_keys_t indicates exchange of security keys completed after encryption
- A received event rsi_ble_on_smp_failed_t indicated SMP procedure have failed
  - Pre-conditions:

Call rsi_ble_smp_pair_request and rsi_ble_smp_pair_response before calling this API. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command
- 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `2246` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_get_le_ping_timeout

> int32_t rsi_ble_get_le_ping_timeout (uint8_t *remote_dev_address, uint16_t *time_out)

Get the timeout value of the LE ping.

Parameters

[in]	remote_dev_address	- This is the remote device address
[out]	time_out	- This a response parameter which holds timeout value for • authentication payload command.

This is a Blocking API

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command
  - 0x4D05 BLE socket not available
  - 0x4E62 Invalid Parameters
  - 0x4D04 BLE not connected
  -

Note

- Refer Error Codes section for above error codes error-codes .
- Currently Get ping is not supported.

Definition at line `2276` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_le_ping_timeout

> int32_t rsi_ble_set_le_ping_timeout (uint8_t *remote_dev_address, uint16_t time_out)

Set the timeout value of the LE ping.

Parameters

[in]	remote_dev_address	- This is the remote device address
[out]	time_out	- This input parameter sets timeout value for authentication

This is a Blocking API

- A received event of rsi_ble_on_le_ping_payload_timeout_t indicates LE ping payload timeout expired
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API.
- payload command.(in milliseconds) **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command
- 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `2307` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_clear_acceptlist

```
int32_t rsi_ble_clear_acceptlist (void)
```

Clear all the BD address present in accept list.

Parameters

[in]		

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2326` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_addto_acceptlist

```
int32_t rsi_ble_addto_acceptlist (int8_t *dev_address, uint8_t dev_addr_type)
```

Add BD address to accept list.

Parameters

[in]	dev_address	- Address of the device which is going to add in accept list
[in]	dev_addr_type	- address type of BD address

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Maximum number of device address that firmware can store is 10.
  - Refer Error Codes section for above error codes error-codes .

Definition at line `2347` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_deletefrom_acceptlist

```
int32_t rsi_ble_deletefrom_acceptlist (int8_t *dev_address, uint8_t dev_addr_type)
```

Delete particular BD address from accept list.

Parameters

[in]	dev_address	- Address of the device which is going to delete from accept list
[in]	dev_addr_type	- address type of BD address

This is a Blocking API

- Pre-conditions:
  - rsi_ble_addto_acceptlist() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2368` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_resolvlist

```
int32_t rsi_ble_resolvlist (uint8_t process_type, uint8_t remote_dev_addr_type, uint8_t *remote_dev_address, uint8_t *peer_irk, uint8_t *local_irk)
```

resolvlist API used for multiple purpose based on the process type.

Parameters

[in]	process_type	- Indicates which type of process this is, as follows: • 1 - add a device to the resolve list • 2 - remove a device from the resolve list • 3 - clear the entire resolve list
[in]	remote_dev_addr_type	- typr of the remote device address
[in]	remote_dev_address	- remote device address • 0 - Public identity address • 1 - Random (static) identity address •
[in]	peer_irk	- 16-byte IRK of the peer device
[in]	local_irk	- 16-byte IRK of the local device

It will be used to add/remove/clear a device to/from the list. This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 2406 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_resolving_list_size

```
int32_t rsi_ble_get_resolving_list_size (uint8_t *resp)
```

Request to get resolving list size.

Parameters

[out]	resp	- output parameter which consists of supported resolving the list size.

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 : Buffer not available to serve the command

Note

Refer Error Codes section for above error codes error-codes .

Definition at line 2430 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_addr_resolution_enable

> int32_t rsi_ble_set_addr_resolution_enable (uint8_t enable, uint16_t tout)

Request to enable address resolution, and to set resolvable private address timeout.

Parameters

[in]	enable	- value to enable/disable address resolution • 1 - enables address resolution • 0 - disables address resolution
[in]	tout	- the period for changing address of our local device in seconds • Value ranges from 0x0001 to 0xA1B8 (1 s to approximately 11.5 hours)

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 : Buffer not available to serve the command

Definition at line 2455 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_privacy_mode

> int32_t rsi_ble_set_privacy_mode (uint8_t remote_dev_addr_type, uint8_t *remote_dev_address, uint8_t privacy_mode)

Request to set privacy mode for particular device.

Parameters

[in]	remote_dev_addr_type	- type of the remote device address • 0 - Public Identity Address • 1 - Random (static) Identity Address
[in]	remote_dev_address	- remote device address
[in]	privacy_mode	- type of the privacy mode • 0 - Network privacy mode • 1 - Device privacy mode

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 : Buffer not available to serve the command

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 2485 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_readphy

> int32_t rsi_ble_readphy (int8_t *remote_dev_address, rsi_ble_resp_read_phy_t *resp)

Reads the TX and RX PHY rates of the Connection.

Parameters

[in]	remote_dev_address	- remote device address
[out]	resp	- pointer to store the response please refer rsi_ble_resp_read_phy_s structure for more info.

This is a Blocking API

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 2502 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_setphy

> int32_t rsi_ble_setphy (int8_t *remote_dev_address, uint8_t tx_phy, uint8_t rx_phy, uint16_t coded_phy)

Set TX and RX PHY.

Parameters

[in]	remote_dev_address	- TX/RX coded PHY rate
N/A	tx_phy	
N/A	rx_phy	
N/A	coded_phy	

This is a Blocking API

- A received event rsi_ble_on_phy_update_complete_t indicates PHY rate update complete.
  - Pre-conditions:

rsi_ble_connect() API needs to be called before this API.

-
- BIT(0) - Host prefers to use the LE 1M transmitter PHY (possibly among others)
- BIT(1) - Host prefers to use the LE 2M transmitter PHY (possibly among others)
- BIT(2) - Host prefers to use the LE Coded transmitter PHY (possibly among others)
- BIT(3) - BIT(7) Reserved for future use
-
- BIT(0) - Host prefers to use the LE 1M receiver PHY (possibly among others)
- BIT(1) - Host prefers to use the LE 2M receiver PHY (possibly among others)
- BIT(2) - Host prefers to use the LE Coded receiver PHY (possibly among others)
- BIT(3) - BIT(7) Reserved for future use
-
- 0 = Host has no preferred coding when transmitting on the LE Coded PHY
- 1 = Host prefers that S=2 coding be used when transmitting on the LE Coded PHY
- 2 = Host prefers that S=8 coding be used when transmitting on the LE Coded PHY
- 3 = Reserved for future use
- **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line 2558 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_conn_params_update

> int32_t rsi_ble_conn_params_update (uint8_t *remote_dev_address, uint16_t min_int, uint16_t max_int, uint16_t latency, uint16_t timeout)

Requests the connection parameters change with the remote device.

Parameters

[in]	remote_dev_address	- supervision timeout for the LE Link.
[in]	min_int	- minimum value for the connection interval.
N/A	max_int	
N/A	latency	
N/A	timeout	

- When the Silicon Labs device acts as a central, this API is used to update the connection parameters.
- When the Silicon Labs device acts as a peripheral, this API is used to request the central to initiate the connection update procedure. This is a Blocking API
- A received event rsi_ble_on_conn_update_complete_t indicates connection parameters update complete.
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API.
- this shall be less than or equal to max_int .
- this shall be greater than or equal to min_int.
- Ranges from 0 to 499
- Ranges from 10 to 3200 (Time = N * 10 ms, Time Range: 100 ms to 32 s) **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters

0x4D04 BLE not connected

- **Note**
  - Refer Error Codes section for above error codes error-codes .
  - min_int and max_int values ranges from 6 to 3200 (Time = N * 1.25 ms, Time Range: 7.5 ms to 4 s)
- latency : If latency value is greater than 32 ,Limiting the peripheral latency value to 32
- Max supported peripheral latency is 32 when Device is in peripheral Role.

Definition at line `2608` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_data_len

> int32_t rsi_ble_set_data_len (uint8_t *remote_dev_address, uint16_t tx_octets, uint16_t tx_time)

Sets the TX octets and the TX time of specified link (remote device connection).

Parameters

[in]	remote_dev_address	- preferred maximum number of microseconds that the local Controller
N/A	tx_octets	
N/A	tx_time	

This is a Blocking API.

- A received event rsi_ble_on_data_length_update_t indicates data length update complete.
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API.
-
- should include in a single Link Layer packet on this connection.
- should use to transmit a single Link Layer packet on this connection. **Returns**
  - The following values are returned:
- 0 - LE_Set_Data_Length command succeeded.
- Non-Zero Value - Failure
- 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line `2644` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_read_max_data_len

> int32_t rsi_ble_read_max_data_len (rsi_ble_read_max_data_length_t *blereaddatalen)

reads the max supported values of TX octets, TX time, RX octets and Rx time.

Parameters

[out]	blereaddatalen	- pointer to structure variable, Please refer rsi_ble_resp_read_max_data_length_s structure for more info.

This is a Blocking API

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:

0 - command success
- Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 2660 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_accept_list_using_adv_data

> int32_t rsi_ble_accept_list_using_adv_data (uint8_t enable, uint8_t data_compare_index, uint8_t len_for_compare_data, uint8_t *payload)

Give vendor-specific command to set the acceptlist feature based on the advertisers advertising payload.

Parameters

[in]	enable	- enable/disable
[in]	data_compare_index	- the starting index of the data to compare
[in]	len_for_compare_data	- total length of data to compare
[in]	payload	- Payload

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  - -4 - Buffer not available to serve the command 0x4E62 Invalid Parameters
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 2817 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

BT_LE_ADPacketExtract

> void BT_LE_ADPacketExtract (uint8_t *remote_name, uint8_t *pbuf, uint8_t buf_len)

Used to extract remote Bluetooth device name from the received advertising report.

Parameters

[in]	remote_name	- device name
[in]	pbuf	- advertise data packet buffer pointer
[in]	buf_len	- buffer length

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - void

Definition at line 2834 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_start_encryption

> int32_t rsi_ble_start_encryption (uint8_t *remote_dev_address, uint16_t ediv, uint8_t *rand, uint8_t *ltk)

Start the encryption process with the remote device.

Parameters

[in]	remote_dev_address	- Remote BD address in string format
[in]	ediv	- remote device ediv value.
[in]	rand	- remote device rand value.
[in]	ltk	- remote device ltk value.

This is a Blocking API

- A received event rsi_ble_on_encrypt_started_t indicated encrypted event is received from module
- A received event rsi_ble_on_le_security_keys_t indicates exchange of security keys completed after encryption.
- A received event rsi_ble_on_smp_failed_t indicated SMP procedure have failed
  - Pre-conditions:
- Encryption enabled event should come before calling this API for second time SMP connection. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- If the return value is less than 0
- -4 - Buffer not available to serve the command 0x4D05 BLE socket not available
- 0x4E62 Invalid Parameters
- 0x4D04 BLE not connected
- **Note**
  - Refer Error Codes section for above error codes error-codes .

Definition at line 2869 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_ble_tx_power

> int32_t rsi_ble_set_ble_tx_power (uint8_t role, uint8_t *remote_dev_address, int8_t tx_power)

Set the TX power value per GAP role.

Parameters

[in]	role	ADV_ROLE 0x01 • SCAN_AND_CENTRAL_ROLE 0x02 • PERIPHERAL_ROLE 0x03 • CONN_ROLE 0x04 •
[in]	remote_dev_address	- Remote device address
[in]	tx_power	- power value

This is a Blocking API **Note**

This API is not supported in the current release.

Note

- remote_dev_address is valid only on role=CONN_ROLE
  - #define RSI_BLE_PWR_INX_DBM 0 indicate tx_power in index
  - Default Value for BLE TX Power Index is 31, The range for the BLE TX Power Index is 1 to 75 (0, 32 indexes are invalid)
  - 1 - 31 BLE - 0DBM Mode.
  - 33 - 63 BLE - 10DBM Mode.
  - 64 - 75 BLE - HP Mode.
  - Currently this API is supports only BLE LP mode . i.e. 1 to 63 BLE LP MODE
  - #define RSI_BLE_PWR_INX_DBM 1 indicate tx_power in dBm
  - tx_power in dBm (-8dBm to 15 dBm)
  - Currently this API is supports only BLE LP mode . i.e. -8 dBm to 4dBm BLE LP MODE
  -

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E02 Unknown Connection Identifier
  - 0x4E01 Unknown HCI Command
  - 0x4E0C Command disallowed
  - 0x4046 Invalid Arguments
  - 0x4D04 BLE not connected
  - 0x4D14 BLE parameter out of mandatory range

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2927` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_get_max_adv_data_len

```
int32_t rsi_ble_get_max_adv_data_len (uint8_t *resp)
```

Get maximum advertising data length.

Parameters

[out]	resp	Maximum supported advertising data length returned by the controller. Possible values range from 0x001F to 0x0672.

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Note

- This function requests the controller to return the maximum supported advertising data length.

Definition at line `3864` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_get_max_no_of_supp_adv_sets

```
int32_t rsi_ble_get_max_no_of_supp_adv_sets (uint8_t *resp)
```

Get maximum number of advertising sets.

Parameters

[out]	resp	Number of supported advertising sets returned by the controller. Possible values range from 0x01 to 0xF0.

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Note

- This function requests the controller to return the maximum number of supporting advertising sets.
- The number of supported advertising sets can be configured through the operating modes.

Definition at line `3880` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_ae_set_random_address

> rsi_ble_set_ae_set_random_address (uint8_t handle, uint8_t *rand_addr)

Update AE random address.

Parameters

[in]	handle	The advertising handle used to identify an advertising set
[in]	rand_addr	Random device address set to either a static or private address

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Definition at line `3893` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_ae_data

> int32_t rsi_ble_set_ae_data (void *ble_ae_data)

Update AE advertiser data.

Parameters

[in]	ble_ae_data	Extended Advertising data to be updated

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Note

- This function sets the AE advertiser data used in advertising PDUs.

Refer to Bluetooth specification 5.3 for possible combinations ae_adv/scanresp data can be set for .

Definition at line 3908 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_ae_params

```
int32_t rsi_ble_set_ae_params (void *ble_ae_params, int8_t *sel_tx_pwr)
```

Update AE parameters.

Parameters

[in]	ble_ae_params	Extended Advertising parameters to be updated
[out]	sel_tx_pwr	Output transmit power in dBm, ranging from -127 to +20.

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Definition at line 3921 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_start_ae_advertising

```
int32_t rsi_ble_start_ae_advertising (void *adv_enable)
```

Enable or disable AE advertising.

Parameters

[in]	adv_enable	Parameters to enable or disable specific advertising sets identified by advertising handle

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Definition at line 3933 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_app_adv_set_clear_or_remove

```
int32_t rsi_ble_app_adv_set_clear_or_remove (uint8_t type, uint8_t handle)
```

Clear or remove an advertising set.

Parameters

[in]	type	Set to 1 to clear, or 2 to remove an advertising set
[in]	handle	Advertising handle identifying the advertising set to remove or clear. Possible values range from 0x00 to 0xEF.

Returns

The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Definition at line 3946 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_app_set_periodic_ae_params

```
int32_t rsi_ble_app_set_periodic_ae_params (void *periodic_adv_params)
```

Update periodic AE parameters.

Parameters

[in]	periodic_adv_params	Periodic advertising parameters to be updated

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Definition at line 3958 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_app_set_periodic_ae_enable

```
int32_t rsi_ble_app_set_periodic_ae_enable (uint8_t enable, uint8_t handle)
```

Enable or disable periodic advertising.

Parameters

[in]	enable	Set to 0 to enable, or 1 to include the ADI field in AUX_SYNC_IND PDUs
[in]	handle	Advertising handle of the advertising set to enable or disable

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Note

- This function requests the controller to enable or disable periodic advertising for the specified advertising set.

Definition at line 3973 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_ae_set_scan_params

```
int32_t rsi_ble_ae_set_scan_params (void *ae_scan_params)
```

Update AE scan parameters.

Parameters

[in]	ae_scan_params	Extended scan parameters to be updated

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Note

- This function sets the extended scan parameters to be used on the physical advertising channels.

Definition at line `3987` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_ae_set_scan_enable

```
int32_t rsi_ble_ae_set_scan_enable (void *ae_scan_enable)
```

Enable or disable legacy and extended scanning.

Parameters

[in]	ae_scan_enable	Parameters specify whether to enable or disable both legacy and extended advertising PDUs

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Definition at line `3999` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_ae_set_periodic_sync

```
int32_t rsi_ble_ae_set_periodic_sync (uint8_t type, void *periodic_sync_data)
```

Synchronize periodic advertising with advertiser.

Parameters

[in]	type	Set to 1 to begin, 2 to cancel, or 3 to terminate the periodic advertising sync
[in]	periodic_sync_data	Parameters for starting a perodic advertising sync operation

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - 0x4E42 = unknown advertising identifier
- The following values are returned:
  - 0x4E0C = command not permitted

Note

- This function performs an operation to synchronize with a periodic advertising train from an advertiser and begin receiving periodic advertising packets.
- The operation is either started, cancelled or terminated depending on the type parameter.

Definition at line 4018 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_ae_dev_to_periodic_list

```
int32_t rsi_ble_ae_dev_to_periodic_list (void *dev_to_list)
```

Manage a device in the periodic advertiser list.

Parameters

[in]	dev_to_list	Details of a device to be added to the periodic advertiser list

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Note

- This function adds, removes, or clears a device from the periodic advertiser list.

Definition at line 4032 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_ae_read_periodic_adv_list_size

```
int32_t rsi_ble_ae_read_periodic_adv_list_size (uint8_t *resp)
```

Get periodic advertiser list size.

Parameters

[out]	resp	Periodic advertiser list size returned by the controller

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Definition at line 4044 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_extended_connect_with_params

```
int32_t rsi_ble_extended_connect_with_params (void *ext_create_conn)
```

Establish ACL connection to advertiser.

Parameters

[in]	ble_extended_conn_params	Connection parameters

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:

!0 = failure

Note

- This function establishes an ACL connection to an advertiser, with the local device in the BLE central role.

Definition at line 4058 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_read_transmit_power

```
int32_t rsi_ble_read_transmit_power (void *resp)
```

Get supported transmit power range.

Parameters

[out]	resp	Minimum and maximum supported transmit power, returned by the controller. Power ranges from -127 dBm to +20 dBm.

Returns

- The following values are returned:
  - 0 = success
- The following values are returned:
  - !0 = failure

Note

- This function requests the controller to return the minimum and maximum supported transmit power.

Definition at line 4072 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_smp_pairing_capabilty_data

Public Attributes

uint8_t io_capability
Device input output capability

- 0x00 - Display Only
- 0x01 - Display Yes/No
- 0x02 - Keyboard Only
- 0x03 - No Input No Output 0x04 - Keyboard Display.

uint8_t oob_data_flag
oob_data_flag

- 0 - disable

uint8_t auth_req
Authentication Request contains bonding type

- MITM Request - BIT(2)
- Secure Connections - BIT(3)
- Keypress - BIT(4)
- CT2 - BIT(5)

uint8_t enc_key_size
Supported Encryption key size 7 to 16 bytes.

uint8_t ini_key_distribution
Initiator generates/requires the no .of keys after successful paring

- BIT(0) - EncKey: Initiator distributes the LTK followed by EDIV and Rand
- BIT(1) - IdKey : Initiator distributes the IRK followed by its address
- BIT(2) - Sign : Initiator distributes the CSRK
- BIT(3) - BIT(7): Reserved for future use.

uint8_t rsp_key_distribution
Responder generates/requires the no .of keys after successful paring

- BIT(0) - EncKey: Responder distributes the LTK followed by EDIV and Rand
- BIT(1) - IdKey : Responder distributes the IRK followed by its address
- BIT(2) - Sign : Responder distributes the CSRK
- BIT(3) - BIT(7): Reserved for future use.

Public Attribute Documentation

io_capability

> uint8_t rsi_ble_set_smp_pairing_capabilty_data::io_capability

Device input output capability

- 0x00 - Display Only
- 0x01 - Display Yes/No
- 0x02 - Keyboard Only
- 0x03 - No Input No Output 0x04 - Keyboard Display.

Definition at line 911 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

oob_data_flag

> uint8_t rsi_ble_set_smp_pairing_capabilty_data::oob_data_flag

oob_data_flag

- 0 - disable
- 1 - enable

Definition at line 916 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

auth_req

> uint8_t rsi_ble_set_smp_pairing_capabilty_data::auth_req

Authentication Request contains bonding type

- MITM Request - BIT(2)
- Secure Connections - BIT(3)
- Keypress - BIT(4)
- CT2 - BIT(5)

Definition at line 926 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

enc_key_size

> uint8_t rsi_ble_set_smp_pairing_capabilty_data::enc_key_size

Supported Encryption key size 7 to 16 bytes.

Definition at line 928 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

ini_key_distribution

> uint8_t rsi_ble_set_smp_pairing_capabilty_data::ini_key_distribution

Initiator generates/requires the no .of keys after successful paring

- BIT(0) - EncKey: Initiator distributes the LTK followed by EDIV and Rand
- BIT(1) - IdKey : Initiator distributes the IRK followed by its address
- BIT(2) - Sign : Initiator distributes the CSRK
- BIT(3) - BIT(7): Reserved for future use.

Definition at line 938 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsp_key_distribution

> uint8_t rsi_ble_set_smp_pairing_capabilty_data::rsp_key_distribution

Responder generates/requires the no .of keys after successful paring

- BIT(0) - EncKey: Responder distributes the LTK followed by EDIV and Rand
- BIT(1) - IdKey : Responder distributes the IRK followed by its address
- BIT(2) - Sign : Responder distributes the CSRK
- BIT(3) - BIT(7): Reserved for future use.

Definition at line `948` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_resp_read_phy_s

Public Attributes

uint8_t dev_addr
Remote device Bluetooth Address.

uint8_t tx_phy
TX PHY Rate

- 0x01 The transmitter PHY for the connection is LE 1M
- 0x02 The transmitter PHY for the connection is LE 2M
- 0x03 The transmitter PHY for the connection is LE Coded
- All other values Reserved for future use.

uint8_t rx_phy
RX PHY Rate

- 0x01 The receiver PHY for the connection is LE 1M
- 0x02 The receiver PHY for the connection is LE 2M
- 0x03 The receiver PHY for the connection is LE Coded
- All other values Reserved for future use.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_resp_read_phy_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Remote device Bluetooth Address.

Definition at line `955` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

tx_phy

```
uint8_t rsi_ble_resp_read_phy_s::tx_phy
```

TX PHY Rate

- 0x01 The transmitter PHY for the connection is LE 1M
- 0x02 The transmitter PHY for the connection is LE 2M
- 0x03 The transmitter PHY for the connection is LE Coded
- All other values Reserved for future use.

Definition at line `965` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rx_phy

```
uint8_t rsi_ble_resp_read_phy_s::rx_phy
```

RX PHY Rate

- 0x01 The receiver PHY for the connection is LE 1M
- 0x02 The receiver PHY for the connection is LE 2M
- 0x03 The receiver PHY for the connection is LE Coded
- All other values Reserved for future use.

Definition at line `975` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_resp_read_max_data_length_s

Public Attributes

uint16_t	maxtxoctets maxtxoctets
uint16_t	maxtxtime maxtxtime
uint16_t	maxrxoctets maxrxoctets
uint16_t	maxrxtime maxrxtime

Public Attribute Documentation

maxtxoctets

```
uint16_t rsi_ble_resp_read_max_data_length_s::maxtxoctets
```

maxtxoctets

- Preferred maximum number of payload octets that the local Controller should include in a single Link Layer packet on this connection.

Definition at line `984` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

maxtxtime

```
uint16_t rsi_ble_resp_read_max_data_length_s::maxtxtime
```

maxtxtime

- Preferred maximum number of microseconds that the local Controller should use to transmit a single Link Layer packet on this connection

Definition at line `989` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

maxrxoctets

```
uint16_t rsi_ble_resp_read_max_data_length_s::maxrxoctets
```

maxrxoctets

- Maximum number of payload octets that the local Controller supports for reception of a single Link Layer packet on a data connection

Definition at line 994 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

maxrxtime

> uint16_t rsi_ble_resp_read_max_data_length_s::maxrxtime

maxrxtime

- Maximum time, in microseconds, that the local Controller supports for reception of a single Link Layer packet on a data connection.

Definition at line 999 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

GATT

GATT

Generic Attribute Profile (GATT) functions provide access to the BLE GATT protocol.

The response payload for all the aysnchronous APIs will be indicated to the application through the corresponding callback functions as described in the following sections. The response payload structure is described in the **Callback Declarations** section. This section provides a reference to GATT functions:

- Synchronous Client functions to implement a synchronous Generic Attribute Profile (GATT) client.
- Asynchronous Client functions to implement an asynchronous Generic Attribute Profile (GATT) client.
- Server functions to implement a Generic Attribute Profile (GATT) server.

Modules

Synchronous Client

Asynchronous Client

Server

Synchronous Client

Functions

int32_t rsi_ble_get_profiles(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_profiles_list_t *p_prof_list)
Get the supported profiles / services of the connected remote device.

int32_t rsi_ble_get_profile(uint8_t *dev_addr, uuid_t profile_uuid, profile_descriptors_t *p_profile)
Get the specific profile / service of the connected remote device.

int32_t rsi_ble_get_char_services(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_char_services_t *p_char_serv_list)
Get the service characteristic services of the connected / remote device.

int32_t rsi_ble_get_inc_services(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_inc_services_t *p_inc_serv_list)
Get the supported include services of the connected / remote device.

int32_t rsi_ble_get_char_value_by_uuid(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, uuid_t char_uuid, rsi_ble_resp_att_value_t *p_char_val)
Get the characteristic value by UUID (char_uuid).

int32_t rsi_ble_get_att_descriptors(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_att_descs_t *p_att_desc)
Get the characteristic descriptors list from the remote device.

int32_t rsi_ble_get_att_value(uint8_t *dev_addr, uint16_t handle, rsi_ble_resp_att_value_t *p_att_val)
Get the attribute by handle.

int32_t rsi_ble_get_multiple_att_values(uint8_t *dev_addr, uint8_t num_of_handlers, uint16_t *handles, rsi_ble_resp_att_value_t *p_att_vals)

int32_t rsi_ble_get_long_att_value(uint8_t *dev_addr, uint16_t handle, uint16_t offset, rsi_ble_resp_att_value_t *p_att_vals)
Get the long attribute value by using handle and offset.

int32_t rsi_ble_set_att_value(uint8_t *dev_addr, uint16_t handle, uint8_t data_len, uint8_t *p_data)
Set the attribute value of the remote device.

int32_t rsi_ble_set_att_cmd(uint8_t *dev_addr, uint16_t handle, uint8_t data_len, uint8_t *p_data)
Set the attribute value without waiting for an ACK from the remote device.

int32_t rsi_ble_set_long_att_value(uint8_t *dev_addr, uint16_t handle, uint16_t offset, uint8_t data_len, uint8_t *p_data)
Set the long attribute value of the remote device.

int32_t rsi_ble_prepare_write(uint8_t *dev_addr, uint16_t handle, uint16_t offset, uint8_t data_len, uint8_t *p_data)
Prepare the attribute value.

int32_t rsi_ble_execute_write(uint8_t *dev_addr, uint8_t exe_flag)
Execute the prepared attribute values.

int32_t rsi_ble_mtu_exchange_event(uint8_t *dev_addr, uint8_t mtu_size)
Initiates the MTU exchange request with the remote device.

int32_t rsi_ble_mtu_exchange_resp(uint8_t *dev_addr, uint8_t mtu_size)
This function (Exchange MTU Response) is sent in reply to a received Exchange MTU Request.

Function Documentation

rsi_ble_get_profiles

> nt32_t rsi_ble_get_profiles (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_profiles_list_t *p_prof_list)

Get the supported profiles / services of the connected remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[out]	p_prof_list	- profiles/services information will be filled in this structure after retrieving from the remote device, please refer rsi_ble_resp_profiles_list_s structure for more info.

rsi_ble_on_profiles_list_resp_t callback function will be called after the profiles list response is received. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_profiles_list_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 2965 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_profile

> int32_t rsi_ble_get_profile (uint8_t *dev_addr, uuid_t profile_uuid, profile_descriptors_t *p_profile)

Get the specific profile / service of the connected remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	profile_uuid	- services/profiles which are searched using profile_uuid

[out]	p_profile	- profile / service information filled in this structure after retrieving from the remote device, please refer profile_descriptor_s structure for more info. •

rsi_ble_on_profile_resp_t callback function is called after the service characteristics response is received. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_profile_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 2995 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_char_services

> int32_t rsi_ble_get_char_services (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_char_services_t *p_char_serv_list)

Get the service characteristic services of the connected / remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[out]	p_char_serv_list	- service characteristics details are filled in this structure, please refer rsi_ble_resp_char_serv_s structure for more info. •

rsi_ble_on_char_services_resp_t callback function is called after the characteristic service response is received. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_char_services_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3023 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_inc_services

> int32_t rsi_ble_get_inc_services (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_inc_services_t *p_inc_serv_list)

Get the supported include services of the connected / remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[out]	p_inc_serv_list	- include service characteristics details are filled in this structure, please refer rsi_ble_resp_inc_serv structure for more info.

rsi_ble_on_inc_services_resp_t callback function is called after the include service response is received. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_inc_services_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3054 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_char_value_by_uuid

> int32_t rsi_ble_get_char_value_by_uuid (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, uuid_t char_uuid, rsi_ble_resp_att_value_t *p_char_val)

Get the characteristic value by UUID (char_uuid).

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[in]	char_uuid	- UUID of the characteristic
[out]	p_char_val	- characteristic value is filled in this structure, please refer rsi_ble_resp_att_value_s structure for more info.

rsi_ble_on_read_resp_t callback function is called after the attribute value is received. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_read_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - If the return value is less than 0
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3088 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_att_descriptors

> int32_t rsi_ble_get_att_descriptors (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_att_descs_t *p_att_desc)

Get the characteristic descriptors list from the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[out]	p_att_desc	- pointer to characteristic descriptor structure, Please refer rsi_ble_resp_att_descs_s structure for more info.

rsi_ble_on_att_desc_resp_t callback function is called after the attribute descriptors response is received. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_att_desc_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3119 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_att_value

> int32_t rsi_ble_get_att_value (uint8_t *dev_addr, uint16_t handle, rsi_ble_resp_att_value_t *p_att_val)

Get the attribute by handle.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- handle value of the attribute
[out]	p_att_val	- attribute value is filled in this structure, Please refer rsi_ble_resp_att_value_s structure for more info. •

rsi_ble_on_read_resp_t callback function is called upon receiving the attribute value. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_read_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3145 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_multiple_att_values

```
int32_t rsi_ble_get_multiple_att_values (uint8_t *dev_addr, uint8_t num_of_handlers, uint16_t *handles,
rsi_ble_resp_att_value_t *p_att_vals)
```

Parameters

N/A	dev_addr	
N/A	num_of_handlers	
N/A	handles	
N/A	p_att_vals	

Definition at line 3171 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_long_att_value

```
int32_t rsi_ble_get_long_att_value (uint8_t *dev_addr, uint16_t handle, uint16_t offset, rsi_ble_resp_att_value_t *p_att_vals)
```

Get the long attribute value by using handle and offset.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute handle
[in]	offset	- offset within the attribute value
[out]	p_att_vals	- attribute value filled in this structure, please refer rsi_ble_resp_att_value_s structure for more info.

rsi_ble_on_read_resp_t callback function is called after the attribute value is received. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_read_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3200 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_att_value

```
int32_t rsi_ble_set_att_value (uint8_t *dev_addr, uint16_t handle, uint8_t data_len, uint8_t *p_data)
```

Set the attribute value of the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute value handle
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

rsi_ble_on_write_resp_t callback function is called if the attribute set action is completed. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_write_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3227 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_att_cmd

```
int32_t rsi_ble_set_att_cmd (uint8_t *dev_addr, uint16_t handle, uint8_t data_len, uint8_t *p_data)
```

Set the attribute value without waiting for an ACK from the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute value handle
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

This is a Blocking API. If the API returns RSI_ERROR_BLE_DEV_BUF_FULL (-31) error then wait until the rsi_ble_on_le_more_data_req_t event gets received from the module.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  - 0x4E65 - Invalid Attribute Length When Small Buffer Mode is Configured
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3259 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_long_att_value

> int32_t rsi_ble_set_long_att_value (uint8_t *dev_addr, uint16_t handle, uint16_t offset, uint8_t data_len, uint8_t *p_data)

Set the long attribute value of the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute handle
[in]	offset	- attribute value offset
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

rsi_ble_on_write_resp_t callback function is called after the attribute set action is completed. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_write_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3287 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_prepare_write

> int32_t rsi_ble_prepare_write (uint8_t *dev_addr, uint16_t handle, uint16_t offset, uint8_t data_len, uint8_t *p_data)

Prepare the attribute value.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute handle
[in]	offset	- attribute value offset
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

rsi_ble_on_write_resp_t callback function is called after the prepare attribute write action is completed. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_write_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 3316 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_execute_write

> int32_t rsi_ble_execute_write (uint8_t *dev_addr, uint8_t exe_flag)

Execute the prepared attribute values.

Parameters

[in]	dev_addr	- remote device address
[in]	exe_flag	- execute flag to write, possible values mentioned below • 0 - BLE_ATT_EXECUTE_WRITE_CANCEL • 1 - BLE_ATT_EXECUTE_PENDING_WRITES_IMMEDIATELY •

rsi_ble_on_write_resp_t callback function is called after the execute attribute write action is completed. This is a non-blocking API, Still user need to wait until the callback rsi_ble_on_write_resp_t is received from the device, to initiate further attribute related transactions on this remote device address.

Pre-conditions:
- rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `3342` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_mtu_exchange_event

```
int32_t rsi_ble_mtu_exchange_event (uint8_t *dev_addr, uint8_t mtu_size)
```

Initiates the MTU exchange request with the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	mtu_size	- requested MTU value

- This is a Blocking API and will receive a callback event rsi_ble_on_mtu_event_t as the response for this API.
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure 0x4D04 - BLE not Connected
- 0x4E62 - Invalid Parameters
- **Note**
  - Refer Error Codes section for above error codes error-codes
-

Definition at line `3766` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_mtu_exchange_resp

```
int32_t rsi_ble_mtu_exchange_resp (uint8_t *dev_addr, uint8_t mtu_size)
```

This function (Exchange MTU Response) is sent in reply to a received Exchange MTU Request.

Parameters

[in]	dev_addr	- Remote Device Address
[in]	mtu_size	- requested MTU value

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  -

0x4D0C - When RSI_BLE_MTU_EXCHANGE_FROM_HOST BIT is not SET. 0x4D05 - BLE Socket Not Available. Non-Zero Value - Failure Refer Error Codes section for above error codes error-codes

Definition at line `3785` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

Asynchronous Client

Asynchronous Client

Functions

int32_t rsi_ble_indicate_value_sync(uint8_t *dev_addr, uint16_t handle, uint16_t data_len, uint8_t *p_data)
Indicate the local value to the remote device.

int32_t rsi_ble_indicate_confirm(uint8_t *dev_addr)
Send indicate confirmation to the remote device.

int32_t rsi_ble_get_profiles_async(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_profiles_list_t *p_prof_list)
Get the supported profiles / services of the connected remote device asynchronously.

int32_t rsi_ble_get_profile_async(uint8_t *dev_addr, uuid_t profile_uuid, profile_descriptors_t *p_profile)
Get the specific profile / service of the connected remote device.

int32_t rsi_ble_get_char_services_async(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_char_services_t *p_char_serv_list)
Get the service characteristics of the connected / remote device.

int32_t rsi_ble_get_inc_services_async(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_inc_services_t *p_inc_serv_list)
Get the supported include services of the connected / remote device.

int32_t rsi_ble_get_char_value_by_uuid_async(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, uuid_t char_uuid, rsi_ble_resp_att_value_t *p_char_val)
Get the characteristic value by UUID (char_uuid).

int32_t rsi_ble_get_att_descriptors_async(uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_att_descs_t *p_att_desc)
Get the characteristic descriptors list from the remote device.

int32_t rsi_ble_get_att_value_async(uint8_t *dev_addr, uint16_t handle, rsi_ble_resp_att_value_t *p_att_val)
Get the attribute with a handle.

int32_t rsi_ble_get_multiple_att_values_async(uint8_t *dev_addr, uint8_t num_of_handlers, uint16_t *handles, rsi_ble_resp_att_value_t *p_att_vals)
Get the multiple attribute values by using multiple handles.

int32_t rsi_ble_get_long_att_value_async(uint8_t *dev_addr, uint16_t handle, uint16_t offset, rsi_ble_resp_att_value_t *p_att_vals)
Get the long attribute value by using handle and offset.

int32_t rsi_ble_set_att_value_async(uint8_t *dev_addr, uint16_t handle, uint8_t data_len, uint8_t *p_data)
Set the attribute value of the remote device.

int32_t rsi_ble_prepare_write_async(uint8_t *dev_addr, uint16_t handle, uint16_t offset, uint8_t data_len, uint8_t *p_data)
Prepare the attribute value.

int32_t rsi_ble_execute_write_async(uint8_t *dev_addr, uint8_t exe_flag)
Execute the prepared attribute values.

Function Documentation

rsi_ble_indicate_value_sync

> int32_t rsi_ble_indicate_value_sync (uint8_t *dev_addr, uint16_t handle, uint16_t data_len, uint8_t *p_data)

Indicate the local value to the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- local attribute handle
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

This is a blocking API.

- This will not send any confirmation event to the application instead
- send the status as success on receiving confirmation from remote side.
  - Pre-conditions:
- rsi_ble_connect() API needs to be called before this API. **Returns**
  - The following values are returned:
- 0 - Success
- Non-Zero Value - Failure
- 0x4D05 - BLE socket not available
- 0x4E60 - Invalid Handle Range
- **Note**
  - Refer Error Codes section for above error codes error-codes
-

Definition at line `3583` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_indicate_confirm

> int32_t rsi_ble_indicate_confirm (uint8_t *dev_addr)

Send indicate confirmation to the remote device.

Parameters

[in]	dev_addr	- remote device address

This is a blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure 0x4D05 - BLE socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes
  -

Definition at line 3602 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_profiles_async

> int32_t rsi_ble_get_profiles_async (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_profiles_list_t *p_prof_list)

Get the supported profiles / services of the connected remote device asynchronously.

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[out]	p_prof_list	- profiles/services information will be filled in this structure after retrieving from the remote device, please refer rsi_ble_resp_profiles_list_s structure for more info.

rsi_ble_on_event_profiles_list_t callback function will be called after the profiles list event is received. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_profiles_list_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_prof_list structure should be passed as NULL because nothing will be filled in this structure

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4121 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_profile_async

> int32_t rsi_ble_get_profile_async (uint8_t *dev_addr, uuid_t profile_uuid, profile_descriptors_t *p_profile)

Get the specific profile / service of the connected remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	profile_uuid	- services/profiles which are searched using profile_uuid

[out]	p_profile	- profile / service information filled in this structure after retrieving from the remote device, please refer profile_descriptor_s structure for more info. •

rsi_ble_one_event_profile_by_uuid_t callback function is called after the service characteristics response is received. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_one_event_profile_by_uuid_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_profile structure should be passed as NULL because nothing will be filled in this structure
  -

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `4158` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_get_char_services_async

> rint32_t rsi_ble_get_char_services_async (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_char_services_t *p_char_serv_list)

Get the service characteristics of the connected / remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[out]	p_char_serv_list	- service characteristics details are filled in this structure, please refer rsi_ble_resp_char_serv_s structure for more info. •

rsi_ble_on_event_read_by_char_services_t callback function is called after the included service characteristics response is received. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_read_by_char_services_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_char_services_list structure should be passed as NULL because nothing will be filled in this structure
  -

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4195 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_inc_services_async

```
int32_t rsi_ble_get_inc_services_async (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle,
rsi_ble_resp_inc_services_t *p_inc_serv_list)
```

Get the supported include services of the connected / remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[out]	p_inc_serv_list	- include service characteristics details are filled in this structure, please refer rsi_ble_resp_inc_serv structure for more info. •

rsi_ble_on_event_read_by_inc_services_t callback function is called after the service characteristics response is received. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_read_by_inc_services_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_inc_serv_list structure should be passed as NULL because nothing will be filled in this structure
  -

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4234 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_char_value_by_uuid_async

> int32_t rsi_ble_get_char_value_by_uuid_async (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, uuid_t char_uuid, rsi_ble_resp_att_value_t *p_char_val)

Get the characteristic value by UUID (char_uuid).

Parameters

[in]	dev_addr	- remote device address
[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[in]	char_uuid	- UUID of the characteristic
[out]	p_char_val	- characteristic value is filled in this structure, please refer rsi_ble_resp_att_value_s structure for more info. •

rsi_ble_on_event_read_att_value_t callback function is called after the attribute value is received. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_read_att_value_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_char_value structure should be passed as NULL because nothing will be filled in this structure
  -

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4275 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_att_descriptors_async

> int32_t rsi_ble_get_att_descriptors_async (uint8_t *dev_addr, uint16_t start_handle, uint16_t end_handle, rsi_ble_resp_att_descs_t *p_att_desc)

Get the characteristic descriptors list from the remote device.

Parameters

[in]	dev_addr	- remote device address

[in]	start_handle	- start handle (index) of the remote device's service records
[in]	end_handle	- end handle (index) of the remote device's service records
[out]	p_att_desc	- pointer to characteristic descriptor structure, Please refer rsi_ble_resp_att_descs_s strcuture for more info. •

rsi_ble_on_gatt_desc_val_event_t callback function is called after the attribute descriptors response is received. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_gatt_desc_val_event_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_att_desc structure should be passed as NULL because nothing will be filled in this structure
  -

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4313 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_att_value_async

> int32_t rsi_ble_get_att_value_async (uint8_t *dev_addr, uint16_t handle, rsi_ble_resp_att_value_t *p_att_val)

Get the attribute with a handle.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- handle value of the attribute
[out]	p_att_val	- attribute value is filled in this structure, Please refer rsi_ble_resp_att_value_s structure for more info. •

rsi_ble_on_event_read_resp_t callback function is called upon receiving the attribute value. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_read_resp_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_att_val structure should be passed as NULL because nothing will be filled in this structure
  -

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4348 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_multiple_att_values_async

> int32_t rsi_ble_get_multiple_att_values_async (uint8_t *dev_addr, uint8_t num_of_handlers, uint16_t *handles, rsi_ble_resp_att_value_t *p_att_vals)

Get the multiple attribute values by using multiple handles.

Parameters

[in]	dev_addr	- remote device address
[in]	num_of_handlers	- number of handles in the list
[in]	handles	- list of attribute handles
[out]	p_att_vals	- attribute values filled in this structure, please refer rsi_ble_resp_att_value_s structure for more info. •

rsi_ble_on_event_read_resp_t callback function is called after the attribute value is received. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_read_resp_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_att_vals structure should be passed as NULL because nothing will be filled in this structure
  -

Returns

- The following values are returned:
  - 0 - Success
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4382 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_long_att_value_async

> int32_t rsi_ble_get_long_att_value_async (uint8_t *dev_addr, uint16_t handle, uint16_t offset, rsi_ble_resp_att_value_t *p_att_vals)

Get the long attribute value by using handle and offset.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute handle
[in]	offset	- offset within the attribute value
[out]	p_att_vals	- attribute value filled in this structure, please refer rsi_ble_resp_att_value_s structure for more info.

rsi_ble_on_event_read_resp_t callback function is called after the attribute value is received. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_read_resp_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Note

- p_att_vals structure should be passed as NULL because nothing will be filled in this structure

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4420 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_att_value_async

> int32_t rsi_ble_set_att_value_async (uint8_t *dev_addr, uint16_t handle, uint8_t data_len, uint8_t *p_data)

Set the attribute value of the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute value handle
[in]	data_len	- attribute value length

[in]	p_data	- attribute value

rsi_ble_on_event_write_resp_t callback function is called after the attribute set action is completed. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_write_resp_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4453 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_prepare_write_async

> int32_t rsi_ble_prepare_write_async (uint8_t *dev_addr, uint16_t handle, uint16_t offset, uint8_t data_len, uint8_t *p_data)

Prepare the attribute value.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute handle
[in]	offset	- attribute value offset
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

rsi_ble_on_event_prepare_write_resp_t callback function is called after the prepare attribute write action is completed. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_prepare_write_resp_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API

Returns

- The following values are returned:
  - 0 - Success
  - 0x4E60 - Invalid Handle range
  - 0x4E62 - Invalid Parameters
  - 0x4D04 - BLE not connected
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4485 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_execute_write_async

> int32_t rsi_ble_execute_write_async (uint8_t *dev_addr, uint8_t exe_flag)

Execute the prepared attribute values.

Parameters

[in]	dev_addr	- remote device address
[in]	exe_flag	- execute flag to write, possible values mentioned below • 0 - BLE_ATT_EXECUTE_WRITE_CANCEL • 1 - BLE_ATT_EXECUTE_PENDING_WRITES_IMMEDIATELY •

rsi_ble_on_event_write_resp_t callback function is called after the execute attribute write action is completed. This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_write_resp_t or rsi_ble_on_gatt_error_resp_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4D05 - BLE Socket not available
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 4516 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Server

Server

Functions

int32_t rsi_ble_add_service(uuid_t service_uuid, rsi_ble_resp_add_serv_t *p_resp_serv)
Add a new service to the local GATT Server.

int32_t rsi_ble_add_attribute(rsi_ble_req_add_att_t *p_attribute)
Add a new attribute to a specific service.

int32_t rsi_ble_set_local_att_value(uint16_t handle, uint16_t data_len, uint8_t *p_data)
Change the local attribute value.

int32_t rsi_ble_set_wo_resp_notify_buf_info(uint8_t *dev_addr, uint8_t buf_mode, uint8_t buf_cnt)
Configure the buf mode for Notify and WO response commands for the remote device.

int32_t rsi_ble_notify_value(uint8_t *dev_addr, uint16_t handle, uint16_t data_len, uint8_t *p_data)
Notify the local value to the remote device.

int32_t rsi_ble_indicate_value(uint8_t *dev_addr, uint16_t handle, uint16_t data_len, uint8_t *p_data)
Indicate the local value to the remote device.

int32_t rsi_ble_get_local_att_value(uint16_t handle, rsi_ble_resp_local_att_value_t *p_resp_local_att_val)
Get the local attribute value.

int32_t rsi_ble_gatt_read_response(uint8_t *dev_addr, uint8_t read_type, uint16_t handle, uint16_t offset, uint16_t length, uint8_t *p_data)
Send the response for the read request received from the remote device.

int32_t rsi_ble_remove_gatt_service(uint32_t service_handler)
Remove the GATT service record.

int32_t rsi_ble_remove_gatt_attibute(uint32_t service_handler, uint16_t att_hndl)
Remove the GATT attribute record.

int32_t rsi_ble_att_error_response(uint8_t *dev_addr, uint16_t handle, uint8_t opcode, uint8_t err)
Send attribute error response for any of the att request.

int32_t rsi_ble_gatt_write_response(uint8_t *dev_addr, uint8_t type)
Send the response to the write request received from the remote device.

int32_t rsi_ble_gatt_prepare_write_response(uint8_t *dev_addr, uint16_t handle, uint16_t offset, uint16_t length, uint8_t *data)
Send the response for the prepare write requests received from the remote device.

Function Documentation

rsi_ble_add_service

```
int32_t rsi_ble_add_service (uuid_t service_uuid, rsi_ble_resp_add_serv_t *p_resp_serv)
```

Add a new service to the local GATT Server.

Parameters

[in]	service_uuid	- new service UUID value, please refer uuid_s structure for more info.
[out]	p_resp_serv	- new service handler filled in this structure, please refer rsi_ble_resp_add_serv_s structure for more info.

This is a Blocking API.

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4046 - Invalid Arguments
  - 0x4D08 - Profile record full
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `3372` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_add_attribute

```
int32_t rsi_ble_add_attribute (rsi_ble_req_add_att_t *p_attribute)
```

Add a new attribute to a specific service.

Parameters

[in]	p_attribute	- add a new attribute to the service, please refer rsi_ble_req_add_att_s structure for more info. •

This is a Blocking API.

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4046 - Invalid Arguments
  - 0x4D09 - Attribute record full
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `3394` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_local_att_value

> int32_t rsi_ble_set_local_att_value (uint16_t handle, uint16_t data_len, uint8_t *p_data)

Change the local attribute value.

Parameters

[in]	handle	- attribute value handle
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

This is a Blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4046 - Invalid Arguments
  - 0x4D06 - Attribute record not found
  - 0x4E60 - Invalid Handle Range
  -

Note

- Refer Error Codes section for above error codes error-codes .
- This API can only be used if the service is maintained inside the firmware.
  - The services which are maintained by firmware must follow the below rules.
  - Rule 1: The attribute_data_size is less than 20 bytes during the service_creation
  - Rule 2: while creating the service, don't use the RSI_BLE_ATT_MAINTAIN_IN_HOST bit
  - in the RSI_BLE_ATT_CONFIG_BITMAP macro.
  - Rule 3: The data_len must be less than or equal to the dat_length mentioned while
  - creating the service/attribute
  - Rule 4: If the services are maintained in the Application/Host,
  - then need to use rsi_ble_notify_value() API to send the notifications to the remote devices.
  -

Definition at line `3438` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_set_wo_resp_notify_buf_info

> int32_t rsi_ble_set_wo_resp_notify_buf_info (uint8_t *dev_addr, uint8_t buf_mode, uint8_t buf_cnt)

Configure the buf mode for Notify and WO response commands for the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	buf_mode	- buffer mode configuration • 0 - BLE_SMALL_BUFF_MODE • 1 - BLE_BIG_BUFF_MODE •

[in]	buf_cnt	- no of buffers to be configured • only value 1 and 2 are supported in BLE_SMALL_BUFF_MODE • in BLE_BIG_BUFF_MODE, buffers allocated based on the below notations. intial available_buf_cnt = RSI_BLE_NUM_CONN_EVENTS, a) When connection 1 is formed, the possible range of buffers is (available_buf_cnt - remaining possible number of connections) b) After allocating X buffers using \ref rsi_ble_set_wo_resp_notify_buf_info to the 1st connection remaining available_buf_cnt = (available_buf_cnt - X) •

This is a Blocking API. **Returns**

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure 0x4046 - Invalid Arguments
  - 0x4D05 - BLE socket not available
  - 0x4D06 - Attribute record not found
  - 0x4E60 - Invalid Handle Range
  - 0x4E63 - BLE Buffer Count Exceeded
  - 0x4E64 - BLE Buffer already in use
  -

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `3481` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_notify_value

> int32_t rsi_ble_notify_value (uint8_t *dev_addr, uint16_t handle, uint16_t data_len, uint8_t *p_data)

Notify the local value to the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- local attribute handle
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

This is a Blocking API. If the API returns RSI_ERROR_BLE_DEV_BUF_FULL (-31) error then wait until the rsi_ble_on_le_more_data_req_t event gets received from the module.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4046 - Invalid Arguments
  - 0x4A0D - Invalid attribute value length
  - 0x4D05 - BLE socket not available
  - 0x4D06 - Attribute record not found
  - 0x4E60 - Invalid Handle Range

0x4E65 - Invalid Attribute Length When Small Buffer Mode is Configured

Note

- Refer Error Codes section for above error codes error-codes
- If the services are maintained in the Application/Host,
  - then need to use rsi_ble_notify_value() API instead of using rsi_ble_set_local_att_value() API
  - to send the notifications to the remote devices.

Definition at line 3521 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_indicate_value

```
int32_t rsi_ble_indicate_value (uint8_t *dev_addr, uint16_t handle, uint16_t data_len, uint8_t *p_data)
```

Indicate the local value to the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- local attribute handle
[in]	data_len	- attribute value length
[in]	p_data	- attribute value

This is a blocking API and can unblock the application on the reception of the callback functions either rsi_ble_on_event_indicate_confirmation_t.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4D05 - BLE socket not available
  - 0x4E60 - Invalid Handle Range

Note

- Refer Error Codes section for above error codes error-codes

Definition at line 3548 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_get_local_att_value

```
int32_t rsi_ble_get_local_att_value (uint16_t handle, rsi_ble_resp_local_att_value_t *p_resp_local_att_val)
```

Get the local attribute value.

Parameters

[in]	handle	- local attribute handle
[out]	p_resp_local_att_val	- local attribute value filled in this structure, see rsi_ble_resp_local_att_value_s structure for more info.

This is a Blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4046 - Invalid Arguments
  - 0x4D06 - Attribute record not found
  -

Note

- Refer Error Codes section for above error codes error-codes
  -
- This API can only be used if the service is maintained inside the firmware. The services which are maintained by firmware must follow the below rules.
  - Rule 1: The attribute_data_size is less than 20 bytes during the service_creation
  - Rule 2: While creating the service, don't use the RSI_BLE_ATT_MAINTAIN_IN_HOST bit in the RSI_BLE_ATT_CONFIG_BITMAP macro.

Definition at line `3637` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_gatt_read_response

> int32_t rsi_ble_gatt_read_response (uint8_t *dev_addr, uint8_t read_type, uint16_t handle, uint16_t offset, uint16_t length, uint8_t *p_data)

Send the response for the read request received from the remote device.

Parameters

[in]	dev_addr	- remote device Address
[in]	read_type	- read value type • 0 - Read response • 1 - Read blob response
[in]	handle	- attribute value handle
[in]	offset	- attribute value offset
[in]	length	- attribute value length
[in]	p_data	- attribute value

This is a Blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4D04 - BLE not connected
  -

Note

Refer Error Codes section for above error codes error-codes

Definition at line 3670 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_remove_gatt_service

```
int32_t rsi_ble_remove_gatt_service (uint32_t service_handler)
```

Remove the GATT service record.

Parameters

[in]	service_handler	- GATT service record handle

This is a Blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4D0A - BLE profile not found (profile handler invalid)

Note

- Refer Error Codes section for above error codes error-codes

Definition at line 3694 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_remove_gatt_attibute

```
int32_t rsi_ble_remove_gatt_attibute (uint32_t service_handler, uint16_t att_hndl)
```

Remove the GATT attribute record.

Parameters

[in]	service_handler	- GATT service record handle
[in]	att_hndl	- attribute handle

This is a Blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure
  - 0x4D06 - Attribute record not found

Note

Refer Error Codes section for above error codes error-codes

Definition at line 3714 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_att_error_response

```
int32_t rsi_ble_att_error_response (uint8_t *dev_addr, uint16_t handle, uint8_t opcode, uint8_t err)
```

Send attribute error response for any of the att request.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute handle
[in]	opcode	- error response opcode
[in]	err	- specific error related Gatt

This is a Blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure 0x4D04 - BLE not Connected
  - 0x4E62 - Invalid Parameters

Note

- Refer Error Codes section for above error codes error-codes

Definition at line 3738 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_gatt_write_response

```
int32_t rsi_ble_gatt_write_response (uint8_t *dev_addr, uint8_t type)
```

Send the response to the write request received from the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	type	- response type • 0 - write response, • 1 - execute write response.

This is a Blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

The following values are returned:
- 0 - Success
- Non-Zero Value - Failure 0x4046 - Invalid Arguments
- 0x4D04 - BLE not Connected
-

Note

- Refer Error Codes section for above error codes error-codes
-

Definition at line 3815 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_gatt_prepare_write_response

```
int32_t rsi_ble_gatt_prepare_write_response (uint8_t *dev_addr, uint16_t handle, uint16_t offset, uint16_t length, uint8_t *data)
```

Send the response for the prepare write requests received from the remote device.

Parameters

[in]	dev_addr	- remote device address
[in]	handle	- attribute value handle
[in]	offset	- attribute value offset
[in]	length	- attribute value length
[in]	data	- attribute value

This is a Blocking API.

- Pre-conditions:
  - rsi_ble_connect() API needs to be called before this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure 0x4046 - Invalid Arguments
  - 0x4D04 - BLE not Connected
  -

Note

- Refer Error Codes section for above error codes error-codes
-

Definition at line 3843 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Test Mode

Modules

rsi_ble_per_transmit_s

rsi_ble_per_receive_s

Typedefs

typedef struct rsi_ble_per_transmit_s	rsi_ble_per_transmit_t
typedef struct rsi_ble_per_receive_s	rsi_ble_per_receive_t

Functions

int32_t	rsi_ble_rx_test_mode(uint8_t rx_channel, uint8_t phy, uint8_t modulation) Start the BLE RX test mode in controller.
int32_t	rsi_ble_tx_test_mode(uint8_t tx_channel, uint8_t phy, uint8_t tx_len, uint8_t mode) Start the BLE TX test mode in controller.
int32_t	rsi_ble_end_test_mode(uint16_t *num_of_pkts) Stop the BLE TX / RX test mode in controller.
int32_t	rsi_ble_per_transmit(struct rsi_ble_per_transmit_s *rsi_ble_per_tx) Initiate the BLE transmit PER mode in the controller.
int32_t	rsi_ble_per_receive(struct rsi_ble_per_receive_s *rsi_ble_per_rx) Initiate the BLE receive PER mode in the controller.

Typedef Documentation

rsi_ble_per_transmit_t

```
typedef struct rsi_ble_per_transmit_s rsi_ble_per_transmit_t
```

Definition at line 1128 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_per_receive_t

```
typedef struct rsi_ble_per_receive_s rsi_ble_per_receive_t
```

Definition at line 1220 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Function Documentation

rsi_ble_rx_test_mode

> int32_t rsi_ble_rx_test_mode (uint8_t rx_channel, uint8_t phy, uint8_t modulation)

Start the BLE RX test mode in controller.

Parameters

[in]	rx_channel	- Channel in which packet have to be received (0 - 39)
[in]	phy	- 0x00 Reserved for future use • 0x01 Receiver set to use the LE 1M PHY • 0x02 Receiver set to use the LE 2M PHY • 0x03 Receiver set to use the LE Coded PHY • (0x04 - 0xFF) Reserved for future use.
[in]	modulation	- 0x00 Assume transmitter will have a standard standard modulation index • 0x01 Assume transmitter will have a stable modulation index • (0x02 - 0xFF) Reserved for future use

This is a Blocking API **Returns**

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line 2692 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_tx_test_mode

> int32_t rsi_ble_tx_test_mode (uint8_t tx_channel, uint8_t phy, uint8_t tx_len, uint8_t mode)

Start the BLE TX test mode in controller.

Parameters

[in]	tx_channel	- RF Channel (0-39). •
[in]	phy	- 0x00 Reserved for future use • 0x01 Transmitter set to use the LE 1M PHY • 0x02 Transmitter set to use the LE 2M PHY • 0x03 Transmitter set to use the LE Coded PHY with S=8 data coding • 0x04 Transmitter set to use the LE Coded PHY with S=2 data coding • (0x05 - 0xFF) Reserved for future use.
[in]	tx_len	- Length in bytes of payload data in each packet (1 - 251 bytes).

[in]	mode	- 0x00 PRBS9 sequence '11111111100000111101...' • 0x01 Repeated '11110000' • 0x02 Repeated '10101010' • 0x03 PRBS15 • 0x04 Repeated '11111111' • 0x05 Repeated '00000000' • 0x06 Repeated '00001111' • 0x07 Repeated '01010101' • 0x08 - 0xFF Reserved for future use •

This is a Blocking API **Returns**

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2737` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_end_test_mode

> int32_t rsi_ble_end_test_mode (uint16_t *num_of_pkts)

Stop the BLE TX / RX test mode in controller.

Parameters

[out]	num_of_pkts	- Number of RX packets received are displayed when RX test is stopped •

This is a Blocking API **Returns**

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2751` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_per_transmit

> int32_t rsi_ble_per_transmit (struct rsi_ble_per_transmit_s *rsi_ble_per_tx)

Initiate the BLE transmit PER mode in the controller.

Parameters

[in]	rsi_ble_per_tx	- This parameter is the buffer to hold the structure values • This is a structure variable of struct rsi_ble_per_transmit_s

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2768` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_per_receive

```
int32_t rsi_ble_per_receive (struct rsi_ble_per_receive_s *rsi_ble_per_rx)
```

Initiate the BLE receive PER mode in the controller.

Parameters

[in]	rsi_ble_per_rx	- This parameter is the buffer to hold the structure values • This is a structure variable of struct rsi_ble_per_receive_s

This is a Blocking API

- Pre-conditions:
  - Device should be initialized before calling this API.

Returns

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes .

Definition at line `2785` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_per_transmit_s

Public Attributes

uint8_t **cmd_ix**
Command ID

- Takes per BLE_TRANSMIT_CMD_ID of value 0x13.

uint8_t **transmit_enable**
Enables/disables the BLE per transmit mode

- 1 PER Transmit Enable
- 0 PER Transmit Disable.

uint8_t **access_addr**
Access address with which packets are transmitted.

uint8_t **phy_rate**
Phy rate at which packets are transmitted

- 1 1Mbps
- 2 2 Mbps
- 4 125 Kbps Coded
- 8 500 Kbps Coded.

uint8_t **rx_chnl_num**
Rx channel number (0 - 39)

uint8_t **tx_chnl_num**
Tx channel number (0 - 39)

uint8_t **scrambler_seed**
Initial seed to be used for whitening.

uint8_t **le_chnl_type**
LE channel type (data or advertise channel)

- 0x00 Advertise Channel
- 0x01 Data Channel (to be used by Default)

uint8_t **freq_hop_en**
Frequency hopping type to be used

- 0 No Hopping
- 1 Fixed Hopping
- 2 Random Hopping (rx_chnl_num, tx_chnl_num parameters are unused in this mode)

uint8_t **ant_sel**
Select the antenna to be used.

uint8_t **pll_mode**
pll_mode type to be used

uint8_t rf_type
Selection of RF type (internal/external)

- 0 BT_EXTERNAL_RF
- 1 BT_INTERNAL_RF (to be used by Default)

uint8_t rf_chain
Selection of RF Chain (HP/LP) to be used

- 2 BT_HP_CHAIN
- 3 BT_LP_CHAIN.

uint8_t pkt_len
Length of the packet to be transmitted.

uint8_t payload_type
Type of payload data sequence

- 0x00 PRBS9 sequence 11111111100000111101...
- 0x01 Repeated 11110000
- 0x02 Repeated 10101010
- 0x03 PRBS15
- 0x04 Repeated 11111111
- 0x05 Repeated 00000000
- 0x06 Repeated '00001111'
- 0x07 Repeated '01010101'.

uint8_t tx_power
Transmit Power.

uint8_t transmit_mode
Transmit mode to be used either Burst/Continuous

- 0 BURST_MODE
- 1 CONTINUOUS_MODE
- 2 CONTINUOUS_WAVE_MODE (CW_MODE)

uint8_t inter_pkt_gap
This field takes the value of inter packet gap.

uint8_t num_pkts
This field defines the number of packets to be transmitted, default to zero for continuous transmission.

Public Attribute Documentation

cmd_ix

> uint8_t rsi_ble_per_transmit_s::cmd_ix

Command ID

- Takes per BLE_TRANSMIT_CMD_ID of value 0x13.

Definition at line 1011 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

transmit_enable

> uint8_t rsi_ble_per_transmit_s::transmit_enable

Enables/disables the BLE per transmit mode

- 1 PER Transmit Enable
- 0 PER Transmit Disable.

Definition at line 1017 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

access_addr

```
uint8_t rsi_ble_per_transmit_s::access_addr[4]
```

Access address with which packets are transmitted.

Definition at line 1019 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

phy_rate

```
uint8_t rsi_ble_per_transmit_s::phy_rate
```

Phy rate at which packets are transmitted

- 1 1Mbps
- 2 2 Mbps
- 4 125 Kbps Coded
- 8 500 Kbps Coded.

Definition at line 1029 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rx_chnl_num

```
uint8_t rsi_ble_per_transmit_s::rx_chnl_num
```

Rx channel number (0 - 39)

Definition at line 1031 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

tx_chnl_num

```
uint8_t rsi_ble_per_transmit_s::tx_chnl_num
```

Tx channel number (0 - 39)

Definition at line 1033 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

scrambler_seed

```
uint8_t rsi_ble_per_transmit_s::scrambler_seed
```

Initial seed to be used for whitening.

It should be set to 0 in order to disable whitening.

- In order to enable, one should give the scrambler seed value which is used on the receive side

Definition at line 1037 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

le_chnl_type

```
uint8_t rsi_ble_per_transmit_s::le_chnl_type
```

LE channel type (data or advertise channel)

- 0x00 Advertise Channel
- 0x01 Data Channel (to be used by Default)

Definition at line 1043 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

freq_hop_en

```
uint8_t rsi_ble_per_transmit_s::freq_hop_en
```

Frequency hopping type to be used

- 0 No Hopping
- 1 Fixed Hopping
- 2 Random Hopping (rx_chnl_num, tx_chnl_num parameters are unused in this mode)

Definition at line 1051 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

ant_sel

```
uint8_t rsi_ble_per_transmit_s::ant_sel
```

Select the antenna to be used.

Refer to the data sheet for your hardware to check whether or not it contains an onboard antenna.

- 2 ONBOARD_ANT_SEL
- 3 EXT_ANT_SEL

Definition at line 1057 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

pll_mode

```
uint8_t rsi_ble_per_transmit_s::pll_mode
```

pll_mode type to be used

- 0 PLL_MODE0 (to be used by Default)
- 1 PLL_MODE1

Definition at line 1063 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rf_type

```
uint8_t rsi_ble_per_transmit_s::rf_type
```

Selection of RF type (internal/external)

- 0 BT_EXTERNAL_RF
- 1 BT_INTERNAL_RF (to be used by Default)

Note

- The above macros are applicable for both BT and BLE

Definition at line 1070 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rf_chain

```
uint8_t rsi_ble_per_transmit_s::rf_chain
```

Selection of RF Chain (HP/LP) to be used

- 2 BT_HP_CHAIN
- 3 BT_LP_CHAIN.

Note

- The above macros are applicable for both BT and BLE

Definition at line 1077 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

pkt_len

```
uint8_t rsi_ble_per_transmit_s::pkt_len[2]
```

Length of the packet to be transmitted.

Definition at line 1079 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

payload_type

```
uint8_t rsi_ble_per_transmit_s::payload_type
```

Type of payload data sequence

- 0x00 PRBS9 sequence ‘11111111100000111101...
- 0x01 Repeated ‘11110000‘
- 0x02 Repeated ‘10101010‘
- 0x03 PRBS15
- 0x04 Repeated ‘11111111‘
- 0x05 Repeated ‘00000000‘
- 0x06 Repeated '00001111'
- 0x07 Repeated '01010101'.

Definition at line 1097 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

tx_power

> uint8_t rsi_ble_per_transmit_s::tx_power

Transmit Power.

- Transmit power value for the rf_chain parameter set to the HP chain the values for the TX power indexes are 0 - 20.
- Transmit power value for the rf chain parameter set to LP chain and values are:
- 0 -31 o/p power equation is -2+10log10(power_index/31)
- 32-63 o/p power equation is
- 6 + 10log10((power_index -32)/31)
- TX power for the BLE LP Chain :1 to 31 (0dBm Mode), 33 to 63 (10dBm Mode)
- TX power for the BLE HP chain : 64 to 79

Definition at line `1113` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

transmit_mode

> uint8_t rsi_ble_per_transmit_s::transmit_mode

Transmit mode to be used either Burst/Continuous

- 0 BURST_MODE
- 1 CONTINUOUS_MODE
- 2 CONTINUOUS_WAVE_MODE (CW_MODE)

Definition at line `1121` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

inter_pkt_gap

> uint8_t rsi_ble_per_transmit_s::inter_pkt_gap

This field takes the value of inter packet gap.

- Number of slots to be skipped between two packets - Each slot will be 1250 usec

Definition at line `1125` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

num_pkts

> uint8_t rsi_ble_per_transmit_s::num_pkts[4]

This field defines the number of packets to be transmitted, default to zero for continuous transmission.

Definition at line `1127` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_per_receive_s

Public Attributes

uint8_t **cmd_ix**
Command ID

- Takes per BLE_RECEIVE_CMD_ID of value 0x14.

uint8_t **receive_enable**
Enables/disables the ble per receive mode

- 1 PER Receive Enable
- 0 PER Receive Disable.

uint8_t **access_addr**
Access address with which packets are received.

uint8_t **phy_rate**
Phy rate at which packets are received

- 1 1Mbps
- 2 2 Mbps
- 4 125 Kbps Coded
- 8 500 Kbps Coded.

uint8_t **rx_chnl_num**
Rx channel number (0 - 39)

uint8_t **tx_chnl_num**
Tx channel number (0 - 39)

uint8_t **scrambler_seed**
Initial seed to be used for whitening.

uint8_t **le_chnl_type**
LE channel type (data or advertise channel)

- 0x00 Advertise Channel
- 0x01 Data Channel (to be used by Default)

uint8_t **freq_hop_en**
Frequency hopping type to be used

- 0 No Hopping
- 1 Fixed Hopping
- 2 Random Hopping (rx_chnl_num, tx_chnl_num parameters are unused in this mode)

uint8_t **ant_sel**
Select the antenna to be used.

uint8_t **pll_mode**
pll_mode type to be used

uint8_t rf_type
Selection of RF type (internal/external)

- 0 BT_EXTERNAL_RF
- 1 BT_INTERNAL_RF (to be used by Default)

uint8_t rf_chain
Selection of RF Chain (HP/LP) to be used

- 2 BT_HP_CHAIN
- 3 BT_LP_CHAIN.

uint8_t ext_data_len_indication
This field enables/disables the extended data length

- 0 Extended Data length disabled
- 1 Extended Data length enabled.

uint8_t loop_back_mode
This field defines the loopback to be enable or disable

- 0 LOOP_BACK_MODE_DISABLE
- 1 LOOP_BACK_MODE_ENABLE.

uint8_t duty_cycling_en
This field enables/disables the duty cycling

- 0 Duty Cycling Disabled (to be used by Default)
- 1 Duty Cycling Enabled.

Public Attribute Documentation

cmd_ix

> uint8_t rsi_ble_per_receive_s::cmd_ix

Command ID

- Takes per BLE_RECEIVE_CMD_ID of value 0x14.

Definition at line `1135` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

receive_enable

> uint8_t rsi_ble_per_receive_s::receive_enable

Enables/disables the ble per receive mode

- 1 PER Receive Enable
- 0 PER Receive Disable.

Definition at line `1141` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

access_addr

> uint8_t rsi_ble_per_receive_s::access_addr[4]

Access address with which packets are received.

Definition at line 1143 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

phy_rate

```
uint8_t rsi_ble_per_receive_s::phy_rate
```

Phy rate at which packets are received

- 1 1Mbps
- 2 2 Mbps
- 4 125 Kbps Coded
- 8 500 Kbps Coded.

Definition at line 1153 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rx_chnl_num

```
uint8_t rsi_ble_per_receive_s::rx_chnl_num
```

Rx channel number (0 - 39)

Definition at line 1155 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

tx_chnl_num

```
uint8_t rsi_ble_per_receive_s::tx_chnl_num
```

Tx channel number (0 - 39)

Definition at line 1157 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

scrambler_seed

```
uint8_t rsi_ble_per_receive_s::scrambler_seed
```

Initial seed to be used for whitening.

It should be set to 0 in order to disable whitening.

- In order to enable, one should give the scrambler seed value which is used on the transmit side

Definition at line 1161 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

le_chnl_type

```
uint8_t rsi_ble_per_receive_s::le_chnl_type
```

LE channel type (data or advertise channel)

- 0x00 Advertise Channel
- 0x01 Data Channel (to be used by Default)

Definition at line 1167 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

freq_hop_en

> uint8_t rsi_ble_per_receive_s::freq_hop_en

Frequency hopping type to be used

- 0 No Hopping
- 1 Fixed Hopping
- 2 Random Hopping (rx_chnl_num, tx_chnl_num parameters are unused in this mode)

Definition at line 1175 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

ant_sel

> uint8_t rsi_ble_per_receive_s::ant_sel

Select the antenna to be used.

Refer to the datasheet for your hardware to check whether or not it contains an onboard antenna.

- 2 ONBOARD_ANT_SEL
- 3 EXT_ANT_SEL

Definition at line 1181 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

pll_mode

> uint8_t rsi_ble_per_receive_s::pll_mode

pll_mode type to be used

- 0 PLL_MODE0 (to be used by Default)
- 1 PLL_MODE1

Definition at line 1187 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rf_type

> uint8_t rsi_ble_per_receive_s::rf_type

Selection of RF type (internal/external)

- 0 BT_EXTERNAL_RF
- 1 BT_INTERNAL_RF (to be used by Default)

Note

- The above macros are applicable for both BT and BLE

Definition at line 1194 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rf_chain

```
uint8_t rsi_ble_per_receive_s::rf_chain
```

Selection of RF Chain (HP/LP) to be used

- 2 BT_HP_CHAIN
- 3 BT_LP_CHAIN.

Note

- The above macros are applicable for both BT and BLE

Definition at line 1201 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

ext_data_len_indication

```
uint8_t rsi_ble_per_receive_s::ext_data_len_indication
```

This field enables/disables the extended data length

- 0 Extended Data length disabled
- 1 Extended Data length enabled.

Definition at line 1207 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

loop_back_mode

```
uint8_t rsi_ble_per_receive_s::loop_back_mode
```

This field defines the loopback to be enable or disable

- 0 LOOP_BACK_MODE_DISABLE
- 1 LOOP_BACK_MODE_ENABLE.

Definition at line 1213 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

duty_cycling_en

```
uint8_t rsi_ble_per_receive_s::duty_cycling_en
```

This field enables/disables the duty cycling

- 0 Duty Cycling Disabled (to be used by Default)
- 1 Duty Cycling Enabled.

Definition at line 1219 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Register Callbacks

Register Callbacks

Functions

void rsi_ble_gap_register_callbacks(rsi_ble_on_adv_report_event_t ble_on_adv_report_event, rsi_ble_on_connect_t ble_on_conn_status_event, rsi_ble_on_disconnect_t ble_on_disconnect_event, rsi_ble_on_le_ping_payload_timeout_t timeout_expired_event, rsi_ble_on_phy_update_complete_t ble_on_phy_update_complete_event, rsi_ble_on_data_length_update_t ble_on_data_length_update_complete_event, rsi_ble_on_enhance_connect_t ble_on_enhance_conn_status_event, rsi_ble_on_directed_adv_report_event_t ble_on_directed_adv_report_event, rsi_ble_on_conn_update_complete_t ble_on_conn_update_complete_event, rsi_ble_on_remote_conn_params_request_t ble_on_remote_conn_params_request_event)

Register GAP callbacks.

void rsi_ble_gap_extended_register_callbacks(rsi_ble_on_remote_features_t ble_on_remote_features_event, rsi_ble_on_le_more_data_req_t ble_on_le_more_data_req_event)

Register GAP Extended responses/events callbacks.

uint32_t rsi_ble_enhanced_gap_extended_register_callbacks(uint16_t callback_id, void(*callback_handler_ptr)(uint16_t status, uint8_t *buffer))

int32_t rsi_ble_adv_ext_events_register_callbacks(uint16_t callback_id, void(*callback_handler_ptr)(uint16_t status, uint8_t *buffer))

void rsi_ble_smp_register_callbacks(rsi_ble_on_smp_request_t ble_on_smp_request_event, rsi_ble_on_smp_response_t ble_on_smp_response_event, rsi_ble_on_smp_passkey_t ble_on_smp_passkey_event, rsi_ble_on_smp_failed_t ble_on_smp_fail_event, rsi_ble_on_encrypt_started_t rsi_ble_on_encrypt_started_event, rsi_ble_on_smp_passkey_display_t ble_on_smp_passkey_display_event, rsi_ble_on_sc_passkey_t ble_sc_passkey_event, rsi_ble_on_le_ltk_req_event_t ble_on_le_ltk_req_event, rsi_ble_on_le_security_keys_t ble_on_le_security_keys_event, rsi_ble_on_smp_response_t ble_on_cli_smp_response_event, rsi_ble_on_sc_method_t ble_on_sc_method_event)

Register the SMP callbacks.

void rsi_ble_gatt_register_callbacks(rsi_ble_on_profiles_list_resp_t ble_on_profiles_list_resp, rsi_ble_on_profile_resp_t ble_on_profile_resp, rsi_ble_on_char_services_resp_t ble_on_char_services_resp, rsi_ble_on_inc_services_resp_t ble_on_inc_services_resp, rsi_ble_on_att_desc_resp_t ble_on_att_desc_resp, rsi_ble_on_read_resp_t ble_on_read_resp, rsi_ble_on_write_resp_t ble_on_write_resp, rsi_ble_on_gatt_write_event_t ble_on_gatt_event, rsi_ble_on_gatt_prepare_write_event_t ble_on_gatt_prepare_write_event, rsi_ble_on_execute_write_event_t ble_on_execute_write_event, rsi_ble_on_read_req_event_t ble_on_read_req_event, rsi_ble_on_mtu_event_t ble_on_mtu_event, rsi_ble_on_gatt_error_resp_t ble_on_gatt_error_resp_event, rsi_ble_on_gatt_desc_val_event_t ble_on_gatt_desc_val_resp_event, rsi_ble_on_event_profiles_list_t ble_on_profiles_list_event, rsi_ble_on_event_profile_by_uuid_t ble_on_profile_by_uuid_event, rsi_ble_on_event_read_by_char_services_t ble_on_read_by_char_services_event, rsi_ble_on_event_read_by_inc_services_t ble_on_read_by_inc_services_event, rsi_ble_on_event_read_att_value_t ble_on_read_att_value_event, rsi_ble_on_event_read_resp_t ble_on_read_resp_event, rsi_ble_on_event_write_resp_t ble_on_write_resp_event, rsi_ble_on_event_indicate_confirmation_t ble_on_indicate_confirmation_event, rsi_ble_on_event_prepare_write_resp_t ble_on_prepare_write_resp_event)

Register the GATT callbacks.

void rsi_ble_gatt_extended_register_callbacks(rsi_ble_on_mtu_exchange_info_t ble_on_mtu_exchange_info_event)

Register the GATT Extended responses/events callbacks.

Function Documentation

rsi_ble_gap_register_callbacks

> void rsi_ble_gap_register_callbacks (rsi_ble_on_adv_report_event_t ble_on_adv_report_event, rsi_ble_on_connect_t ble_on_conn_status_event, rsi_ble_on_disconnect_t ble_on_disconnect_event, rsi_ble_on_le_ping_payload_timeout_t timeout_expired_event, rsi_ble_on_phy_update_complete_t ble_on_phy_update_complete_event, rsi_ble_on_data_length_update_t ble_on_data_length_update_complete_event, rsi_ble_on_enhance_connect_t ble_on_enhance_conn_status_event, rsi_ble_on_directed_adv_report_event_t ble_on_directed_adv_report_event, rsi_ble_on_conn_update_complete_t ble_on_conn_update_complete_event, rsi_ble_on_remote_conn_params_request_t ble_on_remote_conn_params_request_event)

Register GAP callbacks.

Parameters

[in]	ble_on_adv_report_event	- Callback function for Advertise events
[in]	ble_on_conn_status_event	- Callback function for Connect events
[in]	ble_on_disconnect_event	- Callback function for Disconnect events
[in]	timeout_expired_event	- Callback function for LE ping timeout events
[in]	ble_on_phy_update_complete_event	- Callback function for PHY update complete events
[in]	ble_on_data_length_update_complete_event	- Callback function for data length update events
[in]	ble_on_enhance_conn_status_event	- Callback function for enhanced connection status events
[in]	ble_on_directed_adv_report_event	- Callback function for directed advertising report events
[in]	ble_on_conn_update_complete_event	- Callback function for conn update complete events
[in]	ble_on_remote_conn_params_request_event	- Callback function to remote conn params request events

Returns

- The following values are returned:
  - void

Definition at line `4922` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_gap_extended_register_callbacks

> void rsi_ble_gap_extended_register_callbacks (rsi_ble_on_remote_features_t ble_on_remote_features_event, rsi_ble_on_le_more_data_req_t ble_on_le_more_data_req_event)

Register GAP Extended responses/events callbacks.

Parameters

[in]	ble_on_remote_features_event	- Call back function for Remote feature request
[in]	ble_on_le_more_data_req_event	- Call back function for LE More data request

- Pre-conditions:
  - Device should be initialized before calling this API.

Note

- For more information about each callback, please refer to GAP Extended callbacks description section.

Returns

- The following values are returned:
  - void

Definition at line `4945` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_enhanced_gap_extended_register_callbacks

```
uint32_t rsi_ble_enhanced_gap_extended_register_callbacks (uint16_t callback_id, void(*callback_handler_ptr)(uint16_t status, uint8_t *buffer))
```

Parameters

N/A	callback_id	
N/A	callback_handler_ptr	

Definition at line 4948 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_adv_ext_events_register_callbacks

```
int32_t rsi_ble_adv_ext_events_register_callbacks (uint16_t callback_id, void(*callback_handler_ptr)(uint16_t status, uint8_t *buffer))
```

Parameters

N/A	callback_id	
N/A	callback_handler_ptr	

Definition at line 4956 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_smp_register_callbacks

```
void rsi_ble_smp_register_callbacks (rsi_ble_on_smp_request_t ble_on_smp_request_event, rsi_ble_on_smp_response_t ble_on_smp_response_event, rsi_ble_on_smp_passkey_t ble_on_smp_passkey_event, rsi_ble_on_smp_failed_t ble_on_smp_fail_event, rsi_ble_on_encrypt_started_t rsi_ble_on_encrypt_started_event, rsi_ble_on_smp_passkey_display_t ble_on_smp_passkey_display_event, rsi_ble_on_sc_passkey_t ble_sc_passkey_event, rsi_ble_on_le_ltk_req_event_t ble_on_le_ltk_req_event, rsi_ble_on_le_security_keys_t ble_on_le_security_keys_event, rsi_ble_on_smp_response_t ble_on_cli_smp_response_event, rsi_ble_on_sc_method_t ble_on_sc_method_event)
```

Register the SMP callbacks.

Parameters

[in]	ble_on_smp_request_event	- smp request callback
[in]	ble_on_smp_response_event	- smp response callback
[in]	ble_on_smp_passkey_event	- smp passkey callback
[in]	ble_on_smp_fail_event	- smp failed callback
[in]	rsi_ble_on_encrypt_started_event	- encryption enabled callback
[in]	ble_on_smp_passkey_display_event	- smp passkey display callback
[in]	ble_sc_passkey_event	- sc passkey display callback
[in]	ble_on_le_ltk_req_event	- This is the SMP LTK request callback
[in]	ble_on_le_security_keys_event	- This is the SMP security keys callback
[in]	ble_on_cli_smp_response_event	- sc method display callback
N/A	ble_on_sc_method_event	

Returns

- The following values are returned:
  - void

Definition at line 4975 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_gatt_register_callbacks

> void rsi_ble_gatt_register_callbacks (rsi_ble_on_profiles_list_resp_t ble_on_profiles_list_resp, rsi_ble_on_profile_resp_t ble_on_profile_resp, rsi_ble_on_char_services_resp_t ble_on_char_services_resp, rsi_ble_on_inc_services_resp_t ble_on_inc_services_resp, rsi_ble_on_att_desc_resp_t ble_on_att_desc_resp, rsi_ble_on_read_resp_t ble_on_read_resp, rsi_ble_on_write_resp_t ble_on_write_resp, rsi_ble_on_gatt_write_event_t ble_on_gatt_event, rsi_ble_on_gatt_prepare_write_event_t ble_on_gatt_prepare_write_event, rsi_ble_on_execute_write_event_t ble_on_execute_write_event, rsi_ble_on_read_req_event_t ble_on_read_req_event, rsi_ble_on_mtu_event_t ble_on_mtu_event, rsi_ble_on_gatt_error_resp_t ble_on_gatt_error_resp_event, rsi_ble_on_gatt_desc_val_event_t ble_on_gatt_desc_val_resp_event, rsi_ble_on_event_profiles_list_t ble_on_profiles_list_event, rsi_ble_on_event_profile_by_uuid_t ble_on_profile_by_uuid_event, rsi_ble_on_event_read_by_char_services_t ble_on_read_by_char_services_event, rsi_ble_on_event_read_by_inc_services_t ble_on_read_by_inc_services_event, rsi_ble_on_event_read_att_value_t ble_on_read_att_value_event, rsi_ble_on_event_read_resp_t ble_on_read_resp_event, rsi_ble_on_event_write_resp_t ble_on_write_resp_event, rsi_ble_on_event_indicate_confirmation_t ble_on_indicate_confirmation_event, rsi_ble_on_event_prepare_write_resp_t ble_on_prepare_write_resp_event)

Register the GATT callbacks.

Parameters

[in]	ble_on_profiles_list_resp	ble_on_profiles_list_resp - Callback for rsi_ble_get_profiles command
[in]	ble_on_profile_resp	ble_on_profile_resp - Callback for rsi_ble_get_profile command
[in]	ble_on_char_services_resp	ble_on_char_services_resp - Callback for rsi_ble_get_char_services command
[in]	ble_on_inc_services_resp	ble_on_inc_services_resp - Callback for rsi_ble_get_inc_services command
[in]	ble_on_att_desc_resp	ble_on_att_desc_resp - Callback for rsi_ble_get_att_descriptors command
[in]	ble_on_read_resp	ble_on_read_resp - Callback for all read requests command
[in]	ble_on_write_resp	ble_on_write_resp - Callback for all write commands
[in]	ble_on_gatt_event	blw_on_gatt_event - Callback for all GATT events
[in]	ble_on_gatt_prepare_write_event	ble_on_gatt_error_resp_event - Callback for GATT error events
[in]	ble_on_execute_write_event	ble_on_gatt_desc_val_resp_event - Callback for GATT descriptor value event
[in]	ble_on_read_req_event	ble_on_profiles_list_event - Callback function for profiles list event
[in]	ble_on_mtu_event	ble_on_profile_by_uuid_event - Callback function for profile event
[in]	ble_on_gatt_error_resp_event	ble_on_read_by_char_services_event- Callback function for char services event
[in]	ble_on_gatt_desc_val_resp_event	ble_on_read_by_inc_services_event - Callback function for inc services event
[in]	ble_on_profiles_list_event	ble_on_read_att_value_event - Callback function for read att value event
[in]	ble_on_profile_by_uuid_event	ble_on_read_resp_event - Callback function for read att event
[in]	ble_on_read_by_char_services_event	ble_on_write_resp_event - Callback function for write event
[in]	ble_on_read_by_inc_services_event	ble_on_indicate_confirmation_event- Callback function for indicate confirmation event
[in]	ble_on_read_att_value_event	ble_on_prepare_write_resp_event - Callback function for prepare write event
N/A	ble_on_read_resp_event	
N/A	ble_on_write_resp_event	
N/A	ble_on_indicate_confirmation_event	
N/A	ble_on_prepare_write_resp_event	

Returns

- The following values are returned:
  - void

Definition at line 5796 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_gatt_extended_register_callbacks

> void rsi_ble_gatt_extended_register_callbacks (rsi_ble_on_mtu_exchange_info_t ble_on_mtu_exchange_info_event)

Register the GATT Extended responses/events callbacks.

Parameters

[in]	ble_on_mtu_exchange_info_event	ble_on_mtu_exchange_info_event - Call back function for MTU Exchange information Event

Returns

- The following values are returned:
  - void

Definition at line 5828 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Callbacks Declarations

Typedefs

typedef void(* rsi_ble_on_adv_report_event_t)(rsi_ble_event_adv_report_t *rsi_ble_event_adv)
The callback function will be called if advertise report event is received.

typedef void(* rsi_ble_on_connect_t)(rsi_ble_event_conn_status_t *rsi_ble_event_conn)
The callback function will be called if BLE connection status is received.

typedef void(* rsi_ble_on_enhance_connect_t)(rsi_ble_event_enhance_conn_status_t *rsi_ble_event_enhance_conn)
The callback function will be called if BLE connection status is received.

typedef void(* rsi_ble_on_disconnect_t)(rsi_ble_event_disconnect_t *rsi_ble_event_disconnect, uint16_t reason)
The callback function will be called if disconnect event is received.

typedef void(* rsi_ble_on_le_ping_payload_timeout_t)(rsi_ble_event_le_ping_time_expired_t *rsi_ble_event_timeout_expired)
The callback function will be called if le ping payload timeout expired event is received.

typedef void(* rsi_ble_on_le_ltk_req_event_t)(rsi_bt_event_le_ltk_request_t *rsi_ble_event_le_ltk_request)
The callback function will be called if LE LTK request event is received.

typedef void(* rsi_ble_on_le_security_keys_t)(rsi_bt_event_le_security_keys_t *rsi_ble_event_le_security_keys)
The callback function will be called if LE security keys event is received.

typedef void(* rsi_ble_on_smp_request_t)(rsi_bt_event_smp_req_t *remote_dev_address)
The callback function will be called if smp request is received in central mode.

typedef void(* rsi_ble_on_smp_response_t)(rsi_bt_event_smp_resp_t *remote_dev_address)
The callback function will be called if smp request is received in peripheral mode.

typedef void(* rsi_ble_on_smp_passkey_t)(rsi_bt_event_smp_passkey_t *remote_dev_address)
The callback function will be called if smp passkey event is received from module.

typedef void(* rsi_ble_on_smp_passkey_display_t)(rsi_bt_event_smp_passkey_display_t *smp_passkey_display)
The callback function will be called if smp passkey event is received from module.

typedef void(* rsi_ble_on_smp_failed_t)(uint16_t resp_status, rsi_bt_event_smp_failed_t *remote_dev_address)
The callback function will be called if smp failed event is received from module.

typedef void(* rsi_ble_on_sc_method_t)(rsi_bt_event_sc_method_t *scmethod)
The callback function will be called if security method event is received from module.

typedef void(* rsi_ble_on_encrypt_started_t)(uint16_t resp_status, rsi_bt_event_encryption_enabled_t *enc_enabled)
The callback function will be called if encrypted event is received from module.

typedef void(* rsi_ble_on_sc_passkey_t)(rsi_bt_event_sc_passkey_t *sc_passkey)
The callback function will be called if BLE Secure connection passkey event received from module.

typedef void(* rsi_ble_on_phy_update_complete_t)(rsi_ble_event_phy_update_t *rsi_ble_event_phy_update_complete)
The callback function will be called when PHY update complete event is received.

typedef void(* rsi_ble_on_conn_update_complete_t)(rsi_ble_event_conn_update_t *rsi_ble_event_conn_update_complete, uint16_t resp_status)
The callback function will be called when conn update complete event is received.

typedef void(* rsi_ble_on_remote_conn_params_request_t)(rsi_ble_event_remote_conn_param_req_t *rsi_ble_event_remote_conn_param, uint16_t resp_status)
The callback function will be called if remote conn params request is received.

typedef void(* rsi_ble_on_remote_features_t)(rsi_ble_event_remote_features_t *rsi_ble_event_remote_features)
Callback function to peer device supported features.

typedef void(* rsi_ble_on_le_more_data_req_t)(rsi_ble_event_le_dev_buf_ind_t *rsi_ble_more_data_evt)
Callback function to LE more data request.

typedef void(* rsi_ble_on_data_length_update_t)(rsi_ble_event_data_length_update_t *remote_dev_address)
This event is raised when data length is update in controller.

typedef void(* rsi_ble_on_directed_adv_report_event_t)(rsi_ble_event_directedadv_report_t *rsi_ble_event_directed)
The callback function will be called if directed advertise report event is received.

typedef void(* rsi_ble_on_gatt_error_resp_t)(uint16_t event_status, rsi_ble_event_error_resp_t *rsi_ble_gatt_error)
The callback function will be called if GATT error event is received.

typedef void(* rsi_ble_on_gatt_desc_val_event_t)(uint16_t event_status, rsi_ble_event_gatt_desc_t *rsi_ble_gatt_desc_val)
The callback function will be called if attribute descriptors event is received.

typedef void(* rsi_ble_on_event_profiles_list_t)(uint16_t event_status, rsi_ble_event_profiles_list_t *rsi_ble_event_profiles)
The callback function will be called if profiles list event is received.

typedef void(* rsi_ble_on_event_profile_by_uuid_t)(uint16_t event_status, rsi_ble_event_profile_by_uuid_t *rsi_ble_event_profile)
The callback function will be called if profile event is received.

typedef void(* rsi_ble_on_event_read_by_char_services_t)(uint16_t event_status, rsi_ble_event_read_by_type1_t *rsi_ble_event_read_type1)
The callback function will be called if characteristic services list event is received.

typedef void(* rsi_ble_on_event_read_by_inc_services_t)(uint16_t event_status, rsi_ble_event_read_by_type2_t *rsi_ble_event_read_type2)
The callback function will be called if include services list event is received.

typedef void(* rsi_ble_on_event_read_att_value_t)(uint16_t event_status, rsi_ble_event_read_by_type3_t *rsi_ble_event_read_type3)
The callback function will be called if attribute value event is received.

typedef void(* rsi_ble_on_event_read_resp_t)(uint16_t event_status, rsi_ble_event_att_value_t *rsi_ble_event_att_val)
The callback function will be called if attribute value event is received.

typedef void(* rsi_ble_on_event_write_resp_t)(uint16_t event_status, rsi_ble_set_att_resp_t *rsi_ble_event_set_att_rsp)
The callback function will be called if GATT write event is received.

typedef void(* rsi_ble_on_event_indicate_confirmation_t)(uint16_t event_status, rsi_ble_set_att_resp_t *rsi_ble_event_set_att_rsp)
Callback function to be called if indication confirmation event is received.

typedef void(* rsi_ble_on_event_prepare_write_resp_t)(uint16_t event_status, rsi_ble_prepare_write_resp_t *rsi_ble_event_prepare_write)
The callback function will be called if indication confirmation event is received.

typedef void(* rsi_ble_on_profiles_list_resp_t)(uint16_t resp_status, rsi_ble_resp_profiles_list_t *rsi_ble_resp_profiles)
The callback function will be called if profiles list response is received.

typedef void(* rsi_ble_on_profile_resp_t)(uint16_t resp_status, profile_descriptors_t *rsi_ble_resp_profile)
The callback function will be called if profile response is received.

typedef void(* rsi_ble_on_char_services_resp_t)(uint16_t resp_status, rsi_ble_resp_char_services_t *rsi_ble_resp_char_serv)
The callback function will be called if service characteristics response is received.

typedef void(* rsi_ble_on_inc_services_resp_t)(uint16_t resp_status, rsi_ble_resp_inc_services_t *rsi_ble_resp_inc_serv)
The callback function will be called if include services response is received.

typedef void(* rsi_ble_on_att_desc_resp_t)(uint16_t resp_status, rsi_ble_resp_att_descs_t *rsi_ble_resp_att_desc)
The callback function will be called if attribute descriptors response is received.

typedef void(* rsi_ble_on_read_resp_t)(uint16_t resp_status, uint16_t resp_id, rsi_ble_resp_att_value_t *rsi_ble_resp_att_val)
The callback function will be called upon receiving the attribute value.

typedef void(* rsi_ble_on_write_resp_t)(uint16_t resp_status, uint16_t resp_id)
The callback function will be called if the attribute set/prepare/execute action is completed.

typedef void(* rsi_ble_on_gatt_write_event_t)(uint16_t event_id, rsi_ble_event_write_t *rsi_ble_write)
The callback function will be called if the GATT write/notify/indicate events are received.

typedef void(* rsi_ble_on_gatt_prepare_write_event_t)(uint16_t event_id, rsi_ble_event_prepare_write_t *rsi_ble_write)
The callback function will be called if the GATT prepare events are received.

typedef void(* rsi_ble_on_execute_write_event_t)(uint16_t event_id, rsi_ble_execute_write_t *rsi_ble_execute_write)
The callback function will be called if the GATT execute events are received.

typedef void(* rsi_ble_on_read_req_event_t)(uint16_t event_id, rsi_ble_read_req_t *rsi_ble_read_req)
The callback function will be called if the GATT read request events are received.

typedef void(* rsi_ble_on_mtu_event_t)(rsi_ble_event_mtu_t *rsi_ble_event_mtu)
The callback function will be called if MTU size request is received.

typedef void(* rsi_ble_on_mtu_exchange_info_t)(rsi_ble_event_mtu_exchange_information_t *rsi_ble_event_mtu_exchange_info)
Callback function to indicate MTU size and who initiated MTU Exchange Request.

typedef void(* rsi_ble_on_remote_device_info_t)(uint16_t status, rsi_ble_event_remote_device_info_t *resp_buffer)
@callback rsi_ble_on_remote_device_info_t

typedef void(* rsi_ble_on_rcp_resp_rcvd_t)(uint16_t status, rsi_ble_event_rcp_rcvd_info_t *resp_buffer)

Typedef Documentation

rsi_ble_on_adv_report_event_t

```
void(* rsi_ble_on_adv_report_event_t)(rsi_ble_event_adv_report_t *rsi_ble_event_adv) )(rsi_ble_event_adv_report_t *rsi_ble_event_adv)
```

The callback function will be called if advertise report event is received.

Parameters

[out]	rsi_ble_event_adv	contains the advertise report information. Please refer rsi_ble_event_adv_report_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the advertise event report is received from the module

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4560 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_connect_t

```
void(* rsi_ble_on_connect_t)(rsi_ble_event_conn_status_t *rsi_ble_event_conn) )(rsi_ble_event_conn_status_t
*rsi_ble_event_conn)
```

The callback function will be called if BLE connection status is received.

Parameters

[out]	rsi_ble_event_conn	contains the BLE connection status. Please refer rsi_ble_event_conn_status_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the BLE connection status is received from the module. For BLE 4.1 and lower version this callback will be called

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4573 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_enhance_connect_t

```
void(* rsi_ble_on_enhance_connect_t)(rsi_ble_event_enhance_conn_status_t *rsi_ble_event_enhance_conn) )
(rsi_ble_event_enhance_conn_status_t *rsi_ble_event_enhance_conn)
```

The callback function will be called if BLE connection status is received.

Parameters

[out]	rsi_ble_event_conn	contains the BLE connection status. Please refer rsi_ble_event_enhance_conn_status_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the BLE connection status is received from the module. For BLE 4.2 and above version this callback will be called

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4586 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_disconnect_t

```
void(* rsi_ble_on_disconnect_t)(rsi_ble_event_disconnect_t *rsi_ble_event_disconnect, uint16_t reason) )
(rsi_ble_event_disconnect_t *rsi_ble_event_disconnect, uint16_t reason)
```

The callback function will be called if disconnect event is received.

Parameters

[out]	rsi_ble_event_disconnect	contains the disconnect status. Please refer rsi_ble_event_disconnect_s for more info.
[out]	reason	contains reason for failure. •

Note

- Few reason for failure are given below
  - 0x4E13 Remote user terminated connection
  - 0x4E14 Remote device terminated connection due to low resources
  - 0x4E15 Remote device terminated connection due to power off
  - 0x4E3D Connection terminated due to MIC failure
  - 0x4E3E Connection Failed to be Established
  - 0x4E60 Invalid Handle Range

Returns

- The following values are returned:
  - void

description

This callback function will be called if the disconnect status event is received from the module

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4614 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_le_ping_payload_timeout_t

```
void(* rsi_ble_on_le_ping_payload_timeout_t)(rsi_ble_event_le_ping_time_expired_t *rsi_ble_event_timeout_expired) )
(rsi_ble_event_le_ping_time_expired_t *rsi_ble_event_timeout_expired)
```

The callback function will be called if le ping payload timeout expired event is received.

Parameters

[out]	rsi_ble_event_disconnect	contains the disconnect status. Please refer rsi_ble_event_le_ping_time_expired_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the le ping time expired event is received from the module

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4627 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_le_ltk_req_event_t

```
void(* rsi_ble_on_le_ltk_req_event_t)(rsi_bt_event_le_ltk_request_t *rsi_ble_event_le_ltk_request) )
(rsi_bt_event_le_ltk_request_t *rsi_ble_event_le_ltk_request)
```

The callback function will be called if LE LTK request event is received.

Parameters

[out]	rsi_ble_event_le_ltk_request	contains the LTK request info. Please refer rsi_bt_event_le_ltk_request_s for more info

Returns

- The following values are returned:
  - void

description

This callback function will be called if LE LTK request event is received from the module

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4641 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_le_security_keys_t

```
void(* rsi_ble_on_le_security_keys_t)(rsi_bt_event_le_security_keys_t *rsi_ble_event_le_security_keys) )
(rsi_bt_event_le_security_keys_t *rsi_ble_event_le_security_keys)
```

The callback function will be called if LE security keys event is received.

Parameters

[out]	rsi_bt_event_le_security_keys_t	contains security keys. Please refer rsi_bt_event_le_security_keys_s for more info

Returns

- The following values are returned:
  - void

description

This callback function will be called if LE security keys event is received from the module

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4654 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_smp_request_t

```
void(* rsi_ble_on_smp_request_t)(rsi_bt_event_smp_req_t *remote_dev_address) )(rsi_bt_event_smp_req_t
*remote_dev_address)
```

The callback function will be called if smp request is received in central mode.

Parameters

[out]	resp_status	contains the response status (Success or Error code)

[out]	remote_dev_address	contains the smp requested device address. Please refer rsi_bt_event_smp_req_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the smp request is received from the remote device

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4675 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_smp_response_t

```
void(* rsi_ble_on_smp_response_t)(rsi_bt_event_smp_resp_t *remote_dev_address) )(rsi_bt_event_smp_resp_t *remote_dev_address)
```

The callback function will be called if smp request is received in peripheral mode.

Parameters

[out]	remote_dev_address	contains the smp resp information. please refer rsi_bt_event_smp_resp_s for more info

Returns

- The following values are returned:
  - void

description

This callback function will be called if the smp request is received from the remote device

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4689 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_smp_passkey_t

```
void(* rsi_ble_on_smp_passkey_t)(rsi_bt_event_smp_passkey_t *remote_dev_address) )(rsi_bt_event_smp_passkey_t *remote_dev_address)
```

The callback function will be called if smp passkey event is received from module.

Parameters

[out]	resp_statuscontains	the response status (Success or Error code)
[out]	remote_dev_address	contains the remote device address. please refer rsi_bt_event_smp_passkey_s for more info

Returns

- The following values are returned:
  - void

description

This callback function will be called if the smp passkey is received from the module

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4704 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_smp_passkey_display_t

```
void(* rsi_ble_on_smp_passkey_display_t)(rsi_bt_event_smp_passkey_display_t *smp_passkey_display) )
(rsi_bt_event_smp_passkey_display_t *smp_passkey_display)
```

The callback function will be called if smp passkey event is received from module.

Parameters

[out]	resp_status	contains the response status (Success or Error code)
[out]	smp_passkey_display	contains the smp passkey display information. Please refer rsi_bt_event_smp_passkey_display_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the smp passkey display is received from the module

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4719 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_smp_failed_t

```
void(* rsi_ble_on_smp_failed_t)(uint16_t resp_status, rsi_bt_event_smp_failed_t *remote_dev_address) )(uint16_t resp_status,
rsi_bt_event_smp_failed_t *remote_dev_address)
```

The callback function will be called if smp failed event is received from module.

Parameters

[out]	resp_status	contains the remote device address. Please refer rsi_bt_event_smp_failed_s for more info.

Note

- Error codes for SMP FAILED are given below
  - 0x4B01 SMP Passkey entry failed
  - 0x4B02 SMP OOB not available
  - 0x4B03 SMP Authentication Requirements
  - 0x4B04 SMP confirm value failed
  - 0x4B05 SMP Pairing not supported
  - 0x4B06 SMP Encryption key size insufficient
  - 0x4B07 SMP command not supported
  - 0x4B08 SMP Unspecified Reason
  - 0x4B09 SMP repeated attempts
  - 0x4B0C SMP Numeric Comparison Failed
  - 0x4B0B DHKEY Check Failed

Returns

- The following values are returned:
  - void

description

This callback function will be called if the smp process is failed with remote device

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4758 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_sc_method_t

```
void(* rsi_ble_on_sc_method_t)(rsi_bt_event_sc_method_t *scmethod) )(rsi_bt_event_sc_method_t *scmethod)
```

The callback function will be called if security method event is received from module.

Parameters

[out]	scmethod	contains Security Method 1 means Just works or 2 means Passkey. Please refer rsi_bt_event_sc_method_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the SC method is done with remote device

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4772 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_encrypt_started_t

```
void(* rsi_ble_on_encrypt_started_t)(uint16_t resp_status, rsi_bt_event_encryption_enabled_t *enc_enabled) )(uint16_t resp_status, rsi_bt_event_encryption_enabled_t *enc_enabled)
```

The callback function will be called if encrypted event is received from module.

Parameters

[out]	resp_status	contains the response status (Success or Error code)
[out]	enc_enabled	contains encryption information. Please refer rsi_bt_event_encryption_enabled_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the encryption process is started with remote device

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4787 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_sc_passkey_t

```
void(* rsi_ble_on_sc_passkey_t)(rsi_bt_event_sc_passkey_t *sc_passkey) )(rsi_bt_event_sc_passkey_t *sc_passkey)
```

The callback function will be called if BLE Secure connection passkey event received from module.

Parameters

[out]	sc_passkey	contains LE SC Passkey information. Please refer rsi_bt_event_encryption_enabled_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the BLE Secure connection passkey event received

- This callback has to be registered using rsi_ble_smp_register_callbacks API

Definition at line 4801 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_phy_update_complete_t

```
void(* rsi_ble_on_phy_update_complete_t)(rsi_ble_event_phy_update_t *rsi_ble_event_phy_update_complete) )
(rsi_ble_event_phy_update_t *rsi_ble_event_phy_update_complete)
```

The callback function will be called when PHY update complete event is received.

Parameters

[out]	rsi_ble_event_phy_update_complete	contains the controller support PHY information. Please refer rsi_ble_event_phy_update_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called when PHY update complete event is received

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4814 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_conn_update_complete_t

```
void(* rsi_ble_on_conn_update_complete_t)(rsi_ble_event_conn_update_t *rsi_ble_event_conn_update_complete, uint16_t
resp_status) )(rsi_ble_event_conn_update_t *rsi_ble_event_conn_update_complete, uint16_t resp_status)
```

The callback function will be called when conn update complete event is received.

Parameters

[out]	rsi_ble_event_conn_update_complete	contains the controller support conn information. Please refer rsi_ble_event_conn_update_s for more info.
[out]	resp_status	contains the response status (Success or Error code)

Returns

- The following values are returned:
  - void

description

This callback function will be called when conn update complete event is received

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4829 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_remote_conn_params_request_t

```
void(* rsi_ble_on_remote_conn_params_request_t)(rsi_ble_event_remote_conn_param_req_t
*rsi_ble_event_remote_conn_param, uint16_t resp_status) )(rsi_ble_event_remote_conn_param_req_t
*rsi_ble_event_remote_conn_param, uint16_t resp_status)
```

The callback function will be called if remote conn params request is received.

Parameters

[out]	resp_status	contains the response status (Success or Error code)
[out]	rsi_ble_event_remote_features	contains the remote device supported features. Please refer rsi_ble_event_remote_features_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called when remote conn params request is received

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4845 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_remote_features_t

```
void(* rsi_ble_on_remote_features_t)(rsi_ble_event_remote_features_t *rsi_ble_event_remote_features) )
(rsi_ble_event_remote_features_t *rsi_ble_event_remote_features)
```

Callback function to peer device supported features.

Parameters

[out]	rsi_ble_event_remote_features	contains the remote device supported features. Please refer rsi_ble_event_remote_features_s for more info.

Returns

- The following values are returned:

void

description

This callback function will be called when conn update complete event is received

- This callback has to be registered using rsi_ble_gap_extended_register_callbacks API

Definition at line 4860 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_le_more_data_req_t

```
void(* rsi_ble_on_le_more_data_req_t)(rsi_ble_event_le_dev_buf_ind_t *rsi_ble_more_data_evt) )
(rsi_ble_event_le_dev_buf_ind_t *rsi_ble_more_data_evt)
```

Callback function to LE more data request.

Parameters

[out]	rsi_ble_more_data_evt	contains the LE Device Buffer Indication information. Please refer rsi_ble_event_le_dev_buf_ind_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called when LE more data event is received

- This callback has to be registered using rsi_ble_gap_extended_register_callbacks API

Definition at line 4873 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_data_length_update_t

```
void(* rsi_ble_on_data_length_update_t)(rsi_ble_event_data_length_update_t *remote_dev_address) )
(rsi_ble_event_data_length_update_t *remote_dev_address)
```

This event is raised when data length is update in controller.

Parameters

[out]	remote_dev_address	contains the controller support TX and RX length information. Please refer rsi_ble_event_data_length_update_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called when data length update complete event is received

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line 4887 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_directed_adv_report_event_t

```
void(* rsi_ble_on_directed_adv_report_event_t)(rsi_ble_event_directedadv_report_t *rsi_ble_event_directed) )
(rsi_ble_event_directedadv_report_t *rsi_ble_event_directed)
```

The callback function will be called if directed advertise report event is received.

Parameters

[in]	rsi_ble_event_directedcontains	the advertise report information

Returns

- The following values are returned:
  - void

description

This callback function will be called if the advertise event report is received from the module

- This callback has to be registered using rsi_ble_gap_register_callbacks API

Definition at line `4900` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_on_gatt_error_resp_t

```
void(* rsi_ble_on_gatt_error_resp_t)(uint16_t event_status, rsi_ble_event_error_resp_t *rsi_ble_gatt_error) )(uint16_t
event_status, rsi_ble_event_error_resp_t *rsi_ble_gatt_error)
```

The callback function will be called if GATT error event is received.

Parameters

[out]	event_status	contains the GATT error information. Please refer rsi_ble_event_error_resp_s for more info

Note

- Attribute protocol error codes
  - 0x4A01 - Invalid Handle
  - 0x4A06 - Request not supported
  - 0x4A0A - Attribute not found
  - 0x4A05 - Insufficient authentication
  - 0x4A08 - Insufficient authorization
  - 0x4A0C - Insufficient encryption key size
  - 0x4A0F - Insufficient encryption
  - 0x4A02 - Read not permitted
  - 0x4A03 - Write not permitted
  - 0x4A07 - Invalid offset
  - 0x4A0B - Attribute not Long
  -

Returns

- The following values are returned:
  - void

description

This callback function will be called if the GATT error event is received from the module

This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5034 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_gatt_desc_val_event_t

```
void(* rsi_ble_on_gatt_desc_val_event_t)(uint16_t event_status, rsi_ble_event_gatt_desc_t *rsi_ble_gatt_desc_val) )(uint16_t event_status, rsi_ble_event_gatt_desc_t *rsi_ble_gatt_desc_val)
```

The callback function will be called if attribute descriptors event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_gatt_desc_val	contains the profiles list event information. Please refer rsi_ble_event_gatt_desc_s for more info

Returns

- The following values are returned:
  - void

description

This callback function will be called if the attribute descriptors event is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5053 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_profiles_list_t

```
void(* rsi_ble_on_event_profiles_list_t)(uint16_t event_status, rsi_ble_event_profiles_list_t *rsi_ble_event_profiles) )(uint16_t event_status, rsi_ble_event_profiles_list_t *rsi_ble_event_profiles)
```

The callback function will be called if profiles list event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_event_profiles	contains the profiles list event information. Please refer rsi_ble_event_profiles_list_s for more info

Returns

- The following values are returned:
  - void

description

This callback function will be called if the profiles list response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5074 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_profile_by_uuid_t

```
void(* rsi_ble_on_event_profile_by_uuid_t)(uint16_t event_status, rsi_ble_event_profile_by_uuid_t *rsi_ble_event_profile) )
(uint16_t event_status, rsi_ble_event_profile_by_uuid_t *rsi_ble_event_profile)
```

The callback function will be called if profile event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_event_profile	contains the profile response information. Please refer rsi_ble_event_profile_by_uuid_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the profile response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5095 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_read_by_char_services_t

```
void(* rsi_ble_on_event_read_by_char_services_t)(uint16_t event_status, rsi_ble_event_read_by_type1_t
*rsi_ble_event_read_type1) )(uint16_t event_status, rsi_ble_event_read_by_type1_t *rsi_ble_event_read_type1)
```

The callback function will be called if characteristic services list event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_event_read_type1	contains the char services event information. Please refer rsi_ble_event_read_by_type1_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the characteristic services list response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5115 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_read_by_inc_services_t

```
void(* rsi_ble_on_event_read_by_inc_services_t)(uint16_t event_status, rsi_ble_event_read_by_type2_t
*rsi_ble_event_read_type2) )(uint16_t event_status, rsi_ble_event_read_by_type2_t *rsi_ble_event_read_type2)
```

The callback function will be called if include services list event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_event_read_type2	contains the inc services information. Please refer rsi_ble_event_read_by_type2_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the include services list response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5136 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_read_att_value_t

```
void(* rsi_ble_on_event_read_att_value_t)(uint16_t event_status, rsi_ble_event_read_by_type3_t
*rsi_ble_event_read_type3) )(uint16_t event_status, rsi_ble_event_read_by_type3_t *rsi_ble_event_read_type3)
```

The callback function will be called if attribute value event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_event_read_type3	contains the char services event information. Please refer rsi_ble_event_read_by_type3_s for more info.

Returns

The following values are returned:
- void

description

This callback function will be called if the attribute value response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5156 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_read_resp_t

```
void(* rsi_ble_on_event_read_resp_t)(uint16_t event_status, rsi_ble_event_att_value_t *rsi_ble_event_att_val) )(uint16_t event_status, rsi_ble_event_att_value_t *rsi_ble_event_att_val)
```

The callback function will be called if attribute value event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_event_att_val	contains the profile response information. Please refer rsi_ble_event_att_value_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the attribute value is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5176 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_write_resp_t

```
void(* rsi_ble_on_event_write_resp_t)(uint16_t event_status, rsi_ble_set_att_resp_t *rsi_ble_event_set_att_rsp) )(uint16_t event_status, rsi_ble_set_att_resp_t *rsi_ble_event_set_att_rsp)
```

The callback function will be called if GATT write event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_event_set_att_rsp	contains the profile response information. Please refer rsi_ble_set_att_resp_t for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the GATT write response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5195 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_indicate_confirmation_t

```
void(* rsi_ble_on_event_indicate_confirmation_t)(uint16_t event_status, rsi_ble_set_att_resp_t *rsi_ble_event_set_att_rsp) )
(uint16_t event_status, rsi_ble_set_att_resp_t *rsi_ble_event_set_att_rsp)
```

Callback function to be called if indication confirmation event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •
[out]	rsi_ble_event_set_att_rsp	contains the profile response information. Please refer rsi_ble_set_att_resp_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the indication confirmation response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5214 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_event_prepare_write_resp_t

```
void(* rsi_ble_on_event_prepare_write_resp_t)(uint16_t event_status, rsi_ble_prepare_write_resp_t
*rsi_ble_event_prepare_write) )(uint16_t event_status, rsi_ble_prepare_write_resp_t *rsi_ble_event_prepare_write)
```

The callback function will be called if indication confirmation event is received.

Parameters

[out]	event_status	contains the response status • 0 - Success • Non-Zero Value - Failure •

[out]	rsi_ble_event_prepare_write	contains the char services event information. Please refer rsi_ble_prepare_write_resp_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the GATT prepare response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5234 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_profiles_list_resp_t

```
void(* rsi_ble_on_profiles_list_resp_t)(uint16_t resp_status, rsi_ble_resp_profiles_list_t *rsi_ble_resp_profiles) )(uint16_t resp_status, rsi_ble_resp_profiles_list_t *rsi_ble_resp_profiles)
```

The callback function will be called if profiles list response is received.

Parameters

[out]	resp_status	contains the profiles list response information. Please refer rsi_ble_resp_profiles_list_s for more info.

Note

- Attribute protocol error codes
  - 0x4A01 - Invalid Handle
  - 0x4A0A - Attribute not found
  -

Returns

- The following values are returned:
  - void

description

This callback function will be called if the profiles list response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5261 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_profile_resp_t

```
void(* rsi_ble_on_profile_resp_t)(uint16_t resp_status, profile_descriptors_t *rsi_ble_resp_profile) )(uint16_t resp_status, profile_descriptors_t *rsi_ble_resp_profile)
```

The callback function will be called if profile response is received.

Parameters

[out]	resp_status	contains the profile response information. Please refer profile_descriptors_s for more info

Note

-

Attribute protocol error codes
- 0x4A01 - Invalid Handle
- 0x4A06 - Request not supported
- 0x4A0A - Attribute not found
-

Returns

- The following values are returned:
  - void

description

This callback function will be called if the profile response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5289 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_char_services_resp_t

> void(* rsi_ble_on_char_services_resp_t)(uint16_t resp_status, rsi_ble_resp_char_services_t *rsi_ble_resp_char_serv))(uint16_t resp_status, rsi_ble_resp_char_services_t *rsi_ble_resp_char_serv)

The callback function will be called if service characteristics response is received.

Parameters

[out]	resp_status	contains the service characteristics response information. Please refer rsi_ble_resp_char_services_s for more info

Note

- Attribute protocol error codes
  - 0x4A01 - Invalid Handle
  - 0x4A06 - Request not supported
  - 0x4A0A - Attribute not found
  - 0x4A05 - Insufficient authentication
  - 0x4A08 - Insufficient authorization
  - 0x4A0C - Insufficient encryption key size
  - 0x4A0F - Insufficient encryption
  - 0x4A02 - Read not permitted
  -

Returns

- The following values are returned:
  - void

description

This callback function will be called if the service characteristics response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5327 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_inc_services_resp_t

```
void(* rsi_ble_on_inc_services_resp_t)(uint16_t resp_status, rsi_ble_resp_inc_services_t *rsi_ble_resp_inc_serv) )(uint16_t resp_status, rsi_ble_resp_inc_services_t *rsi_ble_resp_inc_serv)
```

The callback function will be called if include services response is received.

Parameters

[out]	resp_status	contains the include services response information. Please refer rsi_ble_resp_inc_services_s for more info

Note

- Attribute protocol error codes
  - 0x4A01 - Invalid Handle
  - 0x4A06 - Request not supported
  - 0x4A0A - Attribute not found
  - 0x4A05 - Insufficient authentication
  - 0x4A08 - Insufficient authorization
  - 0x4A0C - Insufficient encryption key size
  - 0x4A0F - Insufficient encryption
  - 0x4A02 - Read not permitted
  -

Returns

- The following values are returned:
  - void

description

This callback function will be called if the include service response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5366 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_att_desc_resp_t

```
void(* rsi_ble_on_att_desc_resp_t)(uint16_t resp_status, rsi_ble_resp_att_descs_t *rsi_ble_resp_att_desc) )(uint16_t resp_status, rsi_ble_resp_att_descs_t *rsi_ble_resp_att_desc)
```

The callback function will be called if attribute descriptors response is received.

Parameters

[out]	resp_status	contains the attribute descriptors response information. Please refer rsi_ble_resp_att_descs_s for more info

Note

- Attribute protocol error codes
  - 0x4A01 - Invalid Handle
  - 0x4A0A - Attribute not found
  - 0x4A05 - Insufficient authentication
  - 0x4A08 - Insufficient authorization
  - 0x4A0C - Insufficient encryption key size
  - 0x4A0F - Insufficient encryption
  - 0x4A02 - Read not permitted
  -

Returns

- The following values are returned:
  - void

description

This callback function will be called if the attribute descriptors response is received from the module

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5402 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_read_resp_t

```
void(* rsi_ble_on_read_resp_t)(uint16_t resp_status, uint16_t resp_id, rsi_ble_resp_att_value_t *rsi_ble_resp_att_val) )(uint16_t resp_status, uint16_t resp_id, rsi_ble_resp_att_value_t *rsi_ble_resp_att_val)
```

The callback function will be called upon receiving the attribute value.

Parameters

[out]	resp_status	contains the response id because of which, this callback is called response ids: (RSI_BLE_RSP_READ_VAL, RSI_BLE_RSP_READ_BY_UUID, RSI_BLE_RSP_LONG_READ, RSI_BLE_RSP_MULTIPLE_READ)
[out]	rsi_ble_resp_att_val	contains the attribute value. Please refer rsi_ble_resp_att_value_s for more info

Note

- Attribute protocol error codes
  - 0x4A01 - Invalid Handle
  - 0x4A0A - Attribute not found
  - 0x4A05 - Insufficient authentication
  - 0x4A08 - Insufficient authorization
  - 0x4A0C - Insufficient encryption key size
  - 0x4A0F - Insufficient encryption
  - 0x4A02 - Read not permitted
  - 0x4A06 - Request not supported
  - 0x4A07 - Invalid offset
  - 0x4A0B - Attribute not Long
  -

Returns

- The following values are returned:
  - void

description

This callback function will be called upon receiving the attribute value

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5447 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_write_resp_t

```
void(* rsi_ble_on_write_resp_t)(uint16_t resp_status, uint16_t resp_id) )(uint16_t resp_status, uint16_t resp_id)
```

The callback function will be called if the attribute set/prepare/execute action is completed.

Parameters

[out]	resp_status	contains the response id because of which, this callback is called response ids: (RSI_BLE_RSP_WRITE, RSI_BLE_RSP_WRITE_NO_ACK, RSI_BLE_RSP_LONG_WRITE, RSI_BLE_RSP_EXECUTE_WRITE)

Note

- Attribute protocol error codes
  - 0x4A01 - Invalid Handle
  - 0x4A0A - Attribute not found
  - 0x4A05 - Insufficient authentication
  - 0x4A08 - Insufficient authorization
  - 0x4A0C - Insufficient encryption key size
  - 0x4A0F - Insufficient encryption
  - 0x4A03 - Write not permitted
  - 0x4A07 - Invalid offset
  - 0x4A0D - Invalid attribute value length
  -

Returns

- The following values are returned:
  - void

description

This callback function will be called if the attribute set/prepare/execute action is completed

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5489 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_gatt_write_event_t

```
void(* rsi_ble_on_gatt_write_event_t)(uint16_t event_id, rsi_ble_event_write_t *rsi_ble_write) )(uint16_t event_id,
rsi_ble_event_write_t *rsi_ble_write)
```

The callback function will be called if the GATT write/notify/indicate events are received.

Parameters

[out]	event_id	contains the gatt_write event id (RSI_BLE_EVENT_GATT_WRITE) •
[out]	rsi_ble_write	contains the GATT event information. Please refer rsi_ble_event_write_s for more info

Returns

- The following values are returned:
  - void

description

This callback function will be called if the GATT write/notify/indicate events are received

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5511 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_gatt_prepare_write_event_t

```
void(* rsi_ble_on_gatt_prepare_write_event_t)(uint16_t event_id, rsi_ble_event_prepare_write_t *rsi_ble_write) )(uint16_t event_id, rsi_ble_event_prepare_write_t *rsi_ble_write)
```

The callback function will be called if the GATT prepare events are received.

Parameters

[out]	event_id	contains the gatt_prepare_write event id (RSI_BLE_EVENT_PREPARE_WRITE) •
[out]	rsi_ble_write	contains the GATT prepare event information. Please refer rsi_ble_event_prepare_write_s for more info

Returns

- The following values are returned:
  - void

description

This callback function will be called if the GATT prepare event is received

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5526 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_execute_write_event_t

```
void(* rsi_ble_on_execute_write_event_t)(uint16_t event_id, rsi_ble_execute_write_t *rsi_ble_execute_write) )(uint16_t event_id, rsi_ble_execute_write_t *rsi_ble_execute_write)
```

The callback function will be called if the GATT execute events are received.

Parameters

[out]	event_id	contains the gatt_execute_write event id (RSI_BLE_EVENT_EXECUTE_WRITE) •
[out]	rsi_ble_write	contains the GATT event information. Please refer rsi_ble_execute_write_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the GATT execute event is received

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5541 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_read_req_event_t

```
void(* rsi_ble_on_read_req_event_t)(uint16_t event_id, rsi_ble_read_req_t *rsi_ble_read_req) )(uint16_t event_id,
rsi_ble_read_req_t *rsi_ble_read_req)
```

The callback function will be called if the GATT read request events are received.

Parameters

[out]	event_id	contains the gatt_read_req_event id (RSI_BLE_EVENT_READ_REQ) •
[out]	rsi_ble_write	contains the GATT event information. Please refer rsi_ble_read_req_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called if the GATT read request event is received

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5556 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_mtu_event_t

```
void(* rsi_ble_on_mtu_event_t)(rsi_ble_event_mtu_t *rsi_ble_event_mtu) )(rsi_ble_event_mtu_t *rsi_ble_event_mtu)
```

The callback function will be called if MTU size request is received.

Parameters

[out]	rsi_ble_event_mtu	contains the MTU size information. Please refer rsi_ble_event_mtu_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called when connected to indicate MTU size

- This callback has to be registered using rsi_ble_gatt_register_callbacks API

Definition at line 5569 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_mtu_exchange_info_t

```
void(* rsi_ble_on_mtu_exchange_info_t)(rsi_ble_event_mtu_exchange_information_t *rsi_ble_event_mtu_exchange_info) )
(rsi_ble_event_mtu_exchange_information_t *rsi_ble_event_mtu_exchange_info)
```

Callback function to indicate MTU size and who initiated MTU Exchange Request.

Parameters

[out]	rsi_ble_event_mtu_exchange_info	contains the MTU exchange information. Please refer rsi_ble_event_mtu_exchange_information_s for more info.

Returns

- The following values are returned:
  - void

description

This callback function will be called when connected, this event will contain MTU Exchange Information

- This callback has to be registered using rsi_ble_gatt_extended_register_callbacks API

Definition at line 5582 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_remote_device_info_t

```
typedef void(* rsi_ble_on_remote_device_info_t) (uint16_t status, rsi_ble_event_remote_device_info_t *resp_buffer) )
(uint16_t status, rsi_ble_event_remote_device_info_t *resp_buffer)
```

@callback rsi_ble_on_remote_device_info_t

Parameters

[out]	resp_statuscontains	the remote device version information.

Callback function to peer device information. **Note**

- Refer Bluetooth Generic Error Codes section up to 0x4FF8 from error-codes.

Returns

- The following values are returned:
  - void

description

This callback function will be called when conn update complete event is received This callback has to be registered using rsi_ble_enhanced_gap_extended_register_callbacks API

Definition at line 5601 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_on_rcp_resp_rcvd_t

```
typedef void(* rsi_ble_on_rcp_resp_rcvd_t) (uint16_t status, rsi_ble_event_rcp_rcvd_info_t *resp_buffer) )(uint16_t status,
rsi_ble_event_rcp_rcvd_info_t *resp_buffer)
```

Definition at line 5602 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Data Structures

Data Structures

This section provides a reference to Bluetooth Low Energy (BLE) API data types.

Modules

rsi_ble_gatt_remove_att_s

rsi_ble_vendor_rf_type_s

rsi_ble_mtu_exchange_s

rsi_ble_mtu_exchange_resp_s

rsi_ble_ae_get_supported_no_of_adv_sets_s

rsi_ble_ae_read_supported_max_adv_data_s

rsi_ble_ae_set_random_address_s

ae_adv_params_s

rsi_ble_ae_data_s

rsi_ble_ae_adv_enabel_s

rsi_ble_ae_adv_set_clear_or_remove_s

ae_periodic_adv_params

ae_periodic_adv_enable

ae_scan_params_s

rsi_ble_ae_set_scan_params_s

rsi_ble_ae_set_scan_enable_s

rsi_ble_ae_set_periodic_adv_create_sync_s

rsi_ble_ae_set_periodic_adv_terminate_sync_s

rsi_ble_ae_set_periodic_sync_s

rsi_ble_ae_dev_to_periodic_list_s

rsi_ble_initiation_params_s

rsi_ble_ae_extended_create_connect_s

rsi_ble_tx_pwr_s

rsi_ble_query_rf_path_comp_s

rsi_ble_write_rf_path_comp_s

rsi_ble_ae_pdu

profile_descriptor_s

rsi_ble_req_add_att_s

Enumerations

enum rsi_ble_gap_extended_callbacks_s {

RSI_BLE_ON_REMOTE_DEVICE_INFORMATION = 1
RSI_BLE_ON_RCP_EVENT = 2

}

Typedefs

typedef struct rsi_ble_req_rand_s	rsi_ble_req_rand_t
typedef struct rsi_ble_req_adv_s	rsi_ble_req_adv_t
typedef struct rsi_ble_req_adv_data_s	rsi_ble_req_adv_data_t
typedef struct rsi_ble_req_acceptlist_using_payload_s	rsi_ble_req_acceptlist_using_payload_t
typedef struct rsi_ble_set_ble_tx_power_s	rsi_ble_set_ble_tx_power_t
typedef struct rsi_ble_req_scanrsp_data_s	rsi_ble_req_scanrsp_data_t
typedef struct rsi_ble_req_scan_s	rsi_ble_req_scan_t
typedef struct rsi_ble_encrypt_s	rsi_ble_encrypt_t
typedef struct rsi_data_packet_s	rsi_data_packet_t
typedef struct rsi_ble_accept_list_s	rsi_ble_accept_list_t
typedef struct rsi_ble_req_conn_s	rsi_ble_req_conn_t
typedef struct rsi_ble_req_enhance_conn_s	rsi_ble_req_enhance_conn_t
typedef struct rsi_ble_req_disconnect_s	rsi_ble_req_disconnect_t
typedef struct rsi_ble_start_encryption_s	rsi_ble_strat_encryption_t
typedef struct rsi_ble_req_smp_pair_s	rsi_ble_req_smp_pair_t
typedef struct rsi_ble_smp_response_s	rsi_ble_smp_response_t
typedef struct rsi_ble_smp_passkey_s	rsi_ble_smp_passkey_t

typedef struct rsi_ble_get_le_ping_timeout_s	rsi_ble_get_le_ping_timeout_t
typedef struct rsi_ble_rsp_get_le_ping_timeout_s	rsi_ble_rsp_get_le_ping_timeout_t
typedef struct rsi_ble_set_le_ping_timeout_s	rsi_ble_set_le_ping_timeout_t
typedef struct rsi_ble_resolvlist_s	rsi_ble_resolvlist_t
typedef struct rsi_ble_get_resolving_list_size_s	rsi_ble_get_resolving_list_size_t
typedef struct rsi_ble_set_addr_resolution_enable_s	rsi_ble_set_addr_resolution_enable_t
typedef struct rsi_ble_cmd_conn_params_update_s	rsi_ble_cmd_conn_params_update_t
typedef struct rsi_ble_req_read_phy_s	rsi_ble_req_read_phy_t
typedef struct rsi_ble_set_phy_s	rsi_ble_set_phy_t
typedef struct rsi_ble_setdatalength_s	rsi_ble_setdatalength_t
typedef struct rsi_ble_set_privacy_mode_s	rsi_ble_set_privacy_mode_t
typedef struct rsi_ble_cbfc_conn_req_s	rsi_ble_cbfc_conn_req_t
typedef struct rsi_ble_tx_test_mode_s	rsi_ble_tx_test_mode_t
typedef struct rsi_ble_end_test_mode_s	rsi_ble_end_test_mode_t
typedef struct rsi_ble_set_le_ltkreqreply_s	rsi_ble_set_le_ltkreqreply_t
typedef struct rsi_ble_req_smp_pair_failed_s	rsi_ble_req_smp_pair_failed_t

typedef struct rsi_ble_req_profiles_list_s	rsi_ble_req_profiles_list_t
typedef struct rsi_ble_req_profile_s	rsi_ble_req_profile_t
typedef struct rsi_ble_req_char_services_s	rsi_ble_req_char_services_t
typedef struct rsi_ble_req_inc_services_s	rsi_ble_req_inc_services_t
typedef struct rsi_ble_req_char_val_by_uuid_s	rsi_ble_req_char_val_by_uuid_t
typedef struct rsi_ble_req_att_descs_s	rsi_ble_req_att_descs_t
typedef struct rsi_ble_req_att_value_s	rsi_ble_req_att_value_t
typedef struct rsi_ble_req_multiple_att_val_s	rsi_ble_req_multi_att_values_t
typedef struct rsi_ble_req_long_att_value_s	rsi_ble_req_long_att_value_t
typedef struct rsi_ble_set_att_val_s	rsi_ble_set_att_value_t
typedef struct rsi_ble_set_att_cmd_s	rsi_ble_set_att_cmd_t
typedef struct rsi_ble_set_long_att_val_s	rsi_ble_set_long_att_value_t
typedef struct rsi_ble_req_prepare_write_s	rsi_ble_req_prepare_write_t
typedef struct rsi_ble_req_execute_write_s	rsi_ble_req_execute_write_t
typedef struct rsi_ble_cmd_conn_param_resp	rsi_ble_cmd_conn_param_resp_t
typedef struct rsi_ble_req_add_serv_s	rsi_ble_req_add_serv_t

typedef struct rsi_ble_set_local_att_value_s	rsi_ble_set_local_att_value_t
typedef struct rsi_ble_notify_att_value_s	rsi_ble_notify_att_value_t
typedef struct rsi_ble_set_wo_resp_notify_buf_info_s	rsi_ble_set_wo_resp_notify_buf_info_t
typedef struct rsi_ble_indicate_confirm_s	rsi_ble_indicate_confirm_t
typedef struct rsi_ble_get_local_att_value_s	rsi_ble_get_local_att_value_t
typedef struct rsi_ble_gatt_read_response_s	rsi_ble_gatt_read_response_t
typedef struct rsi_ble_gatt_write_response_s	rsi_ble_gatt_write_response_t
typedef struct rsi_ble_gatt_prepare_write_response_s	rsi_ble_gatt_prepare_write_response_t
typedef struct rsi_ble_set_local_irk_s	rsi_ble_set_local_irk_t
typedef enum rsi_ble_gap_extended_callbacks_s	rsi_ble_gap_extended_callbacks_t
typedef struct rsi_ble_att_error_response_s	rsi_ble_att_error_response_t
typedef struct rsi_ble_gatt_remove_serv_s	rsi_ble_gatt_remove_serv_t
typedef struct rsi_ble_gatt_remove_att_s	rsi_ble_gatt_remove_att_t
typedef struct rsi_ble_vendor_rf_type_s	rsi_ble_vendor_rf_type_t
typedef struct rsi_ble_mtu_exchange_s	rsi_ble_mtu_exchange_t
typedef struct rsi_ble_mtu_exchange_resp_s	rsi_ble_mtu_exchange_resp_t

typedef struct profile_descriptor_s	profile_descriptors_t
typedef struct rsi_ble_req_add_att_s	rsi_ble_req_add_att_t

Functions

struct rsi_ble_ae_get_supported_no_of_adv_sets_s	__attribute__((__packed__)) rsi_ble_ae_adv_params_t AE Advertising Params.

Macros

#define	BLE_PROTOCOL 0×01
#define	PROP_PROTOCOL 0×02
#define	ADV_ROLE 0×01
#define	SCAN_AND_CENTRAL_ROLE 0×02
#define	PERIPHERAL_ROLE 0×03
#define	CONN_ROLE 0×04
#define	RSI_BLE_ATT_EXCHANGE_MTU_REQUEST 0×02
#define	RSI_BLE_ATT_FIND_INFORMATION_REQUEST 0×04
#define	RSI_BLE_ATT_FIND_BY_TYPE_VALUE_REQUEST 0×06
#define	RSI_BLE_ATT_READ_BY_TYPE_REQUEST 0×08
#define	RSI_BLE_ATT_READ_REQUEST 0×0A
#define	RSI_BLE_ATT_READ_BLOB_REQUEST 0×0C
#define	RSI_BLE_ATT_READ_MULTIPLE_REQUEST 0×0E
#define	RSI_BLE_ATT_READ_BY_GROUP_TYPE_REQUEST 0×10
#define	RSI_BLE_ATT_WRITE_REQUEST 0×12
#define	RSI_BLE_ATT_PREPARE_WRITE_REQUEST 0×16
#define	RSI_BLE_ATT_EXECUTE_WRITE_REQUEST 0×18
#define	SUPPORTED_SCNNING_PHYS 2

Variables

struct rsi_ble_cb_s	__attribute__ AE Advertising Params.

Enumeration Documentation

rsi_ble_gap_extended_callbacks_s

> rsi_ble_gap_extended_callbacks_s

Enumerator	
RSI_BLE_ON_REMOTE_DEVICE_INFORMATION	
RSI_BLE_ON_RCP_EVENT	

Definition at line `1100` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

Typedef Documentation

rsi_ble_req_rand_t

> typedef struct rsi_ble_req_rand_s rsi_ble_req_rand_t

Definition at line `352` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_adv_t

> typedef struct rsi_ble_req_adv_s rsi_ble_req_adv_t

Definition at line `461` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_adv_data_t

> typedef struct rsi_ble_req_adv_data_s rsi_ble_req_adv_data_t

Definition at line `469` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_acceptlist_using_payload_t

> typedef struct rsi_ble_req_acceptlist_using_payload_s rsi_ble_req_acceptlist_using_payload_t

Definition at line `479` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_set_ble_tx_power_t

> typedef struct rsi_ble_set_ble_tx_power_s rsi_ble_set_ble_tx_power_t

Definition at line `496` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_scanrsp_data_t

```
typedef struct rsi_ble_req_scanrsp_data_s rsi_ble_req_scanrsp_data_t
```

Definition at line 504 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_scan_t

```
typedef struct rsi_ble_req_scan_s rsi_ble_req_scan_t
```

Definition at line 571 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_encrypt_t

```
typedef struct rsi_ble_encrypt_s rsi_ble_encrypt_t
```

Definition at line 580 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_data_packet_t

```
typedef struct rsi_data_packet_s rsi_data_packet_t
```

Definition at line 584 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_accept_list_t

```
typedef struct rsi_ble_accept_list_s rsi_ble_accept_list_t
```

Definition at line 595 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_conn_t

```
typedef struct rsi_ble_req_conn_s rsi_ble_req_conn_t
```

Definition at line 615 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_enhance_conn_t

```
typedef struct rsi_ble_req_enhance_conn_s rsi_ble_req_enhance_conn_t
```

Definition at line 642 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_disconnect_t

```
typedef struct rsi_ble_req_disconnect_s rsi_ble_req_disconnect_t
```

Definition at line 652 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_strat_encryption_t

```
typedef struct rsi_ble_start_encryption_s rsi_ble_strat_encryption_t
```

Definition at line 666 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_smp_pair_t

```
typedef struct rsi_ble_req_smp_pair_s rsi_ble_req_smp_pair_t
```

Definition at line 673 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_smp_response_t

```
typedef struct rsi_ble_smp_response_s rsi_ble_smp_response_t
```

Definition at line 680 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_smp_passkey_t

```
typedef struct rsi_ble_smp_passkey_s rsi_ble_smp_passkey_t
```

Definition at line 687 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_get_le_ping_timeout_t

```
typedef struct rsi_ble_get_le_ping_timeout_s rsi_ble_get_le_ping_timeout_t
```

Definition at line 692 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_rsp_get_le_ping_timeout_t

```
typedef struct rsi_ble_rsp_get_le_ping_timeout_s rsi_ble_rsp_get_le_ping_timeout_t
```

Definition at line 698 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_le_ping_timeout_t

```
typedef struct rsi_ble_set_le_ping_timeout_s rsi_ble_set_le_ping_timeout_t
```

Definition at line 704 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_resolvlist_t

```
typedef struct rsi_ble_resolvlist_s rsi_ble_resolvlist_t
```

Definition at line 713 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_get_resolving_list_size_t

```
typedef struct rsi_ble_get_resolving_list_size_s rsi_ble_get_resolving_list_size_t
```

Definition at line 719 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_addr_resolution_enable_t

```
typedef struct rsi_ble_set_addr_resolution_enable_s rsi_ble_set_addr_resolution_enable_t
```

Definition at line 727 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_cmd_conn_params_update_t

```
typedef struct rsi_ble_cmd_conn_params_update_s rsi_ble_cmd_conn_params_update_t
```

Definition at line 736 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_read_phy_t

```
typedef struct rsi_ble_req_read_phy_s rsi_ble_req_read_phy_t
```

Definition at line 741 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_phy_t

```
typedef struct rsi_ble_set_phy_s rsi_ble_set_phy_t
```

Definition at line 751 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_setdatalength_t

```
typedef struct rsi_ble_setdatalength_s rsi_ble_setdatalength_t
```

Definition at line 758 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_privacy_mode_t

```
typedef struct rsi_ble_set_privacy_mode_s rsi_ble_set_privacy_mode_t
```

Definition at line 765 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_cbfc_conn_req_t

```
typedef struct rsi_ble_cbfc_conn_req_s rsi_ble_cbfc_conn_req_t
```

Definition at line 771 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_tx_test_mode_t

```
typedef struct rsi_ble_tx_test_mode_s rsi_ble_tx_test_mode_t
```

Definition at line 810 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_end_test_mode_t

```
typedef struct rsi_ble_end_test_mode_s rsi_ble_end_test_mode_t
```

Definition at line 815 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_le_ltkreqreply_t

```
typedef struct rsi_ble_set_le_ltkreqreply_s rsi_ble_set_le_ltkreqreply_t
```

Definition at line 821 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_smp_pair_failed_t

```
typedef struct rsi_ble_req_smp_pair_failed_s rsi_ble_req_smp_pair_failed_t
```

Definition at line 827 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_profiles_list_t

```
typedef struct rsi_ble_req_profiles_list_s rsi_ble_req_profiles_list_t
```

Definition at line `839` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_profile_t

```
typedef struct rsi_ble_req_profile_s rsi_ble_req_profile_t
```

Definition at line `849` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_char_services_t

```
typedef struct rsi_ble_req_char_services_s rsi_ble_req_char_services_t
```

Definition at line `862` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_inc_services_t

```
typedef struct rsi_ble_req_inc_services_s rsi_ble_req_inc_services_t
```

Definition at line `872` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_char_val_by_uuid_t

```
typedef struct rsi_ble_req_char_val_by_uuid_s rsi_ble_req_char_val_by_uuid_t
```

Definition at line `886` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_att_descs_t

```
typedef struct rsi_ble_req_att_descs_s rsi_ble_req_att_descs_t
```

Definition at line `899` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_att_value_t

```
typedef struct rsi_ble_req_att_value_s rsi_ble_req_att_value_t
```

Definition at line `907` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_multi_att_values_t

```
typedef struct rsi_ble_req_multiple_att_val_s rsi_ble_req_multi_att_values_t
```

Definition at line 920 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_long_att_value_t

```
typedef struct rsi_ble_req_long_att_value_s rsi_ble_req_long_att_value_t
```

Definition at line 933 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_att_value_t

```
typedef struct rsi_ble_set_att_val_s rsi_ble_set_att_value_t
```

Definition at line 948 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_att_cmd_t

```
typedef struct rsi_ble_set_att_cmd_s rsi_ble_set_att_cmd_t
```

Definition at line 960 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_long_att_value_t

```
typedef struct rsi_ble_set_long_att_val_s rsi_ble_set_long_att_value_t
```

Definition at line 974 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_prepare_write_t

```
typedef struct rsi_ble_req_prepare_write_s rsi_ble_req_prepare_write_t
```

Definition at line 988 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_execute_write_t

```
typedef struct rsi_ble_req_execute_write_s rsi_ble_req_execute_write_t
```

Definition at line 996 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_cmd_conn_param_resp_t

```
typedef struct rsi_ble_cmd_conn_param_resp rsi_ble_cmd_conn_param_resp_t
```

Definition at line 1004 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_add_serv_t

```
typedef struct rsi_ble_req_add_serv_s rsi_ble_req_add_serv_t
```

Definition at line 1017 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_local_att_value_t

```
typedef struct rsi_ble_set_local_att_value_s rsi_ble_set_local_att_value_t
```

Definition at line 1027 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_notify_att_value_t

```
typedef struct rsi_ble_notify_att_value_s rsi_ble_notify_att_value_t
```

Definition at line 1038 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_wo_resp_notify_buf_info_t

```
typedef struct rsi_ble_set_wo_resp_notify_buf_info_s rsi_ble_set_wo_resp_notify_buf_info_t
```

Definition at line 1048 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_indicate_confirm_t

```
typedef struct rsi_ble_indicate_confirm_s rsi_ble_indicate_confirm_t
```

Definition at line 1053 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_get_local_att_value_t

```
typedef struct rsi_ble_get_local_att_value_s rsi_ble_get_local_att_value_t
```

Definition at line 1059 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_read_response_t

```
typedef struct rsi_ble_gatt_read_response_s rsi_ble_gatt_read_response_t
```

Definition at line 1073 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_write_response_t

```
typedef struct rsi_ble_gatt_write_response_s rsi_ble_gatt_write_response_t
```

Definition at line 1081 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_prepare_write_response_t

```
typedef struct rsi_ble_gatt_prepare_write_response_s rsi_ble_gatt_prepare_write_response_t
```

Definition at line 1091 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_local_irk_t

```
typedef struct rsi_ble_set_local_irk_s rsi_ble_set_local_irk_t
```

Definition at line 1097 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gap_extended_callbacks_t

```
typedef enum rsi_ble_gap_extended_callbacks_s rsi_ble_gap_extended_callbacks_t
```

Definition at line 1104 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_att_error_response_t

```
typedef struct rsi_ble_att_error_response_s rsi_ble_att_error_response_t
```

Definition at line 1125 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_remove_serv_t

```
typedef struct rsi_ble_gatt_remove_serv_s rsi_ble_gatt_remove_serv_t
```

Definition at line 1130 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_remove_att_t

```
typedef struct rsi_ble_gatt_remove_att_s rsi_ble_gatt_remove_att_t
```

Definition at line 1136 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_vendor_rf_type_t

```
typedef struct rsi_ble_vendor_rf_type_s rsi_ble_vendor_rf_type_t
```

Definition at line 1142 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_mtu_exchange_t

```
typedef struct rsi_ble_mtu_exchange_s rsi_ble_mtu_exchange_t
```

Definition at line 1148 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_mtu_exchange_resp_t

```
typedef struct rsi_ble_mtu_exchange_resp_s rsi_ble_mtu_exchange_resp_t
```

Definition at line 1154 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

profile_descriptors_t

```
typedef struct profile_descriptor_s profile_descriptors_t
```

Definition at line 441 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_req_add_att_t

```
typedef struct rsi_ble_req_add_att_s rsi_ble_req_add_att_t
```

Definition at line 513 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Function Documentation

__attribute__

```
struct rsi_ble_ae_get_supported_no_of_adv_sets_s __attribute__ ((__packed__)) rsi_ble_ae_adv_params_t
```

AE Advertising Params.

Parameters

N/A		

AE Advertising enable.

Definition at line 1 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

Macro Definition Documentation

BLE_PROTOCOL

```
#define BLE_PROTOCOL
```

Value:

```
0x01
```

Definition at line 481 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

PROP_PROTOCOL

```
#define PROP_PROTOCOL
```

Value:

```
0x02
```

Definition at line 482 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ADV_ROLE

```
#define ADV_ROLE
```

Value:

```
0x01
```

Definition at line 484 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

SCAN_AND_CENTRAL_ROLE

```
#define SCAN_AND_CENTRAL_ROLE
```

Value:

```
0x02
```

Definition at line 485 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

PERIPHERAL_ROLE

```
#define PERIPHERAL_ROLE
```

Value:

```
0x03
```

Definition at line 487 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

CONN_ROLE

```
#define CONN_ROLE
```

Value:

```
0x04
```

Definition at line 489 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_EXCHANGE_MTU_REQUEST

```
#define RSI_BLE_ATT_EXCHANGE_MTU_REQUEST
```

Value:

```
0x02
```

Definition at line 1107 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_FIND_INFORMATION_REQUEST

```
#define RSI_BLE_ATT_FIND_INFORMATION_REQUEST
```

Value:

```
0x04
```

Definition at line 1108 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_FIND_BY_TYPE_VALUE_REQUEST

```
#define RSI_BLE_ATT_FIND_BY_TYPE_VALUE_REQUEST
```

Value:

```
0x06
```

Definition at line 1109 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_READ_BY_TYPE_REQUEST

```
#define RSI_BLE_ATT_READ_BY_TYPE_REQUEST
```

Value:

```
0x08
```

Definition at line 1110 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_READ_REQUEST

```
#define RSI_BLE_ATT_READ_REQUEST
```

Value:

```
0x0A
```

Definition at line 1111 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_READ_BLOB_REQUEST

```
#define RSI_BLE_ATT_READ_BLOB_REQUEST
```

Value:

```
0x0C
```

Definition at line 1112 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_READ_MULTIPLE_REQUEST

```
#define RSI_BLE_ATT_READ_MULTIPLE_REQUEST
```

Value:

```
0x0E
```

Definition at line 1113 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_READ_BY_GROUP_TYPE_REQUEST

```
#define RSI_BLE_ATT_READ_BY_GROUP_TYPE_REQUEST
```

Value:

```
0x10
```

Definition at line 1114 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_WRITE_REQUEST

```
#define RSI_BLE_ATT_WRITE_REQUEST
```

Value:

```
0x12
```

Definition at line 1115 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_PREPARE_WRITE_REQUEST

```
#define RSI_BLE_ATT_PREPARE_WRITE_REQUEST
```

Value:

```
0x16
```

Definition at line 1116 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

RSI_BLE_ATT_EXECUTE_WRITE_REQUEST

```
#define RSI_BLE_ATT_EXECUTE_WRITE_REQUEST
```

Value:

```
0x18
```

Definition at line 1117 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

SUPPORTED_SCNNING_PHYS

```
#define SUPPORTED_SCNNING_PHYS
```

Value:

```
2
```

Definition at line 1366 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

Variable Documentation

__attribute__

```
union @3 __attribute__
```

AE Advertising Params.

AE Advertising enable.

Definition at line 1696 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_rand_s

Public Attributes

uint8_t	rand_addr

Public Attribute Documentation

rand_addr

```
uint8_t rsi_ble_req_rand_s::rand_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 351 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_adv_s

Public Attributes

uint8_t	status Advertising Status -.
uint8_t	adv_type Advertising type used during advertising -.
uint8_t	filter_type Advertising filter type -.
uint8_t	direct_addr_type Address type of the device to which directed advertising has to be done -.
uint8_t	direct_addr Address of the device to which directed advertising has to be done.
uint16_t	adv_int_min Advertising interval min 0x0020 to 0x4000.
uint16_t	adv_int_max Advertising interval max 0x0020 to 0x4000.
uint8_t	own_addr_type Address of the local device.
uint8_t	adv_channel_map Advertising channel map.

Public Attribute Documentation

status

```
uint8_t rsi_ble_req_adv_s::status
```

Advertising Status -.

0 - disable

- 1 - enable

Definition at line 362 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_type

```
uint8_t rsi_ble_req_adv_s::adv_type
```

Advertising type used during advertising -.

1. Advertising will be visible(discoverable) to all the devices. Scanning/Connection is also accepted from all devices.

- #define UNDIR_CONN 0×80
- 2. Advertising will be visible(discoverable) to the particular device mentioned in RSI_BLE_ADV_DIR_ADDR only.
- Scanning and Connection will be accepted from that device only.
- #define DIR_CONN 0×81
- 3. Advertising will be visible(discoverable) to all the devices. Scanning will be accepted from all the devices.
- Connection will be not be accepted from any device.
- #define UNDIR_SCAN 0×82
- 4. Advertising will be visible(discoverable) to all the devices. Scanning and Connection will not be accepted from any device.
- #define UNDIR_NON_CONN 0×83
- 5. Advertising will be visible(discoverable) to the particular device mentioned in RSI_BLE_ADV_DIR_ADDR only.
- Scanning and Connection will be accepted from that device only.
- #define DIR_CONN_LOW_DUTY_CYCLE 0×84

Definition at line 403 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

filter_type

uint8_t rsi_ble_req_adv_s::filter_type

Advertising filter type -.

#define ALLOW_SCAN_REQ_ANY_CONN_REQ_ANY 0×00

- #define ALLOW_SCAN_REQ_ACCEPT_LIST_CONN_REQ_ANY 0×01
- #define ALLOW_SCAN_REQ_ANY_CONN_REQ_ACCEPT_LIST 0×02
- #define ALLOW_SCAN_REQ_ACCEPT_LIST_CONN_REQ_ACCEPT_LIST 0×03

Definition at line 417 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

direct_addr_type

uint8_t rsi_ble_req_adv_s::direct_addr_type

Address type of the device to which directed advertising has to be done -.

```
#define LE_PUBLIC_ADDRESS 0x00
```

- #define LE_RANDOM_ADDRESS 0x01
- #define LE_RESOLVABLE_PUBLIC_ADDRESS 0x02
- #define LE_RESOLVABLE_RANDOM_ADDRESS 0x03

Definition at line 431 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

direct_addr

uint8_t rsi_ble_req_adv_s::direct_addr[RSI_DEV_ADDR_LEN]

Address of the device to which directed advertising has to be done.

Definition at line 433 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_int_min

uint16_t rsi_ble_req_adv_s::adv_int_min

Advertising interval min 0x0020 to 0x4000.

Definition at line 436 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_int_max

uint16_t rsi_ble_req_adv_s::adv_int_max

Advertising interval max 0x0020 to 0x4000.

Definition at line 439 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

own_addr_type

uint8_t rsi_ble_req_adv_s::own_addr_type

Address of the local device.

- #define LE_PUBLIC_ADDRESS 0x00

- #define LE_RANDOM_ADDRESS 0×01
- #define LE_RESOLVABLE_PUBLIC_ADDRESS 0×02
- #define LE_RESOLVABLE_RANDOM_ADDRESS 0×03

Definition at line 454 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_channel_map

> uint8_t rsi_ble_req_adv_s::adv_channel_map

Advertising channel map.

- #define RSI_BLE_ADV_CHANNEL_MAP 0×01 or 0×03 or 0×07

Definition at line 460 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_adv_data_s

Public Attributes

uint8_t	data_len
uint8_t	adv_data

Public Attribute Documentation

data_len

```
uint8_t rsi_ble_req_adv_data_s::data_len
```

Definition at line 466 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_data

```
uint8_t rsi_ble_req_adv_data_s::adv_data[31]
```

Definition at line 468 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_acceptlist_using_payload_s

Public Attributes

uint8_t	opcode
uint8_t	enable
uint8_t	total_len
uint8_t	data_compare_index
uint8_t	len_for_compare_data
uint8_t	adv_data_payload

Public Attribute Documentation

opcode

```
uint8_t rsi_ble_req_acceptlist_using_payload_s::opcode[2]
```

Definition at line 473 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

enable

```
uint8_t rsi_ble_req_acceptlist_using_payload_s::enable
```

Definition at line 474 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

total_len

```
uint8_t rsi_ble_req_acceptlist_using_payload_s::total_len
```

Definition at line 475 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data_compare_index

```
uint8_t rsi_ble_req_acceptlist_using_payload_s::data_compare_index
```

Definition at line 476 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

len_for_compare_data

> uint8_t rsi_ble_req_acceptlist_using_payload_s::len_for_compare_data

Definition at line 477 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_data_payload

> uint8_t rsi_ble_req_acceptlist_using_payload_s::adv_data_payload[31]

Definition at line 478 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_ble_tx_power_s

Public Attributes

uint8_t	role
uint8_t	dev_addr
uint8_t	tx_power

Public Attribute Documentation

role

```
uint8_t rsi_ble_set_ble_tx_power_s::role
```

Definition at line `492` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

dev_addr

```
uint8_t rsi_ble_set_ble_tx_power_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `494` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

tx_power

```
uint8_t rsi_ble_set_ble_tx_power_s::tx_power
```

Definition at line `495` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_scanrsp_data_s

Public Attributes

uint8_t	data_len
uint8_t	scanrsp_data

Public Attribute Documentation

data_len

```
uint8_t rsi_ble_req_scanrsp_data_s::data_len
```

Definition at line 501 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

scanrsp_data

```
uint8_t rsi_ble_req_scanrsp_data_s::scanrsp_data[31]
```

Definition at line 503 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_scan_s

Public Attributes

uint8_t	status Scanning Status -.
uint8_t	scan_type Scanning type -.
uint8_t	filter_type To filter incoming advertising reports -.
uint8_t	own_addr_type Address type of the local device -.
uint16_t	scan_int Scan interval -.
uint16_t	scan_win Scan window -.

Public Attribute Documentation

status

```
uint8_t rsi_ble_req_scan_s::status
```

Scanning Status -.

0 - disable

- 1 - enable

Definition at line 515 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

scan_type

```
uint8_t rsi_ble_req_scan_s::scan_type
```

Scanning type -.

#define SCAN_TYPE_ACTIVE 0x01

- #define SCAN_TYPE_PASSIVE 0x00

Definition at line 524 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

filter_type

uint8_t rsi_ble_req_scan_s::filter_type

To filter incoming advertising reports -.

FILTERING_DISABLED = 0 (default)

- ACCEPTLIST_FILTERING = 1 **Note**
  - In order to allow only acceptlisted devices, need to add bd_addr into acceptlist by calling rsi_ble_addto_acceptlist() API

Definition at line 535 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

own_addr_type

uint8_t rsi_ble_req_scan_s::own_addr_type

Address type of the local device -.

```
#define LE_PUBLIC_ADDRESS              0x00
```

- #define LE_RANDOM_ADDRESS 0x01
- #define LE_RESOLVABLE_PUBLIC_ADDRESS 0x02
- #define LE_RESOLVABLE_RANDOM_ADDRESS 0x03

Definition at line 550 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

scan_int

uint16_t rsi_ble_req_scan_s::scan_int

Scan interval -.

This is defined as the time interval from when the Controller started
its last LE scan until it begins the subsequent LE scan.

- Range: 0x0004 to 0x4000

Definition at line 560 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

scan_win

uint16_t rsi_ble_req_scan_s::scan_win

Scan window -.

The duration of the LE scan. LE_Scan_Window shall be less than or equal to LE_Scan_Interval

- Range: 0×0004 to 0×4000

Definition at line 569 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_encrypt_s

Public Attributes

uint8_t	key
uint8_t	data

Public Attribute Documentation

key

```
uint8_t rsi_ble_encrypt_s::key[16]
```

Definition at line 577 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data

```
uint8_t rsi_ble_encrypt_s::data[16]
```

Definition at line 578 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_data_packet_s

Public Attributes

uint8_t data

Public Attribute Documentation

data

```
uint8_t rsi_data_packet_s::data[1024]
```

Definition at line 583 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_accept_list_s

Public Attributes

uint8_t	addordeltowhitlist
uint8_t	dev_addr
uint8_t	bdaddressType

Public Attribute Documentation

addordeltowhitlist

```
uint8_t rsi_ble_accept_list_s::addordeltowhitlist
```

Definition at line 589 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

dev_addr

```
uint8_t rsi_ble_accept_list_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 591 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

bdaddressType

```
uint8_t rsi_ble_accept_list_s::bdaddressType
```

Definition at line 593 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_conn_s

Public Attributes

uint8_t	dev_addr_type
uint8_t	dev_addr
uint16_t	le_scan_interval
uint16_t	le_scan_window
uint16_t	conn_interval_min
uint16_t	conn_interval_max
uint16_t	conn_latency
uint16_t	supervision_tout

Public Attribute Documentation

dev_addr_type

```
uint8_t rsi_ble_req_conn_s::dev_addr_type
```

Definition at line `600` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

dev_addr

```
uint8_t rsi_ble_req_conn_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `602` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

le_scan_interval

```
uint16_t rsi_ble_req_conn_s::le_scan_interval
```

Definition at line `604` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

le_scan_window

```
uint16_t rsi_ble_req_conn_s::le_scan_window
```

Definition at line `606` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

conn_interval_min

```
uint16_t rsi_ble_req_conn_s::conn_interval_min
```

Definition at line 608 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

conn_interval_max

```
uint16_t rsi_ble_req_conn_s::conn_interval_max
```

Definition at line 610 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

conn_latency

```
uint16_t rsi_ble_req_conn_s::conn_latency
```

Definition at line 612 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

supervision_tout

```
uint16_t rsi_ble_req_conn_s::supervision_tout
```

Definition at line 614 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_enhance_conn_s

Public Attributes

uint8_t	dev_addr_type
uint8_t	dev_addr
uint8_t	filter_policy
uint8_t	own_addr_type
uint16_t	le_scan_interval
uint16_t	le_scan_window
uint16_t	conn_interval_min
uint16_t	conn_interval_max
uint16_t	conn_latency
uint16_t	supervision_tout
uint16_t	min_ce_length
uint16_t	max_ce_length

Public Attribute Documentation

dev_addr_type

```
uint8_t rsi_ble_req_enhance_conn_s::dev_addr_type
```

Definition at line 619 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

dev_addr

```
uint8_t rsi_ble_req_enhance_conn_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 621 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

filter_policy

```
uint8_t rsi_ble_req_enhance_conn_s::filter_policy
```

Definition at line 623 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

own_addr_type

uint8_t rsi_ble_req_enhance_conn_s::own_addr_type

Definition at line 625 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

le_scan_interval

uint16_t rsi_ble_req_enhance_conn_s::le_scan_interval

Definition at line 627 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

le_scan_window

uint16_t rsi_ble_req_enhance_conn_s::le_scan_window

Definition at line 629 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

conn_interval_min

uint16_t rsi_ble_req_enhance_conn_s::conn_interval_min

Definition at line 631 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

conn_interval_max

uint16_t rsi_ble_req_enhance_conn_s::conn_interval_max

Definition at line 633 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

conn_latency

uint16_t rsi_ble_req_enhance_conn_s::conn_latency

Definition at line 635 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

supervision_tout

uint16_t rsi_ble_req_enhance_conn_s::supervision_tout

Definition at line 637 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

min_ce_length

> uint16_t rsi_ble_req_enhance_conn_s::min_ce_length

Definition at line `639` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

max_ce_length

> uint16_t rsi_ble_req_enhance_conn_s::max_ce_length

Definition at line `641` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_disconnect_s

Public Attributes

uint8_t	dev_addr
uint8_t	type

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_disconnect_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 647 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

type

```
uint8_t rsi_ble_req_disconnect_s::type
```

Definition at line 651 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_start_encryption_s

Public Attributes

uint8_t	dev_addr
uint16_t	ediv
uint8_t	rand
uint8_t	ltk

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_start_encryption_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `659` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

ediv

```
uint16_t rsi_ble_start_encryption_s::ediv
```

Definition at line `661` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rand

```
uint8_t rsi_ble_start_encryption_s::rand[8]
```

Definition at line `663` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

ltk

```
uint8_t rsi_ble_start_encryption_s::ltk[16]
```

Definition at line `665` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_smp_pair_s

Public Attributes

uint8_t	dev_addr
uint8_t	io_capability
uint8_t	mitm_req

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_smp_pair_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `670` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

io_capability

```
uint8_t rsi_ble_req_smp_pair_s::io_capability
```

Definition at line `671` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

mitm_req

```
uint8_t rsi_ble_req_smp_pair_s::mitm_req
```

Definition at line `672` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_smp_response_s

Public Attributes

uint8_t	dev_addr
uint8_t	io_capability
uint8_t	mitm_req

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_smp_response_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 677 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

io_capability

```
uint8_t rsi_ble_smp_response_s::io_capability
```

Definition at line 678 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

mitm_req

```
uint8_t rsi_ble_smp_response_s::mitm_req
```

Definition at line 679 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_smp_passkey_s

Public Attributes

uint8_t	dev_addr
uint8_t	reserved
uint32_t	passkey

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_smp_passkey_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `684` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

reserved

```
uint8_t rsi_ble_smp_passkey_s::reserved[2]
```

Definition at line `685` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

passkey

```
uint32_t rsi_ble_smp_passkey_s::passkey
```

Definition at line `686` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_get_le_ping_timeout_s

Public Attributes

uint8_t	dev_addr

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_get_le_ping_timeout_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 691 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_rsp_get_le_ping_timeout_s

Public Attributes

uint8_t	dev_addr
uint16_t	time_out

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_rsp_get_le_ping_timeout_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `696` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

time_out

```
uint16_t rsi_ble_rsp_get_le_ping_timeout_s::time_out
```

Definition at line `697` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_set_le_ping_timeout_s

Public Attributes

uint8_t	dev_addr
uint16_t	time_out

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_set_le_ping_timeout_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `702` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

time_out

```
uint16_t rsi_ble_set_le_ping_timeout_s::time_out
```

Definition at line `703` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_resolvlist_s

Public Attributes

uint8_t	process_type
uint8_t	remote_dev_addr_type
uint8_t	remote_dev_addr
uint8_t	peer_irk
uint8_t	local_irk

Public Attribute Documentation

process_type

```
uint8_t rsi_ble_resolvlist_s::process_type
```

Definition at line 708 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

remote_dev_addr_type

```
uint8_t rsi_ble_resolvlist_s::remote_dev_addr_type
```

Definition at line 709 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

remote_dev_addr

```
uint8_t rsi_ble_resolvlist_s::remote_dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 710 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

peer_irk

```
uint8_t rsi_ble_resolvlist_s::peer_irk[16]
```

Definition at line 711 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

local_irk

```
uint8_t rsi_ble_resolvlist_s::local_irk[16]
```

Definition at line 712 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_get_resolving_list_size_s

Public Attributes

uint8_t size

Public Attribute Documentation

size

```
uint8_t rsi_ble_get_resolving_list_size_s::size
```

Definition at line 718 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_addr_resolution_enable_s

Public Attributes

uint8_t	enable
uint8_t	reserved
uint16_t	tout

Public Attribute Documentation

enable

```
uint8_t rsi_ble_set_addr_resolution_enable_s::enable
```

Definition at line 724 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

reserved

```
uint8_t rsi_ble_set_addr_resolution_enable_s::reserved
```

Definition at line 725 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

tout

```
uint16_t rsi_ble_set_addr_resolution_enable_s::tout
```

Definition at line 726 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_cmd_conn_params_update_s

Public Attributes

uint8_t	dev_addr
uint16_t	min_interval
uint16_t	max_interval
uint16_t	latency
uint16_t	timeout

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_cmd_conn_params_update_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `731` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

min_interval

```
uint16_t rsi_ble_cmd_conn_params_update_s::min_interval
```

Definition at line `732` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

max_interval

```
uint16_t rsi_ble_cmd_conn_params_update_s::max_interval
```

Definition at line `733` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

latency

```
uint16_t rsi_ble_cmd_conn_params_update_s::latency
```

Definition at line `734` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

timeout

```
uint16_t rsi_ble_cmd_conn_params_update_s::timeout
```

Definition at line 735 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_read_phy_s

Public Attributes

uint8_t dev_addr

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_read_phy_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 740 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_phy_s

Public Attributes

uint8_t	dev_addr
uint8_t	all_phy
uint8_t	tx_phy
uint8_t	rx_phy
uint8_t	reserved
uint16_t	phy_options

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_set_phy_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 745 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

all_phy

```
uint8_t rsi_ble_set_phy_s::all_phy
```

Definition at line 746 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

tx_phy

```
uint8_t rsi_ble_set_phy_s::tx_phy
```

Definition at line 747 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rx_phy

```
uint8_t rsi_ble_set_phy_s::rx_phy
```

Definition at line 748 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

reserved

```
uint8_t rsi_ble_set_phy_s::reserved
```

Definition at line 749 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

phy_options

> uint16_t rsi_ble_set_phy_s::phy_options

Definition at line 750 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_setdatalength_s

Public Attributes

uint8_t	dev_addr
uint16_t	txoctets
uint16_t	txtime

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_setdatalength_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 755 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

txoctets

```
uint16_t rsi_ble_setdatalength_s::txoctets
```

Definition at line 756 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

txtime

```
uint16_t rsi_ble_setdatalength_s::txtime
```

Definition at line 757 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_privacy_mode_s

Public Attributes

uint8_t	remote_dev_addr_type
uint8_t	remote_dev_addr
uint8_t	privacy_mode

Public Attribute Documentation

remote_dev_addr_type

```
uint8_t rsi_ble_set_privacy_mode_s::remote_dev_addr_type
```

Definition at line `762` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

remote_dev_addr

```
uint8_t rsi_ble_set_privacy_mode_s::remote_dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `763` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

privacy_mode

```
uint8_t rsi_ble_set_privacy_mode_s::privacy_mode
```

Definition at line `764` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_cbfc_conn_req_s

Public Attributes

uint8_t	dev_addr
uint8_t	psm

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_cbfc_conn_req_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `769` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

psm

```
uint8_t rsi_ble_cbfc_conn_req_s::psm
```

Definition at line `770` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_tx_test_mode_s

Public Attributes

uint8_t	tx_channel
uint8_t	phy
uint8_t	tx_len
uint8_t	tx_data_mode

Public Attribute Documentation

tx_channel

```
uint8_t rsi_ble_tx_test_mode_s::tx_channel
```

Definition at line 806 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

phy

```
uint8_t rsi_ble_tx_test_mode_s::phy
```

Definition at line 807 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

tx_len

```
uint8_t rsi_ble_tx_test_mode_s::tx_len
```

Definition at line 808 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

tx_data_mode

```
uint8_t rsi_ble_tx_test_mode_s::tx_data_mode
```

Definition at line 809 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_end_test_mode_s

Public Attributes

uint16_t	num_of_pkts

Public Attribute Documentation

num_of_pkts

```
uint16_t rsi_ble_end_test_mode_s::num_of_pkts
```

Definition at line 814 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_le_ltkreqreply_s

Public Attributes

uint8_t	dev_addr
uint8_t	replytype
uint8_t	localltk

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_set_le_ltkreqreply_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 818 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

replytype

```
uint8_t rsi_ble_set_le_ltkreqreply_s::replytype
```

Definition at line 819 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

localltk

```
uint8_t rsi_ble_set_le_ltkreqreply_s::localltk[16]
```

Definition at line 820 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_smp_pair_failed_s

Public Attributes

uint8_t	dev_addr
uint8_t	reason

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_smp_pair_failed_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 825 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

reason

```
uint8_t rsi_ble_req_smp_pair_failed_s::reason
```

Definition at line 826 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_profiles_list_s

Public Attributes

uint8_t	dev_addr
uint16_t	start_handle
uint16_t	end_handle

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_profiles_list_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `834` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

start_handle

```
uint16_t rsi_ble_req_profiles_list_s::start_handle
```

Definition at line `836` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

end_handle

```
uint16_t rsi_ble_req_profiles_list_s::end_handle
```

Definition at line `838` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_profile_s

Public Attributes

uint8_t	dev_addr
uint8_t	reserved
uuid_t	profile_uuid

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_profile_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 844 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

reserved

```
uint8_t rsi_ble_req_profile_s::reserved[2]
```

Definition at line 846 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

profile_uuid

```
uuid_t rsi_ble_req_profile_s::profile_uuid
```

Definition at line 848 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_char_services_s

Public Attributes

uint8_t	dev_addr
uint8_t	start_handle
uint8_t	end_handle

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_char_services_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 857 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

start_handle

```
uint8_t rsi_ble_req_char_services_s::start_handle[2]
```

Definition at line 859 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

end_handle

```
uint8_t rsi_ble_req_char_services_s::end_handle[2]
```

Definition at line 861 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_inc_services_s

Public Attributes

uint8_t	dev_addr
uint8_t	start_handle
uint8_t	end_handle

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_inc_services_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `867` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

start_handle

```
uint8_t rsi_ble_req_inc_services_s::start_handle[2]
```

Definition at line `869` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

end_handle

```
uint8_t rsi_ble_req_inc_services_s::end_handle[2]
```

Definition at line `871` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_char_val_by_uuid_s

Public Attributes

uint8_t	dev_addr
uint8_t	start_handle
uint8_t	end_handle
uint8_t	reserved
uuid_t	char_uuid

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_char_val_by_uuid_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 877 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

start_handle

```
uint8_t rsi_ble_req_char_val_by_uuid_s::start_handle[2]
```

Definition at line 879 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

end_handle

```
uint8_t rsi_ble_req_char_val_by_uuid_s::end_handle[2]
```

Definition at line 881 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

reserved

```
uint8_t rsi_ble_req_char_val_by_uuid_s::reserved[2]
```

Definition at line 883 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

char_uuid

```
uuid_t rsi_ble_req_char_val_by_uuid_s::char_uuid
```

Definition at line 885 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_att_descs_s

Public Attributes

uint8_t	dev_addr
uint8_t	start_handle
uint8_t	end_handle

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_att_descs_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `894` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

start_handle

```
uint8_t rsi_ble_req_att_descs_s::start_handle[2]
```

Definition at line `896` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

end_handle

```
uint8_t rsi_ble_req_att_descs_s::end_handle[2]
```

Definition at line `898` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_req_att_value_s

Public Attributes

uint8_t	dev_addr
uint8_t	handle

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_att_value_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 904 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

handle

```
uint8_t rsi_ble_req_att_value_s::handle[2]
```

Definition at line 906 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_multiple_att_val_s

Public Attributes

uint8_t	dev_addr
uint8_t	num_of_handles
uint8_t	reserved
uint16_t	handles

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_multiple_att_val_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 912 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

num_of_handles

```
uint8_t rsi_ble_req_multiple_att_val_s::num_of_handles
```

Definition at line 914 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

reserved

```
uint8_t rsi_ble_req_multiple_att_val_s::reserved
```

Definition at line 916 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

handles

```
uint16_t rsi_ble_req_multiple_att_val_s::handles[RSI_BLE_MAX_RESP_LIST]
```

Definition at line 918 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_long_att_value_s

Public Attributes

uint8_t	dev_addr
uint16_t	handle
uint16_t	offset

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_long_att_value_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `928` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

handle

```
uint16_t rsi_ble_req_long_att_value_s::handle
```

Definition at line `930` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

offset

```
uint16_t rsi_ble_req_long_att_value_s::offset
```

Definition at line `932` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_set_att_val_s

Public Attributes

uint8_t	dev_addr
uint8_t	handle
uint8_t	length
uint8_t	att_value

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_set_att_val_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 941 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

handle

```
uint8_t rsi_ble_set_att_val_s::handle[2]
```

Definition at line 943 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

length

```
uint8_t rsi_ble_set_att_val_s::length
```

Definition at line 945 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

att_value

```
uint8_t rsi_ble_set_att_val_s::att_value[RSI_DEV_ATT_LEN]
```

Definition at line 947 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_att_cmd_s

Public Attributes

uint8_t	dev_addr
uint8_t	handle
uint8_t	length
uint8_t	att_value

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_set_att_cmd_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 953 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

handle

```
uint8_t rsi_ble_set_att_cmd_s::handle[2]
```

Definition at line 955 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

length

```
uint8_t rsi_ble_set_att_cmd_s::length
```

Definition at line 957 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

att_value

```
uint8_t rsi_ble_set_att_cmd_s::att_value[RSI_DEV_ATT_LEN]
```

Definition at line 959 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_long_att_val_s

Public Attributes

uint8_t	dev_addr
uint8_t	handle
uint8_t	offset
uint8_t	length
uint8_t	att_value

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_set_long_att_val_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `965` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

handle

```
uint8_t rsi_ble_set_long_att_val_s::handle[2]
```

Definition at line `967` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

offset

```
uint8_t rsi_ble_set_long_att_val_s::offset[2]
```

Definition at line `969` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

length

```
uint8_t rsi_ble_set_long_att_val_s::length
```

Definition at line `971` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

att_value

```
uint8_t rsi_ble_set_long_att_val_s::att_value[RSI_DEV_ATT_LEN]
```

Definition at line 973 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_prepare_write_s

Public Attributes

uint8_t	dev_addr
uint8_t	handle
uint8_t	offset
uint8_t	length
uint8_t	att_value

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_prepare_write_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `979` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

handle

```
uint8_t rsi_ble_req_prepare_write_s::handle[2]
```

Definition at line `981` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

offset

```
uint8_t rsi_ble_req_prepare_write_s::offset[2]
```

Definition at line `983` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

length

```
uint8_t rsi_ble_req_prepare_write_s::length
```

Definition at line `985` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

att_value

```
uint8_t rsi_ble_req_prepare_write_s::att_value[RSI_DEV_ATT_LEN]
```

Definition at line 987 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_execute_write_s

Public Attributes

uint8_t	dev_addr
uint8_t	flag

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_req_execute_write_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `993` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

flag

```
uint8_t rsi_ble_req_execute_write_s::flag
```

Definition at line `995` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

rsi_ble_cmd_conn_param_resp

Public Attributes

uint8_t	dev_addr
uint8_t	status

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_cmd_conn_param_resp::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1001 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

status

```
uint8_t rsi_ble_cmd_conn_param_resp::status
```

Definition at line 1003 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_req_add_serv_s

Public Attributes

uuid_t	service_uuid
uint8_t	num_of_attributes
uint8_t	total_att_datasize

Public Attribute Documentation

service_uuid

```
uuid_t rsi_ble_req_add_serv_s::service_uuid
```

Definition at line 1012 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

num_of_attributes

```
uint8_t rsi_ble_req_add_serv_s::num_of_attributes
```

Definition at line 1014 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

total_att_datasize

```
uint8_t rsi_ble_req_add_serv_s::total_att_datasize
```

Definition at line 1016 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_local_att_value_s

Public Attributes

uint16_t	handle
uint16_t	data_len
uint8_t	data

Public Attribute Documentation

handle

```
uint16_t rsi_ble_set_local_att_value_s::handle
```

Definition at line 1022 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data_len

```
uint16_t rsi_ble_set_local_att_value_s::data_len
```

Definition at line 1024 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data

```
uint8_t rsi_ble_set_local_att_value_s::data[RSI_DEV_ATT_LEN]
```

Definition at line 1026 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_notify_att_value_s

Public Attributes

uint8_t	dev_addr
uint16_t	handle
uint16_t	data_len
uint8_t	data

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_notify_att_value_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1031 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

handle

```
uint16_t rsi_ble_notify_att_value_s::handle
```

Definition at line 1033 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data_len

```
uint16_t rsi_ble_notify_att_value_s::data_len
```

Definition at line 1035 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data

```
uint8_t rsi_ble_notify_att_value_s::data[RSI_DEV_ATT_LEN]
```

Definition at line 1037 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_wo_resp_notify_buf_info_s

Public Attributes

uint8_t	dev_addr
uint8_t	buf_mode
uint8_t	buf_count

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_set_wo_resp_notify_buf_info_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1042 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

buf_mode

```
uint8_t rsi_ble_set_wo_resp_notify_buf_info_s::buf_mode
```

Definition at line 1044 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

buf_count

```
uint8_t rsi_ble_set_wo_resp_notify_buf_info_s::buf_count
```

Definition at line 1046 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_indicate_confirm_s

Public Attributes

uint8_t	dev_addr

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_indicate_confirm_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1052 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_get_local_att_value_s

Public Attributes

uint16_t handle

Public Attribute Documentation

handle

```
uint16_t rsi_ble_get_local_att_value_s::handle
```

Definition at line 1058 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_read_response_s

Public Attributes

uint8_t	dev_addr
uint8_t	type
uint8_t	reserved
uint16_t	data_len
uint8_t	data

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_gatt_read_response_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1063 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

type

```
uint8_t rsi_ble_gatt_read_response_s::type
```

Definition at line 1067 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

reserved

```
uint8_t rsi_ble_gatt_read_response_s::reserved
```

Definition at line 1068 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data_len

```
uint16_t rsi_ble_gatt_read_response_s::data_len
```

Definition at line 1070 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data

```
uint8_t rsi_ble_gatt_read_response_s::data[RSI_DEV_ATT_LEN]
```

Definition at line 1072 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_write_response_s

Public Attributes

uint8_t	dev_addr
uint8_t	type

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_gatt_write_response_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1078 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

type

```
uint8_t rsi_ble_gatt_write_response_s::type
```

Definition at line 1080 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_prepare_write_response_s

Public Attributes

uint8_t	dev_addr
uint16_t	handle
uint16_t	offset
uint16_t	data_len
uint8_t	data

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_gatt_prepare_write_response_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1086 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

handle

```
uint16_t rsi_ble_gatt_prepare_write_response_s::handle
```

Definition at line 1087 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

offset

```
uint16_t rsi_ble_gatt_prepare_write_response_s::offset
```

Definition at line 1088 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data_len

```
uint16_t rsi_ble_gatt_prepare_write_response_s::data_len
```

Definition at line 1089 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data

```
uint8_t rsi_ble_gatt_prepare_write_response_s::data[RSI_DEV_ATT_LEN]
```

Definition at line 1090 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_set_local_irk_s

Public Attributes

uint8_t irk

Public Attribute Documentation

irk

```
uint8_t rsi_ble_set_local_irk_s::irk[16]
```

Definition at line 1096 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_att_error_response_s

Public Attributes

uint8_t	dev_addr
uint8_t	req_opcode
uint16_t	att_handle
uint8_t	err_code

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_att_error_response_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1121 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

req_opcode

```
uint8_t rsi_ble_att_error_response_s::req_opcode
```

Definition at line 1122 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

att_handle

```
uint16_t rsi_ble_att_error_response_s::att_handle
```

Definition at line 1123 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

err_code

```
uint8_t rsi_ble_att_error_response_s::err_code
```

Definition at line 1124 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_remove_serv_s

Public Attributes

uint32_t	serv_hndler

Public Attribute Documentation

serv_hndler

```
uint32_t rsi_ble_gatt_remove_serv_s::serv_hndler
```

Definition at line 1129 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_gatt_remove_att_s

Public Attributes

uint32_t	serv_hndler
uint16_t	att_hndl

Public Attribute Documentation

serv_hndler

```
uint32_t rsi_ble_gatt_remove_att_s::serv_hndler
```

Definition at line 1134 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

att_hndl

```
uint16_t rsi_ble_gatt_remove_att_s::att_hndl
```

Definition at line 1135 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_vendor_rf_type_s

Public Attributes

uint8_t	opcode
uint8_t	ble_power_index

Public Attribute Documentation

opcode

```
uint8_t rsi_ble_vendor_rf_type_s::opcode[2]
```

Definition at line 1140 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ble_power_index

```
uint8_t rsi_ble_vendor_rf_type_s::ble_power_index
```

Definition at line 1141 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_mtu_exchange_s

Public Attributes

uint8_t	dev_addr
uint16_t	req_mtu_size

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_mtu_exchange_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1146 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

req_mtu_size

```
uint16_t rsi_ble_mtu_exchange_s::req_mtu_size
```

Definition at line 1147 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_mtu_exchange_resp_s

Public Attributes

uint8_t	dev_addr
uint16_t	req_mtu_size

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_mtu_exchange_resp_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 1152 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

req_mtu_size

```
uint16_t rsi_ble_mtu_exchange_resp_s::req_mtu_size
```

Definition at line 1153 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_get_supported_no_of_adv_sets_s

Public Attributes

uint16_t reserved

Public Attribute Documentation

reserved

```
uint16_t rsi_ble_ae_get_supported_no_of_adv_sets_s::reserved
```

Definition at line 1157 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_read_supported_max_adv_data_s

Public Attributes

uint16_t reserved

Public Attribute Documentation

reserved

```
uint16_t rsi_ble_ae_read_supported_max_adv_data_s::reserved
```

Definition at line 1161 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_set_random_address_s

Public Attributes

uint8_t adv_handle
uint8_t Advertising_Handle, Used to identify an advertising set , Range : 0x00 to 0xEF

uint8_t addr
uint8[6] Random_Address , The Random Address may be of either Static Address or Private Address

Public Attribute Documentation

adv_handle

```
uint8_t rsi_ble_ae_set_random_address_s::adv_handle
```

uint8_t Advertising_Handle, Used to identify an advertising set , Range : 0x00 to 0xEF

Definition at line 1167 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

addr

```
uint8_t rsi_ble_ae_set_random_address_s::addr[RSI_DEV_ADDR_LEN]
```

uint8[6] Random_Address , The Random Address may be of either Static Address or Private Address

Definition at line 1169 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_adv_params_s

AE Advertising Params.

Public Attributes

uint8_t	adv_handle uint8_t, Advertising Handle, Used to identify an Advertising set , Range : 0x00 to 0xEF
uint16_t	adv_event_prop
uint32_t	primary_adv_intterval_min uint32_t, Primary Advertising Interval Minimum, Minimum advertising interval for undirected and low duty cycle directed advertising
uint32_t	primary_adv_intterval_max uint32_t, Primary Advertising Interval Maximum, Maximum advertising interval for undirected and low duty cycle directed advertising primary_adv_intterval_min <= primary_adv_intterval_max
uint8_t	primary_adv_chnl_map uint8_t, Primary Advertising Channel Map, It specifies on which channel it shall advertise Bit Number Parameter Description 0 Channel 37 shall be used 1 Channel 38 Shall be used 2 Channel 39 shall be used
uint8_t	own_addr_type uint8_t, Own_Address_type, Indicates the type of the Address 0x00 - Public Device Address 0x01 - Random Device Address 0x02 - Controller generates the Resolvable Private Address based on the local IRK from the resolving list.
uint8_t	peer_addr_type uint8_t, Peer_Address_Type, Specifies Peer Address Type 0x00 - Public Device Address or Public Identity Address 0x01 - Random Device Address or Random (static) Identity Address
uint8_t	peer_dev_addr uint8[6], Peer_Device_Address, Address of the Peer_Address_Type
uint8_t	adv_filter_policy uint8_t, Advertising_Filter_Policy 0x00 - Process scan and connection requests from all devices (i.e., the Filter Accept List is not in use) 0x01 - Process connection requests from all devices and scan requests only from devices that are in the Filter Accept List.
uint8_t	adv_tx_power uint8_t Advertising_TX_Power, Advertising TX Power ranges from -127 to +20 and units are in dBm
uint8_t	primary_adv_phy uint8_t Primary_Advertising_PHY, This parameter specifies the PHY used for the periodic advertising.
uint8_t	sec_adv_max_skip uint8_t Secondary_Advertising_Max_Skip 0x00 AUX_ADV_IND shall be sent prior to the next advertising event 0x01 to 0xFF Maximum advertising events the Controller can skip before sending the AUX_ADV_IND packets on the secondary advertising physical channel
uint8_t	sec_adv_phy uint8_t Secondary_Advertising_PHY, This parameter specifies the PHY used for the periodic advertising.
uint8_t	adv_sid uint8_t Advertising_Sid, Value of the Advertising SID subfield in the ADI field of the PDU, Range : 0x00 to 0x0F

uint8_t scan_req_notify_enable
uint8_t Scan Request Notification Enable 0x00 Scan request notifications disabled 0x01 Scan request notifications enabled

Public Attribute Documentation

adv_handle

> uint8_t ae_adv_params_s::adv_handle

uint8_t, Advertising Handle, Used to identify an Advertising set , Range : 0x00 to 0xEF

Definition at line 1175 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_event_prop

> uint16_t ae_adv_params_s::adv_event_prop

uint16_t, Advertising Event Properties, indicates the properties of Advertising Event

Bit Number	Parameter Description
0	Connectable Advertising
1	Scannable Advertising
2	Direct Advertising
3	High Duty cycle Directed Connectable advertising (≤ 3.75 ms Advertising interval)
4	Use legacy advertising PDUs
5	Omit advertiser's address from all PDUs("anonymous advertising")
6	Include Tx Power in the extended header of at least one advertising PDU

Definition at line 1189 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

primary_adv_intterval_min

> uint32_t ae_adv_params_s::primary_adv_intterval_min

uint32_t, Primary Advertising Interval Minimum, Minimum advertising interval for undirected and low duty cycle directed advertising

Definition at line 1191 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

primary_adv_intterval_max

> uint32_t ae_adv_params_s::primary_adv_intterval_max

uint32_t, Primary Advertising Interval Maximum, Maximum advertising interval for undirected and low duty cycle directed advertising primary_adv_intterval_min <= primary_adv_intterval_max

Definition at line 1194 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

primary_adv_chnl_map

```
uint8_t ae_adv_params_s::primary_adv_chnl_map
```

uint8_t, Primary Advertising Channel Map, It specifies on which channel it shall advertise Bit Number Parameter Description 0 Channel 37 shall be used 1 Channel 38 Shall be used 2 Channel 39 shall be used

Definition at line 1201 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

own_addr_type

```
uint8_t ae_adv_params_s::own_addr_type
```

uint8_t, Own_Address_type, Indicates the type of the Address 0x00 - Public Device Address 0x01 - Random Device Address 0x02 - Controller generates the Resolvable Private Address based on the local IRK from the resolving list.

If the resolving list contains no matching entry, use the public address 0x03 - Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the random address from LE_Set_Advertising_Set_Random_Address

Definition at line 1213 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

peer_addr_type

```
uint8_t ae_adv_params_s::peer_addr_type
```

uint8_t, Peer_Address_Type, Specifies Peer Address Type 0x00 - Public Device Address or Public Identity Address 0x01 - Random Device Address or Random (static) Identity Address

Definition at line 1218 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

peer_dev_addr

```
uint8_t ae_adv_params_s::peer_dev_addr[RSI_DEV_ADDR_LEN]
```

uint8[6], Peer_Device_Address, Address of the Peer_Address_Type

Definition at line 1220 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_filter_policy

```
uint8_t ae_adv_params_s::adv_filter_policy
```

uint8_t, Advertising_Filter_Policy 0x00 - Process scan and connection requests from all devices (i.e., the Filter Accept List is not in use) 0x01 - Process connection requests from all devices and scan requests only from devices that are in the Filter Accept List.

0x02 - Process scan requests from all devices and connection requests only from devices that are in the Filter Accept List. 0x03 - Process scan and connection requests only from devices in the Filter Accept List.

Definition at line 1227 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_tx_power

```
uint8_t ae_adv_params_s::adv_tx_power
```

uint8_t Advertising_TX_Power, Advertising TX Power ranges from -127 to +20 and units are in dBm

Definition at line 1229 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

primary_adv_phy

```
uint8_t ae_adv_params_s::primary_adv_phy
```

uint8_t Primary_Advertising_PHY, This parameter specifies the PHY used for the periodic advertising.

0x01 - Advertiser PHY is LE 1M 0x03 - Advertiser PHY is LE Coded

Definition at line 1234 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

sec_adv_max_skip

```
uint8_t ae_adv_params_s::sec_adv_max_skip
```

uint8_t Secondary_Advertising_Max_Skip 0x00 AUX_ADV_IND shall be sent prior to the next advertising event 0x01 to 0xFF Maximum advertising events the Controller can skip before sending the AUX_ADV_IND packets on the secondary advertising physical channel

Definition at line 1239 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

sec_adv_phy

```
uint8_t ae_adv_params_s::sec_adv_phy
```

uint8_t Secondary_Advertising_PHY, This parameter specifies the PHY used for the periodic advertising.

0x01 - Advertiser PHY is LE 1M 0x02 - Advertiser PHY is LE 2M 0x03 - Advertiser PHY is LE Coded

Definition at line 1245 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_sid

```
uint8_t ae_adv_params_s::adv_sid
```

uint8_t Advertising_Sid, Value of the Advertising SID subfield in the ADI field of the PDU, Range : 0x00 to 0x0F

Definition at line 1247 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

scan_req_notify_enable

```
uint8_t ae_adv_params_s::scan_req_notify_enable
```

uint8_t Scan Request Notification Enable 0x00 Scan request notifications disabled 0x01 Scan request notifications enabled

Definition at line 1252 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_data_s

Public Attributes

uint8_t	type uint8_t AE_ADV_DATA_TYPE 1, AE_PERIODIC_ADV_DATA_TYPE 2, AE_SCAN_RSP_DATA_TYPE 3
uint8_t	adv_handle uint8_t Advertising Handle, used to identify an Advertising set, Ranges from 0x00 to 0xEF
uint8_t	operation uint8_t Operation 0x00 - Intermediate fragment of fragmented extended advertising data 0x01 - First fragment of fragmented extended advertising data 0x02 - Last fragment of fragmented extended advertising data 0x03 - Complete extended advertising data 0x04 - Unchanged data (just update the Advertising DID)
uint8_t	frag_pref uint8_t Fragment_Preference, Specifies the controller on where to fragment the Host advertising Data 0x00 - The Controller may fragment all Host advertising data 0x01 - The Controller should not fragment or should minimize fragmentation of Host advertising data
uint8_t	data_len uint8_t Data Length, Specifies Advertising_Data_Length , This parameter ranges from 0 to 251
uint8_t	data uint8_t Data ,Specifies Advertising_Data.

Public Attribute Documentation

type

```
uint8_t rsi_ble_ae_data_s::type
```

uint8_t AE_ADV_DATA_TYPE 1, AE_PERIODIC_ADV_DATA_TYPE 2, AE_SCAN_RSP_DATA_TYPE 3

Definition at line 1261 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_handle

```
uint8_t rsi_ble_ae_data_s::adv_handle
```

uint8_t Advertising Handle, used to identify an Advertising set, Ranges from 0x00 to 0xEF

Definition at line 1263 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

operation

```
uint8_t rsi_ble_ae_data_s::operation
```

uint8_t Operation 0x00 - Intermediate fragment of fragmented extended advertising data 0x01 - First fragment of fragmented extended advertising data 0x02 - Last fragment of fragmented extended advertising data 0x03 - Complete extended advertising data 0x04 - Unchanged data (just update the Advertising DID)

Definition at line 1271 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

frag_pref

```
uint8_t rsi_ble_ae_data_s::frag_pref
```

uint8_t Fragment_Preference, Specifies the controller on where to fragment the Host advertising Data 0x00 - The Controller may fragment all Host advertising data 0x01 - The Controller should not fragment or should minimize fragmentation of Host advertising data

Definition at line 1277 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data_len

```
uint8_t rsi_ble_ae_data_s::data_len
```

uint8_t Data Length, Specifies Advertising_Data_Length , This parameter ranges from 0 to 251

Definition at line 1279 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

data

```
uint8_t rsi_ble_ae_data_s::data[0xC8]
```

uint8_t Data ,Specifies Advertising_Data.

Definition at line 1281 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_adv_enabel_s

AE Advertising enable.

Public Attributes

uint8_t	enable uint8_t Enable, This parameter specifies whether to disable or Enable Advertising 0x00 - Advertising is disabled 0x01 - Advertising is Enabled
uint8_t	no_of_sets uint8_t Num_of_Sets , Indicates the number of Advertising sets to be disabled or enabled for Advertising 0x00 - Disable all advertising sets 0x01 to 0x3F - Number of advertising sets to enable or disable
uint8_t	adv_handle uint8_t Advertising_Handle, used to identify Advertising set, Ranges from 0x00 to 0xEF
uint16_t	duration uint16_t Duration, specifies the duration to continue advertising 0x00 - No Advertising 0x0001 to 0xFFFF , Advertising Duration
uint8_t	max_ae_events uint8_t Maximum Extended Advertising Events, It specifies the Maximum number of extended advertising events the Controller shall attempt to send prior to terminating the extended advertising

Public Attribute Documentation

enable

```
uint8_t rsi_ble_ae_adv_enabel_s::enable
```

uint8_t Enable, This parameter specifies whether to disable or Enable Advertising 0x00 - Advertising is disabled 0x01 - Advertising is Enabled

Definition at line 1291 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

no_of_sets

```
uint8_t rsi_ble_ae_adv_enabel_s::no_of_sets
```

uint8_t Num_of_Sets , Indicates the number of Advertising sets to be disabled or enabled for Advertising 0x00 - Disable all advertising sets 0x01 to 0x3F - Number of advertising sets to enable or disable

Definition at line 1297 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_handle

```
uint8_t rsi_ble_ae_adv_enabel_s::adv_handle
```

uint8_t Advertising_Handle, used to identify Advertising set, Ranges from 0x00 to 0xEF

Definition at line 1299 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

duration

> uint16_t rsi_ble_ae_adv_enabel_s::duration

uint16_t Duration, specifies the duration to continue advertising 0x00 - No Advertising 0x0001 to 0xFFFF , Advertising Duration

Definition at line 1305 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

max_ae_events

> uint8_t rsi_ble_ae_adv_enabel_s::max_ae_events

uint8_t Maximum Extended Advertising Events, It specifies the Maximum number of extended advertising events the Controller shall attempt to send prior to terminating the extended advertising

Definition at line 1308 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_adv_set_clear_or_remove_s

Public Attributes

uint8_t type
type - Specifies whether to remove or clear the advertising sets.

uint8_t adv_handle
uint8_t Advertising_Handle, used to identify Advertising set, Ranges from 0x00 to 0xEF

Public Attribute Documentation

type

```
uint8_t rsi_ble_ae_adv_set_clear_or_remove_s::type
```

type - Specifies whether to remove or clear the advertising sets.

{1} - clear {2} - remove

Definition at line 1318 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_handle

```
uint8_t rsi_ble_ae_adv_set_clear_or_remove_s::adv_handle
```

uint8_t Advertising_Handle, used to identify Advertising set, Ranges from 0x00 to 0xEF

Definition at line 1320 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_periodic_adv_params

Public Attributes

uint8_t	adv_handle uint8_t, Advertising Handle , this parameter identifies the advertising set whose periodic advertising parameters are being configured Rang : 0x00 to 0xEF
uint16_t	min_interval uint16_t, Minimum Interval,Minimum advertising interval for periodic advertising.Range: 0x0006 to 0xFFFF
uint16_t	max_interval uint16_t, Maximum Interval,Maximum advertising interval for periodic advertising.Range: 0x0006 to 0xFFFF
uint16_t	properties uint16_t, Periodic Advertising Properties, this parameter indicates which fields should be included in the advertising packet Bit Number, 6: Include TxPower in the advertising PDU All other Values - Reserved For future use

Public Attribute Documentation

adv_handle

> uint8_t ae_periodic_adv_params::adv_handle

uint8_t, Advertising Handle , this parameter identifies the advertising set whose periodic advertising parameters are being configured Rang : 0x00 to 0xEF

Definition at line 1327 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

min_interval

> uint16_t ae_periodic_adv_params::min_interval

uint16_t, Minimum Interval,Minimum advertising interval for periodic advertising.Range: 0x0006 to 0xFFFF

Definition at line 1329 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

max_interval

> uint16_t ae_periodic_adv_params::max_interval

uint16_t, Maximum Interval,Maximum advertising interval for periodic advertising.Range: 0x0006 to 0xFFFF

Definition at line 1331 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

properties

uint16_t ae_periodic_adv_params::properties

uint16_t, Periodic Advertising Properties, this parameter indicates which fields should be included in the advertising packet Bit Number, 6: Include TxPower in the advertising PDU All other Values - Reserved For future use

Definition at line 1335 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_periodic_adv_enable

Public Attributes

uint8_t enable
uint8_t, enable, If this parameter is set Periodic Advertising starts 0 - Enable Periodic Advertising 1 - Include the ADI field in AUX_SYNC_IND PDUs

uint8_t adv_handle
uint8_t, Advertising Handle, Used to identify an advertising set Range : 0x00 to 0xEF

Public Attribute Documentation

enable

```
uint8_t ae_periodic_adv_enable::enable
```

uint8_t, enable, If this parameter is set Periodic Advertising starts 0 - Enable Periodic Advertising 1 - Include the ADI field in AUX_SYNC_IND PDUs

Definition at line 1344 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_handle

```
uint8_t ae_periodic_adv_enable::adv_handle
```

uint8_t, Advertising Handle, Used to identify an advertising set Range : 0x00 to 0xEF

Definition at line 1348 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_scan_params_s

Public Attributes

uint8_t	ScanType uint8_t, Scan Type, this parameter specifies the type of scan to perform 0x00 - Passive Scanning.
uint16_t	ScanInterval uint16_t, Scan Interval, this parameter is a recommendation from the Host on how frequently the Controller should scan Range : 0x0004 to 0xFFFF
uint16_t	ScanWindow uint16_t, Scan Window, this parameter is a recommendation from the Host on how long the Controller should scan Range : 0x0004 to 0xFFFF

Public Attribute Documentation

ScanType

```
uint8_t ae_scan_params_s::ScanType
```

uint8_t, Scan Type, this parameter specifies the type of scan to perform 0x00 - Passive Scanning.

No scan request PDUs shall be sent. 0x01 - Active Scanning. Scan request PDUs may be sent.

Definition at line 1356 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ScanInterval

```
uint16_t ae_scan_params_s::ScanInterval
```

uint16_t, Scan Interval, this parameter is a recommendation from the Host on how frequently the Controller should scan Range : 0x0004 to 0xFFFF

Definition at line 1359 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ScanWindow

```
uint16_t ae_scan_params_s::ScanWindow
```

uint16_t, Scan Window, this parameter is a recommendation from the Host on how long the Controller should scan Range : 0x0004 to 0xFFFF

Definition at line 1362 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_set_scan_params_s

Public Attributes

uint8_t	own_addr_type unit8_t,The Own Address Type parameter indicates the type of address being used in the scan request packets Value Parameter Description 0x00 Public Device Address 0x01 Random Device Address 0x02 Controller generates the Resolvable Private Address based on the local IRK from the resolving list.
uint8_t	scanning_filter_policy uint8_t, It is used to determine whether the Filter Accept List is used
uint8_t	scanning_phys uint8_t, The Scanning_PHYs parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising physical channel.
ae_scan_params_t	ScanParams ScanParams is an array of variable of structure ae_scan_params_s.

Public Attribute Documentation

own_addr_type

```
uint8_t rsi_ble_ae_set_scan_params_s::own_addr_type
```

unit8_t,The Own Address Type parameter indicates the type of address being used in the scan request packets Value Parameter Description 0x00 Public Device Address 0x01 Random Device Address 0x02 Controller generates the Resolvable Private Address based on the local IRK from the resolving list.

If the resolving list contains no matching entry, then use the public address. 0x03 Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the random address from LE_Set_Random_Address. All other values Reserved for future use

Definition at line 1379 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

scanning_filter_policy

```
uint8_t rsi_ble_ae_set_scan_params_s::scanning_filter_policy
```

uint8_t, It is used to determine whether the Filter Accept List is used

0x00	Basic unfiltered scanning filter policy
0x01	Basic filtered scanning filter policy
0x02	Extended unfiltered scanning filter policy
0x03	Extended filtered scanning filter policy
All other values	Reserved for future use

Definition at line 1388 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

scanning_phys

uint8_t rsi_ble_ae_set_scan_params_s::scanning_phys

uint8_t, The Scanning_PHYs parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising physical channel.

```
    0          Scan advertisements on the LE 1M PHY
    2          Scan advertisements on the LE Coded PHY
All other bits     Reserved for future use
```

Definition at line 1396 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ScanParams

ae_scan_params_t rsi_ble_ae_set_scan_params_s::ScanParams[SUPPORTED_SCNNING_PHYS]

ScanParams is an array of variable of structure ae_scan_params_s.

Definition at line 1398 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_set_scan_enable_s

Public Attributes

uint8_t	enable uint8_t, Enable, this Parameter determines whether scanning is enabled or disabled
uint8_t	filter_duplicates uint8_t, Filter duplicates, this parameter controls whether the Link Layer should filter out duplicate advertising reports to the Host or if the Link Layer should generate advertising reports for each packet received
uint16_t	duration uint16_t, Duration, The duration of a scan period refers to the time spent scanning on both the primary and secondary advertising physical channels Range : 0x0001 to 0xFFFF
uint16_t	period uint16_t, Period , Time interval from when the Controller started its last Scan_Duration until it begins the subsequent Scan_Duration Range : 0x0001 to 0xFFFF

Public Attribute Documentation

enable

```
uint8_t rsi_ble_ae_set_scan_enable_s::enable
```

uint8_t, Enable, this Parameter determines whether scanning is enabled or disabled

0x00	Scanning disabled
0x01	Scanning enabled

All other values Reserved for future use

Definition at line 1409 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

filter_duplicates

```
uint8_t rsi_ble_ae_set_scan_enable_s::filter_duplicates
```

uint8_t, Filter duplicates, this parameter controls whether the Link Layer should filter out duplicate advertising reports to the Host or if the Link Layer should generate advertising reports for each packet received

0x00 Duplicate Filtering Disabled 0x01 Duplicate Filtering Enabled 0x02 Duplicate filtering enabled, reset for each scan period All other Values Reserved for future use

Definition at line 1418 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

duration

> uint16_t rsi_ble_ae_set_scan_enable_s::duration

uint16_t, Duration, The duration of a scan period refers to the time spent scanning on both the primary and secondary advertising physical channels Range : 0x0001 to 0xFFFF

Definition at line 1422 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

period

> uint16_t rsi_ble_ae_set_scan_enable_s::period

uint16_t, Period , Time interval from when the Controller started its last Scan_Duration until it begins the subsequent Scan_Duration Range : 0x0001 to 0xFFFF

Definition at line 1426 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_set_periodic_adv_create_sync_s

Public Attributes

uint8_t	fil_policy
uint8_t	adv_sid uint8_t, Advertising SID subfield in the ADI field used to identify the Periodic Advertising.
uint8_t	adv_addr_type uint8_t, Advertiser Address Type, this parameter indicates the type of address being used in the connection request packets
uint8_t	adv_addr uint8_t, Advertiser Address[6]
uint16_t	skip uint16_t, Skip,The maximum number of periodic advertising events that can be skipped after a successful receive Range : 0x0000 to 0x01F3
uint16_t	sync_timeout uint16_t, Sync Timeout, Synchronization timeout for the periodic advertising train Range : 0x000A to 0x4000
uint8_t	reserved

Public Attribute Documentation

fil_policy

```
uint8_t rsi_ble_ae_set_periodic_adv_create_sync_s::fil_policy
```

Definition at line 1432 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_sid

```
uint8_t rsi_ble_ae_set_periodic_adv_create_sync_s::adv_sid
```

uint8_t, Advertising SID subfield in the ADI field used to identify the Periodic Advertising.

Range : 0x00 to 0x0F, All other bits - Reserved for future use

Definition at line 1436 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_addr_type

```
uint8_t rsi_ble_ae_set_periodic_adv_create_sync_s::adv_addr_type
```

uint8_t, Advertiser Address Type, this parameter indicates the type of address being used in the connection request packets

0x00	Public Device Address or Public Identity Address
0x01	Random Device Address or Random (static) Identity Address

All other values Reserved for future use

Definition at line 1443 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_addr

```
uint8_t rsi_ble_ae_set_periodic_adv_create_sync_s::adv_addr[RSI_DEV_ADDR_LEN]
```

uint8_t, Advertiser Address[6]

Definition at line 1445 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

skip

```
uint16_t rsi_ble_ae_set_periodic_adv_create_sync_s::skip
```

uint16_t, Skip,The maximum number of periodic advertising events that can be skipped after a successful receive Range : 0x0000 to 0x01F3

Definition at line 1448 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

sync_timeout

```
uint16_t rsi_ble_ae_set_periodic_adv_create_sync_s::sync_timeout
```

uint16_t, Sync Timeout, Synchronization timeout for the periodic advertising train Range : 0x000A to 0x4000

Definition at line 1451 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

reserved

```
uint8_t rsi_ble_ae_set_periodic_adv_create_sync_s::reserved
```

Definition at line 1452 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_set_periodic_adv_terminate_sync_s

Public Attributes

uint16_t	sync_handle uint16_t, Sync Handle, identifies the periodic Advertising Train Range : 0x0000 to 0x0EFF

Public Attribute Documentation

sync_handle

```
uint16_t rsi_ble_ae_set_periodic_adv_terminate_sync_s::sync_handle
```

uint16_t, Sync Handle, identifies the periodic Advertising Train Range : 0x0000 to 0x0EFF

Definition at line 1458 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_set_periodic_sync_s

Public Attributes

uint8_t	type
rsi_ble_ae_set_periodic_adv_create_sync_t	create_sync
rsi_ble_ae_set_periodic_adv_terminate_sync_t	terminate_sync

Public Functions

union rsi_ble_ae_set_periodic_sync_s::@0	__attribute__((__packed__)) sync_type

Public Attribute Documentation

type

```
uint8_t rsi_ble_ae_set_periodic_sync_s::type
```

Definition at line 1467 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

create_sync

```
rsi_ble_ae_set_periodic_adv_create_sync_t rsi_ble_ae_set_periodic_sync_s::create_sync
```

Definition at line 1469 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

terminate_sync

```
rsi_ble_ae_set_periodic_adv_terminate_sync_t rsi_ble_ae_set_periodic_sync_s::terminate_sync
```

Definition at line 1470 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

Public Function Documentation

__attribute__

union rsi_ble_ae_set_periodic_sync_s::@0 rsi_ble_ae_set_periodic_sync_s::__attribute__ ((__packed__)) sync_type

Parameters

N/A		

Definition at line 1467 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_dev_to_periodic_list_s

Public Attributes

uint8_t	type uint8_t, Type Type Values Description 1 Adding Device to Periodic Advertising list 2 Removing Device from Periodic Advertising list 3 Clearing Periodic Advertising List
uint8_t	adv_addr_type uint8_t, Advertiser Address Type, this parameter indicates the type of address being used in the connection request packets
uint8_t	adv_addr uint8_t, Advertiser Address[6]
uint8_t	adv_sid uint8_t, Advertising_Sid, Value of the Advertising SID subfield in the ADI field of the PDU, Range : 0x00 to 0x0F

Public Attribute Documentation

type

```
uint8_t rsi_ble_ae_dev_to_periodic_list_s::type
```

uint8_t, Type Type Values Description 1 Adding Device to Periodic Advertising list 2 Removing Device from Periodic Advertising list 3 Clearing Periodic Advertising List

Definition at line `1482` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

adv_addr_type

```
uint8_t rsi_ble_ae_dev_to_periodic_list_s::adv_addr_type
```

uint8_t, Advertiser Address Type, this parameter indicates the type of address being used in the connection request packets

0×00	Public Device Address or Public Identity Address
0×01	Random Device Address or Random (static) Identity Address

All other values Reserved for future use

Definition at line `1489` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

adv_addr

```
uint8_t rsi_ble_ae_dev_to_periodic_list_s::adv_addr[RSI_DEV_ADDR_LEN]
```

uint8_t, Advertiser Address[6]

Definition at line 1491 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

adv_sid

```
uint8_t rsi_ble_ae_dev_to_periodic_list_s::adv_sid
```

uint8_t, Advertising_Sid, Value of the Advertising SID subfield in the ADI field of the PDU, Range : 0x00 to 0x0F

Definition at line 1493 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_initiation_params_s

Public Attributes

uint16_t	ScanInterval uint16_t, ScanInterval, It is the Time interval from when the Controller started its last scan until it begins the subsequent scan on the primary advertising physical channel.
uint16_t	ScanWindow uint16_t, Scan Window parameter is a recommendation from the host on how long the controller should scan.
uint16_t	ConnIntervalMin uint16_t, Connection interval minimum parameter defines the minimum allowed connection interval.
uint16_t	ConnIntervalMax uint16_t, Connection interval maximum parameter defines the maximum allowed connection interval.
uint16_t	ConnLatency uint16_t, Connection Latency or Maximum Latency parameter defines the maximum allowed Peripheral latency.
uint16_t	ConnSTO uint16_t,Connection Timeout or Supervision Timeout parameter defines the link supervision timeout for the connection.
uint16_t	MinCELen uint16_t,The Min CE Length parameter provide the Controller with the expected minimum length of the connection events.
uint16_t	MaxCELen uint16_t,The Max CE Length parameter provide the controller with the expected maximum length of the connection events.

Public Attribute Documentation

ScanInterval

> uint16_t rsi_ble_initiation_params_s::ScanInterval

uint16_t, ScanInterval, It is the Time interval from when the Controller started its last scan until it begins the subsequent scan on the primary advertising physical channel.

Range : 0x0004 to 0xFFFF

Definition at line `1499` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h`

ScanWindow

> uint16_t rsi_ble_initiation_params_s::ScanWindow

uint16_t, Scan Window parameter is a recommendation from the host on how long the controller should scan.

Range : 0x0004 to 0xFFFF

Definition at line 1502 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ConnIntervalMin

```
uint16_t rsi_ble_initiation_params_s::ConnIntervalMin
```

uint16_t, Connection interval minimum parameter defines the minimum allowed connection interval.

Range: 0x0006 to 0x0C80

Definition at line 1505 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ConnIntervalMax

```
uint16_t rsi_ble_initiation_params_s::ConnIntervalMax
```

uint16_t, Connection interval maximum parameter defines the maximum allowed connection interval.

Range: 0x0006 to 0x0C80

Definition at line 1508 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ConnLatency

```
uint16_t rsi_ble_initiation_params_s::ConnLatency
```

uint16_t, Connection Latency or Maximum Latency parameter defines the maximum allowed Peripheral latency.

Range: 0x0000 to 0x01F3

Definition at line 1511 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ConnSTO

```
uint16_t rsi_ble_initiation_params_s::ConnSTO
```

uint16_t,Connection Timeout or Supervision Timeout parameter defines the link supervision timeout for the connection.

Range: 0x000A to 0x0C80

Definition at line 1514 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

MinCELen

```
uint16_t rsi_ble_initiation_params_s::MinCELen
```

uint16_t,The Min CE Length parameter provide the Controller with the expected minimum length of the connection events.

Range: 0x0000 to 0xFFFF

Definition at line 1517 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

MaxCELen

> uint16_t rsi_ble_initiation_params_s::MaxCELen

uint16_t,The Max CE Length parameter provide the controller with the expected maximum length of the connection events.

Range: 0x0000 to 0xFFFF

Definition at line 1520 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_extended_create_connect_s

Public Attributes

uint8_t	initiator_filter_policy uint8_t, Initiator Filter Policy,It is used to determine whether the Filter Accept List is used Value Parameter Description 0x00 Filter Accept List is not used to determine which advertiser to connect to Peer_Address_Type and Peer_Address shall be used.
uint8_t	own_addr_type uint8_t, Own Address Type, this parameter indicates the type of address being used in the connection request packets
uint8_t	remote_addr_type uint8_t, Remote Address Type or Peer Address Type, this parameter indicates the type of address used in the connectable advertisement sent by the peer Value Parameter Description 0x00 Public Device Address or Public Identity Address 0x01 Random Device Address or Random (static) Identity Address All other values Reserved for future use
uint8_t	remote_addr uint8_t, Remote Address or Peer Address, this parameter indicates the Peer's Public Device Address, Random (static) Device Address, Non-Resolvable Private Address, or Resolvable Private Address depending on the Peer_Address_Type parameter
uint8_t	init_phys uint8_t, Initiating PHYs, this parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising physical channel and the PHYs for which connection parameters have been specified
rsi_ble_initiation_params_t	init_params init_params is an array of Variable of Structure rsi_ble_initiation_params_s

Public Attribute Documentation

initiator_filter_policy

```
uint8_t rsi_ble_ae_extended_create_connect_s::initiator_filter_policy
```

uint8_t, Initiator Filter Policy,It is used to determine whether the Filter Accept List is used Value Parameter Description 0x00 Filter Accept List is not used to determine which advertiser to connect to Peer_Address_Type and Peer_Address shall be used.

0x01 Filter Accept List is used to determine which advertiser to connect to Peer_Address_Type and Peer_Address shall be ignored. All other values Reserved for future use

Definition at line 1531 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

own_addr_type

```
uint8_t rsi_ble_ae_extended_create_connect_s::own_addr_type
```

uint8_t, Own Address Type, this parameter indicates the type of address being used in the connection request packets

Value Parameter Description 0x00 Public Device Address 0x01 Random Device Address 0x02 Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the public address. 0x03 Controller generates the Resolvable Private Address based on the local IRK from the

resolving list. If the resolving list contains no matching entry, then use the random address from the most recent successful HCI_LE_Set_Random_Address command. All other values Reserved for future use

Definition at line 1544 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

remote_addr_type

> uint8_t rsi_ble_ae_extended_create_connect_s::remote_addr_type

uint8_t, Remote Address Type or Peer Address Type, this parameter indicates the type of address used in the connectable advertisement sent by the peer Value Parameter Description 0x00 Public Device Address or Public Identity Address 0x01 Random Device Address or Random (static) Identity Address All other values Reserved for future use

Definition at line 1552 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

remote_addr

> uint8_t rsi_ble_ae_extended_create_connect_s::remote_addr[RSI_DEV_ADDR_LEN]

uint8_t, Remote Address or Peer Address, this parameter indicates the Peer's Public Device Address, Random (static) Device Address, Non-Resolvable Private Address, or Resolvable Private Address depending on the Peer_Address_Type parameter

Definition at line 1555 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

init_phys

> uint8_t rsi_ble_ae_extended_create_connect_s::init_phys

uint8_t, Initiating PHYs, this parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising physical channel and the PHYs for which connection parameters have been specified

Bit number	Parameter Description
0	Scan connectable advertisements on the LE 1M PHY. Connection parameters for the LE 1M PHY are provided.
1	Connection parameters for the LE 2M PHY are provided.
2	Scan connectable advertisements on the LE Coded PHY. Connection parameters for the LE Coded PHY are provided.
All other bits	Reserved for future use

Definition at line 1565 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

init_params

> rsi_ble_initiation_params_t rsi_ble_ae_extended_create_connect_s::init_params[3]

init_params is an array of Variable of Structure rsi_ble_initiation_params_s

Definition at line 1567 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_tx_pwr_s

Public Attributes

int8_t	min_tx_pwr int8_t, Minimum TX Power, Range: -127 to +20
int8_t	max_tx_pwr int8_t, Maximum TX Power, Range: -127 to +20

Public Attribute Documentation

min_tx_pwr

```
int8_t rsi_ble_tx_pwr_s::min_tx_pwr
```

int8_t, Minimum TX Power, Range: -127 to +20

Definition at line 1573 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

max_tx_pwr

```
int8_t rsi_ble_tx_pwr_s::max_tx_pwr
```

int8_t, Maximum TX Power, Range: -127 to +20

Definition at line 1575 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_query_rf_path_comp_s

Public Attributes

int16_t	tx_path_value int16_t, RF TX Path Compensation Value, Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)
int16_t	rx_path_value int16_t, RF RX Path Compensation Value, Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)

Public Attribute Documentation

tx_path_value

```
int16_t rsi_ble_query_rf_path_comp_s::tx_path_value
```

int16_t, RF TX Path Compensation Value, Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)

Definition at line 1581 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rx_path_value

```
int16_t rsi_ble_query_rf_path_comp_s::rx_path_value
```

int16_t, RF RX Path Compensation Value, Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)

Definition at line 1583 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_write_rf_path_comp_s

Public Attributes

int16_t	tx_path_value int16_t, RF TX Path Compensation Value, Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)
int16_t	rx_path_value int16_t, RF RX Path Compensation Value, Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)

Public Attribute Documentation

tx_path_value

```
int16_t rsi_ble_write_rf_path_comp_s::tx_path_value
```

int16_t, RF TX Path Compensation Value, Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)

Definition at line 1589 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rx_path_value

```
int16_t rsi_ble_write_rf_path_comp_s::rx_path_value
```

int16_t, RF RX Path Compensation Value, Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)

Definition at line 1591 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

rsi_ble_ae_pdu

Public Attributes

uint16_t	cmd_sub_opcode
rsi_ble_ae_get_supported_no_of_adv_sets_t	ae_supported_no_of_sets
rsi_ble_ae_read_supported_max_adv_data_t	ae_supported_max_data
rsi_ble_ae_set_random_address_t	ae_random_address
rsi_ble_ae_adv_params_t	ae_adv_params
rsi_ble_ae_data_t	ae_adv_or_scn_rsp_data
rsi_ble_ae_adv_enable_t	ae_adv_enable
rsi_ble_ae_adv_set_clear_or_remove_t	ae_adv_set_clear_or_remove
rsi_ble_ae_periodic_adv_params_t	ae_periodic_adv_params
rsi_ble_ae_periodic_adv_enable_t	ae_periodic_adv_enable
rsi_ble_ae_set_scan_params_t	ae_scan_params
rsi_ble_ae_set_scan_enable_t	ae_scan_enable
rsi_ble_ae_set_periodic_sync_t	ae_periodic_sync
rsi_ble_ae_dev_to_periodic_list_t	dev_to_periodic_list
rsi_ble_ae_extended_create_connect_t	extended_create_conn

Public Functions

union rsi_ble_ae_pdu::@2	__attribute__((__packed__)) pdu_type

Public Attribute Documentation

cmd_sub_opcode

uint16_t rsi_ble_ae_pdu::cmd_sub_opcode

Definition at line 1595 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_supported_no_of_sets

rsi_ble_ae_get_supported_no_of_adv_sets_t rsi_ble_ae_pdu::ae_supported_no_of_sets

Definition at line 1597 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_supported_max_data

rsi_ble_ae_read_supported_max_adv_data_t rsi_ble_ae_pdu::ae_supported_max_data

Definition at line 1598 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_random_address

rsi_ble_ae_set_random_address_t rsi_ble_ae_pdu::ae_random_address

Definition at line 1599 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_adv_params

rsi_ble_ae_adv_params_t rsi_ble_ae_pdu::ae_adv_params

Definition at line 1600 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_adv_or_scn_rsp_data

rsi_ble_ae_data_t rsi_ble_ae_pdu::ae_adv_or_scn_rsp_data

Definition at line 1601 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_adv_enable

rsi_ble_ae_adv_enable_t rsi_ble_ae_pdu::ae_adv_enable

Definition at line 1602 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_adv_set_clear_or_remove

> rsi_ble_ae_adv_set_clear_or_remove_t rsi_ble_ae_pdu::ae_adv_set_clear_or_remove

Definition at line 1603 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_periodic_adv_params

> rsi_ble_ae_periodic_adv_params_t rsi_ble_ae_pdu::ae_periodic_adv_params

Definition at line 1604 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_periodic_adv_enable

> rsi_ble_ae_periodic_adv_enable_t rsi_ble_ae_pdu::ae_periodic_adv_enable

Definition at line 1605 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_scan_params

> rsi_ble_ae_set_scan_params_t rsi_ble_ae_pdu::ae_scan_params

Definition at line 1607 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_scan_enable

> rsi_ble_ae_set_scan_enable_t rsi_ble_ae_pdu::ae_scan_enable

Definition at line 1608 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

ae_periodic_sync

> rsi_ble_ae_set_periodic_sync_t rsi_ble_ae_pdu::ae_periodic_sync

Definition at line 1610 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

dev_to_periodic_list

> rsi_ble_ae_dev_to_periodic_list_t rsi_ble_ae_pdu::dev_to_periodic_list

Definition at line 1611 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

extended_create_conn

> rsi_ble_ae_extended_create_connect_t rsi_ble_ae_pdu::extended_create_conn

Definition at line 1613 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

Public Function Documentation

__attribute__

> union rsi_ble_ae_pdu::@2 rsi_ble_ae_pdu::__attribute__ ((__packed__)) pdu_type

Parameters

N/A		

Definition at line 1595 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble.h

profile_descriptor_s

Public Attributes

uint8_t	start_handle Start handle.
uint8_t	end_handle End handle.
uuid_t	profile_uuid profile uuid.

Public Attribute Documentation

start_handle

```
uint8_t profile_descriptor_s::start_handle[2]
```

Start handle.

Definition at line 436 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

end_handle

```
uint8_t profile_descriptor_s::end_handle[2]
```

End handle.

Definition at line 438 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

profile_uuid

```
uuid_t profile_descriptor_s::profile_uuid
```

profile uuid.

Definition at line 440 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_req_add_att_s

Public Attributes

void *	serv_handler service handler
uint16_t	handle Attribute handle.
uint16_t	config_bitmap If this variable is 1, Host has to maintain attributes and records in the application.
uuid_t	att_uuid Attribute type UUID.
uint8_t	property Attribute property.
uint16_t	data_len Attribute data length.
uint8_t	data Attribute data.

Public Attribute Documentation

serv_handler

```
void* rsi_ble_req_add_att_s::serv_handler
```

service handler

Definition at line `498` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

handle

```
uint16_t rsi_ble_req_add_att_s::handle
```

Attribute handle.

Definition at line `500` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

config_bitmap

```
uint16_t rsi_ble_req_add_att_s::config_bitmap
```

If this variable is 1, Host has to maintain attributes and records in the application.

- If 0, Stack will maintain the attributes and records

Definition at line 504 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_uuid

> uuid_t rsi_ble_req_add_att_s::att_uuid

Attribute type UUID.

Definition at line 506 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

property

> uint8_t rsi_ble_req_add_att_s::property

Attribute property.

Definition at line 508 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

data_len

> uint16_t rsi_ble_req_add_att_s::data_len

Attribute data length.

Definition at line 510 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

data

> uint8_t rsi_ble_req_add_att_s::data[RSI_DEV_ATT_LEN]

Attribute data.

The maximum value is 240, please refer RSI_DEV_ATT_LEN Macro

Definition at line 512 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Event Types

Event Types

This section provides a reference to Bluetooth Low Energy (BLE) API events.

Modules

rsi_ble_event_read_by_type1_s

rsi_ble_event_read_by_type2_s

rsi_ble_event_read_by_type3_s

rsi_ble_event_att_value_s

rsi_ble_set_att_resp_s

rsi_ble_prepare_write_resp_s

rsi_ble_resp_profiles_list_s

rsi_ble_resp_query_profile_descriptor_s

rsi_ble_resp_char_serv_s

rsi_ble_resp_inc_serv

rsi_ble_resp_att_value_s

rsi_ble_resp_att_descs_s

rsi_ble_resp_add_serv_s

rsi_ble_resp_local_att_value_s

rsi_ble_event_remote_device_info_s

rsi_ble_event_rcp_rcvd_info_s

rsi_ble_event_write_s

rsi_ble_event_prepare_write_s

rsi_ble_execute_write_s

rsi_ble_read_req_s

rsi_ble_event_mtu_s

rsi_ble_event_mtu_exchange_information_s

rsi_ble_event_notify_s

rsi_ble_event_indication_s

rsi_ble_event_directedadv_report_s

Typedefs

typedef struct rsi_ble_event_adv_report_s	rsi_ble_event_adv_report_t
typedef struct rsi_ble_event_conn_status_s	rsi_ble_event_conn_status_t
typedef struct rsi_ble_event_enhance_conn_status_s	rsi_ble_event_enhance_conn_status_t

typedef struct rsi_ble_event_disconnect_s	rsi_ble_event_disconnect_t
typedef struct rsi_ble_event_le_ping_time_expired_s	rsi_ble_event_le_ping_time_expired_t
typedef struct rsi_bt_event_le_ltk_request_s	rsi_bt_event_le_ltk_request_t
typedef struct rsi_bt_event_le_security_keys_s	rsi_bt_event_le_security_keys_t
typedef struct rsi_bt_event_encryption_enabled_s	rsi_bt_event_encryption_enabled_t
typedef struct rsi_bt_event_smp_req_s	rsi_bt_event_smp_req_t
typedef struct rsi_bt_event_smp_resp_s	rsi_bt_event_smp_resp_t
typedef struct rsi_bt_event_smp_passkey_s	rsi_bt_event_smp_passkey_t
typedef struct rsi_bt_event_smp_passkey_display_s	rsi_bt_event_smp_passkey_display_t
typedef struct rsi_bt_event_sc_passkey_s	rsi_bt_event_sc_passkey_t
typedef struct rsi_bt_event_smp_failed_s	rsi_bt_event_smp_failed_t
typedef struct rsi_bt_event_sc_method_s	rsi_bt_event_sc_method_t
typedef struct rsi_bt_event_ctkd_s	rsi_ble_event_ctkd_t
typedef struct rsi_ble_event_phy_update_s	rsi_ble_event_phy_update_t
typedef struct rsi_ble_event_conn_update_s	rsi_ble_event_conn_update_t
typedef struct rsi_ble_event_remote_conn_param_req_s	rsi_ble_event_remote_conn_param_req_t

typedef struct rsi_ble_event_remote_features_s	rsi_ble_event_remote_features_t
typedef struct rsi_ble_event_le_dev_buf_ind_s	rsi_ble_event_le_dev_buf_ind_t
typedef struct rsi_ble_event_data_length_update_s	rsi_ble_event_data_length_update_t
typedef struct rsi_ble_event_error_resp_s	rsi_ble_event_error_resp_t
typedef struct rsi_ble_event_gatt_desc_s	rsi_ble_event_gatt_desc_t
typedef struct rsi_ble_event_profiles_list_s	rsi_ble_event_profiles_list_t
typedef struct rsi_ble_event_profile_by_uuid_s	rsi_ble_event_profile_by_uuid_t
typedef struct rsi_ble_event_read_by_type1_s	rsi_ble_event_read_by_type1_t
typedef struct rsi_ble_event_read_by_type2_s	rsi_ble_event_read_by_type2_t
typedef struct rsi_ble_event_read_by_type3_s	rsi_ble_event_read_by_type3_t
typedef struct rsi_ble_event_att_value_s	rsi_ble_event_att_value_t
typedef struct rsi_ble_set_att_resp_s	rsi_ble_set_att_resp_t
typedef struct rsi_ble_prepare_write_resp_s	rsi_ble_prepare_write_resp_t
typedef struct rsi_ble_resp_profiles_list_s	rsi_ble_resp_profiles_list_t
typedef struct rsi_ble_resp_query_profile_descriptor_s	rsi_ble_resp_query_profile_descriptor_t
typedef struct rsi_ble_resp_char_serv_s	rsi_ble_resp_char_services_t

typedef struct rsi_ble_resp_inc_serv	rsi_ble_resp_inc_services_t
typedef struct rsi_ble_resp_att_value_s	rsi_ble_resp_att_value_t
typedef struct rsi_ble_resp_att_descs_s	rsi_ble_resp_att_descs_t
typedef struct rsi_ble_resp_add_serv_s	rsi_ble_resp_add_serv_t
typedef struct rsi_ble_resp_local_att_value_s	rsi_ble_resp_local_att_value_t
typedef struct rsi_ble_event_remote_device_info_s	rsi_ble_event_remote_device_info_t
typedef struct rsi_ble_event_rcp_rcvd_info_s	rsi_ble_event_rcp_rcvd_info_t
typedef struct rsi_ble_event_write_s	rsi_ble_event_write_t
typedef struct rsi_ble_event_prepare_write_s	rsi_ble_event_prepare_write_t
typedef struct rsi_ble_execute_write_s	rsi_ble_execute_write_t
typedef struct rsi_ble_read_req_s	rsi_ble_read_req_t
typedef struct rsi_ble_event_mtu_s	rsi_ble_event_mtu_t
typedef struct rsi_ble_event_mtu_exchange_information_s	rsi_ble_event_mtu_exchange_information_t
typedef struct rsi_ble_event_notify_s	rsi_ble_event_notify_t
typedef struct rsi_ble_event_indication_s	rsi_ble_event_indication_t
typedef struct rsi_ble_event_directedadv_report_s	rsi_ble_event_directedadv_report_t

Macros

#define PEER_DEVICE_INITATED_MTU_EXCHANGE 0×1

#define LOCAL_DEVICE_INITATED_MTU_EXCHANGE 0×2

Typedef Documentation

rsi_ble_event_adv_report_t

```
typedef struct rsi_ble_event_adv_report_s rsi_ble_event_adv_report_t
```

Definition at line 84 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_conn_status_t

```
typedef struct rsi_ble_event_conn_status_s rsi_ble_event_conn_status_t
```

Definition at line 95 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_enhance_conn_status_t

```
typedef struct rsi_ble_event_enhnace_conn_status_s rsi_ble_event_enhance_conn_status_t
```

Definition at line 119 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_disconnect_t

```
typedef struct rsi_ble_event_disconnect_s rsi_ble_event_disconnect_t
```

Definition at line 128 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_le_ping_time_expired_t

```
typedef struct rsi_ble_event_le_ping_time_expired_s rsi_ble_event_le_ping_time_expired_t
```

Definition at line 135 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_le_ltk_request_t

```
typedef struct rsi_bt_event_le_ltk_request_s rsi_bt_event_le_ltk_request_t
```

Definition at line 147 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_le_security_keys_t

```
typedef struct rsi_bt_event_le_security_keys_s rsi_bt_event_le_security_keys_t
```

Definition at line 175 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_encryption_enabled_t

```
typedef struct rsi_bt_event_encryption_enabled_s rsi_bt_event_encryption_enabled_t
```

Definition at line 205 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_req_t

```
typedef struct rsi_bt_event_smp_req_s rsi_bt_event_smp_req_t
```

Definition at line 214 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_resp_t

```
typedef struct rsi_bt_event_smp_resp_s rsi_bt_event_smp_resp_t
```

Definition at line 265 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_passkey_t

```
typedef struct rsi_bt_event_smp_passkey_s rsi_bt_event_smp_passkey_t
```

Definition at line 271 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_passkey_display_t

```
typedef struct rsi_bt_event_smp_passkey_display_s rsi_bt_event_smp_passkey_display_t
```

Definition at line 279 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_sc_passkey_t

```
typedef struct rsi_bt_event_sc_passkey_s rsi_bt_event_sc_passkey_t
```

Definition at line 288 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_failed_t

```
typedef struct rsi_bt_event_smp_failed_s rsi_bt_event_smp_failed_t
```

Definition at line 294 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_sc_method_t

```
typedef struct rsi_bt_event_sc_method_s rsi_bt_event_sc_method_t
```

Definition at line 304 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_ctkd_t

```
typedef struct rsi_bt_event_ctkd_s rsi_ble_event_ctkd_t
```

Definition at line 309 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_phy_update_t

```
typedef struct rsi_ble_event_phy_update_s rsi_ble_event_phy_update_t
```

Definition at line 335 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_conn_update_t

```
typedef struct rsi_ble_event_conn_update_s rsi_ble_event_conn_update_t
```

Definition at line 347 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_remote_conn_param_req_t

```
typedef struct rsi_ble_event_remote_conn_param_req_s rsi_ble_event_remote_conn_param_req_t
```

Definition at line 361 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_remote_features_t

```
typedef struct rsi_ble_event_remote_features_s rsi_ble_event_remote_features_t
```

Definition at line 371 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_le_dev_buf_ind_t

```
typedef struct rsi_ble_event_le_dev_buf_ind_s rsi_ble_event_le_dev_buf_ind_t
```

Definition at line 379 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_data_length_update_t

```
typedef struct rsi_ble_event_data_length_update_s rsi_ble_event_data_length_update_t
```

Definition at line 393 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_error_resp_t

```
typedef struct rsi_ble_event_error_resp_s rsi_ble_event_error_resp_t
```

Definition at line 544 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_gatt_desc_t

```
typedef struct rsi_ble_event_gatt_desc_s rsi_ble_event_gatt_desc_t
```

Definition at line 555 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_profiles_list_t

```
typedef struct rsi_ble_event_profiles_list_s rsi_ble_event_profiles_list_t
```

Definition at line 566 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_profile_by_uuid_t

```
typedef struct rsi_ble_event_profile_by_uuid_s rsi_ble_event_profile_by_uuid_t
```

Definition at line 575 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_read_by_type1_t

```
typedef struct rsi_ble_event_read_by_type1_s rsi_ble_event_read_by_type1_t
```

Definition at line 586 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_read_by_type2_t

```
typedef struct rsi_ble_event_read_by_type2_s rsi_ble_event_read_by_type2_t
```

Definition at line 597 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_read_by_type3_t

```
typedef struct rsi_ble_event_read_by_type3_s rsi_ble_event_read_by_type3_t
```

Definition at line 609 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_att_value_t

```
typedef struct rsi_ble_event_att_value_s rsi_ble_event_att_value_t
```

Definition at line 619 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_att_resp_t

```
typedef struct rsi_ble_set_att_resp_s rsi_ble_set_att_resp_t
```

Definition at line 625 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_prepare_write_resp_t

```
typedef struct rsi_ble_prepare_write_resp_s rsi_ble_prepare_write_resp_t
```

Definition at line 639 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_profiles_list_t

```
typedef struct rsi_ble_resp_profiles_list_s rsi_ble_resp_profiles_list_t
```

Definition at line 651 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_query_profile_descriptor_t

```
typedef struct rsi_ble_resp_query_profile_descriptor_s rsi_ble_resp_query_profile_descriptor_t
```

Definition at line 660 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_char_services_t

```
typedef struct rsi_ble_resp_char_serv_s rsi_ble_resp_char_services_t
```

Definition at line 672 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_inc_services_t

```
typedef struct rsi_ble_resp_inc_serv rsi_ble_resp_inc_services_t
```

Definition at line 684 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_att_value_t

```
typedef struct rsi_ble_resp_att_value_s rsi_ble_resp_att_value_t
```

Definition at line 694 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_att_descs_t

```
typedef struct rsi_ble_resp_att_descs_s rsi_ble_resp_att_descs_t
```

Definition at line 706 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_add_serv_t

```
typedef struct rsi_ble_resp_add_serv_s rsi_ble_resp_add_serv_t
```

Definition at line 714 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_local_att_value_t

```
typedef struct rsi_ble_resp_local_att_value_s rsi_ble_resp_local_att_value_t
```

Definition at line 724 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_remote_device_info_t

```
typedef struct rsi_ble_event_remote_device_info_s rsi_ble_event_remote_device_info_t
```

Definition at line 731 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_rcp_rcvd_info_t

```
typedef struct rsi_ble_event_rcp_rcvd_info_s rsi_ble_event_rcp_rcvd_info_t
```

Definition at line 735 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_write_t

```
typedef struct rsi_ble_event_write_s rsi_ble_event_write_t
```

Definition at line 764 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_prepare_write_t

```
typedef struct rsi_ble_event_prepare_write_s rsi_ble_event_prepare_write_t
```

Definition at line 778 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_execute_write_t

```
typedef struct rsi_ble_execute_write_s rsi_ble_execute_write_t
```

Definition at line 786 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_read_req_t

```
typedef struct rsi_ble_read_req_s rsi_ble_read_req_t
```

Definition at line 801 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_mtu_t

```
typedef struct rsi_ble_event_mtu_s rsi_ble_event_mtu_t
```

Definition at line 809 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_mtu_exchange_information_t

```
typedef struct rsi_ble_event_mtu_exchange_information_s rsi_ble_event_mtu_exchange_information_t
```

Definition at line 827 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_notify_t

```
typedef struct rsi_ble_event_notify_s rsi_ble_event_notify_t
```

Definition at line 838 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_indication_t

```
typedef struct rsi_ble_event_indication_s rsi_ble_event_indication_t
```

Definition at line 850 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_directedadv_report_t

```
typedef struct rsi_ble_event_directedadv_report_s rsi_ble_event_directedadv_report_t
```

Definition at line 867 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Macro Definition Documentation

PEER_DEVICE_INITATED_MTU_EXCHANGE

```
#define PEER_DEVICE_INITATED_MTU_EXCHANGE
```

Value:

```
0x1
```

Definition at line 811 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

LOCAL_DEVICE_INITATED_MTU_EXCHANGE

```
#define LOCAL_DEVICE_INITATED_MTU_EXCHANGE
```

Value:

```
0x2
```

Definition at line 812 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_adv_report_s

Public Attributes

uint8_t **dev_addr_type**
Address type of the advertising device.

uint8_t **dev_addr**
Address of the advertising device.

uint8_t **adv_data_len**
Raw advertisement data length.

uint8_t **adv_data**
advertisement data

int8_t **rssi**
Signal strength.

uint8_t **report_type**
Report type

- 0x00 Connectable and scannable undirected advertising (ADV_IND)
- 0x01 Connectable directed advertising (ADV_DIRECT_IND)
- 0x02 Scannable undirected advertising (ADV_SCAN_IND)
- 0x03 Non connectable undirected advertising (ADV_NONCONN_IND)
- 0x04 Scan Response (SCAN_RSP)
- All other values Reserved for future use.

Public Attribute Documentation

dev_addr_type

```
uint8_t rsi_ble_event_adv_report_s::dev_addr_type
```

Address type of the advertising device.

Definition at line `61` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

dev_addr

```
uint8_t rsi_ble_event_adv_report_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Address of the advertising device.

Definition at line `63` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

adv_data_len

uint8_t rsi_ble_event_adv_report_s::adv_data_len

Raw advertisement data length.

Definition at line 65 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

adv_data

uint8_t rsi_ble_event_adv_report_s::adv_data[RSI_MAX_ADV_REPORT_SIZE]

advertisement data

Definition at line 67 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rssi

int8_t rsi_ble_event_adv_report_s::rssi

Signal strength.

Definition at line 69 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

report_type

uint8_t rsi_ble_event_adv_report_s::report_type

Report type

- 0x00 Connectable and scannable undirected advertising (ADV_IND)
- 0x01 Connectable directed advertising (ADV_DIRECT_IND)
- 0x02 Scannable undirected advertising (ADV_SCAN_IND)
- 0x03 Non connectable undirected advertising (ADV_NONCONN_IND)
- 0x04 Scan Response (SCAN_RSP)
- All other values Reserved for future use.

Definition at line 83 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_conn_status_s

Public Attributes

uint8_t	dev_addr_type Address type of the connected device.
uint8_t	dev_addr Address of the connected device.
uint8_t	status status of the connection - success/failure

Public Attribute Documentation

dev_addr_type

```
uint8_t rsi_ble_event_conn_status_s::dev_addr_type
```

Address type of the connected device.

Definition at line 89 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

dev_addr

```
uint8_t rsi_ble_event_conn_status_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Address of the connected device.

Definition at line 91 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

status

```
uint8_t rsi_ble_event_conn_status_s::status
```

status of the connection - success/failure

Definition at line 93 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_enhnace_conn_status_s

Public Attributes

uint8_t	dev_addr_type Device address type of the Connected Remote Device.
uint8_t	dev_addr Device address of the remote device.
uint8_t	local_resolvlable_addr Local Device resolvable address.
uint8_t	peer_resolvlable_addr Remote Device resolvable address.
uint8_t	role The role of the device - central/ peripheral.
uint16_t	conn_interval Connection interval used on this connection.
uint16_t	conn_latency Peripheral latency for the connection in number of connection events.
uint16_t	supervision_timeout Connection supervision timeout.
uint8_t	master_clock_accuracy Only applicable for peripheral, for central this value is set to 0x00.
uint8_t	status Status of the Connection - success/failure.

Public Attribute Documentation

dev_addr_type

```
uint8_t rsi_ble_event_enhnace_conn_status_s::dev_addr_type
```

Device address type of the Connected Remote Device.

Definition at line `100` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

dev_addr

```
uint8_t rsi_ble_event_enhnace_conn_status_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Device address of the remote device.

Definition at line `102` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

local_resolvlable_addr

```
uint8_t rsi_ble_event_enhnace_conn_status_s::local_resolvlable_addr[RSI_DEV_ADDR_LEN]
```

Local Device resolvable address.

Definition at line 104 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

peer_resolvlable_addr

```
uint8_t rsi_ble_event_enhnace_conn_status_s::peer_resolvlable_addr[RSI_DEV_ADDR_LEN]
```

Remote Device resolvable address.

Definition at line 106 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

role

```
uint8_t rsi_ble_event_enhnace_conn_status_s::role
```

The role of the device - central/ peripheral.

Definition at line 108 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

conn_interval

```
uint16_t rsi_ble_event_enhnace_conn_status_s::conn_interval
```

Connection interval used on this connection.

Range: 0x0006 to 0x0C80

Definition at line 110 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

conn_latency

```
uint16_t rsi_ble_event_enhnace_conn_status_s::conn_latency
```

Peripheral latency for the connection in number of connection events.

Range: 0x0000 to 0x01F3

Definition at line 112 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

supervision_timeout

```
uint16_t rsi_ble_event_enhnace_conn_status_s::supervision_timeout
```

Connection supervision timeout.

Range: 0x000A to 0x0C80

Definition at line 114 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

master_clock_accuracy

> uint8_t rsi_ble_event_enhnace_conn_status_s::master_clock_accuracy

Only applicable for peripheral, for central this value is set to 0x00.

Definition at line 116 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

status

> uint8_t rsi_ble_event_enhnace_conn_status_s::status

Status of the Connection - success/failure.

Definition at line 118 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_disconnect_s

Public Attributes

uint8_t	dev_addr device address of the disconnected device
uint8_t	dev_type The type of the disconnected device -(Classic/LE)

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_disconnect_s::dev_addr[RSI_DEV_ADDR_LEN]
```

device address of the disconnected device

Definition at line 124 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

dev_type

```
uint8_t rsi_ble_event_disconnect_s::dev_type
```

The type of the disconnected device -(Classic/LE)

Definition at line 126 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_le_ping_time_expired_s

Public Attributes

uint8_t	dev_addr Device address of the disconnected device.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_le_ping_time_expired_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Device address of the disconnected device.

Definition at line 133 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_le_ltk_request_s

Public Attributes

uint8_t	dev_addr BD Address of the remote device.
uint16_t	localediv ediv of local device
uint8_t	localrand rand of local device
uint8_t	dev_addr_type Address type of remote device.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_le_ltk_request_s::dev_addr[RSI_DEV_ADDR_LEN]
```

BD Address of the remote device.

Definition at line 140 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

localediv

```
uint16_t rsi_bt_event_le_ltk_request_s::localediv
```

ediv of local device

Definition at line 142 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

localrand

```
uint8_t rsi_bt_event_le_ltk_request_s::localrand[8]
```

rand of local device

Definition at line 144 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

dev_addr_type

```
uint8_t rsi_bt_event_le_ltk_request_s::dev_addr_type
```

Address type of remote device.

Definition at line 146 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_le_security_keys_s

Public Attributes

uint8_t	dev_addr BD Address of the remote device.
uint8_t	local_irk 16 byte irk of the local device
uint8_t	remote_irk 16 byte irk of the remote device
uint16_t	remote_ediv remote device ediv value
uint8_t	remote_rand remote device rand value
uint8_t	remote_ltk remote device ltk value
uint8_t	Identity_addr_type Identity address type - public/random • 0x00 --> Public Identity Address • 0x01 --> Random (static) Identity Address • All other values Reserved for future use.
uint8_t	Identity_addr Identity address which is resolved after security keys exchange.
uint8_t	dev_addr_type Device address type.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_le_security_keys_s::dev_addr[RSI_DEV_ADDR_LEN]
```

BD Address of the remote device.

Definition at line 152 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

local_irk

```
uint8_t rsi_bt_event_le_security_keys_s::local_irk[16]
```

16 byte irk of the local device

Definition at line 154 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_irk

```
uint8_t rsi_bt_event_le_security_keys_s::remote_irk[16]
```

16 byte irk of the remote device

Definition at line 156 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_ediv

```
uint16_t rsi_bt_event_le_security_keys_s::remote_ediv
```

remote device ediv value

Definition at line 158 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_rand

```
uint8_t rsi_bt_event_le_security_keys_s::remote_rand[16]
```

remote device rand value

Definition at line 160 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_ltk

```
uint8_t rsi_bt_event_le_security_keys_s::remote_ltk[16]
```

remote device ltk value

Definition at line 162 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Identity_addr_type

```
uint8_t rsi_bt_event_le_security_keys_s::Identity_addr_type
```

Identity address type - public/random

- 0x00 --> Public Identity Address
- 0x01 --> Random (static) Identity Address
- All other values Reserved for future use.

Definition at line 170 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Identity_addr

```
uint8_t rsi_bt_event_le_security_keys_s::Identity_addr[6]
```

Identity address which is resolved after security keys exchange.

Definition at line 172 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

dev_addr_type

> uint8_t rsi_bt_event_le_security_keys_s::dev_addr_type

Device address type.

Definition at line 174 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_encryption_enabled_s

Public Attributes

uint8_t **dev_addr**
Remote device Address.

uint8_t **enabled**
Status of the Encryption

- ENCRYPT_ENABLED BIT(0) --> To indicate or set encryption is enabled AUTH_LTK_OR_STK_ENC
- BIT(1) --> To indicate or set Authenticated Pairing and Encryption UN_AUTH_LTK_OR_STK_ENC
- BIT(2) --> To indicate or set UnAuthenticated pairing and Encryption AUTH_LTK_WITH_LE_SC_ENC
- BIT(3) --> To indicate or set Authenticated Pairing and Enc with LE SC.

uint8_t **sc_enable**
BLE Secure Connections Enable/disable indication

- 0 --> Disable
- 1 --> Enable.

uint16_t **localediv**
Local device EDIV.

uint8_t **localrand**
Local RAND.

uint8_t **localltk**
Local Long term Key.

uint8_t **dev_addr_type**
Remote Device Address type.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_encryption_enabled_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Remote device Address.

Definition at line 180 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

enabled

```
uint8_t rsi_bt_event_encryption_enabled_s::enabled
```

Status of the Encryption

- ENCRYPT_ENABLED BIT(0) --> To indicate or set encryption is enabled AUTH_LTK_OR_STK_ENC
- BIT(1) --> To indicate or set Authenticated Pairing and Encryption UN_AUTH_LTK_OR_STK_ENC
- BIT(2) --> To indicate or set UnAuthenticated pairing and Encryption AUTH_LTK_WITH_LE_SC_ENC

- BIT(3) --> To indicate or set Authenticated Pairing and Enc with LE SC.

Definition at line 190 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

sc_enable

> uint8_t rsi_bt_event_encryption_enabled_s::sc_enable

BLE Secure Connections Enable/disable indication

- 0 --> Disable
- 1 --> Enable.

Definition at line 196 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

localediv

> uint16_t rsi_bt_event_encryption_enabled_s::localediv

Local device EDIV.

Definition at line 198 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

localrand

> uint8_t rsi_bt_event_encryption_enabled_s::localrand[8]

Local RAND.

Definition at line 200 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

localltk

> uint8_t rsi_bt_event_encryption_enabled_s::localltk[16]

Local Long term Key.

Definition at line 202 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

dev_addr_type

> uint8_t rsi_bt_event_encryption_enabled_s::dev_addr_type

Remote Device Address type.

Definition at line 204 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_req_s

Public Attributes

uint8_t	dev_addr address of remote device
uint8_t	auth_req auth req of remote device

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_smp_req_s::dev_addr[6]
```

address of remote device

Definition at line 211 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

auth_req

```
uint8_t rsi_bt_event_smp_req_s::auth_req
```

auth req of remote device

Definition at line 213 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_resp_s

Public Attributes

uint8_t dev_addr
address of remote device

uint8_t io_cap
Device input output capability

- 0x00 - Display Only
- 0x01 - Display Yes/No
- 0x02 - Keyboard Only
- 0x03 - No Input No Output 0x04 - Keyboard Display.

uint8_t oob_data
Out Of the Band data.

uint8_t auth_req
Authentication Request contains bonding type

- MITM Request - BIT(2)
- Secure Connections - BIT(3)
- Keypress - BIT(4)
- CT2 - BIT(5)

uint8_t min_req_key_size
Minimum required key size.

uint8_t ini_key_distrb
Initiator generates/requires the no .of keys after successful paring

- BIT(0) - EncKey: Initiator distributes the LTK followed by EDIV and Rand
- BIT(1) - IdKey : Initiator distributes the IRK followed by its address
- BIT(2) - Sign : Initiator distributes the CSRK
- BIT(3) - BIT(7): Reserved for future use.

uint8_t resp_key_distrb
Responder generates/requires the no .of keys after successful paring

- BIT(0) - EncKey: Responder distributes the LTK followed by EDIV and Rand
- BIT(1) - IdKey : Responder distributes the IRK followed by its address
- BIT(2) - Sign : Responder distributes the CSRK
- BIT(3) - BIT(7): Reserved for future use.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_smp_resp_s::dev_addr[6]
```

address of remote device

Definition at line 219 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

io_cap

uint8_t rsi_bt_event_smp_resp_s::io_cap

Device input output capability

- 0x00 - Display Only
- 0x01 - Display Yes/No
- 0x02 - Keyboard Only
- 0x03 - No Input No Output 0x04 - Keyboard Display.

Definition at line 230 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

oob_data

uint8_t rsi_bt_event_smp_resp_s::oob_data

Out Of the Band data.

Definition at line 232 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

auth_req

uint8_t rsi_bt_event_smp_resp_s::auth_req

Authentication Request contains bonding type

- MITM Request - BIT(2)
- Secure Connections - BIT(3)
- Keypress - BIT(4)
- CT2 - BIT(5)

Definition at line 242 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

min_req_key_size

uint8_t rsi_bt_event_smp_resp_s::min_req_key_size

Minimum required key size.

Definition at line 244 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

ini_key_distrb

uint8_t rsi_bt_event_smp_resp_s::ini_key_distrb

Initiator generates/requires the no .of keys after successful paring

- BIT(0) - EncKey: Initiator distributes the LTK followed by EDIV and Rand
- BIT(1) - IdKey : Initiator distributes the IRK followed by its address
- BIT(2) - Sign : Initiator distributes the CSRK

- BIT(3) - BIT(7): Reserved for future use.

Definition at line 254 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

resp_key_distrb

> uint8_t rsi_bt_event_smp_resp_s::resp_key_distrb

Responder generates/requires the no .of keys after successful paring

- BIT(0) - EncKey: Responder distributes the LTK followed by EDIV and Rand
- BIT(1) - IdKey : Responder distributes the IRK followed by its address
- BIT(2) - Sign : Responder distributes the CSRK
- BIT(3) - BIT(7): Reserved for future use.

Definition at line 264 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_passkey_s

Public Attributes

uint8_t	dev_addr address of remote device

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_smp_passkey_s::dev_addr[6]
```

address of remote device

Definition at line 270 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_passkey_display_s

Public Attributes

uint8_t	dev_addr address of remote device
uint8_t	passkey This is the key required in pairing process(6 bytes)

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_smp_passkey_display_s::dev_addr[RSI_DEV_ADDR_LEN]
```

address of remote device

Definition at line 276 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

passkey

```
uint8_t rsi_bt_event_smp_passkey_display_s::passkey[BLE_PASSKEY_SIZE]
```

This is the key required in pairing process(6 bytes)

Definition at line 278 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_sc_passkey_s

Public Attributes

uint8_t	dev_addr address of remote device
uint8_t	reserved
uint32_t	passkey This is the key required in pairing process.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_sc_passkey_s::dev_addr[RSI_DEV_ADDR_LEN]
```

address of remote device

Definition at line 284 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_bt_event_sc_passkey_s::reserved[2]
```

Definition at line 285 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

passkey

```
uint32_t rsi_bt_event_sc_passkey_s::passkey
```

This is the key required in pairing process.

Definition at line 287 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_smp_failed_s

Public Attributes

uint8_t	dev_addr device address of the disconnected device

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_smp_failed_s::dev_addr[6]
```

device address of the disconnected device

Definition at line 293 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_sc_method_s

Public Attributes

uint8_t sc_method
Security Method --> Justworks or Passkey

- RSI_BT_LE_SC_JUST_WORKS 0x01
- RSI_BT_LE_SC_PASSKEY 0x02.

Public Attribute Documentation

sc_method

```
uint8_t rsi_bt_event_sc_method_s::sc_method
```

Security Method --> Justworks or Passkey

- RSI_BT_LE_SC_JUST_WORKS 0x01
- RSI_BT_LE_SC_PASSKEY 0x02.

Definition at line 303 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_bt_event_ctkd_s

Public Attributes

uint8_t	dev_addr
uint8_t	key

Public Attribute Documentation

dev_addr

```
uint8_t rsi_bt_event_ctkd_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 307 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

key

```
uint8_t rsi_bt_event_ctkd_s::key[16]
```

Definition at line 308 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_phy_update_s

Public Attributes

uint8_t dev_addr
Device address of the remote device.

uint8_t TxPhy
Transmission PHY rate(1 byte)

- BIT(0) - The Host prefers to use the LE 1M transmitter PHY (possibly among others)
- BIT(1) - The Host prefers to use the LE 2M transmitter PHY (possibly among others)
- BIT(2) - The Host prefers to use the LE Coded transmitter PHY (possibly among others)
- BIT(3) - BIT(7) Reserved for future use.

uint8_t RxPhy
Reception PHY rate(1 byte)

- BIT(0) - The Host prefers to use the LE 1M transmitter PHY (possibly among others)
- BIT(1) - The Host prefers to use the LE 2M transmitter PHY (possibly among others)
- BIT(2) - The Host prefers to use the LE Coded transmitter PHY (possibly among others)
- BIT(3) - BIT(7) Reserved for future use.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_phy_update_s::dev_addr[6]
```

Device address of the remote device.

Definition at line 314 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

TxPhy

```
uint8_t rsi_ble_event_phy_update_s::TxPhy
```

Transmission PHY rate(1 byte)

- BIT(0) - The Host prefers to use the LE 1M transmitter PHY (possibly among others)
- BIT(1) - The Host prefers to use the LE 2M transmitter PHY (possibly among others)
- BIT(2) - The Host prefers to use the LE Coded transmitter PHY (possibly among others)
- BIT(3) - BIT(7) Reserved for future use.

Definition at line 324 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

RxPhy

```
uint8_t rsi_ble_event_phy_update_s::RxPhy
```

Reception PHY rate(1 byte)

- BIT(0) - The Host prefers to use the LE 1M transmitter PHY (possibly among others)
- BIT(1) - The Host prefers to use the LE 2M transmitter PHY (possibly among others)
- BIT(2) - The Host prefers to use the LE Coded transmitter PHY (possibly among others)
- BIT(3) - BIT(7) Reserved for future use.

Definition at line `334` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_event_conn_update_s

Public Attributes

uint8_t	dev_addr Device address of the remote device.
uint16_t	conn_interval Connection Interval.
uint16_t	conn_latency Connection Latency.
uint16_t	timeout Supervision Timeout.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_conn_update_s::dev_addr[6]
```

Device address of the remote device.

Definition at line `340` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

conn_interval

```
uint16_t rsi_ble_event_conn_update_s::conn_interval
```

Connection Interval.

Definition at line `342` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

conn_latency

```
uint16_t rsi_ble_event_conn_update_s::conn_latency
```

Connection Latency.

Definition at line `344` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

timeout

```
uint16_t rsi_ble_event_conn_update_s::timeout
```

Supervision Timeout.

Definition at line 346 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_remote_conn_param_req_s

Public Attributes

uint8_t	dev_addr Device address of the remote device.
uint16_t	conn_interval_min Minimum connection interval.
uint16_t	conn_interval_max Maximum connection interval.
uint16_t	conn_latency Connection Latency.
uint16_t	timeout Supervision Timeout.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_remote_conn_param_req_s::dev_addr[6]
```

Device address of the remote device.

Definition at line `352` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

conn_interval_min

```
uint16_t rsi_ble_event_remote_conn_param_req_s::conn_interval_min
```

Minimum connection interval.

Definition at line `354` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

conn_interval_max

```
uint16_t rsi_ble_event_remote_conn_param_req_s::conn_interval_max
```

Maximum connection interval.

Definition at line `356` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

conn_latency

uint16_t rsi_ble_event_remote_conn_param_req_s::conn_latency

Connection Latency.

Definition at line 358 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

timeout

uint16_t rsi_ble_event_remote_conn_param_req_s::timeout

Supervision Timeout.

Definition at line 360 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_remote_features_s

Public Attributes

uint8_t	dev_addr Remote device address.
uint8_t	remote_features Remote device supported features -.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_remote_features_s::dev_addr[6]
```

Remote device address.

Definition at line 366 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_features

```
uint8_t rsi_ble_event_remote_features_s::remote_features[8]
```

Remote device supported features -.

Note

- please refer spec for the supported features list

Definition at line 370 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_le_dev_buf_ind_s

Public Attributes

uint8_t	remote_dev_bd_addr Remote device address.
uint8_t	avail_buf_cnt No.

Public Attribute Documentation

remote_dev_bd_addr

```
uint8_t rsi_ble_event_le_dev_buf_ind_s::remote_dev_bd_addr[RSI_DEV_ADDR_LEN]
```

Remote device address.

Definition at line 376 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

avail_buf_cnt

```
uint8_t rsi_ble_event_le_dev_buf_ind_s::avail_buf_cnt
```

No.

of Available buffer

Definition at line 378 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_data_length_update_s

Public Attributes

uint8_t	dev_addr Device address of the remote device.
uint16_t	MaxTxOctets Maximum TX Octets to be transmitted.
uint16_t	MaxTxTime Maximum TX time to transmit the MaxTxOctets.
uint16_t	MaxRxOctets Maximum Rx Octets to be received.
uint16_t	MaxRxTime Maximum Rx time to receive the MaxRxOctets.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_data_length_update_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Device address of the remote device.

Definition at line `384` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

MaxTxOctets

```
uint16_t rsi_ble_event_data_length_update_s::MaxTxOctets
```

Maximum TX Octets to be transmitted.

Definition at line `386` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

MaxTxTime

```
uint16_t rsi_ble_event_data_length_update_s::MaxTxTime
```

Maximum TX time to transmit the MaxTxOctets.

Definition at line `388` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

MaxRxOctets

> uint16_t rsi_ble_event_data_length_update_s::MaxRxOctets

Maximum Rx Octets to be received.

Definition at line `390` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

MaxRxTime

> uint16_t rsi_ble_event_data_length_update_s::MaxRxTime

Maximum Rx time to receive the MaxRxOctets.

Definition at line `392` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_event_error_resp_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	handle attribute handle
uint8_t	error Error indicates the type of Gatt Error.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_error_resp_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 539 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

handle

```
uint8_t rsi_ble_event_error_resp_s::handle[2]
```

attribute handle

Definition at line 541 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

error

```
uint8_t rsi_ble_event_error_resp_s::error[2]
```

Error indicates the type of Gatt Error.

Definition at line 543 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_gatt_desc_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	num_of_att number of descriptors found
uint8_t	reserved
att_desc_t	att_desc Attribute descriptors list.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_gatt_desc_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 549 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

num_of_att

```
uint8_t rsi_ble_event_gatt_desc_s::num_of_att
```

number of descriptors found

Definition at line 551 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_ble_event_gatt_desc_s::reserved
```

Definition at line 552 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_desc

```
att_desc_t rsi_ble_event_gatt_desc_s::att_desc[RSI_BLE_MAX_RESP_LIST]
```

Attribute descriptors list.

The maximum value is 5

Definition at line 554 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_profiles_list_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	number_of_profiles number of profiles found
uint8_t	reserved
profile_descriptors_t	profile_desc list of found profiles.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_profiles_list_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 560 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

number_of_profiles

```
uint8_t rsi_ble_event_profiles_list_s::number_of_profiles
```

number of profiles found

Definition at line 562 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_ble_event_profiles_list_s::reserved
```

Definition at line 563 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

profile_desc

```
profile_descriptors_t rsi_ble_event_profiles_list_s::profile_desc[RSI_BLE_MAX_RESP_LIST]
```

list of found profiles.

The maximum value is 5

Definition at line 565 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_profile_by_uuid_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	start_handle profile start handle
uint8_t	end_handle profile end handle

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_profile_by_uuid_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 570 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

start_handle

```
uint8_t rsi_ble_event_profile_by_uuid_s::start_handle[2]
```

profile start handle

Definition at line 572 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

end_handle

```
uint8_t rsi_ble_event_profile_by_uuid_s::end_handle[2]
```

profile end handle

Definition at line 574 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_read_by_type1_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	num_of_services number of characteristic services found
uint8_t	reserved
char_serv_t	char_services It contains the characteristic service list.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_read_by_type1_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 580 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

num_of_services

```
uint8_t rsi_ble_event_read_by_type1_s::num_of_services
```

number of characteristic services found

Definition at line 582 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_ble_event_read_by_type1_s::reserved
```

Definition at line 583 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

char_services

```
char_serv_t rsi_ble_event_read_by_type1_s::char_services[RSI_BLE_MAX_RESP_LIST]
```

It contains the characteristic service list.

The maximum value is 5

Definition at line 585 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_read_by_type2_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	num_of_services number of characteristic services found
uint8_t	reserved
inc_serv_t	services list of included services.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_read_by_type2_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 591 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

num_of_services

```
uint8_t rsi_ble_event_read_by_type2_s::num_of_services
```

number of characteristic services found

Definition at line 593 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_ble_event_read_by_type2_s::reserved
```

Definition at line 594 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

services

```
inc_serv_t rsi_ble_event_read_by_type2_s::services[RSI_BLE_MAX_RESP_LIST]
```

list of included services.

The maximum value is 5

Definition at line 596 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_read_by_type3_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	handle attribute handle
uint16_t	length length of attribute value
uint8_t	att_value This contains the attribute value.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_read_by_type3_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line `602` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

handle

```
uint8_t rsi_ble_event_read_by_type3_s::handle[2]
```

attribute handle

Definition at line `604` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

length

```
uint16_t rsi_ble_event_read_by_type3_s::length
```

length of attribute value

Definition at line `606` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

att_value

```
uint8_t rsi_ble_event_read_by_type3_s::att_value[RSI_DEV_ATT_LEN]
```

This contains the attribute value.

The maximum value is 240, please refer RSI_DEV_ATT_LEN Macro

Definition at line 608 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_att_value_s

Public Attributes

uint8_t	dev_addr remote device address
uint16_t	length length of attribute value
uint8_t	att_value This contains the attribute value.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_att_value_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 614 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

length

```
uint16_t rsi_ble_event_att_value_s::length
```

length of attribute value

Definition at line 616 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_value

```
uint8_t rsi_ble_event_att_value_s::att_value[RSI_DEV_ATT_LEN]
```

This contains the attribute value.

The maximum value is 240, please refer RSI_DEV_ATT_LEN Macro

Definition at line 618 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_set_att_resp_s

Public Attributes

uint8_t	dev_addr remote device address

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_set_att_resp_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 624 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_prepare_write_resp_s

Public Attributes

uint8_t	dev_addr	remote device address
uint8_t	handle	attribute handle
uint8_t	offset	attribute value offset
uint8_t	length	length of attribute value
uint8_t	att_value	This contains the attribute value.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_prepare_write_resp_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line `630` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

handle

```
uint8_t rsi_ble_prepare_write_resp_s::handle[2]
```

attribute handle

Definition at line `632` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

offset

```
uint8_t rsi_ble_prepare_write_resp_s::offset[2]
```

attribute value offset

Definition at line `634` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

length

> uint8_t rsi_ble_prepare_write_resp_s::length

length of attribute value

Definition at line 636 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_value

> uint8_t rsi_ble_prepare_write_resp_s::att_value[RSI_DEV_ATT_LEN]

This contains the attribute value.

The maximum value is 240, please refer RSI_DEV_ATT_LEN Macro

Definition at line 638 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_profiles_list_s

Public Attributes

uint8_t	number_of_profiles Number of profiles found.
uint8_t	reserved Reserved.
profile_descriptors_t	profile_desc List of found profiles • The maximum value is 5.

Public Attribute Documentation

number_of_profiles

```
uint8_t rsi_ble_resp_profiles_list_s::number_of_profiles
```

Number of profiles found.

Definition at line 644 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_ble_resp_profiles_list_s::reserved[3]
```

Reserved.

Definition at line 646 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

profile_desc

```
profile_descriptors_t rsi_ble_resp_profiles_list_s::profile_desc[RSI_BLE_MAX_RESP_LIST]
```

List of found profiles

- The maximum value is 5.

Definition at line 650 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_query_profile_descriptor_s

Public Attributes

uint8_t	dev_addr remote device address
profile_descriptors_t	profile_desc List of found profiles • The maximum value is 5.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_resp_query_profile_descriptor_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 655 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

profile_desc

```
profile_descriptors_t rsi_ble_resp_query_profile_descriptor_s::profile_desc[RSI_BLE_MAX_RESP_LIST]
```

List of found profiles

- The maximum value is 5.

Definition at line 659 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_char_serv_s

Public Attributes

uint8_t	num_of_services The number of profiles found.
uint8_t	reserved Reserved.
char_serv_t	char_services Characteristic service array.

Public Attribute Documentation

num_of_services

```
uint8_t rsi_ble_resp_char_serv_s::num_of_services
```

The number of profiles found.

Definition at line 665 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_ble_resp_char_serv_s::reserved[3]
```

Reserved.

Definition at line 667 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

char_services

```
char_serv_t rsi_ble_resp_char_serv_s::char_services[RSI_BLE_MAX_RESP_LIST]
```

Characteristic service array.

- The maximum value is 5.

Definition at line 671 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_inc_serv

Public Attributes

uint8_t	num_of_services Number of profiles found.
uint8_t	reserved Reserved.
inc_serv_t	services Include service list.

Public Attribute Documentation

num_of_services

```
uint8_t rsi_ble_resp_inc_serv::num_of_services
```

Number of profiles found.

Definition at line 677 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_ble_resp_inc_serv::reserved[3]
```

Reserved.

Definition at line 679 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

services

```
inc_serv_t rsi_ble_resp_inc_serv::services[RSI_BLE_MAX_RESP_LIST]
```

Include service list.

- The maximum value is 5.

Definition at line 683 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_att_value_s

Public Attributes

uint8_t	len Length of attribute value.
uint8_t	att_value Attribute values list.

Public Attribute Documentation

len

```
uint8_t rsi_ble_resp_att_value_s::len
```

Length of attribute value.

Definition at line `689` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

att_value

```
uint8_t rsi_ble_resp_att_value_s::att_value[RSI_DEV_ATT_LEN]
```

Attribute values list.

- Each attribute value is maximum of size 240, see RSI_DEV_ATT_LEN Macro

Definition at line `693` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_resp_att_descs_s

Public Attributes

uint8_t	num_of_att Number of descriptors found.
uint8_t	reserved Reserved.
att_desc_t	att_desc Attribute descriptors list.

Public Attribute Documentation

num_of_att

```
uint8_t rsi_ble_resp_att_descs_s::num_of_att
```

Number of descriptors found.

Definition at line 699 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

```
uint8_t rsi_ble_resp_att_descs_s::reserved[3]
```

Reserved.

Definition at line 701 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_desc

```
att_desc_t rsi_ble_resp_att_descs_s::att_desc[RSI_BLE_MAX_RESP_LIST]
```

Attribute descriptors list.

- The maximum value is 5.

Definition at line 705 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_add_serv_s

Public Attributes

void *	serv_handler Contains the address of the service record stored in the Silicon Labs stack.
uint16_t	start_handle The handle from where the service starts.

Public Attribute Documentation

serv_handler

```
void* rsi_ble_resp_add_serv_s::serv_handler
```

Contains the address of the service record stored in the Silicon Labs stack.

Definition at line 711 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

start_handle

```
uint16_t rsi_ble_resp_add_serv_s::start_handle
```

The handle from where the service starts.

Definition at line 713 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_resp_local_att_value_s

Public Attributes

uint16_t	handle Attribute handle.
uint16_t	data_len Attribute value length.
uint8_t	data Attribute value (data).

Public Attribute Documentation

handle

```
uint16_t rsi_ble_resp_local_att_value_s::handle
```

Attribute handle.

Definition at line 719 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

data_len

```
uint16_t rsi_ble_resp_local_att_value_s::data_len
```

Attribute value length.

Definition at line 721 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

data

```
uint8_t rsi_ble_resp_local_att_value_s::data[RSI_DEV_ATT_LEN]
```

Attribute value (data).

The maximum value is 240, see RSI_DEV_ATT_LEN Macro

Definition at line 723 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_remote_device_info_s

Public Attributes

uint8_t	dev_addr
uint8_t	remote_version
uint16_t	remote_company_id
uint16_t	remote_sub_version

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_remote_device_info_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line 727 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_version

```
uint8_t rsi_ble_event_remote_device_info_s::remote_version
```

Definition at line 728 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_company_id

```
uint16_t rsi_ble_event_remote_device_info_s::remote_company_id
```

Definition at line 729 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_sub_version

```
uint16_t rsi_ble_event_remote_device_info_s::remote_sub_version
```

Definition at line 730 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_rcp_rcvd_info_s

Public Attributes

uint8_t	data

Public Attribute Documentation

data

```
uint8_t rsi_ble_event_rcp_rcvd_info_s::data[1024]
```

Definition at line 734 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_write_s

Public Attributes

uint8_t **dev_addr**
remote device address

uint8_t **reserved**

uint8_t **pkt_type**
Type of the event received from the remote device

- RSI_BLE_WRITE_CMD_EVENT 0x01
- RSI_BLE_WRITE_REQUEST_EVENT 0x02
- RSI_BLE_NOTIFICATION_EVENT 0x03
- RSI_BLE_INDICATION_EVENT 0x04.

uint8_t **handle**
attribute handle

uint8_t **length**
length of attribute value

uint8_t **att_value**
This contains the attribute value.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_write_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line `742` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

reserved

```
uint8_t rsi_ble_event_write_s::reserved
```

Definition at line `743` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

pkt_type

```
uint8_t rsi_ble_event_write_s::pkt_type
```

Type of the event received from the remote device

- RSI_BLE_WRITE_CMD_EVENT 0x01

- RSI_BLE_WRITE_REQUEST_EVENT 0x02
- RSI_BLE_NOTIFICATION_EVENT 0x03
- RSI_BLE_INDICATION_EVENT 0x04.

Definition at line 757 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

handle

> uint8_t rsi_ble_event_write_s::handle[2]

attribute handle

Definition at line 759 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

length

> uint8_t rsi_ble_event_write_s::length

length of attribute value

Definition at line 761 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_value

> uint8_t rsi_ble_event_write_s::att_value[RSI_DEV_ATT_LEN]

This contains the attribute value.

The maximum value is 240, see RSI_DEV_ATT_LEN Macro

Definition at line 763 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_prepare_write_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	handle attribute handle
uint8_t	offset attribute value offset
uint16_t	length length of attribute value
uint8_t	att_value This contains the attribute value.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_prepare_write_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line `769` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

handle

```
uint8_t rsi_ble_event_prepare_write_s::handle[2]
```

attribute handle

Definition at line `771` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

offset

```
uint8_t rsi_ble_event_prepare_write_s::offset[2]
```

attribute value offset

Definition at line `773` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

length

uint16_t rsi_ble_event_prepare_write_s::length

length of attribute value

Definition at line 775 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_value

uint8_t rsi_ble_event_prepare_write_s::att_value[RSI_DEV_ATT_LEN]

This contains the attribute value.

The maximum value is 240, see RSI_DEV_ATT_LEN Macro

Definition at line 777 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_execute_write_s

Public Attributes

uint8_t	dev_addr
uint8_t	exeflag

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_execute_write_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Definition at line `783` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

exeflag

```
uint8_t rsi_ble_execute_write_s::exeflag
```

Definition at line `785` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

rsi_ble_read_req_s

Public Attributes

uint8_t	dev_addr remote device address
uint16_t	handle attribute handle
uint8_t	type 0 Read request
uint8_t	reserved
uint16_t	offset offset of attribute value to be read

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_read_req_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 791 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

handle

```
uint16_t rsi_ble_read_req_s::handle
```

attribute handle

Definition at line 793 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

type

```
uint8_t rsi_ble_read_req_s::type
```

0 Read request

- 1 Read Blob request

Definition at line 797 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

reserved

> uint8_t rsi_ble_read_req_s::reserved

Definition at line 798 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

offset

> uint16_t rsi_ble_read_req_s::offset

offset of attribute value to be read

Definition at line 800 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_mtu_s

Public Attributes

uint8_t	dev_addr remote device address
uint16_t	mtu_size MTU size.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_mtu_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 806 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

mtu_size

```
uint16_t rsi_ble_event_mtu_s::mtu_size
```

MTU size.

Definition at line 808 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_mtu_exchange_information_s

Public Attributes

uint8_t	dev_addr uint8_t[6], remote device address
uint16_t	remote_mtu_size uint8_t[2], Remote MTU Size
uint16_t	local_mtu_size uint8_t[2], Local MTU Size
uint8_t	initiated_role uint8_t Initiated role, who initiated MTU exchange • PEER_DEVICE_INITATED_MTU_EXCHANGE 0x01 • LOCAL_DEVICE_INITATED_MTU_EXCHANGE 0x02

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_mtu_exchange_information_s::dev_addr[RSI_DEV_ADDR_LEN]
```

uint8_t[6], remote device address

Definition at line 816 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

remote_mtu_size

```
uint16_t rsi_ble_event_mtu_exchange_information_s::remote_mtu_size
```

uint8_t[2], Remote MTU Size

Definition at line 818 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

local_mtu_size

```
uint16_t rsi_ble_event_mtu_exchange_information_s::local_mtu_size
```

uint8_t[2], Local MTU Size

Definition at line 820 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

initiated_role

> uint8_t rsi_ble_event_mtu_exchange_information_s::initiated_role

uint8_t Initiated role, who initiated MTU exchange

- PEER_DEVICE_INITATED_MTU_EXCHANGE 0x01
- LOCAL_DEVICE_INITATED_MTU_EXCHANGE 0x02

Definition at line 826 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_notify_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	handle attribute handle
uint8_t	length length of attribute value
uint8_t	att_value This contains the attribute value.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_notify_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 831 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

handle

```
uint8_t rsi_ble_event_notify_s::handle[2]
```

attribute handle

Definition at line 833 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

length

```
uint8_t rsi_ble_event_notify_s::length
```

length of attribute value

Definition at line 835 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_value

```
uint8_t rsi_ble_event_notify_s::att_value[RSI_DEV_ATT_LEN]
```

This contains the attribute value.

The maximum value is 240, see RSI_DEV_ATT_LEN Macro

Definition at line 837 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_indication_s

Public Attributes

uint8_t	dev_addr remote device address
uint8_t	handle attribute handle
uint8_t	length length of attribute value
uint8_t	att_value This contains the attribute value.

Public Attribute Documentation

dev_addr

```
uint8_t rsi_ble_event_indication_s::dev_addr[RSI_DEV_ADDR_LEN]
```

remote device address

Definition at line 843 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

handle

```
uint8_t rsi_ble_event_indication_s::handle[2]
```

attribute handle

Definition at line 845 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

length

```
uint8_t rsi_ble_event_indication_s::length
```

length of attribute value

Definition at line 847 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

att_value

```
uint8_t rsi_ble_event_indication_s::att_value[RSI_DEV_ATT_LEN]
```

This contains the attribute value.

The maximum value is 240, see RSI_DEV_ATT_LEN Macro

Definition at line 849 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rsi_ble_event_directedadv_report_s

Public Attributes

uint16_t	event_type Event type • 0x01 Connectable directed advertising (ADV_DIRECT_IND)
uint8_t	dev_addr_type Address type of remote device.
uint8_t	dev_addr Address of the remote device.
uint8_t	directed_addr_type Directed address type.
uint8_t	directed_dev_addr Directed device address.
int8_t	rssi rssi value

Public Attribute Documentation

event_type

```
uint16_t rsi_ble_event_directedadv_report_s::event_type
```

Event type

- 0x01 Connectable directed advertising (ADV_DIRECT_IND)

Definition at line `856` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

dev_addr_type

```
uint8_t rsi_ble_event_directedadv_report_s::dev_addr_type
```

Address type of remote device.

Definition at line `858` of file `components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h`

dev_addr

```
uint8_t rsi_ble_event_directedadv_report_s::dev_addr[RSI_DEV_ADDR_LEN]
```

Address of the remote device.

Definition at line 860 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

directed_addr_type

> uint8_t rsi_ble_event_directedadv_report_s::directed_addr_type

Directed address type.

Definition at line 862 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

directed_dev_addr

> uint8_t rsi_ble_event_directedadv_report_s::directed_dev_addr[RSI_DEV_ADDR_LEN]

Directed device address.

Definition at line 864 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

rssi

> int8_t rsi_ble_event_directedadv_report_s::rssi

rssi value

Definition at line 866 of file components/device/silabs/si91x/wireless/ble/inc/rsi_ble_apis.h

Overview

Overview

Introduction

The Network Management component provides a generic API to manage the state of all IP-based network interfaces. This includes, but is not restricted to, Ethernet, Wi-Fi, Thread, Matter, Bluetooth and Wi-SUN.

Profiles

Network interface configurations are described by "profiles" that hold all the information needed to configure a particular interface. Every type of network interface has a unique profile structure to optimally store the relevant information.

For example, a Wi-Fi client interface requires an SSID, a security mode, and passphrase (for a security enabled access point).

Credentials

Many network interfaces require secure information such as passphrases or keys to define the network. It is important that the secure information is kept separate from the network profile and so the SL Network Manager defines a credential management API that provides access to secure content via a credential ID. The storage of the credential data can be implemented in different ways depending on the support provided by the host platform.

Certificates

In addition to credential information, some networks and network services also require access to certificate storage to verify remote hosts or prove client identity. The Network Management component provides API to manage individual certificates as well as a bulk certificate store.

APIs

APIs

This section provides a reference to the Network Management API including functions, data types, and constants.

Modules

Functions

Functions

This section provides a reference to Network Management API functions:

- Network Interface functions to manage network interface configuration.
- Network Profiles functions to manage network connection and configuration profiles.
- Network Credential functions to manage network credentials including Wi-Fi client credentials, MQTT client credentials, TLS client certificates, and others.
- Multicast functions to send and receive local area network (LAN) multicast packets.

Modules

Network Interface

Network Profiles

Network Credential

Multicast

Network Interface

Functions

sl_status_t	sl_net_init(sl_net_interface_t interface, const void *configuration, void *network_context, sl_net_event_handler_t event_handler) Initialize a network interface.
sl_status_t	sl_net_deinit(sl_net_interface_t interface) De-initialize a network interface.
sl_status_t	sl_net_up(sl_net_interface_t interface, sl_net_profile_id_t profile_id) Bring a network interface up.
sl_status_t	sl_net_down(sl_net_interface_t interface) Bring a network interface down.
sl_status_t	sl_net_host_get_by_name(const char *host_name, const uint32_t timeout, const sl_net_dns_resolution_ip_type_t dns_resolution_ip, sl_ip_address_t *ip_address) Resolve given host name to IP address.

Function Documentation

sl_net_init

```
sl_status_t sl_net_init (sl_net_interface_t interface, const void *configuration, void *network_context,
sl_net_event_handler_t event_handler)
```

Initialize a network interface.

Parameters

[in]	interface	Interface identified by sl_net_interface_t
[in]	configuration	Configuration object specific to network interface of type sl_wifi_device_configuration_t. If configuration = NULL, then following configuration is used internally by SDK

sl_net_interface_t	Default configuration
SL_NET_WIFI_CLIENT_INTERFACE	sl_wifi_default_client_configuration
SL_NET_WIFI_AP_INTERFACE	sl_wifi_default_ap_configuration

[in]	network_context	Runtime context specific to network interface.
[in]	event_handler	Processes all network events related to network interface as identified by sl_net_event_handler_t

This API should be called before calling any sl_net API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.4

Note

- For Wi-Fi events, sl_net uses the wifi callback framework. Register the corresponding Wi-Fi event handlers using sl_wifi_set_callback API.

Parameter network_context is only being used when module is acting as station in external stack mode[lwIP]. In such case, network_context is supposed to refer a valid sl_net_wifi_lwip_context_t variable.

Definition at line 66 of file components/service/network_manager/inc/sl_net.h

sl_net_deinit

```
sl_status_t sl_net_deinit (sl_net_interface_t interface)
```

De-initialize a network interface.

Parameters

[in]	interface	Interface identified by sl_net_interface_t

- Pre-conditions:
  - sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 82 of file components/service/network_manager/inc/sl_net.h

sl_net_up

```
sl_status_t sl_net_up (sl_net_interface_t interface, sl_net_profile_id_t profile_id)
```

Bring a network interface up.

Parameters

[in]	interface	Interface identified by sl_net_interface_t
[in]	profile_id	Network profile identifier for specific interface of type sl_net_profile_id_t

- Pre-conditions:
  - sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- By default, profile and credential configurations in sl_net_defaults.h are used by SDK. User can define their profile and credential configurations for an interface, by calling sl_net_set_profile and sl_net_set_credentials APIs before calling sl_net_up API".

Definition at line 100 of file components/service/network_manager/inc/sl_net.h

sl_net_down

```
sl_status_t sl_net_down (sl_net_interface_t interface)
```

Bring a network interface down.

Parameters

[in]	interface	Interface identified by sl_net_interface_t

- Pre-conditions:
  - sl_net_up should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 113 of file components/service/network_manager/inc/sl_net.h

sl_net_host_get_by_name

> sl_status_t sl_net_host_get_by_name (const char *host_name, const uint32_t timeout, const sl_net_dns_resolution_ip_type_t dns_resolution_ip, sl_ip_address_t *ip_address)

Resolve given host name to IP address.

Parameters

[in]	host_name	Host name which needs to be resolved.
[in]	timeout	timeout in millisecs.
[in]	dns_resolution_ip	DNS resolution by IP of type sl_net_dns_resolution_ip_type_t
[out]	ip_address	IP address object to store resolved IP address of type sl_ip_address_t

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- If timeout value is greater than zero, caller would be blocked till timeout milliseconds to get the response. If the values is equal's to zero, response would sent through sl_net_event_handler_t.

Definition at line 40 of file components/service/network_manager/inc/sl_net_dns.h

Network Profiles

Functions

sl_status_t sl_net_set_profile(sl_net_interface_t interface, sl_net_profile_id_t id, const sl_net_profile_t *profile)
Store a network profile for a given interface.

sl_status_t sl_net_get_profile(sl_net_interface_t interface, sl_net_profile_id_t id, sl_net_profile_t *profile)
Retrieve a stored network profile for a given interface.

sl_status_t sl_net_delete_profile(sl_net_interface_t interface, sl_net_profile_id_t id)
Delete a stored network profile for a given interface.

Function Documentation

sl_net_set_profile

```
sl_status_t sl_net_set_profile (sl_net_interface_t interface, sl_net_profile_id_t id, const sl_net_profile_t *profile)
```

Store a network profile for a given interface.

Parameters

[in]	interface	Interface identified by sl_net_interface_t
[in]	id	Profile storage index / identifier of type sl_net_profile_id_t
[in]	profile	Pointer to profile data of type sl_net_profile_t

- Pre-conditions:
  - sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 137 of file components/service/network_manager/inc/sl_net.h

sl_net_get_profile

```
sl_status_t sl_net_get_profile (sl_net_interface_t interface, sl_net_profile_id_t id, sl_net_profile_t *profile)
```

Retrieve a stored network profile for a given interface.

Parameters

[in]	interface	Interface identified by sl_net_interface_t
[in]	id	Profile storage index / identifier of type sl_net_profile_id_t
[out]	profile	Pointer to sl_net_profile_t object that stores data.

- Pre-conditions:

sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 154 of file components/service/network_manager/inc/sl_net.h

sl_net_delete_profile

> sl_status_t sl_net_delete_profile (sl_net_interface_t interface, sl_net_profile_id_t id)

Delete a stored network profile for a given interface.

Parameters

[in]	interface	Interface identified by sl_net_interface_t
[in]	id	Profile storage index / identifier of type sl_net_profile_id_t

- Pre-conditions:
  - sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 169 of file components/service/network_manager/inc/sl_net.h

Network Credential

Network Credential

Functions

sl_status_t sl_net_set_credential(sl_net_credential_id_t id, sl_net_credential_type_t type, const void *credential, uint32_t credential_length)
Set a network credential including client credentials, certificates, and keys.

sl_status_t sl_net_get_credential(sl_net_credential_id_t id, sl_net_credential_type_t *type, void *credential, uint32_t *credential_length)
Retrieve a stored network credential.

sl_status_t sl_net_delete_credential(sl_net_credential_id_t id, sl_net_credential_type_t type)
Delete a stored network credential.

Function Documentation

sl_net_set_credential

> sl_status_t sl_net_set_credential (sl_net_credential_id_t id, sl_net_credential_type_t type, const void *credential, uint32_t credential_length)

Set a network credential including client credentials, certificates, and keys.

Parameters

[in]	id	Network credential identifier as identified by sl_net_credential_id_t
[in]	type	Network credential type as identified by sl_net_credential_type_t
[in]	credential	Pointer to the credential data object.
[in]	credential_length	Length of the credential data object.

- Pre-conditions:
  - sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `195` of file `components/service/network_manager/inc/sl_net.h`

sl_net_get_credential

> sl_status_t sl_net_get_credential (sl_net_credential_id_t id, sl_net_credential_type_t *type, void *credential, uint32_t *credential_length)

Retrieve a stored network credential.

Parameters

[in]	id	Network credential identifier as identified by sl_net_credential_id_t

[out]	type	Network credential type as identified by sl_net_credential_type_t
[out]	credential	Pointer to location where credential data is stored.
[inout]	credential_length	in: Number of bytes available at credential, out: Number of bytes written.

- Pre-conditions:
  - sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `218` of file `components/service/network_manager/inc/sl_net.h`

sl_net_delete_credential

> sl_status_t sl_net_delete_credential (sl_net_credential_id_t id, sl_net_credential_type_t type)

Delete a stored network credential.

Parameters

[in]	id	Network credential identifier as identified by sl_net_credential_id_t
[out]	type	Network credential type as identified by sl_net_credential_type_t

- Pre-conditions:
  - sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `236` of file `components/service/network_manager/inc/sl_net.h`

Multicast

Functions

sl_status_t	sl_net_join_multicast_address(sl_net_interface_t interface, const sl_ip_address_t *ip_address) Enable multicast for the given IP address.
sl_status_t	sl_net_leave_multicast_address(sl_net_interface_t interface, const sl_ip_address_t *ip_address) Disable multicast for the given IP address.

Function Documentation

sl_net_join_multicast_address

```
sl_status_t sl_net_join_multicast_address (sl_net_interface_t interface, const sl_ip_address_t *ip_address)
```

Enable multicast for the given IP address.

Parameters

[in]	interface	Interface identified by sl_net_interface_t
[in]	ip_address	Multicast IP address of type sl_ip_address_t

- Pre-conditions:
  - sl_net_up should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `302` of file `components/service/network_manager/inc/sl_net.h`

sl_net_leave_multicast_address

```
sl_status_t sl_net_leave_multicast_address (sl_net_interface_t interface, const sl_ip_address_t *ip_address)
```

Disable multicast for the given IP address.

Parameters

[in]	interface	Interface identified by sl_net_interface_t
[in]	ip_address	Multicast IP address of type sl_ip_address_t

- Pre-conditions:
  - sl_net_up should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 317 of file components/service/network_manager/inc/sl_net.h

Types

Types

This section provides a reference to Network Management API data types.

Modules

sl_net_wifi_lwip_context_t

sl_net_ipv4_setting_t

sl_net_ipv6_setting_t

sl_net_ip_configuration_t

sl_si91x_ping_response_t

sl_net_wifi_client_profile_t

sl_net_wifi_ap_profile_t

sl_net_wifi_psk_credential_entry_t

sl_net_wifi_eap_credential_entry_t

Typedefs

typedef sl_status_t(*	sl_net_event_handler_t)(sl_net_event_t event, sl_status_t status, void *data, uint32_t data_length) SL Net event handler.
typedef void	sl_net_profile_t SL Net abstract profile.

Typedef Documentation

sl_net_event_handler_t

```
sl_net_event_handler_t )(sl_net_event_t event, sl_status_t status, void *data, uint32_t data_length)
```

SL Net event handler.

Parameters

N/A	event	Network event of type sl_net_event_t sl_net_event_t / DataType SL_NET_PING_RESPONSE_EVENT — sl_si91x_ping_response_t SL_NET_DNS_RESOLVE_EVENT — sl_ip_address_t SL_NET_OTA_FW_UPDATE_EVENT — NULL incase of success else uint16_t chunk number incase of failure SL_NET_EVENT_COUNT — Not Applicable, Internally used by SDK
N/A	status	sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details
N/A	data	Data received.
N/A	data_length	Data length

Generic callback for network event **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `63` of file `components/service/network_manager/inc/sl_net_types.h`

sl_net_profile_t

```
typedef void sl_net_profile_t
```

SL Net abstract profile.

Definition at line `69` of file `components/service/network_manager/inc/sl_net_types.h`

sl_net_wifi_lwip_context_t

LwIP network context.

Public Attributes

struct netif	netif lwIP network interfaces

Public Attribute Documentation

netif

```
struct netif sl_net_wifi_lwip_context_t::netif
```

lwIP network interfaces

Definition at line 29 of file components/service/network_manager/inc/sl_net_for_lwip.h

sl_net_ipv4_setting_t

IPv4 address settings for a network interface.

Public Attributes

sl_ipv4_address_t	ip_address IPv4 address of type sl_ipv4_address_t.
sl_ipv4_address_t	gateway IPv4 gateway address of type sl_ipv4_address_t.
sl_ipv4_address_t	netmask IPv4 netmask of type of sl_ipv4_address_t.

Public Attribute Documentation

ip_address

```
sl_ipv4_address_t sl_net_ipv4_setting_t::ip_address
```

IPv4 address of type sl_ipv4_address_t.

Definition at line 20 of file components/service/network_manager/inc/sl_net_ip_types.h

gateway

```
sl_ipv4_address_t sl_net_ipv4_setting_t::gateway
```

IPv4 gateway address of type sl_ipv4_address_t.

Definition at line 21 of file components/service/network_manager/inc/sl_net_ip_types.h

netmask

```
sl_ipv4_address_t sl_net_ipv4_setting_t::netmask
```

IPv4 netmask of type of sl_ipv4_address_t.

Definition at line 22 of file components/service/network_manager/inc/sl_net_ip_types.h

sl_net_ipv6_setting_t

IPv6 address settings for a network interface.

Public Attributes

sl_ipv6_address_t	link_local_address IPv6 link local address of type sl_ipv6_address_t.
sl_ipv6_address_t	global_address IPv6 global address of type sl_ipv6_address_t.
sl_ipv6_address_t	gateway IPv6 gateway address of type of sl_ipv6_address_t.

Public Attribute Documentation

link_local_address

```
sl_ipv6_address_t sl_net_ipv6_setting_t::link_local_address
```

IPv6 link local address of type sl_ipv6_address_t.

Definition at line 27 of file components/service/network_manager/inc/sl_net_ip_types.h

global_address

```
sl_ipv6_address_t sl_net_ipv6_setting_t::global_address
```

IPv6 global address of type sl_ipv6_address_t.

Definition at line 28 of file components/service/network_manager/inc/sl_net_ip_types.h

gateway

```
sl_ipv6_address_t sl_net_ipv6_setting_t::gateway
```

IPv6 gateway address of type of sl_ipv6_address_t.

Definition at line 29 of file components/service/network_manager/inc/sl_net_ip_types.h

sl_net_ip_configuration_t

IP configuration for a network interface.

Public Attributes

sl_ip_management_t	mode IP Assignment Type of sl_ip_management_t.
sl_ip_address_type_t	type IP Address Type of sl_ip_address_type_t.
char *	host_name Host name visible on network.
sl_net_ipv4_setting_t	v4 IPv4 setting to be used in case of static IP address assignment of type sl_net_ipv4_setting_t.
sl_net_ipv6_setting_t	v6 IPv6 setting to be used in case of static IP address assignment of type sl_net_ipv6_setting_t.
union sl_net_ip_configuration_t::@0	ip IP setting to be used for static IP address assignment.

Public Attribute Documentation

mode

```
sl_ip_management_t sl_net_ip_configuration_t::mode
```

IP Assignment Type of sl_ip_management_t.

Definition at line 34 of file components/service/network_manager/inc/sl_net_ip_types.h

type

```
sl_ip_address_type_t sl_net_ip_configuration_t::type
```

IP Address Type of sl_ip_address_type_t.

Definition at line 35 of file components/service/network_manager/inc/sl_net_ip_types.h

host_name

```
char* sl_net_ip_configuration_t::host_name
```

Host name visible on network.

Definition at line 36 of file components/service/network_manager/inc/sl_net_ip_types.h

v4

> sl_net_ipv4_setting_t sl_net_ip_configuration_t::v4

IPv4 setting to be used in case of static IP address assignment of type sl_net_ipv4_setting_t.

Definition at line 39 of file components/service/network_manager/inc/sl_net_ip_types.h

v6

> sl_net_ipv6_setting_t sl_net_ip_configuration_t::v6

IPv6 setting to be used in case of static IP address assignment of type sl_net_ipv6_setting_t.

Definition at line 41 of file components/service/network_manager/inc/sl_net_ip_types.h

ip

> union sl_net_ip_configuration_t::@0 sl_net_ip_configuration_t::ip

IP setting to be used for static IP address assignment.

Definition at line 42 of file components/service/network_manager/inc/sl_net_ip_types.h

sl_si91x_ping_response_t

Ping Response structure.

Public Attributes

uint16_t	ip_version IP version.
uint16_t	ping_size ping size
uint8_t	ipv4_address IPv4 address.
uint8_t	ipv6_address IPv6 address.
union sl_si91x_ping_response_t::@1	ping_address Pinged IP Address.

Public Attribute Documentation

ip_version

```
uint16_t sl_si91x_ping_response_t::ip_version
```

IP version.

Definition at line 73 of file components/service/network_manager/inc/sl_net_types.h

ping_size

```
uint16_t sl_si91x_ping_response_t::ping_size
```

ping size

Definition at line 74 of file components/service/network_manager/inc/sl_net_types.h

ipv4_address

```
uint8_t sl_si91x_ping_response_t::ipv4_address[4]
```

IPv4 address.

Definition at line 76 of file components/service/network_manager/inc/sl_net_types.h

ipv6_address

```
uint8_t sl_si91x_ping_response_t::ipv6_address[16]
```

IPv6 address.

Definition at line 77 of file components/service/network_manager/inc/sl_net_types.h

ping_address

```
union sl_si91x_ping_response_t::@1 sl_si91x_ping_response_t::ping_address
```

Pinged IP Address.

Definition at line 78 of file components/service/network_manager/inc/sl_net_types.h

sl_net_wifi_client_profile_t

Network Wi-Fi client Profile.

Public Attributes

sl_wifi_client_configuration_t	config Wi-Fi client configuration of type sl_wifi_client_configuration_t
sl_net_ip_configuration_t	ip Network IP configuration of type sl_net_ip_configuration_t.

Public Attribute Documentation

config

```
sl_wifi_client_configuration_t sl_net_wifi_client_profile_t::config
```

Wi-Fi client configuration of type sl_wifi_client_configuration_t

Definition at line 29 of file components/service/network_manager/inc/sl_net_wifi_types.h

ip

```
sl_net_ip_configuration_t sl_net_wifi_client_profile_t::ip
```

Network IP configuration of type sl_net_ip_configuration_t.

Definition at line 30 of file components/service/network_manager/inc/sl_net_wifi_types.h

sl_net_wifi_ap_profile_t

Network Wi-Fi AP profile.

Public Attributes

sl_wifi_ap_configuration_t	config Wi-Fi AP configuration of type sl_wifi_client_configuration_t
sl_net_ip_configuration_t	ip Network IP configuration of type sl_net_ip_configuration_t.

Public Attribute Documentation

config

```
sl_wifi_ap_configuration_t sl_net_wifi_ap_profile_t::config
```

Wi-Fi AP configuration of type sl_wifi_client_configuration_t

Definition at line 36 of file components/service/network_manager/inc/sl_net_wifi_types.h

ip

```
sl_net_ip_configuration_t sl_net_wifi_ap_profile_t::ip
```

Network IP configuration of type sl_net_ip_configuration_t.

Definition at line 37 of file components/service/network_manager/inc/sl_net_wifi_types.h

sl_net_wifi_psk_credential_entry_t

Network Wi-Fi PSK credential entry.

Public Attributes

sl_net_credential_type_t	type Network credential type of sl_net_credential_type_t.
uint16_t	data_length Data length.
uint8_t	data Data.

Public Attribute Documentation

type

```
sl_net_credential_type_t sl_net_wifi_psk_credential_entry_t::type
```

Network credential type of sl_net_credential_type_t.

Definition at line 42 of file components/service/network_manager/inc/sl_net_wifi_types.h

data_length

```
uint16_t sl_net_wifi_psk_credential_entry_t::data_length
```

Data length.

Definition at line 43 of file components/service/network_manager/inc/sl_net_wifi_types.h

data

```
uint8_t sl_net_wifi_psk_credential_entry_t::data[196]
```

Data.

Definition at line 44 of file components/service/network_manager/inc/sl_net_wifi_types.h

sl_net_wifi_eap_credential_entry_t

Network Wi-Fi EAP credential entry.

Public Attributes

sl_net_credential_type_t	type Network credential type of sl_net_credential_type_t.
uint16_t	data_length Data length.
sl_wifi_eap_credential_t	data data of type sl_wifi_eap_credential_t

Public Attribute Documentation

type

```
sl_net_credential_type_t sl_net_wifi_eap_credential_entry_t::type
```

Network credential type of sl_net_credential_type_t.

Definition at line 49 of file components/service/network_manager/inc/sl_net_wifi_types.h

data_length

```
uint16_t sl_net_wifi_eap_credential_entry_t::data_length
```

Data length.

Definition at line 50 of file components/service/network_manager/inc/sl_net_wifi_types.h

data

```
sl_wifi_eap_credential_t sl_net_wifi_eap_credential_entry_t::data
```

data of type sl_wifi_eap_credential_t

Definition at line 52 of file components/service/network_manager/inc/sl_net_wifi_types.h

Constants

Constants

This section provides a reference to Network Management API constants.

Enumerations

enum sl_net_interface_t {

SL_NET_WIFI_CLIENT_INTERFACE = (1 << 3)
SL_NET_WIFI_AP_INTERFACE = (2 << 3)
SL_NET_ETHERNET_INTERFACE = (3 << 3)
SL_NET_THREAD_INTERFACE = (4 << 3)
SL_NET_BLUETOOTH_INTERFACE = (5 << 3)
SL_NET_ZWAVE_INTERFACE = (6 << 3)

}

Enumeration of network interfaces.

enum sl_net_packet_type_t {

SL_NET_UDP_PACKET
SL_NET_TCP_PACKET
SL_NET_ETHERNET_PACKET
SL_NET_TLS_PACKET
SL_NET_DTLS_PACKET
SL_NET_THREAD_PACKET
SL_NET_BLUETOOTH_PACKET
SL_NET_ZWAVE_PACKET
SL_NET_ZIGBEE_PACKET
SL_NET_6LOWPAN_PACKET
SL_NET_RAW_PACKET

}

Enumeration of network packet types.

enum sl_net_address_resolution_t {

SL_NET_AUTOMATIC_ADDRESS_RESOLUTION
SL_NET_STATIC_ADDRESS_RESOLUTION
SL_NET_DHCP_ADDRESS_RESOLUTION = SL_NET_AUTOMATIC_ADDRESS_RESOLUTION
SL_NET_LINK_LOCAL_ADDRESS_RESOLUTION

}

Enumeration of IP address resolution methods.

enum sl_net_dns_resolution_ip_type_t {

SL_NET_DNS_TYPE_IPV4
SL_NET_DNS_TYPE_IPV6

}

Enumeration of DNS resolution IP type.

enum sl_net_event_t {

SL_NET_PING_RESPONSE_EVENT
SL_NET_DNS_RESOLVE_EVENT
SL_NET_OTA_FW_UPDATE_EVENT
SL_NET_EVENT_COUNT

}

Enumeration of SL-Net Event.

enum sl_net_profile_id_t {

SL_NET_PROFILE_ID_0 = 0
SL_NET_PROFILE_ID_1 = 1
SL_NET_PROFILE_ID_2 = 2
SL_NET_PROFILE_ID_3 = 3
SL_NET_PROFILE_ID_4 = 4
SL_NET_PROFILE_ID_5 = 5
SL_NET_PROFILE_ID_6 = 6
SL_NET_PROFILE_ID_7 = 7
SL_NET_PROFILE_ID_8 = 8
SL_NET_PROFILE_ID_9 = 9
SL_NET_PROFILE_ID_10 = 10
SL_NET_DEFAULT_WIFI_CLIENT_PROFILE_ID = SL_NET_PROFILE_ID_0
SL_NET_DEFAULT_WIFI_AP_PROFILE_ID = SL_NET_PROFILE_ID_0
SL_NET_DEFAULT_ETHERNET_PROFILE_ID = SL_NET_PROFILE_ID_0
SL_NET_DEFAULT_THREAD_PROFILE_ID = SL_NET_PROFILE_ID_0
SL_NET_DEFAULT_ZWAVE_PROFILE_ID = SL_NET_PROFILE_ID_0

}

SL Network profile ID.

enum sl_net_credential_type_t {

SL_NET_INVALID_CREDENTIAL_TYPE
SL_NET_WIFI_PSK
SL_NET_WIFI_PMK
SL_NET_WIFI_WEP
SL_NET_CERTIFICATE
SL_NET_PUBLIC_KEY
SL_NET_PRIVATE_KEY
SL_NET_SIGNING_CERTIFICATE
SL_NET_HTTP_CLIENT_CREDENTIAL
SL_NET_EAP_CLIENT_CREDENTIAL
SL_NET_MQTT_CLIENT_CREDENTIAL

}

Enumeration of network credential types.

enum sl_net_credential_id_t {

SL_NET_INVALID_CREDENTIAL_ID = 0
SL_NET_DEFAULT_WIFI_CLIENT_CREDENTIAL_ID = 1
SL_NET_DEFAULT_WIFI_AP_CREDENTIAL_ID = 2
SL_NET_WIFI_EAP_CLIENT_CREDENTIAL_ID = 3
SL_NET_WIFI_EAP_SERVER_CREDENTIAL_ID = 4
SL_NET_USER_CREDENTIAL_ID = 5
SL_NET_TLS_CLIENT_CREDENTIAL_START = (1 << 8)
SL_NET_TLS_SERVER_CREDENTIAL_START = (2 << 8)
SL_NET_MQTT_SERVER_CREDENTIAL_START = (3 << 8)
SL_NET_MQTT_CLIENT_CREDENTIAL_START = (4 << 8)
SL_NET_HTTP_SERVER_CREDENTIAL_START = (5 << 8)
SL_NET_HTTP_CLIENT_CREDENTIAL_START = (6 << 8)

}

Enumeration of network credential identifiers.

Typedefs

typedef uint32_t sl_net_certificate_id_t
Unique Certificate store Id.

Macros

#define SL_NET_CREDENTIAL_GROUP_MASK 0xFF00
Credential Group Mask.

#define SL_NET_TLS_CLIENT_CREDENTIAL_ID (x)
TLS Client Credential ID.

#define SL_NET_TLS_SERVER_CREDENTIAL_ID (x)
TLS Server Credential ID.

#define SL_NET_MQTT_SERVER_CREDENTIAL_ID (x)
MQTT Server Credential ID.

#define SL_NET_MQTT_CLIENT_CREDENTIAL_ID (x)
MQTT Client Credential ID.

#define SL_NET_HTTP_SERVER_CREDENTIAL_ID (x)
HTTP Server Credential ID.

#define SL_NET_HTTP_CLIENT_CREDENTIAL_ID (x)
HTTP Client Credential ID.

Enumeration Documentation

sl_net_interface_t

> sl_net_interface_t

Enumeration of network interfaces.

Note

- Only Wi-Fi client and Wi-Fi access point interfaces currently supported.

	Enumerator
SL_NET_WIFI_CLIENT_INTERFACE	Wi-Fi Client Interface.
SL_NET_WIFI_AP_INTERFACE	Wi-Fi Access Point Interface.
SL_NET_ETHERNET_INTERFACE	Ethernet Interface (not currently supported)
SL_NET_THREAD_INTERFACE	Thread Interface (not currently supported)
SL_NET_BLUETOOTH_INTERFACE	Bluetooth Interface (not currently supported)
SL_NET_ZWAVE_INTERFACE	Z-Wave Interface (not currently supported)

Definition at line 47 of file components/service/network_manager/inc/sl_net_constants.h

sl_net_packet_type_t

> sl_net_packet_type_t

Enumeration of network packet types.

	Enumerator

SL_NET_UDP_PACKET	
SL_NET_TCP_PACKET	UDP Packet.
SL_NET_ETHERNET_PACKET	TCP Packet.
SL_NET_TLS_PACKET	Ethernet Packet.
SL_NET_DTLS_PACKET	TLS Packet.
SL_NET_THREAD_PACKET	DTLS Packet.
SL_NET_BLUETOOTH_PACKET	Thread Packet.
SL_NET_ZWAVE_PACKET	Bluetooth Packet.
SL_NET_ZIGBEE_PACKET	Zwave Packet.
SL_NET_6LOWPAN_PACKET	Zigbee Packet.
SL_NET_RAW_PACKET	LOWPAN Packet.

Definition at line `80` of file `components/service/network_manager/inc/sl_net_constants.h`

sl_net_address_resolution_t

> sl_net_address_resolution_t

Enumeration of IP address resolution methods.

Note

- Link local address resolution not currently supported.

Enumerator	
SL_NET_AUTOMATIC_ADDRESS_RESOLUTION	Automatic Address Resolution.
SL_NET_STATIC_ADDRESS_RESOLUTION	Static Address Resolution.
SL_NET_DHCP_ADDRESS_RESOLUTION	DHCP Address Resolution.
SL_NET_LINK_LOCAL_ADDRESS_RESOLUTION	Link Local Address Resolution (not currently supported)

Definition at line `96` of file `components/service/network_manager/inc/sl_net_constants.h`

sl_net_dns_resolution_ip_type_t

> sl_net_dns_resolution_ip_type_t

Enumeration of DNS resolution IP type.

Enumerator	
SL_NET_DNS_TYPE_IPV4	IPV4 DNS Address resolution.
SL_NET_DNS_TYPE_IPV6	IPV6 DNS Address resolution.

Definition at line `104` of file `components/service/network_manager/inc/sl_net_constants.h`

sl_net_event_t

> sl_net_event_t

Enumeration of SL-Net Event.

Enumerator	
SL_NET_PING_RESPONSE_EVENT	Ping Response Event.
SL_NET_DNS_RESOLVE_EVENT	DNS Address resolution Event.
SL_NET_OTA_FW_UPDATE_EVENT	OTA Firmware Update Event.
SL_NET_EVENT_COUNT	Maximum event count.

Definition at line 110 of file components/service/network_manager/inc/sl_net_constants.h

sl_net_profile_id_t

> sl_net_profile_id_t

SL Network profile ID.

Note

- Ethernet, Thread and Z-Wave profiles not currently supported.

Enumerator	
SL_NET_PROFILE_ID_0	Profile Id 0.
SL_NET_PROFILE_ID_1	Profile Id 1.
SL_NET_PROFILE_ID_2	Profile Id 2.
SL_NET_PROFILE_ID_3	Profile Id 3.
SL_NET_PROFILE_ID_4	Profile Id 4.
SL_NET_PROFILE_ID_5	Profile Id 5.
SL_NET_PROFILE_ID_6	Profile Id 6.
SL_NET_PROFILE_ID_7	Profile Id 7.
SL_NET_PROFILE_ID_8	Profile Id 8.
SL_NET_PROFILE_ID_9	Profile Id 9.
SL_NET_PROFILE_ID_10	Profile Id 10.
SL_NET_DEFAULT_WIFI_CLIENT_PROFILE_ID	Wi-Fi Client Default Profile.
SL_NET_DEFAULT_WIFI_AP_PROFILE_ID	Wi-Fi Access Point Default Profile.
SL_NET_DEFAULT_ETHERNET_PROFILE_ID	Ethernet Default Profile (not currently supported)
SL_NET_DEFAULT_THREAD_PROFILE_ID	Thread Default Profile (not currently supported)
SL_NET_DEFAULT_ZWAVE_PROFILE_ID	Zwave Default Profile (not currently supported)

Definition at line 119 of file components/service/network_manager/inc/sl_net_constants.h

sl_net_credential_type_t

> sl_net_credential_type_t

Enumeration of network credential types.

Enumerator	
SL_NET_INVALID_CREDENTIAL_TYPE	Invalid Credential Type.
SL_NET_WIFI_PSK	Wi-Fi PSk Credential.
SL_NET_WIFI_PMK	Wi-Fi PMK Credential.

SL_NET_WIFI_WEP	Wi-Fi WEP Credential.
SL_NET_CERTIFICATE	TLS Client Certificate.
SL_NET_PUBLIC_KEY	TLS Certificate Public key.
SL_NET_PRIVATE_KEY	TLS Certificate Private key.
SL_NET_SIGNING_CERTIFICATE	TLS CA Certificate.
SL_NET_HTTP_CLIENT_CREDENTIAL	HTTP Client Credential.
SL_NET_EAP_CLIENT_CREDENTIAL	Wi-Fi EAP Credential.
SL_NET_MQTT_CLIENT_CREDENTIAL	MQTT Client Credential.

Definition at line 140 of file components/service/network_manager/inc/sl_net_constants.h

sl_net_credential_id_t

```
sl_net_credential_id_t
```

Enumeration of network credential identifiers.

Enumerator	
SL_NET_INVALID_CREDENTIAL_ID	Invalid Credential Id.
SL_NET_DEFAULT_WIFI_CLIENT_CREDENTIAL_ID	Wi-Fi Client Credential Id.
SL_NET_DEFAULT_WIFI_AP_CREDENTIAL_ID	Wi-Fi Access Point Credential Id.
SL_NET_WIFI_EAP_CLIENT_CREDENTIAL_ID	Wi-Fi EAP Client Credential Id.
SL_NET_WIFI_EAP_SERVER_CREDENTIAL_ID	Wi-Fi EAP Server Credential Id.
SL_NET_USER_CREDENTIAL_ID	User Credential Id.
SL_NET_TLS_CLIENT_CREDENTIAL_START	TLS Client Credential Id.
SL_NET_TLS_SERVER_CREDENTIAL_START	TLS Server Credential Id.
SL_NET_MQTT_SERVER_CREDENTIAL_START	MQTT Server Credential Id.
SL_NET_MQTT_CLIENT_CREDENTIAL_START	MQTT Client Credential Id.
SL_NET_HTTP_SERVER_CREDENTIAL_START	HTTP Server Credential Id.
SL_NET_HTTP_CLIENT_CREDENTIAL_START	HTTP Client Credential Id.

Definition at line 169 of file components/service/network_manager/inc/sl_net_constants.h

Typedef Documentation

sl_net_certificate_id_t

```
typedef uint32_t sl_net_certificate_id_t
```

Unique Certificate store Id.

Definition at line 184 of file components/service/network_manager/inc/sl_net_constants.h

Macro Definition Documentation

SL_NET_CREDENTIAL_GROUP_MASK

```
#define SL_NET_CREDENTIAL_GROUP_MASK
```

Value:

```
0xFF00
```

Credential Group Mask.

Definition at line 154 of file components/service/network_manager/inc/sl_net_constants.h

SL_NET_TLS_CLIENT_CREDENTIAL_ID

```
#define SL_NET_TLS_CLIENT_CREDENTIAL_ID
```

Value:

```
(x)
```

TLS Client Credential ID.

Definition at line 156 of file components/service/network_manager/inc/sl_net_constants.h

SL_NET_TLS_SERVER_CREDENTIAL_ID

```
#define SL_NET_TLS_SERVER_CREDENTIAL_ID
```

Value:

```
(x)
```

TLS Server Credential ID.

Definition at line 158 of file components/service/network_manager/inc/sl_net_constants.h

SL_NET_MQTT_SERVER_CREDENTIAL_ID

```
#define SL_NET_MQTT_SERVER_CREDENTIAL_ID
```

Value:

```
(x)
```

MQTT Server Credential ID.

Definition at line 160 of file components/service/network_manager/inc/sl_net_constants.h

SL_NET_MQTT_CLIENT_CREDENTIAL_ID

```
#define SL_NET_MQTT_CLIENT_CREDENTIAL_ID
```

Value:

```
(x)
```

MQTT Client Credential ID.

Definition at line 162 of file components/service/network_manager/inc/sl_net_constants.h

SL_NET_HTTP_SERVER_CREDENTIAL_ID

```
#define SL_NET_HTTP_SERVER_CREDENTIAL_ID
```

Value:

```
(x)
```

HTTP Server Credential ID.

Definition at line 164 of file components/service/network_manager/inc/sl_net_constants.h

SL_NET_HTTP_CLIENT_CREDENTIAL_ID

```
#define SL_NET_HTTP_CLIENT_CREDENTIAL_ID
```

Value:

```
(x)
```

HTTP Client Credential ID.

Definition at line 166 of file components/service/network_manager/inc/sl_net_constants.h

Overview

Overview

This Module explains the various socket implementation being provided by the SDK, Which tries to emulate the standards library to the extent possible for embedded offerings from Silabs. The features along with the limitations are documented in subsequent sections.

> **Note**: The maximum number of sockets supported is 10 including client and server sockets.

APIs

APIs

This section provides a reference to the various socket API functions.

Modules

Functions

Functions

Functions

This section provides a reference to various socket implementations.

Modules

BSD Sockets

IOT Sockets

SiWx91x Sockets

Socket Configuration

BSD Sockets

BSD Sockets

The BSD socket implementation tries to emulate the standards library to the extent possible for embedded offerings from Silabs. There are substantial limitation/exceptions however, and those are mentioned in the subsequent documentation of BSD sockets.

Note

- sl_si91x_get_saved_firmware_status() needs to be called to get to know error code returned by firmware in case when API returns -1 and errno being 0.

Functions

int accept(int socket_id, struct sockaddr *addr, socklen_t *addr_len)
The argument socket_id is a socket that has been created with socket(), bound to an address with bind(), and is listening for connections after a listen().

int bind(int socket_id, const struct sockaddr *addr, socklen_t addr_len)
The bind() system call assigns the local protocol address to a socket.

int connect(int socket_id, const struct sockaddr *addr, socklen_t addr_len)
If socket_id is of type SOCK_DGRAM, connect() system call specifies the peer with which the socket is to be associated; If the socket is of type SOCK_STREAM, connect() system call attempts to make a connection to another socket.

int getpeername(int socket_id, struct sockaddr *name, socklen_t *name_len)
The getpeername() system call returns the name of the peer connected to socket socket_id.

int getsockname(int socket_id, struct sockaddr *name, socklen_t *name_len)
The getsockname() system call returns the current name for the specified socket.

int getsockopt(int socket_id, int option_level, int option_name, void *option_value, socklen_t *option_length)
The getsockopt() system call manipulate the options associated with a socket.

int listen(int socket_id, int backlog)
The listen() sysytem call listen for socket connections.

ssize_t recv(int socket_id, void *buf, size_t buf_len, int flags)
The recv() function is normally used only on a connected socket.

ssize_t recvfrom(int socket_id, void *buf, size_t buf_len, int flags, struct sockaddr *from_addr, socklen_t *from_addr_len)
The recvfrom() system calls are used to receive messages from a socket, and may be used to receive data on a socket whether or not it is connection-oriented.

ssize_t send(int socket_id, const void *buf, size_t buf_len, int flags)
The send() system call is used to transmit one or more messages to another socket.

ssize_t sendto(int socket_id, const void *buf, size_t buf_len, int flags, const struct sockaddr *to_addr, socklen_t to_addr_len)
The sendto() system calls are used to transmit one or more messages to another socket.

int setsockopt(int socket_id, int option_level, int option_name, const void *option_value, socklen_t option_length)
The setsockopt() system call set options on sockets.

int close(int socket_id)
The close() system call deletes the file descriptor of the socket.

int socket(int family, int type, int protocol)
The socket() system call creates an endpoint for communication and returns a descriptor.

Function Documentation

accept

> int accept (int socket_id, struct sockaddr *addr, socklen_t *addr_len)

The argument socket_id is a socket that has been created with socket(), bound to an address with bind(), and is listening for connections after a listen().

Parameters

[in]	socket_id	Socket identification number of the socket, which is to be accepted.
[in]	addr	The argument addr is a result argument that is filled-in with the address of the connecting entity, as known to the communications layer. The exact format of the addr argument is determined by the domain in which the communication is occurring. A null pointer may be specified for addr if the address information is not desired; in this case, addrlen is not used and should also be null.
[in]	addr_len	The addrlen argument is a value-result argument; it should initially contain the amount of space pointed to by addr; on return it will contain the actual length (in bytes) of the address returned. This call is used with connection-based socket types, currently with SOCK_STREAM.

The accept() system call extracts the first connection request on the queue of pending connections, creates a new socket, and allocates a new file descriptor for the socket.

If no pending connections are present on the queue, and the original socket is not marked as non-blocking, accept() blocks the caller until a connection is present. If the original socket is marked non-blocking and no pending connections are present on the queue, accept() returns an error. The accepted socket may not be used to accept more connections. The original socket socket_id remains open.

Returns

- These calls return -1 on error. If they succeed, they return a non-negative integer that is a descriptor for the accepted socket.

Note

- The accept() system call only supports blocking.

Definition at line `610` of file `components/service/bsd_socket/inc/socket.h`

bind

> int bind (int socket_id, const struct sockaddr *addr, socklen_t addr_len)

The bind() system call assigns the local protocol address to a socket.

Parameters

[in]	socket_id	Socket identification number.
[in]	addr	Address to be assigned to the socket.

[in]	addr_len	Size of the storage pointed to by addr.

When a socket is created with socket() it exists in an address family space but has no protocol address assigned. The bind() system call requests that addr be assigned to the socket.

Returns

- The bind() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

Definition at line 627 of file components/service/bsd_socket/inc/socket.h

connect

```
int connect (int socket_id, const struct sockaddr *addr, socklen_t addr_len)
```

If socket_id is of type SOCK_DGRAM, connect() system call specifies the peer with which the socket is to be associated; If the socket is of type SOCK_STREAM, connect() system call attempts to make a connection to another socket.

Parameters

[in]	socket_id	The socket_id argument is a socket.
[in]	addr	The addr argument is the address is that to which datagrams are to be sent.
[in]	addr_len	The addr_len argument indicates the amount of space pointed to by addr, in bytes.

The other socket is specified by addr, which is an address in the communications space of the socket. Each communications space interprets the addr argument in its own way. Generally, stream sockets may successfully connect() only once; datagram sockets may use connect() multiple times to change their association.

Returns

- The connect() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

Note

- Connecting to an invalid address, such as null address will result in EFAULT error.

Definition at line 648 of file components/service/bsd_socket/inc/socket.h

getpeername

```
int getpeername (int socket_id, struct sockaddr *name, socklen_t *name_len)
```

The getpeername() system call returns the name of the peer connected to socket socket_id.

Parameters

[in]	socket_id	Socket identification number.
[in]	name	Pointer to the peer name.
[in]	name_len	The name_len argument should be initialized to indicate the amount of space pointed to by name. On return it contains the actual size of the name returned (in bytes).

The name is truncated if the buffer provided is too small.

Returns

- The getpeername() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

Definition at line 664 of file components/service/bsd_socket/inc/socket.h

getsockname

```
int getsockname (int socket_id, struct sockaddr *name, socklen_t *name_len)
```

The getsockname() system call returns the current name for the specified socket.

Parameters

[in]	socket_id	Socket identification number.
[in]	name	Pointer to the socket name.
[out]	name_len	The name_len argument should be initialized to indicate the amount of space pointed to by name. On return it contains the actual size of the name returned (in bytes).

Returns

- The getsockname() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

Definition at line 679 of file components/service/bsd_socket/inc/socket.h

getsockopt

```
int getsockopt (int socket_id, int option_level, int option_name, void *option_value, socklen_t *option_length)
```

The getsockopt() system call manipulate the options associated with a socket.

Parameters

[in]	socket_id	Socket identification number.
[in]	option_level	Level for which the option is set.
[in]	option_name	The option_name argument and any specified options are passed uninterpreted to the appropriate protocol module for interpretation.
[in]	option_value	Pointer to option data.
[in]	option_length	The option_length is a value-result argument, initially containing the size of the buffer pointed to by option_value, and modified on return to indicate the actual size of the value returned. If no option value is to be supplied or returned, optval may be NULL.

Options may exist at multiple protocol levels; they are always present at the uppermost "socket" level. When manipulating socket options the level at which the option resides and the name of the option must be specified. To manipulate options at the socket level, level is specified as SOL_SOCKET. To manipulate options at any other level the protocol number of the appropriate protocol controlling the option is supplied. For example, to indicate that an option is to be interpreted by the TCP protocol, level should be set to the protocol number of TCP; For getsockopt() they identify a buffer in which the value for the requested option(s) are to be returned. The include file <sys/socket.h> contains definitions for socket level options. Options at other protocol levels vary in format and name. Most socket-level options utilize an int argument for option_value.

Returns

- Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

Note

- The following are the options, which are supported currently.
  - SO_RCVTIMEO
  - SO_KEEPALIVE
  - TCP_ULP

SO_MAX_RETRANSMISSION_TIMEOUT_VALUE
- IP_TOS.

Definition at line 714 of file components/service/bsd_socket/inc/socket.h

listen

```
int listen (int socket_id, int backlog)
```

The listen() sysytem call listen for socket connections.

Parameters

[in]	socket_id	Socket identification number.
[in]	backlog	The backlog argument defines the maximum number of the clients supported.

The listen() system call applies only to sockets of type SOCK_STREAM or SOCK_SEQPACKET.

- Pre-conditions:
  - To accept connections, a socket is first created with socket(), a willingness to accept incoming connections and a queue limit for incoming connections are specified with listen(), and then the connections are accepted with accept().

Returns

- The listen() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

Definition at line 730 of file components/service/bsd_socket/inc/socket.h

recv

```
ssize_t recv (int socket_id, void *buf, size_t buf_len, int flags)
```

The recv() function is normally used only on a connected socket.

Parameters

[in]	socket_id	Socket identification number.
[in]	buf	Pointer to the buffer that receives the data.
[in]	buf_len	Length of the buffer pointed to by the buf parameter.
[in]	flags	Controls the reception of the data.

recv() function Receive message(s) from a socket. The recv() return the length of the message on successful completion. If a message is too long to fit in the supplied buffer, excess bytes may be discarded depending on the type of socket the message is received from.

Returns

- ssize_t.

Note

- The recv() system call doesn't support any flags.

Definition at line 751 of file components/service/bsd_socket/inc/socket.h

recvfrom

> ssize_t recvfrom (int socket_id, void *buf, size_t buf_len, int flags, struct sockaddr *from_addr, socklen_t *from_addr_len)

The recvfrom() system calls are used to receive messages from a socket, and may be used to receive data on a socket whether or not it is connection-oriented.

Parameters

[in]	socket_id	Socket identification number.
[in]	buf	Pointer to the buffer that receives the data.
[in]	buf_len	Length of the buffer pointed to by the buf parameter.
[in]	flags	Controls the reception of the data.
[in]	from_addr	Pointer to a socket address structure from which data is received.
[in]	from_addr_len	The from_addr_len argument is a value-result argument, initialized to the size of the buffer associated with from_addr, and modified on return to indicate the actual size of the address stored there. The recvfrom() return the length of the message on successful completion.

If from_addr is not a null pointer and the socket is not connection-oriented, the source address of the message is filled in.

Returns

- ssize_t.

Note

- The recvfrom() system call doesn't support any flags.

Definition at line `778` of file `components/service/bsd_socket/inc/socket.h`

send

> ssize_t send (int socket_id, const void *buf, size_t buf_len, int flags)

The send() system call is used to transmit one or more messages to another socket.

Parameters

[in]	socket_id	Socket identification number.
[in]	buf	Pointer to the buffer containing the message to transmit.
[in]	buf_len	The length of the message is given by buf_len.
[in]	flags	Controls the transmission of the data.

The send() function may be used only when the socket is in a connected state. If the socket is connection-mode, the protocol must support implied connect or the socket must be in a connected state before use. No indication of failure to deliver is implicit in a send(). Locally detected errors are indicated by a return value of -1. If no messages space is available at the socket to hold the message to be transmitted, then send() normally blocks, unless the socket has been placed in non-blocking I/O mode.

Returns

- The send() call return the number of octets sent. If an error occurred a value of -1 is returned.

Note

- Currently, send() system call supports only blocking. The send() system call doesn't guarantees the packets are transmitted to remote note, which are enqueued in the queue. The send() system call doesn't supports any flags. The send() system call can only send max of 1460 bytes incase of plain TCP, UDP. Whereas incase of TLS, the max buffer length is 1370.

Definition at line 806 of file components/service/bsd_socket/inc/socket.h

sendto

> ssize_t sendto (int socket_id, const void *buf, size_t buf_len, int flags, const struct sockaddr *to_addr, socklen_t to_addr_len)

The sendto() system calls are used to transmit one or more messages to another socket.

Parameters

[in]	socket_id	Socket identification number.
[in]	buf	Pointer to the buffer containing the message to transmit.
[in]	buf_len	The length of the message is given by buf_len.
[in]	flags	Controls the transmission of the data.
[in]	to_addr	Address of the target.
[in]	to_addr_len	Size of the address pointed by to_addr.

The functions sendto() may be used at any time if the socket is connectionless-mode. If the socket is connection-mode, the protocol must support implied connect or the socket must be in a connected state before use. The address of the target is given by to with to_addr_len specifying its size. If the socket is in a connected state, the target address passed to sendto() is ignored. If the message is too long to pass atomically through the protocol, the error EMSGSIZE is returned, and the message is not transmitted.

Returns

- The sendto() call return the number of octets sent. If an error occurred a value of -1 is returned.

Note

- Due to firmware limitations, sendto() system call doesn't support any flags. The sendto() system call can only send max of 1460 bytes incase of plain TCP, UDP. Whereas incase of TLS, the max buffer length is 1370.

Definition at line 834 of file components/service/bsd_socket/inc/socket.h

setsockopt

> int setsockopt (int socket_id, int option_level, int option_name, const void *option_value, socklen_t option_length)

The setsockopt() system call set options on sockets.

Parameters

[in]	socket_id	Socket identification number.
[in]	option_level	Level for which the option is being set.
[in]	option_name	The option_name argument and any specified options are passed uninterpreted to the appropriate protocol module for interpretation.
[in]	option_value	Most socket-level options utilize an int argument for option_value. For setsockopt(), the argument should be non-zero to enable a boolean option, or zero if the option is to be disabled.
[in]	option_length	Pointer to the length of the option data.

The setsockopt() system call manipulate the options associated with a socket. Options may exist at multiple protocol levels; they are always present at the uppermost "socket" level. When manipulating socket options the level at which the option resides and the name of the option must be specified. To manipulate options at the socket level, level is specified as SOL_SOCKET. To manipulate options at any other level the protocol number of the appropriate protocol controlling the option is supplied. For example, to indicate that an option is to be interpreted by the TCP protocol, level should be set to

the protocol number of TCP; The option_value and option_length arguments are used to access option values for setsockopt(). The include file <sys/socket.h> contains definitions for socket level options. Options at other protocol levels vary in format and name.

Returns

- Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

Note

- The following are the options, which are supported currently.
  - SO_RCVTIMEO
  - SO_KEEPALIVE
  - TCP_ULP
  - SO_MAX_RETRANSMISSION_TIMEOUT_VALUE
  - IP_TOS.

Definition at line 871 of file components/service/bsd_socket/inc/socket.h

close

```
int close (int socket_id)
```

The close() system call deletes the file descriptor of the socket.

Parameters

[in]	socket_id	Socket identification number.

Returns

- The close() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable errno is set to indicate the error.

Note

- Calling close on server or first client socket would end up closing othet socket as well. Closing an non existing socket, error EBADF (Bad file descriptor) will be thrown.

Definition at line 884 of file components/service/bsd_socket/inc/socket.h

socket

```
int socket (int family, int type, int protocol)
```

The socket() system call creates an endpoint for communication and returns a descriptor.

Parameters

[in]	family	The domain argument specifies a communications domain within which communication will take place; this selects the protocol family which should be used. These families are defined in the include file <sys/socket.h>.
[in]	type	The socket has the indicate type, which specifies the semantics of communication. Currently defined types are SOCK_STREAM, SOCK_DGRAM, SOCK_RAW, SOCK_RDM, and SOCK_SEQPACKET.

[in]	protocol	The protocol argument specifies a particular protocol to be used with the socket. Normally only a single protocol exists to support a particular socket type within a given protocol family. However, it is possible that many protocols may exist, in which case a particular protocol must be specified in this manner. The protocol number to use is particular to the "communication domain" in which communication is to take place. The protocol argument may be set to zero (0) to request the default implementation of a socket type for the protocol, if any.

Returns

- A -1 is returned if an error occurs, otherwise the return value is a descriptor referencing the socket.

Definition at line `905` of file `components/service/bsd_socket/inc/socket.h`

IOT Sockets

IOT Sockets

The IOT socket implementation aims to emulate the standards library to the extent possible for embedded offerings from Silabs. There are substantial limitation/exceptions however, and those are mentioned in the subsequent documentation of IOT sockets.

Functions

int32_t iotSocketCreate(int32_t af, int32_t type, int32_t protocol)
Create a communication socket.

int32_t iotSocketBind(int32_t socket, const uint8_t *ip, uint32_t ip_len, uint16_t port)
Assign a local address to a socket.

int32_t iotSocketListen(int32_t socket, int32_t backlog)
Listen for socket connections.

int32_t iotSocketAccept(int32_t socket, uint8_t *ip, uint32_t *ip_len, uint16_t *port)
Accept a new connection on a socket.

int32_t iotSocketConnect(int32_t socket, const uint8_t *ip, uint32_t ip_len, uint16_t port)
Connect a socket to a remote host.

int32_t iotSocketRecv(int32_t socket, void *buf, uint32_t len)
Receives data from the socket.

int32_t iotSocketRecvFrom(int32_t socket, void *buf, uint32_t len, uint8_t *ip, uint32_t *ip_len, uint16_t *port)
Receives data from the socket.

int32_t iotSocketSend(int32_t socket, const void *buf, uint32_t len)
Send data or check if data can be sent on a connected socket.

int32_t iotSocketSendTo(int32_t socket, const void *buf, uint32_t len, const uint8_t *ip, uint32_t ip_len, uint16_t port)
Send data or check if data can be sent on a socket.

int32_t iotSocketGetSockName(int32_t socket, uint8_t *ip, uint32_t *ip_len, uint16_t *port)
Retrieve local IP address and port of a socket.

int32_t iotSocketGetPeerName(int32_t socket, uint8_t *ip, uint32_t *ip_len, uint16_t *port)
Retrieve remote IP address and port of a socket.

int32_t iotSocketGetOpt(int32_t socket, int32_t opt_id, void *opt_val, uint32_t *opt_len)
Get socket option.

int32_t iotSocketSetOpt(int32_t socket, int32_t opt_id, const void *opt_val, uint32_t opt_len)
Set socket option.

int32_t iotSocketClose(int32_t socket)
Close and release a socket.

int32_t iotSocketGetHostByName(const char *name, int32_t af, uint8_t *ip, uint32_t *ip_len)
Retrieve host IP address from host name.

Function Documentation

iotSocketCreate

> int32_t iotSocketCreate (int32_t af, int32_t type, int32_t protocol)

Create a communication socket.

Parameters

[in]	af	address family.
[in]	type	socket type.
[in]	protocol	socket protocol.

Returns

- status information:
  - Socket identification number (>=0).
  - IOT_SOCKET_EINVAL = Invalid argument.
  - IOT_SOCKET_ENOTSUP = Operation not supported.
  - IOT_SOCKET_ENOMEM = Not enough memory.
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line `105` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketBind

> int32_t iotSocketBind (int32_t socket, const uint8_t *ip, uint32_t ip_len, uint16_t port)

Assign a local address to a socket.

Parameters

[in]	socket	socket identification number.
[in]	ip	pointer to local IP address.
[in]	ip_len	length of 'ip' address in bytes.
[in]	port	local port number.

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (address or socket already bound).
  - IOT_SOCKET_EADDRINUSE = Address already in use.
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line `121` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketListen

> int32_t iotSocketListen (int32_t socket, int32_t backlog)

Listen for socket connections.

Parameters

[in]	socket	socket identification number.
[in]	backlog	maximum number of the clients supported.

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (socket not bound).
  - IOT_SOCKET_ENOTSUP = Operation not supported.
  - IOT_SOCKET_EISCONN = Socket is already connected.
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line 136 of file third_party/iot_socket/include/iot_socket.h

iotSocketAccept

```
int32_t iotSocketAccept (int32_t socket, uint8_t *ip, uint32_t *ip_len, uint16_t *port)
```

Accept a new connection on a socket.

Parameters

[in]	socket	socket identification number.
[out]	ip	pointer to buffer where address of connecting socket shall be returned (NULL for none).
[inout]	ip_len	pointer to length of 'ip' (or NULL if 'ip' is NULL): • length of supplied 'ip' on input. • length of stored 'ip' on output.
[out]	port	pointer to buffer where port of connecting socket shall be returned (NULL for none).

Returns

- status information:
  - socket identification number of accepted socket (>=0).
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (socket not in listen mode).
  - IOT_SOCKET_ENOTSUP = Operation not supported (socket type does not support accepting connections).
  - IOT_SOCKET_ECONNRESET = Connection reset by the peer.
  - IOT_SOCKET_ECONNABORTED = Connection aborted locally.
  - IOT_SOCKET_EAGAIN = Operation would block or timed out (may be called again).
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line 157 of file third_party/iot_socket/include/iot_socket.h

iotSocketConnect

```
int32_t iotSocketConnect (int32_t socket, const uint8_t *ip, uint32_t ip_len, uint16_t port)
```

Connect a socket to a remote host.

Parameters

[in]	socket	socket identification number.

[in]	ip	pointer to remote IP address.
[in]	ip_len	length of 'ip' address in bytes.
[in]	port	remote port number.

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument.
  - IOT_SOCKET_EALREADY = Connection already in progress.
  - IOT_SOCKET_EINPROGRESS = Operation in progress.
  - IOT_SOCKET_EISCONN = Socket is connected.
  - IOT_SOCKET_ECONNREFUSED = Connection rejected by the peer.
  - IOT_SOCKET_ECONNABORTED = Connection aborted locally.
  - IOT_SOCKET_EADDRINUSE = Address already in use.
  - IOT_SOCKET_ETIMEDOUT = Operation timed out.
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line `179` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketRecv

> int32_t iotSocketRecv (int32_t socket, void *buf, uint32_t len)

Receives data from the socket.

Parameters

[in]	socket	socket identification number.
[out]	buf	pointer to buffer where data should be stored.
[in]	len	length of buffer (in bytes).

Returns

- status information:
  - number of bytes received (>=0), if len != 0.
  - 0 = Data is available (len = 0).
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (pointer to buffer or length).
  - IOT_SOCKET_ENOTCONN = Socket is not connected.
  - IOT_SOCKET_ECONNRESET = Connection reset by the peer.
  - IOT_SOCKET_ECONNABORTED = Connection aborted locally.
  - IOT_SOCKET_EAGAIN = Operation would block or timed out (may be called again).
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line `198` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketRecvFrom

> int32_t iotSocketRecvFrom (int32_t socket, void *buf, uint32_t len, uint8_t *ip, uint32_t *ip_len, uint16_t *port)

Receives data from the socket.

Parameters

[in]	socket	socket identification number.

[out]	buf	pointer to buffer where data should be stored.
[in]	len	length of buffer (in bytes).
[out]	ip	pointer to buffer where remote source address shall be returned (NULL for none).
[inout]	ip_len	pointer to length of 'ip' (or NULL if 'ip' is NULL): • length of supplied 'ip' on input. • length of stored 'ip' on output.
[out]	port	pointer to buffer where remote source port shall be returned (NULL for none).

Returns

- status information:
  - number of bytes received (>=0), if len != 0.
  - 0 = Data is available (len = 0).
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (pointer to buffer or length).
  - IOT_SOCKET_ENOTCONN = Socket is not connected.
  - IOT_SOCKET_ECONNRESET = Connection reset by the peer.
  - IOT_SOCKET_ECONNABORTED = Connection aborted locally.
  - IOT_SOCKET_EAGAIN = Operation would block or timed out (may be called again).
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line 222 of file third_party/iot_socket/include/iot_socket.h

iotSocketSend

```
int32_t iotSocketSend (int32_t socket, const void *buf, uint32_t len)
```

Send data or check if data can be sent on a connected socket.

Parameters

[in]	socket	socket identification number.
[in]	buf	pointer to buffer containing data to send.
[in]	len	length of data (in bytes).

Returns

- status information:
  - number of bytes sent (>=0), if len != 0.
  - 0 = Data can be sent (len = 0).
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (pointer to buffer or length).
  - IOT_SOCKET_ENOTCONN = Socket is not connected.
  - IOT_SOCKET_ECONNRESET = Connection reset by the peer.
  - IOT_SOCKET_ECONNABORTED = Connection aborted locally.
  - IOT_SOCKET_EAGAIN = Operation would block or timed out (may be called again).
  - IOT_SOCKET_ERROR = Unspecified error.

Note

- The iotSocketSend() system call doesn't guarantees the packets are transmitted to remote note, which are enqueued in the queue. The iotSocketSend() system call can only send max of 1460 bytes incase of plain TCP, UDP. Whereas incase of TLS, the max buffer length is 1370.

Definition at line 244 of file third_party/iot_socket/include/iot_socket.h

iotSocketSendTo

```
int32_t iotSocketSendTo (int32_t socket, const void *buf, uint32_t len, const uint8_t *ip, uint32_t ip_len, uint16_t port)
```

Send data or check if data can be sent on a socket.

Parameters

[in]	socket	socket identification number.
[in]	buf	pointer to buffer containing data to send.
[in]	len	length of data (in bytes).
[in]	ip	pointer to remote destination IP address.
[in]	ip_len	length of 'ip' address in bytes.
[in]	port	remote destination port number.

Note

- If number of bytes to be send exceeds the MSS size specified by remote node then API will return IOT_SOCKET_ERROR. To know the MSS size for the socket use si91x_get_socket_mss() utility, In case of TCP this should be called after the connect()

Returns

- status information:
  - number of bytes sent (>=0), if len != 0.
  - 0 = Data can be sent (len = 0).
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (pointer to buffer or length).
  - IOT_SOCKET_ENOTCONN = Socket is not connected.
  - IOT_SOCKET_ECONNRESET = Connection reset by the peer.
  - IOT_SOCKET_ECONNABORTED = Connection aborted locally.
  - IOT_SOCKET_EAGAIN = Operation would block or timed out (may be called again).
  - IOT_SOCKET_ERROR = Unspecified error.

Note

- The iotSocketSendTo() system call doesn't guarantees the packets are transmitted to remote note, which are enqueued in the queue. The iotSocketSendTo() system call can only send max of 1460 bytes incase of plain TCP, UDP. Whereas incase of TLS, the max buffer length is 1370.

Definition at line `271` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketGetSockName

```
int32_t iotSocketGetSockName (int32_t socket, uint8_t *ip, uint32_t *ip_len, uint16_t *port)
```

Retrieve local IP address and port of a socket.

Parameters

[in]	socket	socket identification number.
[out]	ip	pointer to buffer where local address shall be returned (NULL for none).
[inout]	ip_len	pointer to length of 'ip' (or NULL if 'ip' is NULL): • length of supplied 'ip' on input. • length of stored 'ip' on output.

[out]	port	pointer to buffer where local port shall be returned (NULL for none).

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (pointer to buffer or length).
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line `288` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketGetPeerName

> int32_t iotSocketGetPeerName (int32_t socket, uint8_t *ip, uint32_t *ip_len, uint16_t *port)

Retrieve remote IP address and port of a socket.

Parameters

[in]	socket	socket identification number.
[out]	ip	pointer to buffer where remote address shall be returned (NULL for none).
[inout]	ip_len	pointer to length of 'ip' (or NULL if 'ip' is NULL): • length of supplied 'ip' on input. • length of stored 'ip' on output.
[out]	port	pointer to buffer where remote port shall be returned (NULL for none).

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument (pointer to buffer or length).
  - IOT_SOCKET_ENOTCONN = Socket is not connected.
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line `306` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketGetOpt

> int32_t iotSocketGetOpt (int32_t socket, int32_t opt_id, void *opt_val, uint32_t *opt_len)

Get socket option.

Parameters

[in]	socket	socket identification number.
[in]	opt_id	option identifier.
[out]	opt_val	pointer to the buffer that will receive the option value.
[inout]	opt_len	pointer to length of the option value: • length of buffer on input. • length of data on output.

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument.
  - IOT_SOCKET_ENOTSUP = Operation not supported.
  - IOT_SOCKET_ERROR = Unspecified error.

Note

- The following are the options, which are supported currently.
  - IOT_SOCKET_IO_FIONBIO
  - IOT_SOCKET_SO_RCVTIMEO
  - IOT_SOCKET_SO_SNDTIMEO
  - IOT_SOCKET_SO_KEEPALIVE
  - IOT_SOCKET_SO_TYPE.

Definition at line `331` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketSetOpt

> int32_t iotSocketSetOpt (int32_t socket, int32_t opt_id, const void *opt_val, uint32_t opt_len)

Set socket option.

Parameters

[in]	socket	socket identification number.
[in]	opt_id	option identifier.
[in]	opt_val	pointer to the option value.
[in]	opt_len	length of the option value in bytes.

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EINVAL = Invalid argument.
  - IOT_SOCKET_ENOTSUP = Operation not supported.
  - IOT_SOCKET_ERROR = Unspecified error.

Note

- The following are the options, which are supported currently.
  - IOT_SOCKET_IO_FIONBIO
  - IOT_SOCKET_SO_RCVTIMEO
  - IOT_SOCKET_SO_SNDTIMEO
  - IOT_SOCKET_SO_KEEPALIVE
  - IOT_SOCKET_SO_TYPE.

Definition at line `354` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketClose

> int32_t iotSocketClose (int32_t socket)

Close and release a socket.

Parameters

[in]	socket	socket identification number.

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_ESOCK = Invalid socket.
  - IOT_SOCKET_EAGAIN = Operation would block (may be called again).
  - IOT_SOCKET_ERROR = Unspecified error.

Note

- Calling close on server or first client socket would end up closing othet socket as well.

Definition at line `368` of file `third_party/iot_socket/include/iot_socket.h`

iotSocketGetHostByName

> int32_t iotSocketGetHostByName (const char *name, int32_t af, uint8_t *ip, uint32_t *ip_len)

Retrieve host IP address from host name.

Parameters

[in]	name	host name.
[in]	af	address family.
[out]	ip	pointer to buffer where resolved IP address shall be returned.
[inout]	ip_len	pointer to length of 'ip': • length of supplied 'ip' on input. • length of stored 'ip' on output.

Returns

- status information:
  - 0 = Operation successful.
  - IOT_SOCKET_EINVAL = Invalid argument.
  - IOT_SOCKET_ENOTSUP = Operation not supported.
  - IOT_SOCKET_ETIMEDOUT = Operation timed out.
  - IOT_SOCKET_EHOSTNOTFOUND = Host not found.
  - IOT_SOCKET_ERROR = Unspecified error.

Definition at line `387` of file `third_party/iot_socket/include/iot_socket.h`

SiWx91x Sockets

SiWx91x Sockets

This sections provides a reference to the SiWx91x chipset's socket API functions.

Functions

int sl_si91x_socket_async(int family, int type, int protocol, receive_data_callback callback)
This sl_si91x_socket_async() function creates an asynchronous socket and registers the provided callback.

int sl_si91x_socket(int family, int type, int protocol)
The sl_si91x_socket() function creates a new socket.

int sl_si91x_setsockopt_async(int32_t sockID, int level, int option_name, const void *option_value, socklen_t option_len)
The sl_si91x_setsockopt_async() function's purpose would be to set the specified socket option on the identified socket asynchronously.

int sl_si91x_bind(int socket, const struct sockaddr *addr, socklen_t addr_len)
The sl_si91x_bind() api assigns the local protocol address to a socket.

int sl_si91x_listen(int socket, int max_number_of_clients)
The sl_si91x_listen() function listen for socket connections.

int sl_si91x_accept(int socket, const struct sockaddr *addr, socklen_t addr_len)
The purpose of the sl_si91x_accept function is to block until a client attempts to connect to the server socket, once a connection request is received.

int sl_si91x_accept_async(int socket, accept_callback callback)
The sl_si91x_accept_async() api primary purpose is to set up the server socket to listen for incoming connections and immediately return without blocking the main program's execution.

int sl_si91x_connect(int socket, const struct sockaddr *addr, socklen_t addr_len)
The purpose of this sl_si91x_connect() function is to initiate a connection to a remote socket specified by the addr parameter.

int sl_si91x_send(int socket, const uint8_t *buffer, size_t buffer_length, int32_t flags)
The sl_si91x_send() function is used to transmit one or more messages to a socket.

int sl_si91x_send_async(int socket, const uint8_t *buffer, size_t buffer_length, int32_t flags, data_transfer_complete_handler callback)
The sl_si91x_send_async() function is used to transmit one or more messages to a socket asynchronously.

int sl_si91x_sendto(int socket, uint8_t *buffer, size_t buffer_length, int32_t flags, const struct sockaddr *addr, socklen_t addr_len)
The sl_si91x_sendto() function is used to transmit one or more messages to another socket.

int sl_si91x_sendto_async(int socket, uint8_t *buffer, size_t buffer_length, int32_t flags, const struct sockaddr *to_addr, socklen_t to_addr_len, data_transfer_complete_handler callback)
The sl_si91x_sendto_async() function is used to transmit one or more messages to another socket asynchronously.

int sl_si91x_send_large_data(int socket, const uint8_t *buffer, size_t buffer_length, int32_t flags)
Basically sl_si91x_send_large_data() function is used to send the data that is greater than MSS size.

int sl_si91x_recv(int socket, uint8_t *buffer, size_t bufferLength, int32_t flags)
The purpose of the sl_si91x_recv function is to receive data from a connected socket.

int sl_si91x_recvfrom(int socket, uint8_t *buffer, size_t buffersize, int32_t flags, struct sockaddr *fromAddr, socklen_t *fromAddrLen)
The purpose of the sl_si91x_recvfrom function is to receive data from an unconnected socket typically like a UDP socket.

int sl_si91x_shutdown(int socket, int how)
The purpose of the sl_si91x_shutdown is to disable sends or receives on a socket.

int sl_si91x_select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout, select_callback callback)
The purpose of the sl_si91x_select function is to monitor multiple file descriptors, including sockets, for various I/O events, such as readiness for reading or writing, and exceptional conditions.

Function Documentation

sl_si91x_socket_async

> int sl_si91x_socket_async (int family, int type, int protocol, receive_data_callback callback)

This sl_si91x_socket_async() function creates an asynchronous socket and registers the provided callback.

Parameters

[in]	family	This specifies a communication domain within which communication will take place; this selects the protocol family which should be used.
[in]	type	The socket has the indicate type, which specifies the semantics of communication. Currently defined types are SOCK_STREAM, SOCK_DGRAM.
[in]	protocol	The protocol argument specifies a particular protocol to be used with the socket.
[in]	callback	A function pointer that will be called when the socket receive the data.

Returns

- int

Definition at line 41 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_socket

> int sl_si91x_socket (int family, int type, int protocol)

The sl_si91x_socket() function creates a new socket.

Parameters

[in]	family	This specifies a communication domain within which communication will take place; this selects the protocol family which should be used.
[in]	type	The socket has the indicate type, which specifies the semantics of communication. Currently defined types are SOCK_STREAM, SOCK_DGRAM.
[in]	protocol	The protocol argument specifies a particular protocol to be used with the socket.

Returns

- int

Definition at line 56 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_setsockopt_async

```
int sl_si91x_setsockopt_async (int32_t sockID, int level, int option_name, const void *option_value, socklen_t option_len)
```

The sl_si91x_setsockopt_async() function's purpose would be to set the specified socket option on the identified socket asynchronously.

Parameters

[in]	sockID	Socket identification number.
[in]	level	Level for which the option is set.
[in]	option_name	This parameter is used to specify the particular socket option that may be set.
[in]	option_value	Pointer to option data.
[in]	option_len	The option_length is a value-result argument, initially containing the size of the buffer pointed to by option_value, and modified on return to indicate the actual size of the value returned. If no option value is to be supplied or returned, optval may be NULL.

Returns

- int

Definition at line `75` of file `components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h`

sl_si91x_bind

```
int sl_si91x_bind (int socket, const struct sockaddr *addr, socklen_t addr_len)
```

The sl_si91x_bind() api assigns the local protocol address to a socket.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	addr	The addr argument is the address is that to which datagrams are to be sent.
[in]	addr_len	The addr_len argument indicates the amount of space pointed to by addr, in bytes.

Returns

- int

Definition at line `92` of file `components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h`

sl_si91x_listen

```
int sl_si91x_listen (int socket, int max_number_of_clients)
```

The sl_si91x_listen() function listen for socket connections.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	max_number_of_clients	The maximum number of clients that the socket can accept.

Returns

- int

Definition at line 103 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_accept

> int sl_si91x_accept (int socket, const struct sockaddr *addr, socklen_t addr_len)

The purpose of the sl_si91x_accept function is to block until a client attempts to connect to the server socket, once a connection request is received.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	addr	The addr argument is used to store the accepted client socket address.
[in]	addr_len	The addr_len argument indicates the amount of space pointed to by addr, in bytes.

Returns

- int

Definition at line 117 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_accept_async

> int sl_si91x_accept_async (int socket, accept_callback callback)

The sl_si91x_accept_async() api primary purpose is to set up the server socket to listen for incoming connections and immediately return without blocking the main program's execution.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	callback	A function pointer that will be called when the new client is connected to the server.

Returns

- int

Definition at line 129 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_connect

> int sl_si91x_connect (int socket, const struct sockaddr *addr, socklen_t addr_len)

The purpose of this sl_si91x_connect() function is to initiate a connection to a remote socket specified by the addr parameter.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	addr	The addr argument is the address is that to which datagrams are to be sent.
[in]	addr_len	The addr_len argument indicates the amount of space pointed to by addr, in bytes.

Returns

- int

Definition at line 142 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_send

```
int sl_si91x_send (int socket, const uint8_t *buffer, size_t buffer_length, int32_t flags)
```

The sl_si91x_send() function is used to transmit one or more messages to a socket.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	buffer	Pointer to the buffer that receives the data.
[in]	buffer_length	Length of the buffer pointed to by the buffer parameter.
[in]	flags	Controls the transmission of the data.

The sl_si91x_send() function may be used only when the socket is in a connected state. **Returns**

- int

Note

- The argument flags are not supported currently.

Definition at line 159 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_send_async

```
int sl_si91x_send_async (int socket, const uint8_t *buffer, size_t buffer_length, int32_t flags,
data_transfer_complete_handler callback)
```

The sl_si91x_send_async() function is used to transmit one or more messages to a socket asynchronously.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	buffer	Pointer to the buffer that receives the data.
[in]	buffer_length	Length of the buffer pointed to by the buffer parameter.
[in]	flags	Controls the transmission of the data.
[in]	callback	A function pointer that will be called when the socket receive the data.

The sl_si91x_send_async() function may be used only when the socket is in a connected state. **Returns**

- int

Note

- The argument flags are not supported currently.

Definition at line 178 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_sendto

```
int sl_si91x_sendto (int socket, uint8_t *buffer, size_t buffer_length, int32_t flags, const struct sockaddr *addr, socklen_t
addr_len)
```

The sl_si91x_sendto() function is used to transmit one or more messages to another socket.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	buffer	Pointer to the buffer that receives the data.
[in]	buffer_length	Length of the buffer pointed to by the buffer parameter.
[in]	flags	Controls the transmission of the data.
[in]	addr	The addr argument is the address is that to which datagrams are to be sent.
[in]	addr_len	The addr_len argument indicates the amount of space pointed to by addr, in bytes.

The function can also be called from an unconnected socket typically like a UDP socket. **Returns**

- int

Note

- The argument flags are not supported currently.

Definition at line `203` of file `components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h`

sl_si91x_sendto_async

```
int sl_si91x_sendto_async (int socket, uint8_t *buffer, size_t buffer_length, int32_t flags, const struct sockaddr *to_addr,
socklen_t to_addr_len, data_transfer_complete_handler callback)
```

The sl_si91x_sendto_async() function is used to transmit one or more messages to another socket asynchronously.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	buffer	Pointer to the buffer that receives the data.
[in]	buffer_length	Length of the buffer pointed to by the buffer parameter.
[in]	flags	Controls the transmission of the data.
[in]	to_addr	Address of the target.
[in]	to_addr_len	Size of the address pointed by to_addr.
[in]	callback	A function pointer that will be called when the socket receive the data.

The function can also be called from an unconnected socket typically like a UDP socket. **Returns**

- int

Note

- The argument flags are not supported currently.

Definition at line `231` of file `components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h`

sl_si91x_send_large_data

```
int sl_si91x_send_large_data (int socket, const uint8_t *buffer, size_t buffer_length, int32_t flags)
```

Basically sl_si91x_send_large_data() function is used to send the data that is greater than MSS size.

Parameters

[in]	socket	The socket_id argument is a socket
[in]	buffer	Pointer to the buffer that receives the data.

[in]	buffer_length	Length of the buffer pointed to by the buffer parameter.
[in]	flags	Controls the transmission of the data.

Returns

- int

Note

- The argument flags are not supported currently.

Definition at line 253 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_recv

```
int sl_si91x_recv (int socket, uint8_t *buffer, size_t bufferLength, int32_t flags)
```

The purpose of the sl_si91x_recv function is to receive data from a connected socket.

Parameters

[in]	socket	The socket_id argument is a socket
[in]	buffer	Pointer to the buffer that receives the data.
[in]	bufferLength	Length of the buffer pointed to by the buffer parameter.
[in]	flags	Controls the reception of the data.

Returns

- int

Note

- The argument flags are not supported currently.

Definition at line 269 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_recvfrom

```
int sl_si91x_recvfrom (int socket, uint8_t *buffer, size_t buffersize, int32_t flags, struct sockaddr *fromAddr, socklen_t *fromAddrLen)
```

The purpose of the sl_si91x_recvfrom function is to receive data from an unconnected socket typically like a UDP socket.

Parameters

[in]	socket	The socket_id argument is a socket.
[in]	buffer	Pointer to the buffer that receives the data.
[in]	buffersize	Size of the buffer pointed to by the buffer parameter.
[in]	flags	Controls the reception of the data.
[in]	fromAddr	Pointer to a socket address structure from which data is received.
[in]	fromAddrLen	The fromAddrLen argument is a value-result argument, initialized to the size of the buffer associated with from_addr, and modified on return to indicate the actual size of the address stored there.

Returns

- int

Note

- The argument flags are not supported currently.

Definition at line 290 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_shutdown

> int sl_si91x_shutdown (int socket, int how)

The purpose of the sl_si91x_shutdown is to disable sends or receives on a socket.

Parameters

[in]	socket	Socket FD that is to be closed.
[in]	how	How determines whether the socket is closed based on given socket ID or port number.

Note

- In case of socket being a server socket, the "how" parameter would be ignored and socket shall always be closed based on port number.

Definition at line 307 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

sl_si91x_select

> int sl_si91x_select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout, select_callback callback)

The purpose of the sl_si91x_select function is to monitor multiple file descriptors, including sockets, for various I/O events, such as readiness for reading or writing, and exceptional conditions.

Parameters

[in]	nfds	The first nfds descriptors are checked in each set; i.e., the descriptors from 0 through nfds-1.
[inout]	readfds	A pointer to a fd_set object that specifies the descriptors to check for files that are ready for reading.
[inout]	writefds	A pointer to a fd_set object that specifies the descriptors to check for files that are ready for writing.
[inout]	exceptfds	A pointer to a fd_set object that will be watched for exceptions
[in]	timeout	If timeout is not a null pointer, it specifies the maximum interval to wait for the selection to complete.
[in]	callback	A function pointer that will be called when the socket receive the data.

It allows you to efficiently manage multiple sockets and determine when they are available for specific I/O operations.
Returns

- int

Note

- The readfds, writefds are modified incase of callback being NULL. exceptfds are not currently being supported.

Definition at line 329 of file components/device/silabs/si91x/wireless/asynchronous_socket/inc/sl_si91x_socket.h

Socket Configuration

Socket Configuration

The module provides an API to set socket configuration parameter.

Functions

sl_status_t	sl_si91x_config_socket(sl_si91x_socket_config_t socket_config) Si917 specific socket configuration.

Function Documentation

sl_si91x_config_socket

```
sl_status_t sl_si91x_config_socket (sl_si91x_socket_config_t socket_config)
```

Si917 specific socket configuration.

Parameters

[in]	socket_config	Socket configuration of type sl_si91x_socket_config_t.

- Pre-conditions:
  - This API to be called before calling sl_si91x_socket_async.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 87 of file components/device/silabs/si91x/wireless/socket/inc/sl_si91x_socket_utility.h

Overview

Overview

The SiWx917 chipset provides various features such a low-power modes, calibration, firmware upgrades, and others.

APIs

APIs

This section provides a reference to the Si91x Device Management API including functions, data types, and constants.

Modules

Functions

Types

Constants

Functions

Functions

This section provides a reference to Si91x Device Management functions:

- Core functions to initialize and configure the device.
- Radio functions to access radio features such as calibration, low-power settings, and others.
- Firmware Update functions to perform firmware updates on the device.
- Network Configuration Functions to perform network configurations on device

Modules

Core

Radio

Firmware Update

Network Configuration

Core

Core

Functions

sl_status_t	sl_si91x_get_saved_firmware_status(void) Get the saved thread specific firmware status value.
sl_status_t	sl_si91x_set_join_configuration(sl_wifi_interface_t interface, uint8_t join_feature_bitmap) Si91X specific set join feature bitmap configuration.
sl_status_t	sl_si91x_get_join_configuration(sl_wifi_interface_t interface, uint8_t *join_feature_bitmap) Si91X specific get join feature bitmap configuration.
sl_status_t	sl_si91x_configure_timeout(sl_si91x_timeout_type_t timeout_type, uint16_t timeout_value) This API is used to set different module timeouts.
void	sl_si91x_set_timeout(sl_si91x_timeout_t *timeout_config) Si91x specific set timeout.
sl_status_t	sl_si91x_assert(void) Signals the occurrence of an assertion in the firmware.
sl_status_t	sl_si91x_get_ram_log(uint32_t address, uint32_t length) Retrieves TA RAM log/dump via Si91x UART/UART2.

Function Documentation

sl_si91x_get_saved_firmware_status

```
static sl_status_t sl_si91x_get_saved_firmware_status (void)
```

Get the saved thread specific firmware status value.

Parameters

N/A		

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `85` of file `components/device/silabs/si91x/wireless/inc/sl_si91x_core_utilities.h`

sl_si91x_set_join_configuration

```
sl_status_t sl_si91x_set_join_configuration (sl_wifi_interface_t interface, uint8_t join_feature_bitmap)
```

Si91X specific set join feature bitmap configuration.

Parameters

[in]	interface	sl_wifi_interface_t Selected interface.
[in]	join_feature_bitmap	Join feature bitmap configuration. One of values from Join Feature Bitmap

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- By default SL_SI91X_JOIN_FEAT_LISTEN_INTERVAL_VALID bitmap is enabled. User can call this API before calling sl_wifi_connect, sl_wifi_start_ap, sl_wifi_start_wps to overwrite the join feature bitmap

Definition at line 558 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_get_join_configuration

> sl_status_t sl_si91x_get_join_configuration (sl_wifi_interface_t interface, uint8_t *join_feature_bitmap)

Si91X specific get join feature bitmap configuration.

Parameters

[in]	interface	sl_wifi_interface_t Selected interface.
[out]	join_feature_bitmap	join feature bitmap configuration. One of values from Join Feature Bitmap

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- By default SL_SI91X_JOIN_FEAT_LISTEN_INTERVAL_VALID bitmap is enabled.

Definition at line 572 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_configure_timeout

> sl_status_t sl_si91x_configure_timeout (sl_si91x_timeout_type_t timeout_type, uint16_t timeout_value)

This API is used to set different module timeouts.

Parameters

[in]	timeout_type	It is used to identify which timeout type to be set. The possible values can be sl_si91x_timeout_type_t •
[in]	timeout_value	timeout value to be set. The time resolution depends on timeout_type.

- Pre-conditions:
  - This API should be called after sl_wifi_init

Note

- After a successful IP configuration, Gratuitous ARP is used as the periodic WLAN Keep-Alive packet with the configured keep_alive_timeout interval.
  - If there is no IP configuration, the NULL Data Packets is used as the WLAN Keep-Alive packet.

Returns

sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 593 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_set_timeout

```
void sl_si91x_set_timeout (sl_si91x_timeout_t *timeout_config)
```

Si91x specific set timeout.

Parameters

[in]	timeout_config	Timeout configuration of type sl_si91x_timeout_t.

This function is used to set active channel scan timeout, authentication association timeout and keep alive timeout of module. **Returns**

- None

Note

- This API should ONLY be called before sl_wifi_init and repeated call to this API will overwrite timeout values stored in SDK, will be applied on next call to sl_wifi_init.

Definition at line 604 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_assert

```
sl_status_t sl_si91x_assert (void)
```

Signals the occurrence of an assertion in the firmware.

Parameters

N/A		

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 698 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_get_ram_log

```
sl_status_t sl_si91x_get_ram_log (uint32_t address, uint32_t length)
```

Retrieves TA RAM log/dump via Si91x UART/UART2.

Parameters

[in]	address	Address in Si91x module.
[in]	length	Chunk length to read from Si91x module.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 710 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

Radio

Radio

Functions

sl_status_t	sl_si91x_transmit_test_start(sl_si91x_request_tx_test_info_t *tx_test_info) Start the transmit test.
sl_status_t	sl_si91x_transmit_test_stop(void) Stop the transmit test.
sl_status_t	sl_si91x_frequency_offset(const sl_si91x_freq_offset_t *frequency_calibration) Used to provide feedback of Frequency error in KHz.
sl_status_t	sl_si91x_set_device_region(sl_si91x_operation_mode_t operation_mode, sl_si91x_band_mode_t band, sl_si91x_region_code_t region_code) Set the device region.
sl_status_t	sl_si91x_calibration_write(sl_si91x_calibration_write_t calib_write) This API will command the firmware to update the existing Flash/EFuse calibration data.
sl_status_t	sl_si91x_calibration_read(sl_si91x_calibration_read_t target, sl_si91x_calibration_read_t *calibration_read) This API reads the calibration data from the Flash/EFuse storage.
sl_status_t	sl_si91x_evm_offset(const sl_si91x_evm_offset_t *evm_offset) Application that offers feedback on the error caused by the EVM offset.
sl_status_t	sl_si91x_evm_write(const sl_si91x_evm_write_t *evm_write) This API will command the firmware to update the existing Flash/EFuse calibration data.
sl_status_t	sl_si91x_dpd_calibration(const sl_si91x_get_dpd_calib_data_t *dpd_calib_data) This API updates Flash/Efuse DPD data.
sl_status_t	sl_si91x_efuse_read(sl_si91x_efuse_read_t *efuse_read, uint8_t *efuse_read_buf) This API will command the firmware to get the data from the Efuse memory location.

Function Documentation

sl_si91x_transmit_test_start

```
sl_status_t sl_si91x_transmit_test_start (sl_si91x_request_tx_test_info_t *tx_test_info)
```

Start the transmit test.

Parameters

[in]	tx_test_info	sl_si91x_request_tx_test_info_t Configurable Tx test mode request structure

This is a blocking API.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 276 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_transmit_test_stop

```
sl_status_t sl_si91x_transmit_test_stop (void)
```

Stop the transmit test.

Parameters

[in]		

This is a blocking API.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 287 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_frequency_offset

```
sl_status_t sl_si91x_frequency_offset (const sl_si91x_freq_offset_t *frequency_calibration)
```

Used to provide feedback of Frequency error in KHz.

Parameters

[in]	frequency_calibration	Frequency in KHz of type sl_si91x_freq_offset_t.

This is a blocking API.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 300 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_set_device_region

```
sl_status_t sl_si91x_set_device_region (sl_si91x_operation_mode_t operation_mode, sl_si91x_band_mode_t band,
sl_si91x_region_code_t region_code)
```

Set the device region.

Parameters

[in]	operation_mode	sl_si91x_operation_mode_t Operation mode of the device.
[in]	band	sl_si91x_band_mode_t Operational band of the device.
[in]	region_code	sl_si91x_region_code_t Region code to be set in the device.

Pre-conditions:
- sl_si91x_driver_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 317 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_calibration_write

```
sl_status_t sl_si91x_calibration_write (sl_si91x_calibration_write_t calib_write)
```

This API will command the firmware to update the existing Flash/EFuse calibration data.

Parameters

[in]	calib_write	Write calibration configuration of type sl_si91x_calibration_write_t

This is a blocking API.

- Pre-conditions:
  - sl_wifi_init, sl_si91x_transmit_test_start and sl_si91x_frequency_offset should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 332 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_calibration_read

```
sl_status_t sl_si91x_calibration_read (sl_si91x_calibration_read_t target, sl_si91x_calibration_read_t *calibration_read)
```

This API reads the calibration data from the Flash/EFuse storage.

Parameters

[in]	target	0 - READ_FROM_EFUSE (read calibration data from the EFuse) 1 - READ_FROM_FLASH (read calibration data from the Flash)
[out]	calibration_read	Read the calibration configuration of type sl_si91x_calibration_read_t

This is a blocking API.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 348 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_evm_offset

```
sl_status_t sl_si91x_evm_offset (const sl_si91x_evm_offset_t *evm_offset)
```

Application that offers feedback on the error caused by the EVM offset.

Parameters

[in]	evm_offset	evm offset of type sl_si91x_evm_offset_t

This is a blocking API.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 362 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_evm_write

```
sl_status_t sl_si91x_evm_write (const sl_si91x_evm_write_t *evm_write)
```

This API will command the firmware to update the existing Flash/EFuse calibration data.

Parameters

[in]	evm_write	Write the evm calibration configuration of type sl_si91x_evm_write_t

This is a blocking API.

- Pre-conditions:
  - sl_wifi_init, sl_si91x_evm_offset and sl_si91x_transmit_test_start should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 375 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_dpd_calibration

```
sl_status_t sl_si91x_dpd_calibration (const sl_si91x_get_dpd_calib_data_t *dpd_calib_data)
```

This API updates Flash/Efuse DPD data.

Parameters

[in]	dpd_calib_data	Write DPD calibration data of type sl_si91x_get_dpd_calib_data_t

This is a synchronous API

- Pre-conditions:
  - sl_wifi_init, sl_si91x_transmit_test_start should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 388 of file components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h

sl_si91x_efuse_read

```
sl_status_t sl_si91x_efuse_read (sl_si91x_efuse_read_t *efuse_read, uint8_t *efuse_read_buf)
```

This API will command the firmware to get the data from the Efuse memory location.

Parameters

[in]	efuse_read	efuse read structure, which contains efuse read address offset and read data length of type sl_si91x_efuse_read_t
[out]	efuse_read_buf	efuse read buffer

This is a blocking API.

- Pre-conditions:
  - sl_wifi_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `403` of file `components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h`

Firmware Update

Firmware Update

Firmware update can be handled on Si91x Chipsets in following ways

- Firmware Update from Host functions to perform firmware update from the host.
- Firmware Update from Module functions to perform firmware update from the module.

Modules

Firmware Update from Host

Firmware Update from Module

Firmware Update from Host

Firmware Update from Host

This section provides a reference to the APIs used to perform firmware updates from the host.

Functions

sl_status_t	sl_si91x_fwup_start(uint8_t *rps_header) Send the RPS header content of firmware file.This is a blocking API.
sl_status_t	sl_si91x_fwup_load(uint8_t *content, uint16_t length) Send the firmware file content.This is a blocking API.

Function Documentation

sl_si91x_fwup_start

```
sl_status_t sl_si91x_fwup_start (uint8_t *rps_header)
```

Send the RPS header content of firmware file.This is a blocking API.

Parameters

[in]	rps_header	Pointer to the RPS header content.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `508` of file `components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h`

sl_si91x_fwup_load

```
sl_status_t sl_si91x_fwup_load (uint8_t *content, uint16_t length)
```

Send the firmware file content.This is a blocking API.

Parameters

[in]	content	Pointer to the firmware file content.
[in]	length	Length of the content

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `520` of file `components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h`

Firmware Update from Module

Firmware Update from Module

This section provides a reference to the APIs used to perform firmware updates from the module.

Functions

sl_status_t	sl_si91x_ota_firmware_upgradation(sl_ip_address_t server_ip, uint16_t server_port, uint16_t chunk_number, uint16_t timeout, uint16_t tcp_retry_count, bool asynchronous) Create an OTAF client.
sl_status_t	sl_si91x_http_otaf(uint8_t type, uint8_t flags, uint8_t *ip_address, uint16_t port, uint8_t *resource, uint8_t *host_name, uint8_t *extended_header, uint8_t *user_name, uint8_t *password, uint8_t *post_data, uint32_t post_data_length) Post the HTTP data for the requested URL to HTTP server.This is a non-blocking API.

Function Documentation

sl_si91x_ota_firmware_upgradation

> sl_status_t sl_si91x_ota_firmware_upgradation (sl_ip_address_t server_ip, uint16_t server_port, uint16_t chunk_number, uint16_t timeout, uint16_t tcp_retry_count, bool asynchronous)

Create an OTAF client.

Parameters

[in]	server_ip	OTAF server IP address sl_ip_address_t
[in]	server_port	OTAF server port number.
[in]	chunk_number	Firmware content request chunk number.
[in]	timeout	TCP receive packet timeout.
[in]	tcp_retry_count	TCP retransmissions count.
[in]	asynchronous	OTAF upgrade done aynchronously when this is set, else synchronous upgrade.

Initialize the client with a given configuration.

- Pre-conditions:
  - sl_net_up API needs to be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- For safe firmware upgrade via TCP server,
  - it will take approx. 65 sec duration for upgrading the firmware of 1.5 MB file.
- This is an asynchronous API. The response is recieved via sl_net_event_handler_t with SL_NET_OTA_FW_UPDATE_EVENT as event

Definition at line `637` of file `components/device/silabs/si91x/wireless/inc/sl_si91x_driver.h`

sl_si91x_http_otaf

```
sl_status_t sl_si91x_http_otaf (uint8_t type, uint8_t flags, uint8_t *ip_address, uint16_t port, uint8_t *resource, uint8_t *host_name, uint8_t *extended_header, uint8_t *user_name, uint8_t *password, uint8_t *post_data, uint32_t post_data_length)
```

Post the HTTP data for the requested URL to HTTP server.This is a non-blocking API.

Parameters

[in]	type	Server IP address.
[in]	flags	Port number. Default : 80 - HTTP, 443 - HTTPS
[in]	ip_address	Requested resource URL in string format.
[in]	port	Host name.
[in]	resource	Extender header if present, after each header member append \r
[in]	host_name	Username for server authentication.
[in]	extended_header	Password for server authentication.
[in]	user_name	HTTP data to be posted to server.
[in]	password	HTTP data length to be posted to server.
N/A	post_data	
N/A	post_data_length	

Flags	Macro	Description
BIT(0)	HTTPS_SUPPORT	Set this bit to enable HTTPS_SUPPORT to use HTTPS feature.
BIT(1)	SSL_ENABLE	Set this bit to enable SSL feature.
BIT(2)	SI91X_TLS_V_1_0	Set this bit to support SSL TLS Version 1.0 if HTTPS is enabled.
BIT(3)	IPV6	Set this bit to enable IPv6, by default it is configured to IPv4.
BIT(4)	SI91X_TLS_V_1_1	Set this bit to support SSL_TLS Version 1.1 if HTTPS is enabled.
BIT(5)	HTTP_POST_DATA	Set this bit to enable Http_post large data feature.
BIT(6)	HTTP_V_1_1	Set this bit to use HTTP version 1.1

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `77` of file `components/device/silabs/si91x/wireless/firmware_upgrade/firmware_upgradation.h`

Network Configuration

Network Configuration

Enumerations

enum sl_si91x_multicast_address_command_type_t {

SL_WIFI_MULTICAST_LEAVE = 0
SL_WIFI_MULTICAST_JOIN = 1

}

enum sl_si91x_ip_config_mode_t {

SL_SI91X_STATIC = 0
SL_SI91X_DHCP
SL_SI91X_DHCP_RESERVED
SL_SI91X_DHCP_HOSTNAME
SL_SI91X_DHCP_OPTION81
SL_SI91X_DHCP_OPTION77

}

Functions

sl_status_t sl_si91x_configure_ip_address(sl_net_ip_configuration_t *address, uint8_t virtual_ap_id)
Configure IP address.

Enumeration Documentation

sl_si91x_multicast_address_command_type_t

> sl_si91x_multicast_address_command_type_t

Enumerator	
SL_WIFI_MULTICAST_LEAVE	
SL_WIFI_MULTICAST_JOIN	

Definition at line 40 of file components/device/silabs/si91x/wireless/sl_net/inc/sl_net_si91x.h

sl_si91x_ip_config_mode_t

> sl_si91x_ip_config_mode_t

Enumerator	
SL_SI91X_STATIC	
SL_SI91X_DHCP	
SL_SI91X_DHCP_RESERVED	
SL_SI91X_DHCP_HOSTNAME	
SL_SI91X_DHCP_OPTION81	
SL_SI91X_DHCP_OPTION77	

Definition at line 46 of file components/device/silabs/si91x/wireless/sl_net/inc/sl_net_si91x.h

Function Documentation

sl_si91x_configure_ip_address

> sl_status_t sl_si91x_configure_ip_address (sl_net_ip_configuration_t *address, uint8_t virtual_ap_id)

Configure IP address.

Parameters

[in]	address	- Pointer to store assigned Ip address details.
[in]	virtual_ap_id	- Virtual AP ID. One of values from sl_si91x_wifi_vap_id_t

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 63 of file components/device/silabs/si91x/wireless/sl_net/inc/sl_net_si91x.h

Types

Types

This section provides a reference to Si91x Device Management data types.

Note

- Bits 28 - 31 are reserved

Modules

sl_si91x_request_tx_test_info_t

sl_si91x_async_stats_response_t

sl_si91x_advance_stats_response_t

sl_si91x_boot_configuration_t

sl_si91x_timeout_t

sl_si91x_module_state_stats_response_t

sl_wifi_device_configuration_t

sl_wifi_device_context_t

sl_bt_performance_profile_t

sl_wifi_performance_profile_t

Enumerations

enum sl_si91x_ap_keepalive_type_t {

SL_SI91X_AP_KEEP_ALIVE_DISABLE = 0
SL_SI91X_AP_DEAUTH_BASED_KEEP_ALIVE = 1
SL_SI91X_AP_NULL_BASED_KEEP_ALIVE = 3

}

Si91x specific keepalive types.

enum sl_si91x_band_mode_t {

SL_SI91X_WIFI_BAND_2_4GHZ = 0
SL_SI91X_WIFI_BAND_5GHZ = 1
SL_SI91X_WIFI_DUAL_BAND = 2

}

Si91x command types Si91x band mode.

enum sl_si91x_region_code_t {

DEFAULT_REGION
US
EU
JP
WORLD_DOMAIN
KR
SG

}

Si91x region code.

enum sl_si91x_timeout_type_t {

SL_SI91X_AUTHENTICATION_ASSOCIATION_TIMEOUT = 0
SL_SI91X_CHANNEL_ACTIVE_SCAN_TIMEOUT
SL_SI91X_KEEP_ALIVE_TIMEOUT

}

Si91x Timeout types.

enum sl_si91x_wifi_vap_id_t {

SL_SI91X_WIFI_CLIENT_VAP_ID
SL_SI91X_WIFI_AP_VAP_ID

}

Si91x Wi-Fi VAP ID.

enum sl_si91x_operation_mode_t {

SL_SI91X_CLIENT_MODE = 0
SL_SI91X_ENTERPRISE_CLIENT_MODE = 2
SL_SI91X_ACCESS_POINT_MODE = 6
SL_SI91X_TRANSMIT_TEST_MODE = 8
SL_SI91X_CONCURRENT_MODE = 9
__FORCE_OPERATION_ENUM_16BIT = 0xFFFF

}

SiWx91x operating mode.

enum sl_si91x_coex_mode_t {

SL_SI91X_WLAN_ONLY_MODE = 0
SL_SI91X_WLAN_MODE = 1
SL_SI91X_BLUETOOTH_MODE = 4
SL_SI91X_WLAN_BLUETOOTH_MODE = 5
SL_SI91X_DUAL_MODE = 8
SL_SI91X_WLAN_DUAL_MODE = 9
SL_SI91X_BLE_MODE = 12
SL_SI91X_WLAN_BLE_MODE = 13
__FORCE_COEX_ENUM_16BIT = 0xFFFF

}

SiWx91x wireless co-existence mode.

enum sl_si91x_performance_profile_t {

HIGH_PERFORMANCE
ASSOCIATED_POWER_SAVE
ASSOCIATED_POWER_SAVE_LOW_LATENCY
STANDBY_POWER_SAVE
STANDBY_POWER_SAVE_WITH_RAM_RETENTION

}

Performance profile.

Enumeration Documentation

sl_si91x_ap_keepalive_type_t

> sl_si91x_ap_keepalive_type_t

Si91x specific keepalive types.

Enumerator	
SL_SI91X_AP_KEEP_ALIVE_DISABLE	Disable keepalive.
SL_SI91X_AP_DEAUTH_BASED_KEEP_ALIVE	AP performs keep alive functionality based on the RX packets received from its stations. If no packet is received from the station with in time out, AP discards it.
SL_SI91X_AP_NULL_BASED_KEEP_ALIVE	AP performs keep alive functionality by sending NULL DATA packet to the station. If no ACK is received from the station after specific no of retries, AP discards the station.

Definition at line `197` of file `components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h`

sl_si91x_band_mode_t

> sl_si91x_band_mode_t

Si91x command types Si91x band mode.

Note

- Only 2.4 GHz currently supported.

Enumerator	
SL_SI91X_WIFI_BAND_2_4GHZ	2.4GHz WiFi band
SL_SI91X_WIFI_BAND_5GHZ	5GHz WiFi band (not currently supported)
SL_SI91X_WIFI_DUAL_BAND	both 2.4GHz and 5GHZ WiFi band (not currently supported)

Definition at line `94` of file `components/device/silabs/si91x/wireless/inc/sl_si91x_types.h`

sl_si91x_region_code_t

> sl_si91x_region_code_t

Si91x region code.

Note

- Singapore region not currently supported.

Enumerator	
DEFAULT_REGION	Factory default region.
US	United States.
EU	European Union.
JP	Japan.
WORLD_DOMAIN	World wide domain.
KR	Korea.
SG	Singapore (not currently supported)

Definition at line 102 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

sl_si91x_timeout_type_t

```
sl_si91x_timeout_type_t
```

Si91x Timeout types.

Enumerator	
SL_SI91X_AUTHENTICATION_ASSOCIATION_TIMEOUT	Used for setting association and authentication timeout request in millisecs.
SL_SI91X_CHANNEL_ACTIVE_SCAN_TIMEOUT	Used for setting active scan time for per channel in millisecs.
SL_SI91X_KEEP_ALIVE_TIMEOUT	Used for setting WLAN keep alive time in seconds.

Definition at line 113 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

sl_si91x_wifi_vap_id_t

```
sl_si91x_wifi_vap_id_t
```

Si91x Wi-Fi VAP ID.

Enumerator	
SL_SI91X_WIFI_CLIENT_VAP_ID	Wi-Fi Client VAP ID.
SL_SI91X_WIFI_AP_VAP_ID	Wi-Fi Access point VAP ID.

Definition at line 121 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

sl_si91x_operation_mode_t

```
sl_si91x_operation_mode_t
```

SiWx91x operating mode.

Enumerator	
SL_SI91X_CLIENT_MODE	WiFi personal client mode.
SL_SI91X_ENTERPRISE_CLIENT_MODE	WiFi enterprise client mode.
SL_SI91X_ACCESS_POINT_MODE	WiFi access point mode.
SL_SI91X_TRANSMIT_TEST_MODE	WiFi transit test mode.
SL_SI91X_CONCURRENT_MODE	WiFi concurrent mode.
__FORCE_OPERATION_ENUM_16BIT	

Definition at line 851 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

sl_si91x_coex_mode_t

```
sl_si91x_coex_mode_t
```

SiWx91x wireless co-existence mode.

Note

Only BLE, WLAN, and WLAN + BLE modes are supported.

Enumerator	
SL_SI91X_WLAN_ONLY_MODE	Wireless local area network (WLAN) only mode.
SL_SI91X_WLAN_MODE	WLAN mode (not currently supported)
SL_SI91X_BLUETOOTH_MODE	Bluetooth only mode (not currently supported)
SL_SI91X_WLAN_BLUETOOTH_MODE	WLAN and Bluetooth mode (not currently supported)
SL_SI91X_DUAL_MODE	Dual mode (not currently supported)
SL_SI91X_WLAN_DUAL_MODE	WLAN dual mode (not currently supported)
SL_SI91X_BLE_MODE	Bluetooth Low Energy (BLE) only mode, used when power save mode not needed.
SL_SI91X_WLAN_BLE_MODE	WLAN and BLE mode.
__FORCE_COEX_ENUM_16BIT	

Definition at line `864` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

sl_si91x_performance_profile_t

> sl_si91x_performance_profile_t

Performance profile.

Enumerator	
HIGH_PERFORMANCE	Power save is disabled and throughput is maximum.
ASSOCIATED_POWER_SAVE	Power save mode when module is associated with either AP or station.
ASSOCIATED_POWER_SAVE_LOW_LATENCY	Power save mode when module is associated with either AP or station, with higher throughput than ASSOCIATED_POWER_SAVE. This is not supported for BT/BLE.
STANDBY_POWER_SAVE	Power save mode when module is not associated with either AP or station, ram is not retained in this mode.
STANDBY_POWER_SAVE_WITH_RAM_RETENTION	Power save mode when module is not associated with either AP or station, ram is retained in this mode.

Definition at line `940` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

sl_si91x_request_tx_test_info_t

Si91x specific TX test info.

Public Attributes

uint16_t	enable enable/disable tx test mode
uint16_t	power tx test mode power. Range : 2 - 18 dBm.
uint32_t	rate tx test mode rate
uint16_t	length tx test mode length. Range: [24 - 1500] bytes in Burst mode and [24 - 260] bytes in Continuous mode
uint16_t	mode tx test mode mode. 0 - Burst Mode; 1 - Continuous Mode; 2 - CW Mode; 3 - CW Mode center frequency - 2.5MHz; 4 - CW Mode center frequency + 5MHz
uint16_t	channel tx test mode channel
uint16_t	rate_flags tx test mode rate_flags
uint16_t	channel_bw tx test mode tx test_ch_bw
uint16_t	aggr_enable tx test mode aggr_enable
uint16_t	reserved tx test mode reserved
uint16_t	no_of_pkts tx test mode no_of_pkts
uint32_t	delay tx test mode delay

Public Attribute Documentation

enable

```
uint16_t sl_si91x_request_tx_test_info_t::enable
```

enable/disable tx test mode

Definition at line 1504 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

power

uint16_t sl_si91x_request_tx_test_info_t::power

tx test mode power. Range : 2 - 18 dBm.

Definition at line 1505 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

rate

uint32_t sl_si91x_request_tx_test_info_t::rate

tx test mode rate

Definition at line 1506 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

length

uint16_t sl_si91x_request_tx_test_info_t::length

tx test mode length. Range: [24 - 1500] bytes in Burst mode and [24 - 260] bytes in Continuous mode

Definition at line 1508 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

mode

uint16_t sl_si91x_request_tx_test_info_t::mode

tx test mode mode. 0 - Burst Mode; 1 - Continuous Mode; 2 - CW Mode; 3 - CW Mode center frequency - 2.5MHz; 4 - CW Mode center frequency + 5MHz

Definition at line 1510 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

channel

uint16_t sl_si91x_request_tx_test_info_t::channel

tx test mode channel

Definition at line 1511 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

rate_flags

uint16_t sl_si91x_request_tx_test_info_t::rate_flags

tx test mode rate_flags

Definition at line 1512 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

channel_bw

```
uint16_t sl_si91x_request_tx_test_info_t::channel_bw
```

tx test mode tx test_ch_bw

Definition at line 1513 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

aggr_enable

```
uint16_t sl_si91x_request_tx_test_info_t::aggr_enable
```

tx test mode aggr_enable

Definition at line 1514 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

reserved

```
uint16_t sl_si91x_request_tx_test_info_t::reserved
```

tx test mode reserved

Definition at line 1515 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

no_of_pkts

```
uint16_t sl_si91x_request_tx_test_info_t::no_of_pkts
```

tx test mode no_of_pkts

Definition at line 1516 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

delay

```
uint32_t sl_si91x_request_tx_test_info_t::delay
```

tx test mode delay

Definition at line 1517 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

sl_si91x_async_stats_response_t

Si917 specific Wi-Fi asynchronous statistics.

Public Attributes

uint16_t	tx_pkts Number of tx pkts.
uint8_t	reserved_1 Number of rx pkts.
uint16_t	tx_retries Number of tx retries.
uint16_t	crc_pass Number ofpkts that pass crc.
uint16_t	crc_fail Number of pkts failing crc chk.
uint16_t	cca_stk Number of times cca got stuck.
uint16_t	cca_not_stk Number of times cca didn't get stuck.
uint16_t	pkt_abort Number of pkt aborts.
uint16_t	fls_rx_start Number of false rx starts.
uint16_t	cca_idle cca idle time
uint8_t	reserved_2 Reserved fields.
uint16_t	rx_retries Number of rx retries.
uint8_t	reserved_3 rssi value
uint16_t	cal_rssi cal_rssi
uint8_t	reserved_4 lna_gain bb_gain
uint16_t	xretries Number of tx packets dropped after maximum retries.
uint16_t	max_cons_pkts_dropped Consecutive pkts dropped.

uint8_t	reserved_5 Reserved fields.
uint16_t	bss_broadcast_pkts BSSID matched broadcast packets count.
uint16_t	bss_multicast_pkts BSSID matched multicast packets count.
uint16_t	bss_filter_matched_multicast_pkts BSSID & multicast filter matched packets count.

Public Attribute Documentation

tx_pkts

```
uint16_t sl_si91x_async_stats_response_t::tx_pkts
```

Number of tx pkts.

Definition at line 1690 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

reserved_1

```
uint8_t sl_si91x_async_stats_response_t::reserved_1[2]
```

Number of rx pkts.

Definition at line 1691 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

tx_retries

```
uint16_t sl_si91x_async_stats_response_t::tx_retries
```

Number of tx retries.

Definition at line 1692 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

crc_pass

```
uint16_t sl_si91x_async_stats_response_t::crc_pass
```

Number ofpkts that pass crc.

Definition at line 1693 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

crc_fail

```
uint16_t sl_si91x_async_stats_response_t::crc_fail
```

Number of pkts failing crc chk.

Definition at line 1694 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

cca_stk

```
uint16_t sl_si91x_async_stats_response_t::cca_stk
```

Number of times cca got stuck.

Definition at line 1695 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

cca_not_stk

```
uint16_t sl_si91x_async_stats_response_t::cca_not_stk
```

Number of times cca didn't get stuck.

Definition at line 1696 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

pkt_abort

```
uint16_t sl_si91x_async_stats_response_t::pkt_abort
```

Number of pkt aborts.

Definition at line 1697 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

fls_rx_start

```
uint16_t sl_si91x_async_stats_response_t::fls_rx_start
```

Number of false rx starts.

Definition at line 1698 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

cca_idle

```
uint16_t sl_si91x_async_stats_response_t::cca_idle
```

cca idle time

Definition at line 1699 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

reserved_2

```
uint8_t sl_si91x_async_stats_response_t::reserved_2[26]
```

Reserved fields.

Definition at line 1700 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

rx_retries

```
uint16_t sl_si91x_async_stats_response_t::rx_retries
```

Number of rx retries.

Definition at line 1701 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

reserved_3

```
uint8_t sl_si91x_async_stats_response_t::reserved_3[2]
```

rssi value

Definition at line 1702 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

cal_rssi

```
uint16_t sl_si91x_async_stats_response_t::cal_rssi
```

cal_rssi

Definition at line 1703 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

reserved_4

```
uint8_t sl_si91x_async_stats_response_t::reserved_4[4]
```

lna_gain bb_gain

Definition at line 1704 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

xretries

```
uint16_t sl_si91x_async_stats_response_t::xretries
```

Number of tx packets dropped after maximum retries.

Definition at line 1705 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

max_cons_pkts_dropped

```
uint16_t sl_si91x_async_stats_response_t::max_cons_pkts_dropped
```

Consecutive pkts dropped.

Definition at line 1706 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

reserved_5

```
uint8_t sl_si91x_async_stats_response_t::reserved_5[2]
```

Reserved fields.

Definition at line 1707 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

bss_broadcast_pkts

```
uint16_t sl_si91x_async_stats_response_t::bss_broadcast_pkts
```

BSSID matched broadcast packets count.

Definition at line 1708 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

bss_multicast_pkts

```
uint16_t sl_si91x_async_stats_response_t::bss_multicast_pkts
```

BSSID matched multicast packets count.

Definition at line 1709 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

bss_filter_matched_multicast_pkts

```
uint16_t sl_si91x_async_stats_response_t::bss_filter_matched_multicast_pkts
```

BSSID & multicast filter matched packets count.

Definition at line 1710 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

sl_si91x_advance_stats_response_t

Si917 specific Wi-Fi advance statistics.

Public Attributes

uint32_t	beacon_lost_count Number of missed beacons.
uint32_t	beacon_rx_count Number of received beacons.
uint32_t	mcast_rx_count Multicast packets received.
uint32_t	mcast_tx_count Multicast packets transmitted.
uint32_t	ucast_rx_count Unicast packets received.
uint32_t	ucast_tx_count Unicast packets transmitted.
uint32_t	overrun_count Number of packets dropped either at ingress or egress, due to lack of buffer memory to retain all packets.

Public Attribute Documentation

beacon_lost_count

```
uint32_t sl_si91x_advance_stats_response_t::beacon_lost_count
```

Number of missed beacons.

Definition at line 1715 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

beacon_rx_count

```
uint32_t sl_si91x_advance_stats_response_t::beacon_rx_count
```

Number of received beacons.

Definition at line 1716 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

mcast_rx_count

```
uint32_t sl_si91x_advance_stats_response_t::mcast_rx_count
```

Multicast packets received.

Definition at line 1717 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

mcast_tx_count

```
uint32_t sl_si91x_advance_stats_response_t::mcast_tx_count
```

Multicast packets transmitted.

Definition at line 1718 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

ucast_rx_count

```
uint32_t sl_si91x_advance_stats_response_t::ucast_rx_count
```

Unicast packets received.

Definition at line 1719 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

ucast_tx_count

```
uint32_t sl_si91x_advance_stats_response_t::ucast_tx_count
```

Unicast packets transmitted.

Definition at line 1720 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

overrun_count

```
uint32_t sl_si91x_advance_stats_response_t::overrun_count
```

Number of packets dropped either at ingress or egress, due to lack of buffer memory to retain all packets.

Definition at line 1722 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

sl_si91x_boot_configuration_t

Si91x boot configuration structure.

Public Attributes

uint16_t	oper_mode operation mode, one of the values from sl_si91x_operation_mode_t.
uint16_t	coex_mode coex mode, one of the values from sl_si91x_coex_mode_t.
uint32_t	feature_bit_map Feature bit map, Device Feature Bitmap.
uint32_t	tcp_ip_feature_bit_map TCP/IP feature bit map, TCP/IP Feature Bitmap.
uint32_t	custom_feature_bit_map Custom feature bit map, Custom Feature Bitmap.
uint32_t	ext_custom_feature_bit_map Extended custom feature bit map, Extended Custom Feature Bitmap.
uint32_t	bt_feature_bit_map BT featured bit map, Bluetooth Feature Bitmap.
uint32_t	ext_tcp_ip_feature_bit_map Extended tcp/ip feature bit map, Extended TCP/IP Feature Bitmap.
uint32_t	ble_feature_bit_map BLE feature bitmap, BLE Feature Bitmap.
uint32_t	ble_ext_feature_bit_map BLE extended feature bit map, Extended BLE Custom Feature Bitmap.
uint32_t	config_feature_bit_map Config feature bit map, Configuration Feature Bitmap.

Public Attribute Documentation

oper_mode

```
uint16_t sl_si91x_boot_configuration_t::oper_mode
```

operation mode, one of the values from sl_si91x_operation_mode_t.

Definition at line 130 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

coex_mode

```
uint16_t sl_si91x_boot_configuration_t::coex_mode
```

coex mode, one of the values from sl_si91x_coex_mode_t.

Definition at line 131 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::feature_bit_map
```

Feature bit map, Device Feature Bitmap.

Definition at line 132 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

tcp_ip_feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::tcp_ip_feature_bit_map
```

TCP/IP feature bit map, TCP/IP Feature Bitmap.

Definition at line 133 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

custom_feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::custom_feature_bit_map
```

Custom feature bit map, Custom Feature Bitmap.

Definition at line 134 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

ext_custom_feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::ext_custom_feature_bit_map
```

Extended custom feature bit map, Extended Custom Feature Bitmap.

Definition at line 135 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

bt_feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::bt_feature_bit_map
```

BT featured bit map, Bluetooth Feature Bitmap.

Definition at line 136 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

ext_tcp_ip_feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::ext_tcp_ip_feature_bit_map
```

Extended tcp/ip feature bit map, Extended TCP/IP Feature Bitmap.

Definition at line 137 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

ble_feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::ble_feature_bit_map
```

BLE feature bitmap, BLE Feature Bitmap.

Definition at line 138 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

ble_ext_feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::ble_ext_feature_bit_map
```

BLE extended feature bit map, Extended BLE Custom Feature Bitmap.

Definition at line 139 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

config_feature_bit_map

```
uint32_t sl_si91x_boot_configuration_t::config_feature_bit_map
```

Config feature bit map, Configuration Feature Bitmap.

Definition at line 140 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

sl_si91x_timeout_t

Timeout Configuration Structure.

Public Attributes

uint16_t	active_chan_scan_timeout_value Time spent on each channel when performing active scan (milliseconds). Default value of 100 millisecs is used when SL_WIFI_DEFAULT_ACTIVE_CHANNEL_SCAN_TIME is passed.
uint16_t	auth_assoc_timeout_value Authentication and association timeout value. Default value of 300 millisecs is used when SL_WIFI_DEFAULT_AUTH_ASSOCIATION_TIMEOUT is passed.
uint16_t	keep_alive_timeout_value Keep Alive Timeout value. Default value of 30 secs is used when SL_WIFI_DEFAULT_KEEP_ALIVE_TIMEOUT is passed.

Public Attribute Documentation

active_chan_scan_timeout_value

```
uint16_t sl_si91x_timeout_t::active_chan_scan_timeout_value
```

Time spent on each channel when performing active scan (milliseconds). Default value of 100 millisecs is used when SL_WIFI_DEFAULT_ACTIVE_CHANNEL_SCAN_TIME is passed.

Definition at line 146 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

auth_assoc_timeout_value

```
uint16_t sl_si91x_timeout_t::auth_assoc_timeout_value
```

Authentication and association timeout value. Default value of 300 millisecs is used when SL_WIFI_DEFAULT_AUTH_ASSOCIATION_TIMEOUT is passed.

Definition at line 148 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

keep_alive_timeout_value

```
uint16_t sl_si91x_timeout_t::keep_alive_timeout_value
```

Keep Alive Timeout value. Default value of 30 secs is used when SL_WIFI_DEFAULT_KEEP_ALIVE_TIMEOUT is passed.

Definition at line 150 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

sl_si91x_module_state_stats_response_t

Si917 specific Wi-Fi module state statistics.

Public Attributes

uint32_t	timestamp Timestamp.
uint8_t	state_code State code.
uint8_t	reason_code Reason code.
uint8_t	channel Channel number.
uint8_t	rssi RSSI VALUE.
uint8_t	bssid BSSID.

Public Attribute Documentation

timestamp

```
uint32_t sl_si91x_module_state_stats_response_t::timestamp
```

Timestamp.

Definition at line 156 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

state_code

```
uint8_t sl_si91x_module_state_stats_response_t::state_code
```

State code.

Definition at line 157 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

reason_code

```
uint8_t sl_si91x_module_state_stats_response_t::reason_code
```

Reason code.

Definition at line 158 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

channel

```
uint8_t sl_si91x_module_state_stats_response_t::channel
```

Channel number.

Definition at line 159 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

rssi

```
uint8_t sl_si91x_module_state_stats_response_t::rssi
```

RSSI VALUE.

Definition at line 160 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

bssid

```
uint8_t sl_si91x_module_state_stats_response_t::bssid[6]
```

BSSID.

Definition at line 161 of file components/device/silabs/si91x/wireless/inc/sl_si91x_types.h

sl_wifi_device_configuration_t

Device configuration for Si91x device.

Public Attributes

uint8_t	boot_option Boot option. One of the values from Load Image Types.
sl_mac_address_t *	mac_address mac address of type sl_mac_address_t
sl_si91x_band_mode_t	band Si91x Wi-Fi band of type sl_si91x_band_mode_t.
sl_si91x_region_code_t	region_code Si91x region code of type sl_si91x_region_code_t.
sl_si91x_boot_configuration_t	boot_config Si91x boot configuration, Refer Boot Configuration.
sl_si91x_dynamic_pool	ta_pool TA buffer allocation command parameters of tyoe sl_si91x_dynamic_pool.

Public Attribute Documentation

boot_option

```
uint8_t sl_wifi_device_configuration_t::boot_option
```

Boot option. One of the values from Load Image Types.

Definition at line 912 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

mac_address

```
sl_mac_address_t* sl_wifi_device_configuration_t::mac_address
```

mac address of type sl_mac_address_t

Definition at line 913 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

band

```
sl_si91x_band_mode_t sl_wifi_device_configuration_t::band
```

Si91x Wi-Fi band of type sl_si91x_band_mode_t.

Definition at line 914 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

region_code

> sl_si91x_region_code_t sl_wifi_device_configuration_t::region_code

Si91x region code of type sl_si91x_region_code_t.

Definition at line 915 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

boot_config

> sl_si91x_boot_configuration_t sl_wifi_device_configuration_t::boot_config

Si91x boot configuration, Refer Boot Configuration.

Definition at line 916 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

ta_pool

> sl_si91x_dynamic_pool sl_wifi_device_configuration_t::ta_pool

TA buffer allocation command parameters of tyoe sl_si91x_dynamic_pool.

Definition at line 917 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

sl_wifi_device_context_t

SL Wi-Fi device context.

Public Attributes

void *	device_context Reserved for future use.

Public Attribute Documentation

device_context

```
void* sl_wifi_device_context_t::device_context
```

Reserved for future use.

Definition at line 922 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

sl_bt_performance_profile_t

BT performance profile.

Public Attributes

sl_si91x_performance_profile_t	profile Performance profile.

Public Attribute Documentation

profile

```
sl_si91x_performance_profile_t sl_bt_performance_profile_t::profile
```

Performance profile.

Definition at line 950 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

sl_wifi_performance_profile_t

Wi-Fi performance profile.

Public Attributes

sl_si91x_performance_profile_t	profile Performance profile of type sl_si91x_performance_profile_t.
uint8_t	dtim_aligned_type Set DTIM alignment required. DTIM Alignment Types.
uint8_t	num_of_dtim_skip Default value to be used is 0.
uint16_t	listen_interval Listen interval.
uint16_t	monitor_interval Monitor interval will be in millisecs, Default interval 50 milli secs is used if monitor_interval is set to 0. This is only valid when performance profile is set to ASSOCIATED_POWER_SAVE_LOW_LATENCY.
sl_wifi_twt_request_t	twt_request twt request.
sl_wifi_twt_selection_t	twt_selection twt selection request

Public Attribute Documentation

profile

```
sl_si91x_performance_profile_t sl_wifi_performance_profile_t::profile
```

Performance profile of type sl_si91x_performance_profile_t.

Definition at line 955 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

dtim_aligned_type

```
uint8_t sl_wifi_performance_profile_t::dtim_aligned_type
```

Set DTIM alignment required. DTIM Alignment Types.

Definition at line 956 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

num_of_dtim_skip

```
uint8_t sl_wifi_performance_profile_t::num_of_dtim_skip
```

Default value to be used is 0.

Definition at line 957 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

listen_interval

> uint16_t sl_wifi_performance_profile_t::listen_interval

Listen interval.

Definition at line 958 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

monitor_interval

> uint16_t sl_wifi_performance_profile_t::monitor_interval

Monitor interval will be in millisecs, Default interval 50 milli secs is used if monitor_interval is set to 0. This is only valid when performance profile is set to ASSOCIATED_POWER_SAVE_LOW_LATENCY.

Definition at line 960 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

twt_request

> sl_wifi_twt_request_t sl_wifi_performance_profile_t::twt_request

twt request.

Definition at line 961 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

twt_selection

> sl_wifi_twt_selection_t sl_wifi_performance_profile_t::twt_selection

twt selection request

Definition at line 962 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Constants

Constants

This section provides a reference to Si91x Device Management constants.

Modules

Boot Configuration

Boot Configuration

This section provides a reference to Si91x boot configuration parameters.

Modules

TCP/IP Feature Bitmap

TCP/IP Feature Bitmap

This section provides a reference to the Transport Control Protocol and Internet Protocol (TCP/IP) feature bitmap in the boot configuration.

Macros

#define SL_SI91X_TCP_IP_FEAT_BYPASS BIT(0)
TCP/IP bypass feature.

#define SL_SI91X_TCP_IP_FEAT_HTTP_SERVER BIT(1)
Enable HTTP server feature.

#define SL_SI91X_TCP_IP_FEAT_DHCPV4_CLIENT BIT(2)
Enable DHCPv4 client feature.

#define SL_SI91X_TCP_IP_FEAT_DHCPV6_CLIENT BIT(3)
Enable DHCPv6 client feature.

#define SL_SI91X_TCP_IP_FEAT_DHCPV4_SERVER BIT(4)
Enable DHCPv4 server feature.

#define SL_SI91X_TCP_IP_FEAT_DHCPV6_SERVER BIT(5)
Enable DHCPv6 server feature.

#define SL_SI91X_TCP_IP_FEAT_JSON_OBJECTS BIT(6)
Enable JSON objects.

#define SL_SI91X_TCP_IP_FEAT_HTTP_CLIENT BIT(7)
Enable HTTP client.

#define SL_SI91X_TCP_IP_FEAT_DNS_CLIENT BIT(8)
Enable DNS client.

#define SL_SI91X_TCP_IP_FEAT_SNMP_AGENT BIT(9)
Enable SNMP client.

#define SL_SI91X_TCP_IP_FEAT_SSL BIT(10)
Enable SSL feature.

#define SL_SI91X_TCP_IP_FEAT_ICMP BIT(11)
Enable ICMP feature(ping)

#define SL_SI91X_TCP_IP_FEAT_HTTPS_SERVER BIT(12)
Enable HTTP server.

#define SL_SI91X_TCP_IP_FEAT_SEND_CONFIGS_TO_HOST BIT(14)
Enable sending web page configuration to host from wireless config page.

#define SL_SI91X_TCP_IP_FEAT_FTP_CLIENT BIT(15)
Enable FTP client.

#define SL_SI91X_TCP_IP_FEAT_SNTP_CLIENT BIT(16)
Enable SNTP client.

#define SL_SI91X_TCP_IP_FEAT_IPV6 BIT(17)
Enable IPV6 support.

#define SL_SI91X_TCP_IP_FEAT_RAW_DATA BIT(18)
Enable Raw data support.

#define SL_SI91X_TCP_IP_FEAT_MDNSD BIT(19)
Enable MDNSD.

#define SL_SI91X_TCP_IP_FEAT_SMTP_CLIENT BIT(20)
Enable SMTP client.

#define SL_SI91X_TCP_IP_TOTAL_SOCKETS (total_sockets)
Select no of sockets.

#define SL_SI91X_TCP_IP_FEAT_SINGLE_SSL_SOCKET BIT(25)
Enable Single SSL socket.

#define SL_SI91X_TCP_IP_FEAT_LOAD_PUBLIC_PRIVATE_CERTS BIT(26)
Enable to Load public and private keys for TLS and SSL handshake.

#define SL_SI91X_TCP_IP_FEAT_LOAD_CERTS_INTO_RAM BIT(27)
Enable to Load SSL certificates in to RAM.

#define SL_SI91X_TCP_IP_FEAT_POP3_CLIENT BIT(29)
Enable POP3 client.

#define SL_SI91X_TCP_IP_FEAT_OTAF BIT(30)
Enable OTAF client.

#define SL_SI91X_TCP_IP_FEAT_EXTENSION_VALID BIT(31)
Enable TCP IP Extension.

Macro Definition Documentation

SL_SI91X_TCP_IP_FEAT_BYPASS

```
#define SL_SI91X_TCP_IP_FEAT_BYPASS
```

Value:

```
BIT(0)
```

TCP/IP bypass feature.

Definition at line 111 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_HTTP_SERVER

```
#define SL_SI91X_TCP_IP_FEAT_HTTP_SERVER
```

Value:

```
BIT(1)
```

Enable HTTP server feature.

Definition at line 114 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_DHCPV4_CLIENT

```
#define SL_SI91X_TCP_IP_FEAT_DHCPV4_CLIENT
```

Value:

```
BIT(2)
```

Enable DHCPv4 client feature.

Definition at line 117 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_DHCPV6_CLIENT

```
#define SL_SI91X_TCP_IP_FEAT_DHCPV6_CLIENT
```

Value:

```
BIT(3)
```

Enable DHCPv6 client feature.

Definition at line 120 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_DHCPV4_SERVER

```
#define SL_SI91X_TCP_IP_FEAT_DHCPV4_SERVER
```

Value:

```
BIT(4)
```

Enable DHCPv4 server feature.

Definition at line 123 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_DHCPV6_SERVER

```
#define SL_SI91X_TCP_IP_FEAT_DHCPV6_SERVER
```

Value:

```
BIT(5)
```

Enable DHCPv6 server feature.

Definition at line 126 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_JSON_OBJECTS

```
#define SL_SI91X_TCP_IP_FEAT_JSON_OBJECTS
```

Value:

```
BIT(6)
```

Enable JSON objects.

Definition at line 129 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_HTTP_CLIENT

```
#define SL_SI91X_TCP_IP_FEAT_HTTP_CLIENT
```

Value:

```
BIT(7)
```

Enable HTTP client.

Definition at line 132 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_DNS_CLIENT

```
#define SL_SI91X_TCP_IP_FEAT_DNS_CLIENT
```

Value:

```
BIT(8)
```

Enable DNS client.

Definition at line 135 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_SNMP_AGENT

```
#define SL_SI91X_TCP_IP_FEAT_SNMP_AGENT
```

Value:

```
BIT(9)
```

Enable SNMP client.

Definition at line 138 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_SSL

```
#define SL_SI91X_TCP_IP_FEAT_SSL
```

Value:

```
BIT(10)
```

Enable SSL feature.

Definition at line 141 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_ICMP

```
#define SL_SI91X_TCP_IP_FEAT_ICMP
```

Value:

```
BIT(11)
```

Enable ICMP feature(ping)

Definition at line 144 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_HTTPS_SERVER

```
#define SL_SI91X_TCP_IP_FEAT_HTTPS_SERVER
```

Value:

```
BIT(12)
```

Enable HTTP server.

Note

- Only supported in opermode 0

Definition at line 148 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_SEND_CONFIGS_TO_HOST

```
#define SL_SI91X_TCP_IP_FEAT_SEND_CONFIGS_TO_HOST
```

Value:

```
BIT(14)
```

Enable sending web page configuration to host from wireless config page.

Note

- Bit 13 is reserved

Definition at line 153 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_FTP_CLIENT

```
#define SL_SI91X_TCP_IP_FEAT_FTP_CLIENT
```

Value:

```
BIT(15)
```

Enable FTP client.

Definition at line 156 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_SNTP_CLIENT

```
#define SL_SI91X_TCP_IP_FEAT_SNTP_CLIENT
```

Value:

```
BIT(16)
```

Enable SNTP client.

Definition at line 159 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_IPV6

```
#define SL_SI91X_TCP_IP_FEAT_IPV6
```

Value:

```
BIT(17)
```

Enable IPV6 support.

Note

- IPv6 will also get enabled if DHCP v6 client/DHCP v6 server is enabled irrespective of tcp_ip_feature_bit_map[17]

Definition at line 163 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_RAW_DATA

```
#define SL_SI91X_TCP_IP_FEAT_RAW_DATA
```

Value:

```
BIT(18)
```

Enable Raw data support.

Note

- This feature is supported only in AP mode. TCP_BYPASS feature should be disabled for this feature to be supported. If any packet from host with frame type 0x1 is received by firmware, the packet will be sent on air without TCP/IP stack processing. ARP and broadcast packets (other than DHCP packets) which are coming on air will be sent to host

Definition at line 167 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_MDNSD

```
#define SL_SI91X_TCP_IP_FEAT_MDNSD
```

Value:

```
BIT(19)
```

Enable MDNSD.

Definition at line 170 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_SMTP_CLIENT

```
#define SL_SI91X_TCP_IP_FEAT_SMTP_CLIENT
```

Value:

```
BIT(20)
```

Enable SMTP client.

Definition at line 173 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_TOTAL_SOCKETS

```
#define SL_SI91X_TCP_IP_TOTAL_SOCKETS
```

Value:

```
(total_sockets)
```

Select no of sockets.

Note

- Max of 10 sockets are allowed
- Bits 21- 24 are used to set TOTAL_SOCKETS

Definition at line 178 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_SINGLE_SSL_SOCKET

```
#define SL_SI91X_TCP_IP_FEAT_SINGLE_SSL_SOCKET
```

Value:

```
BIT(25)
```

Enable Single SSL socket.

Definition at line 181 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_LOAD_PUBLIC_PRIVATE_CERTS

```
#define SL_SI91X_TCP_IP_FEAT_LOAD_PUBLIC_PRIVATE_CERTS
```

Value:

```
BIT(26)
```

Enable to Load public and private keys for TLS and SSL handshake.

Note

- If Secure handshake is with CA-certificate alone , then disable loading private and public keys and erase these certificates from the flash using load_cert API. Or if Secure handshake is needed for verification of Private and Public keys , then enable loading of private and public keys.

Definition at line 185 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_LOAD_CERTS_INTO_RAM

```
#define SL_SI91X_TCP_IP_FEAT_LOAD_CERTS_INTO_RAM
```

Value:

```
BIT(27)
```

Enable to Load SSL certificates in to RAM.

Definition at line 188 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_POP3_CLIENT

```
#define SL_SI91X_TCP_IP_FEAT_POP3_CLIENT
```

Value:

```
BIT(29)
```

Enable POP3 client.

Note

- Bit 28 is reserved

Definition at line 193 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_OTAF

```
#define SL_SI91X_TCP_IP_FEAT_OTAF
```

Value:

```
BIT(30)
```

Enable OTAF client.

Definition at line 196 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TCP_IP_FEAT_EXTENSION_VALID

```
#define SL_SI91X_TCP_IP_FEAT_EXTENSION_VALID
```

Value:

```
BIT(31)
```

Enable TCP IP Extension.

Definition at line 199 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Extended TCP/IP Feature Bitmap

Extended TCP/IP Feature Bitmap

This section provides a reference to the Transport Control Protocol and Internet Protocol (TCP/IP) extended feature bitmap in the boot configuration.

Macros

#define SL_SI91X_EXT_TCP_FEAT_DHCP_OPT77 BIT(1)
DHCP USER CLASS.

#define SL_SI91X_EXT_TCP_IP_HTTP_SERVER_BYPASS BIT(2)
Enable HTTP server root path (Default configuration page) bypass.

#define SL_SI91X_EXT_TCP_IP_BI_DIR_ACK_UPDATE BIT(3)
TCP bi-dir ack update.

#define SL_SI91X_EXT_TCP_IP_WINDOW_DIV BIT(4)
TCP RX window division.

#define SL_SI91X_EXT_TCP_IP_CERT_BYPASS BIT(5)
SSL server certificate bypass, validation from the host.

#define SL_SI91X_EXT_TCP_IP_SSL_16K_RECORD BIT(6)
SSL 16K record size support.

#define SL_SI91X_EXT_TCP_IP_DNS_CLIENT_BYPASS BIT(7)
Enable DNS client bypass.

#define SL_SI91X_EXT_TCP_IP_WINDOW_SCALING BIT(8)
Enable TCP window scaling feature.

#define SL_SI91X_EXT_TCP_IP_DUAL_MODE_ENABLE BIT(9)
Enables both TCP/IP bypass mode & embedded modes.

#define SL_SI91X_EXT_TCP_IP_ETH_WIFI_BRIDGE BIT(10)
Enables Ethernet to WIFI bridge.

#define SL_SI91X_EXT_DYNAMIC_COEX_MEMORY BIT(11)
Enables the Dynamic coex memory.

#define SL_SI91X_EXT_TCP_IP_TOTAL_SELECTS (total_selects)
Configures the number of selects.

#define SL_SI91X_EXT_TCP_IP_WAIT_FOR_SOCKET_CLOSE BIT(16)
To enable socket wait close.

#define SL_SI91X_EXT_EMB_MQTT_ENABLE BIT(17)
Enable Embedded/internal MQTT.

#define SL_SI91X_EXT_TCP_IP_FEAT_SSL_HIGH_PERFORMANCE BIT(18)
Enables the SSL_HIGH_PERFORMANCE.

#define SL_SI91X_EXT_TCP_DYNAMIC_WINDOW_UPDATE_FROM_HOST BIT(19)
Enabled to update TCP window from host.

#define SL_SI91X_EXT_TCP_MAX_RECV_LENGTH BIT(20)
Enable to update max receive length for TCP.

#define SL_SI91X_EXT_TCP_IP_FEAT_SSL_THREE_SOCKETS BIT(29)
Enable three SSL/TLS sockets.

#define SL_SI91X_EXT_TCP_IP_FEAT_SSL_MEMORY_CLOUD BIT(30)
To configure additional memory for SSL/TLS connection typically to a cloud server.

#define SL_SI91X_CONFIG_FEAT_EXTENTION_VALID BIT(31)
config_feature_bit_map validity

Macro Definition Documentation

SL_SI91X_EXT_TCP_FEAT_DHCP_OPT77

```
#define SL_SI91X_EXT_TCP_FEAT_DHCP_OPT77
```

Value:

```
BIT(1)
```

DHCP USER CLASS.

Note

- bit 0 is reserved

Definition at line 519 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_HTTP_SERVER_BYPASS

```
#define SL_SI91X_EXT_TCP_IP_HTTP_SERVER_BYPASS
```

Value:

```
BIT(2)
```

Enable HTTP server root path (Default configuration page) bypass.

Definition at line 521 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_BI_DIR_ACK_UPDATE

```
#define SL_SI91X_EXT_TCP_IP_BI_DIR_ACK_UPDATE
```

Value:

```
BIT(3)
```

TCP bi-dir ack update.

Note

- Need to enable this bit if user wants to run the bi-directional data transfer.

Definition at line 524 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_WINDOW_DIV

```
#define SL_SI91X_EXT_TCP_IP_WINDOW_DIV
```

Value:

```
BIT(4)
```

TCP RX window division.

Definition at line 526 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_CERT_BYPASS

```
#define SL_SI91X_EXT_TCP_IP_CERT_BYPASS
```

Value:

```
BIT(5)
```

SSL server certificate bypass, validation from the host.

Definition at line 528 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_SSL_16K_RECORD

```
#define SL_SI91X_EXT_TCP_IP_SSL_16K_RECORD
```

Value:

```
BIT(6)
```

SSL 16K record size support.

Definition at line 530 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_DNS_CLIENT_BYPASS

```
#define SL_SI91X_EXT_TCP_IP_DNS_CLIENT_BYPASS
```

Value:

```
BIT(7)
```

Enable DNS client bypass.

Definition at line 532 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_WINDOW_SCALING

```
#define SL_SI91X_EXT_TCP_IP_WINDOW_SCALING
```

Value:

```
BIT(8)
```

Enable TCP window scaling feature.

Note

- If this feature is not enabled, then the maximum possible RX window size is 64 KB. If user wants to use more than 64KB window size, tcp_rx_window_size_cap in socket configuration is used to increase the window size.

Definition at line `535` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_EXT_TCP_IP_DUAL_MODE_ENABLE

```
#define SL_SI91X_EXT_TCP_IP_DUAL_MODE_ENABLE
```

Value:

```
BIT(9)
```

Enables both TCP/IP bypass mode & embedded modes.

Note

- Enabling this feature allows to use both bypass and non bypass modes simultaneously.

Definition at line `538` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_EXT_TCP_IP_ETH_WIFI_BRIDGE

```
#define SL_SI91X_EXT_TCP_IP_ETH_WIFI_BRIDGE
```

Value:

```
BIT(10)
```

Enables Ethernet to WIFI bridge.

Definition at line `540` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_EXT_DYNAMIC_COEX_MEMORY

```
#define SL_SI91X_EXT_DYNAMIC_COEX_MEMORY
```

Value:

```
BIT(11)
```

Enables the Dynamic coex memory.

Note

- To enable or disable the coex and update TCP RX window accordingly.

Definition at line 543 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_TOTAL_SELECTS

```
#define SL_SI91X_EXT_TCP_IP_TOTAL_SELECTS
```

Value:

```
(total_selects)
```

Configures the number of selects.

Note

- Bit 12 -15 are used for TOTAL_SELECTS and max value can be 10

Definition at line 546 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_WAIT_FOR_SOCKET_CLOSE

```
#define SL_SI91X_EXT_TCP_IP_WAIT_FOR_SOCKET_CLOSE
```

Value:

```
BIT(16)
```

To enable socket wait close.

Note

- If it is set socket will not be closed until close() is called from host. It is recommended to enable this bit when using TCP sockets.
- This is always set internally for Si91x chips

Definition at line 551 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_EMB_MQTT_ENABLE

```
#define SL_SI91X_EXT_EMB_MQTT_ENABLE
```

Value:

```
BIT(17)
```

Enable Embedded/internal MQTT.

Note

- If user wants to use AT command for MQTT, enable this bit in the Opermode Command

Definition at line 554 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_FEAT_SSL_HIGH_PERFORMANCE

```
#define SL_SI91X_EXT_TCP_IP_FEAT_SSL_HIGH_PERFORMANCE
```

Value:

```
BIT(18)
```

Enables the SSL_HIGH_PERFORMANCE.

Note

- To do firmware upgrade with HTTP this bit should be enabled

Definition at line 557 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_DYNAMIC_WINDOW_UPDATE_FROM_HOST

```
#define SL_SI91X_EXT_TCP_DYNAMIC_WINDOW_UPDATE_FROM_HOST
```

Value:

```
BIT(19)
```

Enabled to update TCP window from host.

Definition at line 559 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_MAX_RECV_LENGTH

```
#define SL_SI91X_EXT_TCP_MAX_RECV_LENGTH
```

Value:

```
BIT(20)
```

Enable to update max receive length for TCP.

Definition at line 562 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_FEAT_SSL_THREE_SOCKETS

```
#define SL_SI91X_EXT_TCP_IP_FEAT_SSL_THREE_SOCKETS
```

Value:

```
BIT(29)
```

Enable three SSL/TLS sockets.

Note

- Bit 21-28 are reserved
- Set tcp_ip_feature_bit_map[31] and ext_tcp_ip_feature_bit_map[29] to open 3 TLS sockets

Definition at line 568 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_TCP_IP_FEAT_SSL_MEMORY_CLOUD

```
#define SL_SI91X_EXT_TCP_IP_FEAT_SSL_MEMORY_CLOUD
```

Value:

```
BIT(30)
```

To configure additional memory for SSL/TLS connection typically to a cloud server.

Note

- If user connects to a cloud server using two SSL/TLS connections then it is required to set this bit to avoid 0xD2 error

Definition at line 572 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CONFIG_FEAT_EXTENTION_VALID

```
#define SL_SI91X_CONFIG_FEAT_EXTENTION_VALID
```

Value:

```
BIT(31)
```

config_feature_bit_map validity

Definition at line 575 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

BLE Feature Bitmap

BLE Feature Bitmap

This section provides a reference to the Bluetooth Low Energy (BLE) feature bitmap in the boot configuration.

Macros

#define SL_SI91X_BLE_MAX_NBR_ATT_REC (max_num_of_att_rec)
BLE number of attributes,.

#define SL_SI91X_BLE_MAX_NBR_ATT_SERV (max_num_of_att_serv)
BLE number of GATT services.

#define SL_SI91X_BLE_MAX_NBR_PERIPHERALS (max_num_of_peripherals)
BLE number of peripherals.

#define SL_SI91X_BLE_PWR_INX (power_index)
BLE Tx power index.

#define SL_SI91X_BLE_PWR_SAVE_OPTIONS (duty_cycle)
BLE powersave options Bit 24 for BLE_DUTY_CYCLING Bit 25 for BLR_DUTY_CYCLING Bit 26 for BLE_4X_PWR_SAVE_MODE For BLE_DISABLE_DUTY_CYCLING, BITs 24-26 are set to be zero.

#define SL_SI91X_BLE_MAX_NBR_CENTRALS (max_num_of_centrals)
Number of Centrals.

#define SL_SI91X_BLE_GATT_ASYNC_ENABLE BIT(29)
GATT SYNC BIT.

#define SL_SI91X_916_BLE_COMPATIBLE_FEAT_ENABLE BIT(30)
0 for 9113 compatible; 1 for enabling 9116 BLE-compatible features.

#define SL_SI91X_FEAT_BLE_CUSTOM_FEAT_EXTENTION_VALID BIT(31)
Extention valid to use Extended custom feature bitmap.

Macro Definition Documentation

SL_SI91X_BLE_MAX_NBR_ATT_REC

```
#define SL_SI91X_BLE_MAX_NBR_ATT_REC
```

Value:

```
(max_num_of_att_rec)
```

BLE number of attributes,.

Note

- Maximum No of BLE attributes = 80, refer NOTE given below for more info
- Bits 0 -7 are used to set MAX_NBR_ATT_REC

Definition at line 652 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_MAX_NBR_ATT_SERV

```
#define SL_SI91X_BLE_MAX_NBR_ATT_SERV
```

Value:

```
(max_num_of_att_serv)
```

BLE number of GATT services.

Note

- Maximum no services - 10, refer NOTE given below for more info
- Bits 8 - 11 are used to set MAX_NBR_ATT_SERV

Definition at line 657 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_MAX_NBR_PERIPHERALS

```
#define SL_SI91X_BLE_MAX_NBR_PERIPHERALS
```

Value:

```
(max_num_of_peripherals)
```

BLE number of peripherals.

Note

- Maximum No of BLE peripherals = 8, refer NOTE given below for more info
- Bits 12 - 15 are used to set MAX_NBR_PERIPHERALS

Definition at line 661 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_PWR_INX

```
#define SL_SI91X_BLE_PWR_INX
```

Value:

```
(power_index)
```

BLE Tx power index.

Note

- Give 31 as BLE TX power index (e.g.,: 31<<16) This variable is used to select the BLE TX power index value. The following are the possible values. Default Value for BLE Tx Power Index is 31 Range for the BLE Tx Power Index is 1 to 75 (0, 32 index is invalid) 1 - 31 BLE -0DBM Mode 33 - 63 BLE- 10DBM Mode 64- 75 BLE - HP Mode.
- Bits 16 - 23 are used to set PWR_INX

Definition at line 665 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_PWR_SAVE_OPTIONS

```
#define SL_SI91X_BLE_PWR_SAVE_OPTIONS
```

Value:

```
(duty_cycle)
```

BLE powersave options Bit 24 for BLE_DUTY_CYCLING Bit 25 for BLR_DUTY_CYCLING Bit 26 for BLE_4X_PWR_SAVE_MODE For BLE_DISABLE_DUTY_CYCLING, BITs 24-26 are set to be zero.

Note

- This feature is not supported in the current release

Definition at line `673` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BLE_MAX_NBR_CENTRALS

```
#define SL_SI91X_BLE_MAX_NBR_CENTRALS
```

Value:

```
(max_num_of_centrals)
```

Number of Centrals.

Note

- Maximum number of BLE Centrals = 2. refer to the note below for more info.
- Bits 27 - 28 are used to set BLE_PWR_INX

Definition at line `678` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BLE_GATT_ASYNC_ENABLE

```
#define SL_SI91X_BLE_GATT_ASYNC_ENABLE
```

Value:

```
BIT(29)
```

GATT SYNC BIT.

Note

- Default Disabled Expectation of GATT Async Bit Enable: Response structure will be filled in the Event and Event will come later. Not in sync with response for query command.

Definition at line `681` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_916_BLE_COMPATIBLE_FEAT_ENABLE

```
#define SL_SI91X_916_BLE_COMPATIBLE_FEAT_ENABLE
```

Value:

```
BIT(30)
```

0 for 9113 compatible; 1 for enabling 9116 BLE-compatible features.

Definition at line `684` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_FEAT_BLE_CUSTOM_FEAT_EXTENTION_VALID

```
#define SL_SI91X_FEAT_BLE_CUSTOM_FEAT_EXTENTION_VALID
```

Value:

```
BIT(31)
```

Extention valid to use Extended custom feature bitmap.

Definition at line `687` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

Extended BLE Custom Feature Bitmap

Extended BLE Custom Feature Bitmap

This section provides a reference to the Bluetooth Low Energy (BLE) extended custom feature bitmap in the boot configuration.

Macros

#define SL_SI91X_BLE_NUM_CONN_EVENTS (num_conn_events)
BLE number of Connection Events.

#define SL_SI91X_BLE_NUM_REC_BYTES (num_rec_bytes)
BLE number of record size in bytes (n)

#define SL_SI91X_BLE_GATT_INIT BIT(13)
GATT INIT.

#define SL_SI91X_BLE_INDICATE_CONFIRMATION_FROM_HOST BIT(14)
Indication response from APP.

#define SL_SI91X_BLE_MTU_EXCHANGE_FROM_HOST BIT(15)
MTU Exchange request initiation from APP.

#define SL_SI91X_BLE_SET_SCAN_RESP_DATA_FROM_HOST BIT(16)
Set SCAN Resp Data from APP.

#define SL_SI91X_BLE_DISABLE_CODED_PHY_FROM_HOST BIT(17)
Disable Coded PHY from APP.

#define SL_SI91X_BLE_ENABLE_ADV_EXTN BIT(19)

#define SL_SI91X_BLE_AE_MAX_ADV_SETS (num_adv_sets)

Macro Definition Documentation

SL_SI91X_BLE_NUM_CONN_EVENTS

```
#define SL_SI91X_BLE_NUM_CONN_EVENTS
```

Value:

```
(num_conn_events)
```

BLE number of Connection Events.

Note

- Describes the number of buffers need to be allocated for BLE on the opermode. By default each role (central/peripheral) will be allocated with 1 buffer for the notify/write command We increase the buffer capacity for the notify/write cmds to achieve the best throughput. See rsi_ble_set_wo_resp_notify_buf_info() to set more buffers for the notify/write commands
- Bits 0 - 4 are used to set NUM_CONN_EVENTS

Definition at line 698 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_NUM_REC_BYTES

```
#define SL_SI91X_BLE_NUM_REC_BYTES
```

Value:

```
(num_rec_bytes)
```

BLE number of record size in bytes (n)

Note

- n*16 : (n=60, Default 1024 bytes(1K))
- Bits 5 - 12 are used to set NUM_REC_BYTES

Definition at line 703 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_GATT_INIT

```
#define SL_SI91X_BLE_GATT_INIT
```

Value:

```
BIT(13)
```

GATT INIT.

Note

- 0 - GATT Init in Firmware i.e both the GAP service and GATT service will be maintained by Firmware 1 - GATT Init in Host i.e GAP service and GATT service should be created by the APP/Host/User and the ATT transactions like read, write, notify and indicate shall be handled by the APP/Host/User. Default Gatt Init in Firmware

Definition at line 707 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_INDICATE_CONFIRMATION_FROM_HOST

```
#define SL_SI91X_BLE_INDICATE_CONFIRMATION_FROM_HOST
```

Value:

```
BIT(14)
```

Indication response from APP.

Note

- As per ATT protocol for every indication received from the server should be acknowledged (indication response) by the Client. If this bit is disabled then firmware will send the acknowledgment(indication response) and if the bit is enabled then APP/Host/User needs to send the acknowledgment(indication response).

Definition at line 710 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_MTU_EXCHANGE_FROM_HOST

```
#define SL_SI91X_BLE_MTU_EXCHANGE_FROM_HOST
```

Value:

```
BIT(15)
```

MTU Exchange request initiation from APP.

Note

- If this bit is disabled, the firmware will initiate the MTU request to the remote device on the successful connection. And if Peer initiates MTU exchange Request then firmware will send Exchange MTU Response in reply to a received Exchange MTU Request. If this bit enabled then APP/Host/User need to initiate the MTU request by using the rsi_ble_mtu_exchange_event API. And if Peer initiates MTU exchange Request then APP/Host/User shall send Exchange MTU Response in reply to a received Exchange MTU Request using rsi_ble_mtu_exchange_resp API

Definition at line `714` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BLE_SET_SCAN_RESP_DATA_FROM_HOST

```
#define SL_SI91X_BLE_SET_SCAN_RESP_DATA_FROM_HOST
```

Value:

```
BIT(16)
```

Set SCAN Resp Data from APP.

Note

- Device will maintain some default scan response data and will be used in the scan_response controller frame. By enabling this bit we can make the default data as Null(empty).

Definition at line `717` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BLE_DISABLE_CODED_PHY_FROM_HOST

```
#define SL_SI91X_BLE_DISABLE_CODED_PHY_FROM_HOST
```

Value:

```
BIT(17)
```

Disable Coded PHY from APP.

Note

- Device will support the LE-coded phy feature (i.e LR - 125kbps and 500kbps) by default. If this bit is enabled, the device will not the support of the LE-coded phy rates.

Definition at line `720` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BLE_ENABLE_ADV_EXTN

```
#define SL_SI91X_BLE_ENABLE_ADV_EXTN
```

Value:

```
BIT(19)
```

Definition at line 722 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BLE_AE_MAX_ADV_SETS

```
#define SL_SI91X_BLE_AE_MAX_ADV_SETS
```

Value:

```
(num_adv_sets)
```

Definition at line 724 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Bluetooth Feature Bitmap

Bluetooth Feature Bitmap

This section provides a reference to the Bluetooth feature bitmap in the boot configuration.

Macros

#define SL_SI91X_BT_BDR_MODE_ENABLE BIT(0)
BT Mode Enable.

#define SL_SI91X_BT_BDR_MODE_LP_CHAIN_ENABLE BIT(1)
BT Mode LP Chain Enable.

#define SL_SI91X_BT_PWR_CTRL_DISABLE BIT(2)
BT Power Control Disable.

#define SL_SI91X_BT_EDR_3MBPS_DISABLE BIT(8)
BT EDR 3Mbps Feature Disable.

#define SL_SI91X_BT_EDR_2MBPS_DISABLE BIT(9)
BT EDR 2Mbps Feature Disable.

#define SL_SI91X_BT_5_SLOT_PACKETS_DISABLE BIT(10)
BT 5 Slot Packet Feature Disable.

#define SL_SI91X_BT_3_SLOT_PACKETS_DISABLE BIT(11)
BT 3 Slot Packet Feature Disable.

#define SL_SI91X_TA_BASED_ENCODER_ENABLE BIT(14)
TA based encoder.

#define SL_SI91X_HFP_PROFILE_ENABLE BIT(15)
To enable HFP profile.

#define SL_SI91X_A2DP_PROFILE_ENABLE BIT(23)
To enable A2DP profile.

#define SL_SI91X_A2DP_SOURCE_ROLE_ENABLE BIT(24)
To enable A2DP Profile role as source/sink.

#define SL_SI91X_A2DP_ACCELERATOR_MODE_ENABLE BIT(25)
To enable A2DP Accelerator mode.

#define SL_SI91X_BT_BLE_CP_BUFF_SIZE BIT(27)
To enable Buffer Alignment for Test Mode.

#define SL_SI91X_BT_ATT_OVER_CLASSIC_ACL BIT(29)

#define SL_SI91X_BT_RF_TYPE BIT(30)
To bt rf type.

#define SL_SI91X_ENABLE_BLE_PROTOCOL BIT(31)
To enable ble protocol.

Macro Definition Documentation

SL_SI91X_BT_BDR_MODE_ENABLE

```
#define SL_SI91X_BT_BDR_MODE_ENABLE
```

Value:

```
BIT(0)
```

BT Mode Enable.

Note

- If this Bit sets, then our Controller will sends/receives the packets in 1Mbps mode. Else, Controller will be operable in both BR/EDR mode

Definition at line `585` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BT_BDR_MODE_LP_CHAIN_ENABLE

```
#define SL_SI91X_BT_BDR_MODE_LP_CHAIN_ENABLE
```

Value:

```
BIT(1)
```

BT Mode LP Chain Enable.

Definition at line `588` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BT_PWR_CTRL_DISABLE

```
#define SL_SI91X_BT_PWR_CTRL_DISABLE
```

Value:

```
BIT(2)
```

BT Power Control Disable.

Note

- To make Fixed/Adaptive Power

Definition at line `592` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BT_EDR_3MBPS_DISABLE

```
#define SL_SI91X_BT_EDR_3MBPS_DISABLE
```

Value:

```
BIT(8)
```

BT EDR 3Mbps Feature Disable.

Note

- Bit 3-7 are reserved
- 3Mbps Disable means using 2Mbps DataRate

Definition at line 598 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BT_EDR_2MBPS_DISABLE

```
#define SL_SI91X_BT_EDR_2MBPS_DISABLE
```

Value:

```
BIT(9)
```

BT EDR 2Mbps Feature Disable.

@note2Mbps Disable means using 3Mbps DataRate

Definition at line 602 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BT_5_SLOT_PACKETS_DISABLE

```
#define SL_SI91X_BT_5_SLOT_PACKETS_DISABLE
```

Value:

```
BIT(10)
```

BT 5 Slot Packet Feature Disable.

Definition at line 605 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BT_3_SLOT_PACKETS_DISABLE

```
#define SL_SI91X_BT_3_SLOT_PACKETS_DISABLE
```

Value:

```
BIT(11)
```

BT 3 Slot Packet Feature Disable.

Definition at line 608 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_TA_BASED_ENCODER_ENABLE

```
#define SL_SI91X_TA_BASED_ENCODER_ENABLE
```

Value:

```
BIT(14)
```

TA based encoder.

Note

- Bits 12-13 bit are reserved
- To Enable/Disable SBC Encoder in Firmware.This feature is not supported in the current release

Definition at line 614 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_HFP_PROFILE_ENABLE

```
#define SL_SI91X_HFP_PROFILE_ENABLE
```

Value:

```
BIT(15)
```

To enable HFP profile.

Note

- This feature is not supported in the current release

Definition at line 617 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_A2DP_PROFILE_ENABLE

```
#define SL_SI91X_A2DP_PROFILE_ENABLE
```

Value:

```
BIT(23)
```

To enable A2DP profile.

Note

- Bits 16 - 22 are reserved

Definition at line 622 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_A2DP_SOURCE_ROLE_ENABLE

```
#define SL_SI91X_A2DP_SOURCE_ROLE_ENABLE
```

Value:

```
BIT(24)
```

To enable A2DP Profile role as source/sink.

Definition at line 624 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_A2DP_ACCELERATOR_MODE_ENABLE

```
#define SL_SI91X_A2DP_ACCELERATOR_MODE_ENABLE
```

Value:

```
BIT(25)
```

To enable A2DP Accelerator mode.

Note

- This feature is not supported in the current release

Definition at line 627 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BT_BLE_CP_BUFF_SIZE

```
#define SL_SI91X_BT_BLE_CP_BUFF_SIZE
```

Value:

```
BIT(27)
```

To enable Buffer Alignment for Test Mode.

Note

- Bit 26 is reserved
- To get 512 bytes from common pool, need to enable this, else 320 bytes will be fixed

Definition at line 633 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BT_ATT_OVER_CLASSIC_ACL

```
#define SL_SI91X_BT_ATT_OVER_CLASSIC_ACL
```

Value:

```
BIT(29)
```

Note

- Bit 28 is reserved

Definition at line 636 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BT_RF_TYPE

```
#define SL_SI91X_BT_RF_TYPE
```

Value:

```
BIT(30)
```

To bt rf type.

Definition at line 639 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_ENABLE_BLE_PROTOCOL

```
#define SL_SI91X_ENABLE_BLE_PROTOCOL
```

Value:

```
BIT(31)
```

To enable ble protocol.

Definition at line 641 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Device Feature Bitmap

Device Feature Bitmap

This section provides a reference to the SiWx91x device feature bitmap in the boot configuration.

Macros

#define SL_SI91X_FEAT_SECURITY_OPEN BIT(0)
Security type open.

#define SL_SI91X_FEAT_SECURITY_PSK BIT(1)
Security type WPA/WPA2.

#define SL_SI91X_FEAT_AGGREGATION BIT(2)
Aggregation support.

#define SL_SI91X_FEAT_LP_GPIO_BASED_HANDSHAKE BIT(3)
LP mode GPIO handshake.

#define SL_SI91X_FEAT_ULP_GPIO_BASED_HANDSHAKE BIT(4)
ULP mode GPIO based handshake.

#define SL_SI91X_FEAT_DEV_TO_HOST_ULP_GPIO_1 BIT(5)
To select ULP GPIO 1 for wake up indication.

#define SL_SI91X_FEAT_RF_SUPPLY_VOL_3_3_VOLT BIT(6)
To supply 3.3 volt supply.

#define SL_SI91X_FEAT_WPS_DISABLE BIT(7)
To Disable WPS in AP mode.

#define SL_SI91X_FEAT_EAP_LEAP_IN_COEX BIT(8)
To enable EAP-LEAP mode.

#define SL_SI91X_FEAT_HIDE_PSK_CREDENTIALS BIT(9)
To hide sensitive information (PSK, PMK, EAP credentials)

#define SL_SI91X_FEAT_SSL_HIGH_STREAMING_BIT BIT(10)
To enable high SSL streaming throughput.

#define SL_SI91X_FEAT_SECURE_ATTESTATION BIT(30)
Secure Attestation.

Macro Definition Documentation

SL_SI91X_FEAT_SECURITY_OPEN

```
#define SL_SI91X_FEAT_SECURITY_OPEN
```

Value:

```
BIT(0)
```

Security type open.

Note

- Supported in client mode

Definition at line 66 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_SECURITY_PSK

```
#define SL_SI91X_FEAT_SECURITY_PSK
```

Value:

```
BIT(1)
```

Security type WPA/WPA2.

Note

- Supported in client mode

Definition at line 70 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_AGGREGATION

```
#define SL_SI91X_FEAT_AGGREGATION
```

Value:

```
BIT(2)
```

Aggregation support.

Definition at line 73 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_LP_GPIO_BASED_HANDSHAKE

```
#define SL_SI91X_FEAT_LP_GPIO_BASED_HANDSHAKE
```

Value:

```
BIT(3)
```

LP mode GPIO handshake.

Definition at line 76 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_ULP_GPIO_BASED_HANDSHAKE

```
#define SL_SI91X_FEAT_ULP_GPIO_BASED_HANDSHAKE
```

Value:

```
BIT(4)
```

ULP mode GPIO based handshake.

Definition at line 79 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_DEV_TO_HOST_ULP_GPIO_1

```
#define SL_SI91X_FEAT_DEV_TO_HOST_ULP_GPIO_1
```

Value:

```
BIT(5)
```

To select ULP GPIO 1 for wake up indication.

Definition at line 82 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_RF_SUPPLY_VOL_3_3_VOLT

```
#define SL_SI91X_FEAT_RF_SUPPLY_VOL_3_3_VOLT
```

Value:

```
BIT(6)
```

To supply 3.3 volt supply.

Definition at line 85 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_WPS_DISABLE

```
#define SL_SI91X_FEAT_WPS_DISABLE
```

Value:

```
BIT(7)
```

To Disable WPS in AP mode.

Definition at line 88 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_EAP_LEAP_IN_COEX

```
#define SL_SI91X_FEAT_EAP_LEAP_IN_COEX
```

Value:

```
BIT(8)
```

To enable EAP-LEAP mode.

Definition at line 91 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_HIDE_PSK_CREDENTIALS

```
#define SL_SI91X_FEAT_HIDE_PSK_CREDENTIALS
```

Value:

```
BIT(9)
```

To hide sensitive information (PSK, PMK, EAP credentials)

Definition at line 94 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_SSL_HIGH_STREAMING_BIT

```
#define SL_SI91X_FEAT_SSL_HIGH_STREAMING_BIT
```

Value:

```
BIT(10)
```

To enable high SSL streaming throughput.

Note

- bits 11 - 29 are reserved

Definition at line 98 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_SECURE_ATTESTATION

```
#define SL_SI91X_FEAT_SECURE_ATTESTATION
```

Value:

```
BIT(30)
```

Secure Attestation.

Note

- bit 31 is reserved

Definition at line 102 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Calibration Flags

Calibration Flags

This section provides a reference to the calibration flags in the boot configuration.

Macros

#define SL_SI91X_BURN_GAIN_OFFSET BIT(0)
Burn Gain offset into the device.

#define SL_SI91X_BURN_FREQ_OFFSET BIT(1)
Burn Frequency offset into the device.

#define SL_SI91X_SW_XO_CTUNE_VALID BIT(2)
Software XO CTUNE is valid.

#define SL_SI91X_BURN_XO_FAST_DISABLE BIT(3)
Burn Bit to disable XO Fast into the device.

Macro Definition Documentation

SL_SI91X_BURN_GAIN_OFFSET

```
#define SL_SI91X_BURN_GAIN_OFFSET
```

Value:

```
BIT(0)
```

Burn Gain offset into the device.

Definition at line 887 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BURN_FREQ_OFFSET

```
#define SL_SI91X_BURN_FREQ_OFFSET
```

Value:

```
BIT(1)
```

Burn Frequency offset into the device.

Definition at line 888 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_SW_XO_CTUNE_VALID

```
#define SL_SI91X_SW_XO_CTUNE_VALID
```

Value:

```
BIT(2)
```

Software XO CTUNE is valid.

Definition at line `889` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_BURN_XO_FAST_DISABLE

```
#define SL_SI91X_BURN_XO_FAST_DISABLE
```

Value:

```
BIT(3)
```

Burn Bit to disable XO Fast into the device.

Definition at line `890` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

Burn Target Options

Burn Target Options

This section provides a reference to the SiWx91x burn target options in the boot configuration.

Macros

#define SL_SI91X_BURN_INTO_EFUSE 0
Burn data to EFUSE.

#define SL_SI91X_BURN_INTO_FLASH 1
Burn data to Flash.

Macro Definition Documentation

SL_SI91X_BURN_INTO_EFUSE

```
#define SL_SI91X_BURN_INTO_EFUSE
```

Value:

```
0
```

Burn data to EFUSE.

Definition at line 880 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_BURN_INTO_FLASH

```
#define SL_SI91X_BURN_INTO_FLASH
```

Value:

```
1
```

Burn data to Flash.

Definition at line 882 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Configuration Feature Bitmap

Configuration Feature Bitmap

This section provides a reference to the configuration feature bitmap in the boot configuration.

Note

- Bits 21 -31 are reserved

Macros

#define SL_SI91X_FEAT_SLEEP_GPIO_SEL_BITMAP BIT(0)
To select wakeup indication to host. If it is disabled UULP_GPIO_3 is used as a wakeup indication to host. If it is enabled UULP_GPIO_0 is used as a wakeup indication to host.

#define SL_SI91X_FEAT_DVS_SEL_CONFIG_1 BIT(2)
DVS Dynamic Voltage Selection.

#define SL_SI91X_FEAT_DVS_SEL_CONFIG_2 BIT(3)
Dynamic Voltage selection configuration 2.

#define SL_SI91X_FEAT_DVS_SEL_CONFIG_3 BIT(4)
Dynamic Voltage selection configuration 3.

#define SL_SI91X_FEAT_DVS_SEL_CONFIG_4 BIT(5)
Dynamic Voltage selection configuration 4.

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_100us BIT(6)
External PMU Selection.

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_200us BIT(7)
200us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_300us (BIT(6) | BIT(7))
300us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_400us BIT(8)
400us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_500us (BIT(6) | BIT(8))
500us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_600us (BIT(7) | BIT(8))
600us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_700us (BIT(6) | BIT(7) | BIT(8))
700us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_800us BIT(9)
800us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_900us (BIT(6) | BIT(9))
900us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1000us (BIT(7) | BIT(9))
1000us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1100us (BIT(6) | BIT(7) | BIT(9))
1100us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1200us (BIT(8) | BIT(9))
1200us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1300us (BIT(6) | BIT(8) | BIT(9))
1300us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1400us (BIT(7) | BIT(8) | BIT(9))
1400us External PMU Good time

#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1500us (BIT(6) | BIT(7) | BIT(8) | BIT(9))
1500us External PMU Good time

#define SL_SI91X_FEAT_EXTERNAL_LDO_SEL BIT(10)
External LDO voltage selection.

#define SL_SI91X_FEAT_EXTERNAL_LDO_VOL BIT(11)
This field valid only if RSI_FEAT_EXTERNAL_LDO_SEL is enabled i.e BIT(10) is set.

#define SL_SI91X_FEAT_EAP_TLS_V1P0 BIT(14)

#define SL_SI91X_FEAT_EAP_TLS_V1P2 BIT(15)
Enterprise security TLS version 1.2.

#define SL_SI91X_FEAT_CONC_STA_AP_DYN_SWITCH_SEL BIT(17)
Reserved bit.

#define SL_SI91X_ULP_GPIO9_FOR_UART2_TX BIT(18)
Bit to select ULP_GPIO_9 as UART2 port for device network processor debug prints. If this bit is not set, then by default UART2-TX GPIO_6 will be used.

#define SL_SI91X_FEAT_DISABLE_MCS_5_6_7_DATARATES BIT(19)
Bit to disable Short-GI.

#define SL_SI91X_FEAT_DISABLE_SHORT_GI BIT(20)

#define SL_SI91X_PTA_3WIRE_EN BIT(21)
To enable PTA-3WIRE.

#define SL_SI91X_PTA_3WIRE_CONFIG_SEL (config_sel)
To choose PTA-3WIRE.

#define SL_SI91X_XTAL_GOODTIME_1000us 0
XTAL goodtime configurations.

#define SL_SI91X_XTAL_GOODTIME_2000us BIT(24)
2000us XTAL Good Time

#define SL_SI91X_XTAL_GOODTIME_3000us BIT(25)
3000us XTAL Good Time

#define SL_SI91X_XTAL_GOODTIME_600us (BIT(24) | BIT(25))
600us XTAL Good Time

#define SL_SI91X_ENABLE_ENHANCED_MAX_PSP BIT(26)
Bit to enable Enhanced Max PSP.

#define SL_SI91X_ENABLE_DEBUG_BBP_TEST_PINS BIT(27)
Bit to enable BBP Test Pins.

Macro Definition Documentation

SL_SI91X_FEAT_SLEEP_GPIO_SEL_BITMAP

```
#define SL_SI91X_FEAT_SLEEP_GPIO_SEL_BITMAP
```

Value:

```
BIT(0)
```

To select wakeup indication to host. If it is disabled UULP_GPIO_3 is used as a wakeup indication to host. If it is enabled UULP_GPIO_0 is used as a wakeup indication to host.

Definition at line 735 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_DVS_SEL_CONFIG_1

```
#define SL_SI91X_FEAT_DVS_SEL_CONFIG_1
```

Value:

```
BIT(2)
```

DVS Dynamic Voltage Selection.

Note

- Bit 1 is reserved
- These bits are used for dynamic voltage selection Dynamic Voltage selection configuration 1

Definition at line 743 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_DVS_SEL_CONFIG_2

```
#define SL_SI91X_FEAT_DVS_SEL_CONFIG_2
```

Value:

```
BIT(3)
```

Dynamic Voltage selection configuration 2.

Definition at line 745 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_DVS_SEL_CONFIG_3

```
#define SL_SI91X_FEAT_DVS_SEL_CONFIG_3
```

Value:

```
BIT(4)
```

Dynamic Voltage selection configuration 3.

Definition at line 747 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_DVS_SEL_CONFIG_4

```
#define SL_SI91X_FEAT_DVS_SEL_CONFIG_4
```

Value:

```
BIT(5)
```

Dynamic Voltage selection configuration 4.

Definition at line 749 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_100us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_100us
```

Value:

```
BIT(6)
```

External PMU Selection.

Note

- These bits are used to select external PMU good time.1 to 15 means 100 usec to 1500 usec (in 100 usec granularity) 100us External PMU Good time

Definition at line 755 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_200us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_200us
```

Value:

```
BIT(7)
```

200us External PMU Good time

Definition at line 757 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_300us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_300us
```

Value:

```
(BIT(6) | BIT(7))
```

300us External PMU Good time

Definition at line 759 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_400us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_400us
```

Value:

```
BIT(8)
```

400us External PMU Good time

Definition at line 761 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_500us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_500us
```

Value:

```
(BIT(6) | BIT(8))
```

500us External PMU Good time

Definition at line 763 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_600us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_600us
```

Value:

```
(BIT(7) | BIT(8))
```

600us External PMU Good time

Definition at line 765 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_700us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_700us
```

Value:

```
(BIT(6) | BIT(7) | BIT(8))
```

700us External PMU Good time

Definition at line 767 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_800us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_800us
```

Value:

```
BIT(9)
```

800us External PMU Good time

Definition at line 769 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_900us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_900us
```

Value:

```
(BIT(6) | BIT(9))
```

900us External PMU Good time

Definition at line 771 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1000us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1000us
```

Value:

```
(BIT(7) | BIT(9))
```

1000us External PMU Good time

Definition at line 773 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1100us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1100us
```

Value:

```
(BIT(6) | BIT(7) | BIT(9))
```

1100us External PMU Good time

Definition at line 775 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1200us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1200us
```

Value:

```
(BIT(8) | BIT(9))
```

1200us External PMU Good time

Definition at line 777 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1300us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1300us
```

Value:

```
(BIT(6) | BIT(8) | BIT(9))
```

1300us External PMU Good time

Definition at line 779 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1400us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1400us
```

Value:

```
(BIT(7) | BIT(8) | BIT(9))
```

1400us External PMU Good time

Definition at line 781 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1500us

```
#define SL_SI91X_EXTERNAL_PMU_GOOD_TIME_1500us
```

Value:

```
(BIT(6) | BIT(7) | BIT(8) | BIT(9))
```

1500us External PMU Good time

Definition at line 783 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_EXTERNAL_LDO_SEL

```
#define SL_SI91X_FEAT_EXTERNAL_LDO_SEL
```

Value:

```
BIT(10)
```

External LDO voltage selection.

Note

- These bits are used for External LDO selection External PMU : 1.In case of External PMU, User has to set EXTERNAL_PMU_GOOD_TIME_CONFIGURATION value to external PMU good time, If this is zero then it indicates using Internal PMU. 2. Incase of External PMU 1.0v or 1.05v, User has to set both the bits config_feature_bit_map[11] & config_feature_bit_map[10].

Definition at line 787 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_EXTERNAL_LDO_VOL

```
#define SL_SI91X_FEAT_EXTERNAL_LDO_VOL
```

Value:

```
BIT(11)
```

This field valid only if RSI_FEAT_EXTERNAL_LDO_SEL is enabled i.e BIT(10) is set.

If this bit set means 1.0V selected else 1.1V selected

Definition at line 790 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_EAP_TLS_V1P0

```
#define SL_SI91X_FEAT_EAP_TLS_V1P0
```

Value:

```
BIT(14)
```

Note

- Bit 12 -13 are reserved TLS version Enterprise security TLS version 1.0

Definition at line 795 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_EAP_TLS_V1P2

```
#define SL_SI91X_FEAT_EAP_TLS_V1P2
```

Value:

```
BIT(15)
```

Enterprise security TLS version 1.2.

Definition at line 798 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_FEAT_CONC_STA_AP_DYN_SWITCH_SEL

```
#define SL_SI91X_FEAT_CONC_STA_AP_DYN_SWITCH_SEL
```

Value:

```
BIT(17)
```

Reserved bit.

Note

- Bit 16 is reserved

Definition at line `803` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_ULP_GPIO9_FOR_UART2_TX

```
#define SL_SI91X_ULP_GPIO9_FOR_UART2_TX
```

Value:

```
BIT(18)
```

Bit to select ULP_GPIO_9 as UART2 port for device network processor debug prints. If this bit is not set, then by default UART2-TX GPIO_6 will be used.

Bit to disable MCS-5,6,7 data rates

Definition at line `805` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_FEAT_DISABLE_MCS_5_6_7_DATARATES

```
#define SL_SI91X_FEAT_DISABLE_MCS_5_6_7_DATARATES
```

Value:

```
BIT(19)
```

Bit to disable Short-GI.

Definition at line `807` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_FEAT_DISABLE_SHORT_GI

```
#define SL_SI91X_FEAT_DISABLE_SHORT_GI
```

Value:

```
BIT(20)
```

Definition at line `809` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_PTA_3WIRE_EN

```
#define SL_SI91X_PTA_3WIRE_EN
```

Value:

```
BIT(21)
```

To enable PTA-3WIRE.

Note

- Should be set to enable and use the PTA 3 wire feature followed by available configurations

Definition at line 813 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_PTA_3WIRE_CONFIG_SEL

```
#define SL_SI91X_PTA_3WIRE_CONFIG_SEL
```

Value:

```
(config_sel)
```

To choose PTA-3WIRE.

Note

- Configurability options for config selection among 1,2 & 3
- Bit 22 - 23 are used to set NUM_CONN_EVENTS
- 0 kept reserved for future. 3wire used at DUT as ULP_GPIO_0(Grant pin driven by DUT), ULP_GPIO_1(Request i/p pin for dut) and ULP_GPIO_6(Priority i/p pin for dut)

Mode(KB)	BIT[23]	BIT[22]
Reserved	0	0
config1	0	1
config2	1	0
config3	1	1

Definition at line 827 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_XTAL_GOODTIME_1000us

```
#define SL_SI91X_XTAL_GOODTIME_1000us
```

Value:

```
0
```

XTAL goodtime configurations.

Note

- These bits are used to select XTAL good time. These changes are available from Release 2.3.0 onward. Release prior to 2.3.0 these config_feature_bitmap[31:17] are reserved. Its only applicable for customers using chip not the device. Contact Support for more details Default value is 1000 us. 1000us XTAL Good Time

Definition at line 833 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_XTAL_GOODTIME_2000us

```
#define SL_SI91X_XTAL_GOODTIME_2000us
```

Value:

```
BIT(24)
```

2000us XTAL Good Time

Definition at line 835 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_XTAL_GOODTIME_3000us

```
#define SL_SI91X_XTAL_GOODTIME_3000us
```

Value:

```
BIT(25)
```

3000us XTAL Good Time

Definition at line 837 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_XTAL_GOODTIME_600us

```
#define SL_SI91X_XTAL_GOODTIME_600us
```

Value:

```
(BIT(24) | BIT(25))
```

600us XTAL Good Time

Definition at line 839 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_ENABLE_ENHANCED_MAX_PSP

```
#define SL_SI91X_ENABLE_ENHANCED_MAX_PSP
```

Value:

```
BIT(26)
```

Bit to enable Enhanced Max PSP.

Definition at line 842 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_ENABLE_DEBUG_BBP_TEST_PINS

```
#define SL_SI91X_ENABLE_DEBUG_BBP_TEST_PINS
```

Value:

```
BIT(27)
```

Bit to enable BBP Test Pins.

Definition at line 844 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Custom Feature Bitmap

Custom Feature Bitmap

This section provides a reference to the custom feature bitmap in the boot configuration.

Macros

#define SL_SI91X_CUSTOM_FEAT_DISABLE_GATEWAY_IN_RSI_AP BIT(2)
Disables gateway config sent to STA from RSI AP.

#define SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_160MHZ BIT(4)
To configure the clock for NWP SOC 160Mhz If higher performance is needed(like High throughput) then this configuration is needed.

#define SL_SI91X_CUSTOM_FEAT_AP_IN_HIDDEN_MODE BIT(5)
If this bit is set, AP is created in hidden mode This bit is valid only in case of AP mode.

#define SL_SI91X_CUSTOM_FEAT_DNS_SERVER_IN_DHCP_OFFER BIT(6)
DNS server IP address in DHCP offer response in AP mode.

#define SL_SI91X_CUSTOM_FEAT_DFS_CHANNEL_SUPPORT BIT(8)
Support for scanning in DFS channels() in 5GHZ band This bit is valid in WiFi client mode.

#define SL_SI91X_CUSTOM_FEAT_LED_FEATURE BIT(9)
If this bit is set, it enables the LED blinking feature after module initialization.

#define SL_SI91X_CUSTOM_FEAT_ASYNC_CONNECTION_STATUS BIT(10)
If this bit is enabled, module indicates the host the wlan connection status asynchronously This bit is valid in case of Wi-Fi client mode.

#define SL_SI91X_CUSTOM_FEAT_WAKE_ON_WIRELESS BIT(11)
Wake on wireless indication in UART mode.

#define SL_SI91X_CUSTOM_FEAT_ENABLE_AP_BLACKLIST BIT(12)
Enables AP blacklisting in STA mode.

#define SL_SI91X_CUSTOM_FEAT_MAX_NUM_OF_CLIENTS (max_num_of_clients)
Number of clients to support in AP/WFD mode Bit 13 -16 are used to set MAX_NUM_OF_CLIENTS.

#define SL_SI91X_CUSTOM_FEAT_ROAM_WITH_DEAUTH_OR_NULL_DATA BIT(17)
select between de-authentication or null data (with power management bit set) based roaming.

#define SL_SI91X_CUSTOM_FEAT_TRIGGER_AUTO_CONFIG BIT(20)
Trigger Auto Configuration.

#define SL_SI91X_CUSTOM_FEAT_LIMIT_PACKETS_PER_STA BIT(22)
In AP mode, If set only two packets per STA will be buffered when STA is in PS.

#define SL_SI91X_CUSTOM_FEAT_HTTP_HTTPS_AUTH BIT(23)
Bit to set HTTP Authentication.

#define SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_120MHZ BIT(24)
To configure the clock for NWP SOC 120Mhz.

#define SL_SI91X_CUSTOM_FEAT_HTTP_SERVER_CRED_TO_HOST BIT(25)
HTTP server credentials to host in get configuration command.

#define SL_SI91X_CUSTOM_FEAT_REJECT_CONNECT_REQ_IMMEDIATELY BIT(26)
For a LTCP socket when maximum clients are connected if a new connection request is received, then this connection request will be rejected immediately.

#define SL_SI91X_CUSTOM_FEAT_DUAL_BAND_ROAM_VCSAFD BIT(27)
Enables Dual band roaming and vcsafd feature.

#define SL_SI91X_CUSTOM_FEAT_RTC_FROM_HOST BIT(28)
Real time clock from host.

#define SL_SI91X_CUSTOM_FEAT_BT_IAP BIT(29)
Custom feat BT IAP.

#define SL_SI91X_CUSTOM_FEAT_EXTENTION_VALID BIT(31)
Extention valid to use Extended custom feature bitmap.

Macro Definition Documentation

SL_SI91X_CUSTOM_FEAT_DISABLE_GATEWAY_IN_RSI_AP

```
#define SL_SI91X_CUSTOM_FEAT_DISABLE_GATEWAY_IN_RSI_AP
```

Value:

```
BIT(2)
```

Disables gateway config sent to STA from RSI AP.

Note

- Bits 0 -1 are reserved
- If this bit is set to 1, the DHCP server behavior changes when the device is in AP mode. The DHCP server, when it assigns IP addresses to the client nodes, does not send out a Gateway address, and sends only the assigned IP and Subnet values to the client. It is highly recommended to keep this value at '0' as the changed behavior is required in only very specialized use cases and not in normal AP functionality. The default value of this bit is '0'.

Definition at line 211 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_160MHZ

```
#define SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_160MHZ
```

Value:

```
BIT(4)
```

To configure the clock for NWP SOC 160Mhz If higher performance is needed(like High throughput) then this configuration is needed.

Note

- Need to set pll_mode to 1 in feature frame command

Definition at line 217 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_AP_IN_HIDDEN_MODE

```
#define SL_SI91X_CUSTOM_FEAT_AP_IN_HIDDEN_MODE
```

Value:

```
BIT(5)
```

If this bit is set, AP is created in hidden mode This bit is valid only in case of AP mode.

Definition at line `220` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_CUSTOM_FEAT_DNS_SERVER_IN_DHCP_OFFER

```
#define SL_SI91X_CUSTOM_FEAT_DNS_SERVER_IN_DHCP_OFFER
```

Value:

```
BIT(6)
```

DNS server IP address in DHCP offer response in AP mode.

Definition at line `223` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_CUSTOM_FEAT_DFS_CHANNEL_SUPPORT

```
#define SL_SI91X_CUSTOM_FEAT_DFS_CHANNEL_SUPPORT
```

Value:

```
BIT(8)
```

Support for scanning in DFS channels() in 5GHZ band This bit is valid in WiFi client mode.

Note

- Bit 7 is reserved
- it's mandatory to set region before scanning DFS channel.

Definition at line `230` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_CUSTOM_FEAT_LED_FEATURE

```
#define SL_SI91X_CUSTOM_FEAT_LED_FEATURE
```

Value:

```
BIT(9)
```

If this bit is set, it enables the LED blinking feature after module initialization.

LED (GPIO_16) is used for this feature and blinks when a TX packet is sent or when we get a unicast packet addressed to our MAC.

Definition at line `236` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_CUSTOM_FEAT_ASYNC_CONNECTION_STATUS

```
#define SL_SI91X_CUSTOM_FEAT_ASYNC_CONNECTION_STATUS
```

Value:

```
BIT(10)
```

If this bit is enabled, module indicates the host the wlan connection status asynchronously This bit is valid in case of Wi-Fi client mode.

Definition at line 241 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_WAKE_ON_WIRELESS

```
#define SL_SI91X_CUSTOM_FEAT_WAKE_ON_WIRELESS
```

Value:

```
BIT(11)
```

Wake on wireless indication in UART mode.

Definition at line 244 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_ENABLE_AP_BLACKLIST

```
#define SL_SI91X_CUSTOM_FEAT_ENABLE_AP_BLACKLIST
```

Value:

```
BIT(12)
```

Enables AP blacklisting in STA mode.

Note

- By default client maintains AP blacklist internally to avoid some access points. To bypass AP blacklist feature in client mode during roaming or rejoin, this feature should be enabled.

Definition at line 250 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_MAX_NUM_OF_CLIENTS

```
#define SL_SI91X_CUSTOM_FEAT_MAX_NUM_OF_CLIENTS
```

Value:

```
(max_num_of_clients)
```

Number of clients to support in AP/WFD mode Bit 13 -16 are used to set MAX_NUM_OF_CLIENTS.

Definition at line 254 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_ROAM_WITH_DEAUTH_OR_NULL_DATA

```
#define SL_SI91X_CUSTOM_FEAT_ROAM_WITH_DEAUTH_OR_NULL_DATA
```

Value:

```
BIT(17)
```

select between de-authentication or null data (with power management bit set) based roaming.

Definition at line 257 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_TRIGGER_AUTO_CONFIG

```
#define SL_SI91X_CUSTOM_FEAT_TRIGGER_AUTO_CONFIG
```

Value:

```
BIT(20)
```

Trigger Auto Configuration.

Note

- Bit 18 - 19 are reserved

Definition at line 262 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_LIMIT_PACKETS_PER_STA

```
#define SL_SI91X_CUSTOM_FEAT_LIMIT_PACKETS_PER_STA
```

Value:

```
BIT(22)
```

In AP mode, If set only two packets per STA will be buffered when STA is in PS.

Note

- Bit 21 is reserved

Definition at line 267 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_HTTP_HTTPS_AUTH

```
#define SL_SI91X_CUSTOM_FEAT_HTTP_HTTPS_AUTH
```

Value:

```
BIT(23)
```

Bit to set HTTP Authentication.

Definition at line 270 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_120MHZ

```
#define SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_120MHZ
```

Value:

```
BIT(24)
```

To configure the clock for NWP SOC 120Mhz.

If higher performance is needed(like High throughput) then this configuration is needed **Note**

- Need to set pll_mode to 1 in feature frame command

Definition at line `276` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_CUSTOM_FEAT_HTTP_SERVER_CRED_TO_HOST

```
#define SL_SI91X_CUSTOM_FEAT_HTTP_SERVER_CRED_TO_HOST
```

Value:

```
BIT(25)
```

HTTP server credentials to host in get configuration command.

Definition at line `279` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_CUSTOM_FEAT_REJECT_CONNECT_REQ_IMMEDIATELY

```
#define SL_SI91X_CUSTOM_FEAT_REJECT_CONNECT_REQ_IMMEDIATELY
```

Value:

```
BIT(26)
```

For a LTCP socket when maximum clients are connected if a new connection request is received, then this connection request will be rejected immediately.

Note

- By default this bit value is zero. When BIT[26] = 0: For a LTCP socket when maximum clients are connected if a new connection request is received, then this connection request will not be rejected. Instead device will maintain this connection request in LTCP pending list. This request will be served when any of the connected client is disconnected. When BIT[26] = 1: For a LTCP socket when maximum clients are connected if a new connection request is received, then this connection request will be rejected immediately. Device will not maintain this connection request in LTCP pending list.

Definition at line `283` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_CUSTOM_FEAT_DUAL_BAND_ROAM_VCSAFD

```
#define SL_SI91X_CUSTOM_FEAT_DUAL_BAND_ROAM_VCSAFD
```

Value:

```
BIT(27)
```

Enables Dual band roaming and vcsafd feature.

Definition at line 286 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_RTC_FROM_HOST

```
#define SL_SI91X_CUSTOM_FEAT_RTC_FROM_HOST
```

Value:

```
BIT(28)
```

Real time clock from host.

Definition at line 289 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_BT_IAP

```
#define SL_SI91X_CUSTOM_FEAT_BT_IAP
```

Value:

```
BIT(29)
```

Custom feat BT IAP.

Definition at line 292 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_CUSTOM_FEAT_EXTENTION_VALID

```
#define SL_SI91X_CUSTOM_FEAT_EXTENTION_VALID
```

Value:

```
BIT(31)
```

Extention valid to use Extended custom feature bitmap.

Note

- Bit 30 is reserved

Definition at line 297 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Extended Custom Feature Bitmap

Extended Custom Feature Bitmap

This section provides a reference to the extended custom feature bitmap in the boot configuration.

Macros

#define SL_SI91X_EXT_FEAT_RSA_KEY_WITH_4096_SUPPORT BIT(1)
To support 4096 size RSA KEY certificate.

#define SL_SI91X_EXT_FEAT_TELEC_SUPPORT BIT(2)
Extended custom bitmap to support TELEC.

#define SL_SI91X_EXT_FEAT_SSL_CERT_WITH_4096_KEY_SUPPORT BIT(3)
To support 4096 size KEY SSL certificate.

#define SL_SI91X_EXT_FEAT_AP_BROADCAST_PKT_SND_B4_DTIM BIT(4)
Extended custom bitmap for AP Broadcast customization.

#define SL_SI91X_EXT_FEAT_FCC_LOW_PWR BIT(5)
Extended custom bitmap to support FCC.

#define SL_SI91X_EXT_FEAT_PUF BIT(7)
To enable PUF.

#define SL_SI91X_EXT_FEAT_SPECTRAL_MASK_NOKIA BIT(8)
Nokia Spectral mask extended custom bitmap.

#define SL_SI91X_EXT_HTTP_SKIP_DEFAULT_LEADING_CHARACTER BIT(9)
Extended feature bit map to skip default leading character '\' in HTTP header.

#define SL_SI91X_EXT_FEAT_PUF_PRIVATE_KEY BIT(10)
To enable PUF private key.

#define SL_SI91X_EXT_FEAT_ENABLE_11R_OTA BIT(11)
To enable 802.11R Over The Air Roaming.

#define SL_SI91X_EXT_FEAT_IEEE_80211J BIT(12)
To enable 802.11J protocol.

#define SL_SI91X_EXT_FEAT_IEEE_80211W BIT(13)
To enable 802.11W protocol.

#define SL_SI91X_EXT_FEAT_SSL_VERSIONS_SUPPORT BIT(14)
To enable the Multi-version TCP over SSL support.

#define SL_SI91X_EXT_FEAT_16th_STATION_IN_AP_MODE BIT(15)
To Enable 16 client support.

#define SL_SI91X_EXT_FEAT_ENABLE_11R_ODS BIT(16)
To enable 802.11R Over the Distribution System Roaming.

#define SL_SI91X_EXT_FEAT_HTTP_OTAF_SUPPORT BIT(18)
To enable http otaf support.

#define SL_SI91X_EXT_FEAT_LOW_POWER_MODE BIT(19)
To enable low power mode in Wlan.

#define SL_SI91X_EXT_FEAT_256K_MODE BIT(21)
To enable 256K memory for TA.

#define SL_SI91X_RAM_LEVEL_NWP_MEDIUM_MCU_MEDIUM SL_SI91X_EXT_FEAT_256K_MODE

#define SL_SI91X_EXT_FEAT_320K_MODE BIT(20)
To enable 320K memory for TA.

#define SL_SI91X_RAM_LEVEL_NWP_ADV_MCU_BASIC SL_SI91X_EXT_FEAT_320K_MODE

#define SL_SI91X_EXT_FEAT_384K_MODE (BIT(20) | BIT(21))
To enable 384K memory for TA.

#define SL_SI91X_RAM_LEVEL_NWP_ALL_MCU_ZERO SL_SI91X_EXT_FEAT_384K_MODE

#define SL_SI91X_EXT_FEAT_XTAL_CLK_ENABLE (xtal_clk_enable)
To enable crystal clock for TA.

#define SL_SI91X_EXT_FEAT_XTAL_CLK SL_SI91X_EXT_FEAT_XTAL_CLK_ENABLE(2)

#define SL_SI91X_EXT_FEAT_HOMEKIT_WAC_ENABLED BIT(24)
To intimate FW not to modify MDNS text record.

#define SL_SI91X_EXT_FEAT_1P8V_SUPPORT BIT(25)
To enable 1.8v support for TA.

#define SL_SI91X_EXT_FEAT_UART_SEL_FOR_DEBUG_PRINTS BIT(27)
To select UART debug prints pin selection If BIT(27) is enabled,Debug prints are supported on UART1 If BIT(27) is disabled,Debug prints are supported on UART2.

#define SL_SI91X_EXT_FEAT_DISABLE_DEBUG_PRINTS BIT(28)
If this bit is enabled,NWP disables Debug prints support.

#define SL_SI91X_EXT_FEAT_BT_CUSTOM_FEAT_ENABLE BIT(31)
Enable BT Custom Features.

Macro Definition Documentation

SL_SI91X_EXT_FEAT_RSA_KEY_WITH_4096_SUPPORT

```
#define SL_SI91X_EXT_FEAT_RSA_KEY_WITH_4096_SUPPORT
```

Value:

```
BIT(1)
```

To support 4096 size RSA KEY certificate.

Definition at line 308 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_TELEC_SUPPORT

```
#define SL_SI91X_EXT_FEAT_TELEC_SUPPORT
```

Value:

```
BIT(2)
```

Extended custom bitmap to support TELEC.

Definition at line 311 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_SSL_CERT_WITH_4096_KEY_SUPPORT

```
#define SL_SI91X_EXT_FEAT_SSL_CERT_WITH_4096_KEY_SUPPORT
```

Value:

```
BIT(3)
```

To support 4096 size KEY SSL certificate.

Definition at line 314 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_AP_BROADCAST_PKT_SND_B4_DTIM

```
#define SL_SI91X_EXT_FEAT_AP_BROADCAST_PKT_SND_B4_DTIM
```

Value:

```
BIT(4)
```

Extended custom bitmap for AP Broadcast customization.

Note

- If this bit is enable then connected client who is in power save may miss the packet.

Definition at line 318 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_FCC_LOW_PWR

```
#define SL_SI91X_EXT_FEAT_FCC_LOW_PWR
```

Value:

```
BIT(5)
```

Extended custom bitmap to support FCC.

Definition at line 321 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_PUF

```
#define SL_SI91X_EXT_FEAT_PUF
```

Value:

```
BIT(7)
```

To enable PUF.

Definition at line 324 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_SPECTRAL_MASK_NOKIA

```
#define SL_SI91X_EXT_FEAT_SPECTRAL_MASK_NOKIA
```

Value:

```
BIT(8)
```

Nokia Spectral mask extended custom bitmap.

Definition at line 327 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_HTTP_SKIP_DEFAULT_LEADING_CHARACTER

```
#define SL_SI91X_EXT_HTTP_SKIP_DEFAULT_LEADING_CHARACTER
```

Value:

```
BIT(9)
```

Extended feature bit map to skip default leading character '\' in HTTP header.

Definition at line 330 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_PUF_PRIVATE_KEY

```
#define SL_SI91X_EXT_FEAT_PUF_PRIVATE_KEY
```

Value:

```
BIT(10)
```

To enable PUF private key.

Definition at line 333 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_ENABLE_11R_OTA

```
#define SL_SI91X_EXT_FEAT_ENABLE_11R_OTA
```

Value:

```
BIT(11)
```

To enable 802.11R Over The Air Roaming.

Note

- Resource Request Support is not Present

- If both BIT[11] and BIT[16] are not enabled then it will select as Legacy Roaming.

Definition at line 338 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_IEEE_80211J

```
#define SL_SI91X_EXT_FEAT_IEEE_80211J
```

Value:

```
BIT(12)
```

To enable 802.11J protocol.

Note

- If this bit is enable, set region command is mandatory with setting it to Japan region and band value must be 1 (5GHz).

Definition at line 342 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_IEEE_80211W

```
#define SL_SI91X_EXT_FEAT_IEEE_80211W
```

Value:

```
BIT(13)
```

To enable 802.11W protocol.

Note

- This bit must be set for enabling WPA3 Personal Mode and WPA3 Personal Transition mode.

Definition at line 346 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_SSL_VERSIONS_SUPPORT

```
#define SL_SI91X_EXT_FEAT_SSL_VERSIONS_SUPPORT
```

Value:

```
BIT(14)
```

To enable the Multi-version TCP over SSL support.

Definition at line 349 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_16th_STATION_IN_AP_MODE

```
#define SL_SI91X_EXT_FEAT_16th_STATION_IN_AP_MODE
```

Value:

```
BIT(15)
```

To Enable 16 client support.

Note

- If this bit is enable then 16 stations can connect in AP mode otherwise Maximum of 8 stations can connect.

Definition at line `353` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_EXT_FEAT_ENABLE_11R_ODS

```
#define SL_SI91X_EXT_FEAT_ENABLE_11R_ODS
```

Value:

```
BIT(16)
```

To enable 802.11R Over the Distribution System Roaming.

Note

- 1. Resource Request Support is not Present. 2. If both BIT[11] and BIT[16] are not enabled then it will select as Legacy Roaming.

Definition at line `357` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_EXT_FEAT_HTTP_OTAF_SUPPORT

```
#define SL_SI91X_EXT_FEAT_HTTP_OTAF_SUPPORT
```

Value:

```
BIT(18)
```

To enable http otaf support.

Note

- Bit 17 is reserved

Definition at line `362` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_EXT_FEAT_LOW_POWER_MODE

```
#define SL_SI91X_EXT_FEAT_LOW_POWER_MODE
```

Value:

```
BIT(19)
```

To enable low power mode in Wlan.

Note

- EXT_FEAT_LOW_POWER_MODE is not supported for 1.3 version chipset.

Definition at line 366 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_256K_MODE

```
#define SL_SI91X_EXT_FEAT_256K_MODE
```

Value:

```
BIT(21)
```

To enable 256K memory for TA.

Note

- Default memory configuration (RAM) is 192KB. User can set these bits to change the memory configuration as below:

Mode(KB)	BIT[20]	BIT[21]
192	0	0
256	0	1
320	1	0
384	1	1

Definition at line 419 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_RAM_LEVEL_NWP_MEDIUM_MCU_MEDIUM

```
#define SL_SI91X_RAM_LEVEL_NWP_MEDIUM_MCU_MEDIUM
```

Value:

```
SL_SI91X_EXT_FEAT_256K_MODE
```

Definition at line 420 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_320K_MODE

```
#define SL_SI91X_EXT_FEAT_320K_MODE
```

Value:

```
BIT(20)
```

To enable 320K memory for TA.

Definition at line 423 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_RAM_LEVEL_NWP_ADV_MCU_BASIC

```
#define SL_SI91X_RAM_LEVEL_NWP_ADV_MCU_BASIC
```

Value:

```
SL_SI91X_EXT_FEAT_320K_MODE
```

Definition at line 424 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_384K_MODE

```
#define SL_SI91X_EXT_FEAT_384K_MODE
```

Value:

```
(BIT(20) | BIT(21))
```

To enable 384K memory for TA.

Definition at line 427 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_RAM_LEVEL_NWP_ALL_MCU_ZERO

```
#define SL_SI91X_RAM_LEVEL_NWP_ALL_MCU_ZERO
```

Value:

```
SL_SI91X_EXT_FEAT_384K_MODE
```

Definition at line 428 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_XTAL_CLK_ENABLE

```
#define SL_SI91X_EXT_FEAT_XTAL_CLK_ENABLE
```

Value:

```
(xtal_clk_enable)
```

To enable crystal clock for TA.

Based on sleep clock source selection, User can choose one of the following

Selection	BIT[23]	BIT[22]
Use RC clock as sleep clock	0	0
Use 32KHz clock from external XTAL OSCILLATOR	0	1
Use 32KHz bypass clock on UULP_GPIO_3	1	0
Use 32KHz bypass clock on UULP_GPIO_4	1	1

Note

- For 917 V2 radio boards set SL_SI91X_EXT_FEAT_XTAL_CLK_ENABLE to 1, for other variants 2 is recommended

Definition at line 445 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_XTAL_CLK

```
#define SL_SI91X_EXT_FEAT_XTAL_CLK
```

Value:

```
SL_SI91X_EXT_FEAT_XTAL_CLK_ENABLE(2)
```

Definition at line 454 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_HOMEKIT_WAC_ENABLED

```
#define SL_SI91X_EXT_FEAT_HOMEKIT_WAC_ENABLED
```

Value:

```
BIT(24)
```

To intimate FW not to modify MDNS text record.

Definition at line 458 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_1P8V_SUPPORT

```
#define SL_SI91X_EXT_FEAT_1P8V_SUPPORT
```

Value:

```
BIT(25)
```

To enable 1.8v support for TA.

Definition at line 461 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_UART_SEL_FOR_DEBUG_PRINTS

```
#define SL_SI91X_EXT_FEAT_UART_SEL_FOR_DEBUG_PRINTS
```

Value:

```
BIT(27)
```

To select UART debug prints pin selection If BIT(27) is enabled,Debug prints are supported on UART1 If BIT(27) is disabled,Debug prints are supported on UART2.

Note

- Bit 26 is reserved
- By default all the debug prints from device network processor will be coming on UART2 if this bit is not enabled. UART1 pins are mapped to the following pins w.r.t to the device network processor. User needs to ensure that these pins are not used in MCU applications in SoC mode to avoid conflicts of pins usage based on the requirement. This bit is valid only if BIT[28] in ext_custom_feature_bit_map is set to 0. UART1-TX: GPIO_9 UART1-RX: GPIO_8 UART2-TX: GPIO_6 UART2-RX: GPIO_10 There is no functionality on rx pins for debug prints.

Definition at line 469 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_DISABLE_DEBUG_PRINTS

```
#define SL_SI91X_EXT_FEAT_DISABLE_DEBUG_PRINTS
```

Value:

```
BIT(28)
```

If this bit is enabled,NWP disables Debug prints support.

Definition at line 471 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

SL_SI91X_EXT_FEAT_BT_CUSTOM_FEAT_ENABLE

```
#define SL_SI91X_EXT_FEAT_BT_CUSTOM_FEAT_ENABLE
```

Value:

```
BIT(31)
```

Enable BT Custom Features.

Note

- Bit 29 - 30 is reserved

Definition at line 509 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Default Device Configuration

Default Device Configuration

This section provides a reference to Si91x default device configuration.

Variables

const sl_wifi_device_configuration_t	sl_wifi_default_client_configuration Default Wi-Fi client configuration.
const sl_wifi_device_configuration_t	sl_wifi_default_enterprise_client_configuration Default Wi-Fi enterprise client configuration.
const sl_wifi_device_configuration_t	sl_wifi_default_ap_configuration Default Wi-Fi ap configuration.
const sl_wifi_device_configuration_t	sl_wifi_default_concurrent_configuration Default Wi-Fi concurrent (AP + STATION) configuration.
const sl_wifi_device_configuration_t	sl_wifi_transmit_test_configuration Default Wi-Fi transmit configuration.

Variable Documentation

sl_wifi_default_client_configuration

```
const sl_wifi_device_configuration_t sl_wifi_default_client_configuration
```

Default Wi-Fi client configuration.

Definition at line `969` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

sl_wifi_default_enterprise_client_configuration

```
const sl_wifi_device_configuration_t sl_wifi_default_enterprise_client_configuration
```

Default Wi-Fi enterprise client configuration.

Definition at line `1004` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

sl_wifi_default_ap_configuration

```
const sl_wifi_device_configuration_t sl_wifi_default_ap_configuration
```

Default Wi-Fi ap configuration.

Definition at line 1030 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

sl_wifi_default_concurrent_configuration

```
const sl_wifi_device_configuration_t sl_wifi_default_concurrent_configuration
```

Default Wi-Fi concurrent (AP + STATION) configuration.

Definition at line 1050 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

sl_wifi_transmit_test_configuration

```
const sl_wifi_device_configuration_t sl_wifi_transmit_test_configuration
```

Default Wi-Fi transmit configuration.

Definition at line 1074 of file components/device/silabs/si91x/wireless/inc/sl_wifi_device.h

Load Image Types

Load Image Types

This section provides a reference to Si91x load image types.

Macros

#define LOAD_NWP_FW '1'
Load NWP firmware.

#define LOAD_DEFAULT_NWP_FW_ACTIVE_LOW 0x71
Load default NWP firmware active low.

Macro Definition Documentation

LOAD_NWP_FW

```
#define LOAD_NWP_FW
```

Value:

```
'1'
```

Load NWP firmware.

Definition at line 37 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

LOAD_DEFAULT_NWP_FW_ACTIVE_LOW

```
#define LOAD_DEFAULT_NWP_FW_ACTIVE_LOW
```

Value:

```
0x71
```

Load default NWP firmware active low.

Definition at line 40 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

TLS Flags

TLS Flags

This section provides a reference to Si91x flags for use in the Transport Layer Security (TLS) protocol functionality.

Macros

#define SL_SI91X_ENABLE_TLS BIT(0)
Bit to enable SSL feature.

#define SL_SI91X_TLS_V_1_0 BIT(2)
Bitmap to enable TLS version 1.0.

#define SL_SI91X_TLS_V_1_2 BIT(3)
Bitmap to enable TLS version 1.2.

#define SL_SI91X_TLS_V_1_1 BIT(4)
Bitmap to enable TLS version 1.1.

Macro Definition Documentation

SL_SI91X_ENABLE_TLS

```
#define SL_SI91X_ENABLE_TLS
```

Value:

```
BIT(0)
```

Bit to enable SSL feature.

Definition at line 50 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_TLS_V_1_0

```
#define SL_SI91X_TLS_V_1_0
```

Value:

```
BIT(2)
```

Bitmap to enable TLS version 1.0.

Definition at line 53 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_TLS_V_1_2

```
#define SL_SI91X_TLS_V_1_2
```

Value:

```
BIT(3)
```

Bitmap to enable TLS version 1.2.

Definition at line 56 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_TLS_V_1_1

```
#define SL_SI91X_TLS_V_1_1
```

Value:

```
BIT(4)
```

Bitmap to enable TLS version 1.1.

Definition at line 59 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

HTTP Flags

HTTP Flags

This section provides a reference to Si91x flags for use in the Hyper-Text Transfer Protocol (HTTP) functionality.

Macros

#define SL_SI91X_ENABLE_NULL_DELIMETER BIT(1)
Bit to enable NULL delimiter for HTTP buffer instead of comma.

#define SL_SI91X_SUPPORT_HTTP_POST_DATA BIT(5)
HTTP client post big data support feature bitmap.

#define SL_SI91X_HTTP_V_1_1 BIT(6)
HTTP version 1.1 support feature bitmap.

#define SL_SI91X_HTTP_USER_DEFINED_CONTENT_TYPE BIT(7)
Bit to enable user given content type in extended header.

#define SL_SI91X_HTTPS_CERTIFICATE_INDEX_1 BIT(9)
To specify index of SSL cert to be used for HTTPS, for index 0 leave them unset.

#define SL_SI91X_HTTPS_CERTIFICATE_INDEX_2 BIT(10)
To specify index of SSL cert to be used for HTTPS, for index 0 leave them unset.

#define SL_SI91X_HTTPS_USE_SNI BIT(11)
To enable SNI.

Macro Definition Documentation

SL_SI91X_ENABLE_NULL_DELIMETER

```
#define SL_SI91X_ENABLE_NULL_DELIMETER
```

Value:

```
BIT(1)
```

Bit to enable NULL delimiter for HTTP buffer instead of comma.

Definition at line 74 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_SUPPORT_HTTP_POST_DATA

```
#define SL_SI91X_SUPPORT_HTTP_POST_DATA
```

Value:

```
BIT(5)
```

HTTP client post big data support feature bitmap.

Definition at line 77 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_HTTP_V_1_1

```
#define SL_SI91X_HTTP_V_1_1
```

Value:

```
BIT(6)
```

HTTP version 1.1 support feature bitmap.

Definition at line 80 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_HTTP_USER_DEFINED_CONTENT_TYPE

```
#define SL_SI91X_HTTP_USER_DEFINED_CONTENT_TYPE
```

Value:

```
BIT(7)
```

Bit to enable user given content type in extended header.

Definition at line 83 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_HTTPS_CERTIFICATE_INDEX_1

```
#define SL_SI91X_HTTPS_CERTIFICATE_INDEX_1
```

Value:

```
BIT(9)
```

To specify index of SSL cert to be used for HTTPS, for index 0 leave them unset.

Definition at line 86 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_HTTPS_CERTIFICATE_INDEX_2

```
#define SL_SI91X_HTTPS_CERTIFICATE_INDEX_2
```

Value:

```
BIT(10)
```

To specify index of SSL cert to be used for HTTPS, for index 0 leave them unset.

Definition at line 88 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

SL_SI91X_HTTPS_USE_SNI

```
#define SL_SI91X_HTTPS_USE_SNI
```

Value:

```
BIT(11)
```

To enable SNI.

Definition at line 91 of file components/device/silabs/si91x/wireless/inc/sl_si91x_constants.h

Join Feature Bitmap

Join Feature Bitmap

This section provides a reference to the Si91x join feature bitmap.

Macros

#define SL_SI91X_JOIN_FEAT_STA_BG_ONLY_MODE_ENABLE (1 << 0)
To enable b/g only mode in station mode.

#define SL_SI91X_JOIN_FEAT_LISTEN_INTERVAL_VALID (1 << 1)
To take listen interval from join command.

#define SL_SI91X_JOIN_FEAT_QUICK_JOIN (1 << 2)
To enable quick join feature.

#define SL_SI91X_JOIN_FEAT_CCXV2_FEATURE (1 << 3)
To enable CCXV2 feature.

#define SL_SI91X_JOIN_FEAT_BSSID_BASED (1 << 4)
To connect to AP based on BSSID together with configured SSID.

#define SL_SI91X_JOIN_FEAT_MFP_CAPABLE_ONLY (1 << 5)
MFP Capable only.

#define SL_SI91X_JOIN_FEAT_MFP_CAPABLE_REQUIRED ((1 << 5) | (1 << 6))
MFP Capable required.

#define SL_SI91X_JOIN_FEAT_PS_CMD_LISTEN_INTERVAL_VALID (1 << 7)
listen interval from power save command

Macro Definition Documentation

SL_SI91X_JOIN_FEAT_STA_BG_ONLY_MODE_ENABLE

```
#define SL_SI91X_JOIN_FEAT_STA_BG_ONLY_MODE_ENABLE
```

Value:

```
(1 << 0)
```

To enable b/g only mode in station mode.

Definition at line 146 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

SL_SI91X_JOIN_FEAT_LISTEN_INTERVAL_VALID

```
#define SL_SI91X_JOIN_FEAT_LISTEN_INTERVAL_VALID
```

Value:

```
(1 << 1)
```

To take listen interval from join command.

Definition at line 149 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

SL_SI91X_JOIN_FEAT_QUICK_JOIN

```
#define SL_SI91X_JOIN_FEAT_QUICK_JOIN
```

Value:

```
(1 << 2)
```

To enable quick join feature.

Definition at line 152 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

SL_SI91X_JOIN_FEAT_CCXV2_FEATURE

```
#define SL_SI91X_JOIN_FEAT_CCXV2_FEATURE
```

Value:

```
(1 << 3)
```

To enable CCXV2 feature.

Definition at line 155 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

SL_SI91X_JOIN_FEAT_BSSID_BASED

```
#define SL_SI91X_JOIN_FEAT_BSSID_BASED
```

Value:

```
(1 << 4)
```

To connect to AP based on BSSID together with configured SSID.

Definition at line 158 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

SL_SI91X_JOIN_FEAT_MFP_CAPABLE_ONLY

```
#define SL_SI91X_JOIN_FEAT_MFP_CAPABLE_ONLY
```

Value:

```
(1 << 5)
```

MFP Capable only.

Definition at line 161 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

SL_SI91X_JOIN_FEAT_MFP_CAPABLE_REQUIRED

```
#define SL_SI91X_JOIN_FEAT_MFP_CAPABLE_REQUIRED
```

Value:

```
((1 << 5) | (1 << 6))
```

MFP Capable required.

Definition at line 164 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

SL_SI91X_JOIN_FEAT_PS_CMD_LISTEN_INTERVAL_VALID

```
#define SL_SI91X_JOIN_FEAT_PS_CMD_LISTEN_INTERVAL_VALID
```

Value:

```
(1 << 7)
```

listen interval from power save command

Definition at line 167 of file components/device/silabs/si91x/wireless/inc/sl_si91x_protocol_types.h

DTIM Alignment Types

DTIM Alignment Types

This section provides a reference to the Si91x DTIM alignment types.

Macros

#define SL_SI91X_ALIGN_WITH_BEACON 0
Module wakes up at beacon which is just before or equal to listen_interval.

#define SL_SI91X_ALIGN_WITH_DTIM_BEACON 1
Module wakes up at DTIM beacon which is just before or equal to listen_interval.

Macro Definition Documentation

SL_SI91X_ALIGN_WITH_BEACON

```
#define SL_SI91X_ALIGN_WITH_BEACON
```

Value:

```
0
```

Module wakes up at beacon which is just before or equal to listen_interval.

Definition at line `897` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

SL_SI91X_ALIGN_WITH_DTIM_BEACON

```
#define SL_SI91X_ALIGN_WITH_DTIM_BEACON
```

Value:

```
1
```

Module wakes up at DTIM beacon which is just before or equal to listen_interval.

Definition at line `899` of file `components/device/silabs/si91x/wireless/inc/sl_wifi_device.h`

Overview

Overview

Provides the list of External Host Interface APIs.

APIs

APIs

This section provides a reference to the External Host Interface API.

Modules

Functions

Functions

Functions

This section provides a reference to the External Host Interface API Functions.

Functions

void	sl_si91x_host_delay_ms(uint32_t delay_milliseconds) This API used to block MCU for specified time.
sl_si91x_host_timestamp_t	sl_si91x_host_get_timestamp(void) It retrieves a timestamp.
sl_si91x_host_timestamp_t	sl_si91x_host_elapsed_time(uint32_t starting_timestamp) This API calculates the timestamp difference.
void	sl_si91x_host_clear_sleep_indicator(void) Sets sleep Indication GPIO to LOW.

Function Documentation

sl_si91x_host_delay_ms

```
void sl_si91x_host_delay_ms (uint32_t delay_milliseconds)
```

This API used to block MCU for specified time.

Parameters

[in]	delay_milliseconds	time delay in milliseconds

Definition at line `183` of file `components/device/silabs/si91x/wireless/inc/sl_rsi_utility.h`

sl_si91x_host_get_timestamp

```
sl_si91x_host_timestamp_t sl_si91x_host_get_timestamp (void)
```

It retrieves a timestamp.

Parameters

N/A		

Returns

- sl_si91x_host_timestamp_t

Definition at line `190` of file `components/device/silabs/si91x/wireless/inc/sl_rsi_utility.h`

sl_si91x_host_elapsed_time

> sl_si91x_host_timestamp_t sl_si91x_host_elapsed_time (uint32_t starting_timestamp)

This API calculates the timestamp difference.

Parameters

[in]	starting_timestamp	This parameter is used to calculate the elapsed time.

Returns

- sl_si91x_host_timestamp_t

Definition at line `200` of file `components/device/silabs/si91x/wireless/inc/sl_rsi_utility.h`

sl_si91x_host_clear_sleep_indicator

> void sl_si91x_host_clear_sleep_indicator (void)

Sets sleep Indication GPIO to LOW.

Parameters

N/A		

Definition at line `113` of file `components/device/silabs/si91x/wireless/inc/sl_si91x_host_interface.h`

Overview

SiWx91x MCU Host Platform

This section provides the API references for the platform features of the SiWx91x microcontroller unit (MCU) host in System-on-chip (SoC) mode, where the application and connnectivity stack run on the SiWx91x chipset.

Refer to the Getting Started in SoC Mode section to run your first application.

Downloads | Release Notes | Installations

Available Development Environments

- Downloading and Installing Simplicity Studio refer Simplicity Studio
- Companion IDEs : Visual Studio Code VSCode

> *Note-*: Pick the IDE of your choice from the above.

Si91x Platform

Si91x-Platform provides software or functions to control peripherals on and as well as external to Si91x device.

- Peripheral Drivers that could be used to control peripherals present on Si91x device.
- Service layer provides abstraction and ease of use for power management, memory management, timer functionalities etc.
- Hardware Drivers that could be used in controlling the functionality of externally connected components like Button, LED, MEMLCD display etc.

Development Tools

Simplicity Studio as well as Silicon Labs provides the following development tools:

- **Network Analyzer**: SSv5's Network Analyzer enables debugging of complex wireless systems. This tool captures a trace of wireless network activity that can be examined in detail live or at a later time. See the Network Analyzer section of the Simplicity Studio 5 User's Guide for more information.
- **Simplicity Commander**: Simplicity Commander is a single, all-purpose tool to be used in a production environment. It is invoked using a simple Command Line Interface (CLI) that is also scriptable. Simplicity Commander enables customers to complete essential tasks such as configuring and building applications and bootloaders and flashing images to their devices. Simplicity Commander is available through SSv5 or can be downloaded through system-specific installers. The Simplicity Commander User's Guide provides more information.
- **Silicon Labs Configurator (SLC)**: SLC offers command-line access to application configuration and generation functions. Software Project Generation and Configuration with SLC-CLI provides instructions on downloading and using the SLC-CLI tool.

Technical Support and concerns

For any other support or information reach out to Silicon Labs Support team.

Overview

Overview

Peripheral drivers provide a simple, generic and feature oriented APIs, using which the peripherals on Si91x device could be controlled.

Peripheral driver provides polling, interrupt and DMA programming models. It provides Support for multi-instance of a peripheral by allowing concurrent API calls for multiple instances of a given peripheral (Ex: I2C0/I2C1 and or UART1/UART2).

All the interrupt configuartion APIs provides user callback feature.

APIs

APIs

This section provides a reference to the Si91x Peripheral API including the functions, data types, and constants provided for various peripherals on the SiWx917™ chipset.

- ADC functions to use the analog-to-digital converter interface on the device.
- Calendar functions to set or get the current RTC time and date.
- Config Timer functions to create timers to count clocks and events, capture events on the GPIOs in input mode, and output modulated signals.
- Direct Memory Access functions to transfer data from a source peripheral or memory to a destination peripheral or memory over one or more advanced high-performance buses (AHBs).
- Disable UC Config section providing information on disabling certain peripherals that are enabled by default.
- E-Fuse functions to use the e-fuse functionality on the device.
- General-Purpose Input-Output functions to set, clear, and toggle pins, program them as input/output pins, and generate interupts including group interrupts.
- Generic SPI functions to access input/output (I/O) interfaces to a wide variety of SPI-compatible peripherals on the device.
- I2C functions to access the Inter-Integrated Circuit (I2C) controllers on the device.
- I2S functions to access the Inter-Integrated Circuit Sound (I2S) interface on the device.
- PSRAM Driver functions to control PSRAM on the device.
- PWM functions to use the pulse width modulation functionality on device.
- System RTC functions to use SYSRTC functionality.
- SDIO Secondary functions to control SDIO interface on the device.
- Serial Input-Output functions to access the regular GPIO pins and enhanced serial stream processing.
- Synchronous Serial Interface functions to access the SSI controllers on the device.
- Ultra Low-Power Timer functions to use low-power timers to count clocks, microseconds, milliseconds, seconds, and minutes with both ref clock and system (SoC) clock.
- USART functions to access the Universal Synchronous/Asynchronous Receiver/Transmitter (USART) interfaces on the device.
- Watchdog Timer functions to generate interrupts on timeout or a system reset on system failure.

Modules

ADC

Modules

sl_adc_threshold_config_t

sl_adc_fifo_thrld_config_t

sl_adc_clock_config_t

sl_adc_version_t

Enumerations

enum sl_adc_input_type_typedef_t {

SL_ADC_SINGLE_ENDED
SL_ADC_DIFFERENTIAL
SL_ADC_INPUT_TYPE_LAST

}

Enumeration for ADC input type.

enum sl_adc_operation_mode_typedef_t {

SL_ADC_FIFO_MODE = ADC_FIFOMODE_ENABLE
SL_ADC_STATIC_MODE = ADC_STATICMODE_ENABLE
SL_ADC_OPERATION_MODE_LAST

}

Enumeration for ADC operation mode.

enum sl_adc_dma_type_typedef_t {

SL_ADC_INTERNAL_DMA = INTERNAL_DMA_EN
SL_ADC_EXTERNAL_DMA = EXTERNAL_DMA_EN
SL_ADC_DMA_TYPE_LAST

}

Enumeration for ADC DMA type.

enum sl_adc_channel_type_typedef_t {

SL_ADC_SINGLE_CHNL = DYNAMIC_MODE_DI
SL_ADC_MULTI_CHNL = DYNAMIC_MODE_EN
SL_ADC_CHANNEL_TYPE_LAST

}

Enumeration for ADC multiple channel selection.

enum sl_adc_ext_trigger_type_t {

SL_ULP_TIMER_EXT_TRIGGER = ULP_TIMER_EXT_TRIGGER
SL_ULP_GPIO_EXT_TRIGGER = ULP_GPIO_EXT_TRIGGER
SL_M4_CT_EXT_TRIGGER = M4_CT_EXT_TRIGGER
SL_ADC_EXT_TRIGGER_TYPE_LAST

}

Enumeration for ADC external trigger type.

enum sl_adc_ext_trigger_num_t {

SL_ADC_EXT_TRIGGER_1 = DETECTION1
SL_ADC_EXT_TRIGGER_2 = DETECTION2
SL_ADC_EXT_TRIGGER_3 = DETECTION3
SL_ADC_EXT_TRIGGER_4 = DETECTION4
SL_ADC_EXT_TRIGGER_LAST

}

Enumeration for ADC external trigger number.

enum sl_adc_ext_trigger_edge_t {

SL_ADC_EXT_TRIGGER_POS_EDGE = POSITIVE_EDGE
SL_ADC_EXT_TRIGGER_NEG_EDGE = NEGATIVE_EDGE
SL_ADC_EXT_TRIGGER_POS_NEG_EDGE = POS_NEG_EDGE
SL_ADC_EXT_TRIGGER_EDGE_LAST

}

Enumeration for ADC external trigger edge selection.

enum sl_adc_ext_trigger_sel_t {

SL_ADC_EXT_TRIGGER_SEL_1 = EXT_TRIGGER_SEL1
SL_ADC_EXT_TRIGGER_SEL_2 = EXT_TRIGGER_SEL2
SL_ADC_EXT_TRIGGER_SEL_3 = EXT_TRIGGER_SEL3
SL_ADC_EXT_TRIGGER_SEL_4 = EXT_TRIGGER_SEL4
SL_ADC_EXT_TRIGGER_SEL_LAST

}

Enumeration for ADC external trigger selection.

enum sl_adc_channel_id_t {

SL_ADC_CHANNEL_0
SL_ADC_CHANNEL_1
SL_ADC_CHANNEL_2
SL_ADC_CHANNEL_3
SL_ADC_CHANNEL_4
SL_ADC_CHANNEL_5
SL_ADC_CHANNEL_6
SL_ADC_CHANNEL_7
SL_ADC_CHANNEL_8
SL_ADC_CHANNEL_9
SL_ADC_CHANNEL_10
SL_ADC_CHANNEL_11
SL_ADC_CHANNEL_12
SL_ADC_CHANNEL_13
SL_ADC_CHANNEL_14
SL_ADC_CHANNEL_15

}

Enumeration for ADC channel.

Typedefs

typedef sl_adc_channel_config_t adc_ch_config_t

Renamed ADC channel configuration structure.

typedef sl_adc_config_t adc_config_t

Renamed ADC configuration structure.

typedef sl_adc_internal_config_t adc_inter_config_t

Renamed ADC internal configuration structure.

typedef sl_adc_external_config_t adc_extr_config_t

Renamed ADC external trigger configuration structure.

typedef void(* sl_adc_callback_t)(uint8_t channel, uint8_t event)
Typedef for the function pointer of the callback function.

Functions

sl_status_t sl_si91x_adc_configure_clock(sl_adc_clock_config_t *clock_configuration)
Configure the ADC clock.

sl_status_t sl_si91x_adc_init(sl_adc_channel_config_t adc_channel_config, sl_adc_config_t adc_config, float vref_value)
Initialize the ADC peripheral.

sl_status_t sl_si91x_adc_set_channel_configuration(sl_adc_channel_config_t adc_channel_config, sl_adc_config_t adc_config)
Configure ADC channel parameters.

sl_status_t sl_si91x_adc_register_event_callback(sl_adc_callback_t callback_event)
Register the user callback function.

void sl_si91x_adc_unregister_event_callback(void)
Un-register the user callback function.

sl_status_t sl_si91x_adc_configure_external_trigger(sl_adc_external_config_t adc_external_trigger)
Configure the ADC external trigger.

sl_status_t sl_si91x_adc_configure_channel_sampling_rate(sl_adc_internal_config_t adc_internal_config, uint8_t channel_num)
Configure the ADC sampling rate for ADC channels.

sl_status_t sl_si91x_adc_get_external_trigger_status(sl_adc_external_config_t adc_external_trigger, uint8_t *ext_trigger)
Read the ADC external trigger status.

sl_status_t sl_si91x_adc_clear_external_trigger(sl_adc_external_config_t adc_external_trigger)
Clear the ADC external trigger status.

sl_status_t sl_si91x_adc_configure_ping_pong_memory_address(sl_adc_internal_config_t adc_internal_config, uint8_t channel_num)
Configure the ADC ping and pong memory location and length.

sl_status_t sl_si91x_adc_enable_ping_pong(uint8_t channel_num)
Enable ping pong for corresponding ADC channels.

sl_status_t sl_si91x_adc_disable_ping_pong(uint8_t channel_num)
Disable ping pong for corresponding ADC channels.

sl_status_t sl_si91x_adc_internal_per_channel_dma_enable(uint8_t channel_num)
Enable internal DMA for corresponding ADC channels.

sl_status_t sl_si91x_adc_internal_per_channel_dma_disable(uint8_t channel_num)
Disable internal dma channel for corresponding ADC channels.

sl_status_t sl_si91x_adc_configure_static_mode(sl_adc_channel_config_t adc_channel_config, uint8_t channel_num)
Configure the ADC in Static Mode.

sl_status_t sl_si91x_adc_configure_fifo_mode(sl_adc_channel_config_t adc_channel_config, uint8_t channel_num)
Configure the ADC in FIFO Mode.

sl_status_t sl_si91x_adc_channel_enable(uint8_t channel_num)
Enable the ADC channel.

sl_status_t sl_si91x_adc_channel_disable(uint8_t channel_num)
Disable the ADC channel.

sl_status_t	sl_si91x_adc_set_power_mode(POWER_STATE state) Set to Power On and off for ADC.
sl_status_t	sl_si91x_adc_set_noise_average_mode(boolean_t state) Enable or Disable Noise averaging mode.
sl_status_t	sl_si91x_adc_temperature_sensor_enable(void) Enable temp-sensor for ADC.
sl_status_t	sl_si91x_adc_fifo_threshold_configuration(sl_adc_config_t adc_config, sl_adc_fifo_thrld_config_t adc_fifo_threshold) Configuring ADC fifo threshold.
sl_status_t	sl_si91x_adc_threshold_configuration(sl_adc_threshold_config_t adc_threshold) Configure the ADC threshold to compare threshold value with ADC data.
sl_status_t	sl_si91x_adc_read_data(sl_adc_channel_config_t adcchconfig, uint8_t channel_num) Read the ADC samples when ulp memories are used.
sl_status_t	sl_si91x_adc_read_data_static(sl_adc_channel_config_t adc_channel_config, sl_adc_config_t adc_config, uint16_t *adc_value) Read the ADC samples when static mode is enabled.
uint32_t	sl_si91x_adc_get_sampling_rate(uint8_t channel_num) Read the ADC sampling rate when static mode is enabled.
sl_status_t	sl_si91x_adc_deinit(sl_adc_config_t adc_config) De-initialize the ADC.
sl_status_t	sl_si91x_adc_start(sl_adc_config_t adc_config) Start the ADC operation.
sl_status_t	sl_si91x_adc_stop(sl_adc_config_t adc_config) Stop the ADC operation.
sl_adc_version_t	sl_si91x_adc_get_version(void) Get the release, sqa and dev version of ADC.

Macros

#define	SL_INTERNAL_DMA INTERNAL_DMA
#define	SL_ADC_STATIC_MODE_EVENT ADC_STATIC_MODE_CALLBACK
#define	SIGN_BIT BIT(11)

Enumeration Documentation

sl_adc_input_type_typedef_t

> sl_adc_input_type_typedef_t

Enumeration for ADC input type.

	Enumerator
SL_ADC_SINGLE_ENDED	Input type single ended.
SL_ADC_DIFFERENTIAL	Input type differential.
SL_ADC_INPUT_TYPE_LAST	Last member of enum for validation.

Definition at line 67 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_operation_mode_typedef_t

> sl_adc_operation_mode_typedef_t

Enumeration for ADC operation mode.

Enumerator	
SL_ADC_FIFO_MODE	operation mode as fifo mode
SL_ADC_STATIC_MODE	operation mode as static mode
SL_ADC_OPERATION_MODE_LAST	Last member of enum for validation.

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_dma_type_typedef_t

> sl_adc_dma_type_typedef_t

Enumeration for ADC DMA type.

Enumerator	
SL_ADC_INTERNAL_DMA	Internal DMA type.
SL_ADC_EXTERNAL_DMA	External DMA type.
SL_ADC_DMA_TYPE_LAST	Last member of enum for validation.

Definition at line 81 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_channel_type_typedef_t

> sl_adc_channel_type_typedef_t

Enumeration for ADC multiple channel selection.

Enumerator	
SL_ADC_SINGLE_CHNL	Dynamic mode disable.
SL_ADC_MULTI_CHNL	Dynamic mode enable.
SL_ADC_CHANNEL_TYPE_LAST	Last member of enum for validation.

Definition at line 88 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_ext_trigger_type_t

> sl_adc_ext_trigger_type_t

Enumeration for ADC external trigger type.

Enumerator	
SL_ULP_TIMER_EXT_TRIGGER	ULP timer external trigger type.
SL_ULP_GPIO_EXT_TRIGGER	ULP gpio external trigger type.
SL_M4_CT_EXT_TRIGGER	M4 CT external trigger type.
SL_ADC_EXT_TRIGGER_TYPE_LAST	Last member of enum for validation.

Definition at line 95 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_ext_trigger_num_t

> sl_adc_ext_trigger_num_t

Enumeration for ADC external trigger number.

Enumerator	
SL_ADC_EXT_TRIGGER_1	External trigger detection 1.
SL_ADC_EXT_TRIGGER_2	External trigger detection 2.
SL_ADC_EXT_TRIGGER_3	External trigger detection 3.
SL_ADC_EXT_TRIGGER_4	External trigger detection 4.
SL_ADC_EXT_TRIGGER_LAST	Last member of enum for validation.

Definition at line 103 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_ext_trigger_edge_t

> sl_adc_ext_trigger_edge_t

Enumeration for ADC external trigger edge selection.

Enumerator	
SL_ADC_EXT_TRIGGER_POS_EDGE	External trigger positive edge.
SL_ADC_EXT_TRIGGER_NEG_EDGE	External trigger negative edge.
SL_ADC_EXT_TRIGGER_POS_NEG_EDGE	External trigger positive and negative edge.
SL_ADC_EXT_TRIGGER_EDGE_LAST	Last member of enum for validation.

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_ext_trigger_sel_t

> sl_adc_ext_trigger_sel_t

Enumeration for ADC external trigger selection.

Enumerator	
SL_ADC_EXT_TRIGGER_SEL_1	External trigger selection 1.
SL_ADC_EXT_TRIGGER_SEL_2	External trigger selection 2.
SL_ADC_EXT_TRIGGER_SEL_3	External trigger selection 3.
SL_ADC_EXT_TRIGGER_SEL_4	External trigger selection 4.
SL_ADC_EXT_TRIGGER_SEL_LAST	Last member of enum for validation.

Definition at line 120 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_channel_id_t

> sl_adc_channel_id_t

Enumeration for ADC channel.

Enumerator	
SL_ADC_CHANNEL_0	ADC channel 1.
SL_ADC_CHANNEL_1	ADC channel 2.
SL_ADC_CHANNEL_2	ADC channel 3.
SL_ADC_CHANNEL_3	ADC channel 4.
SL_ADC_CHANNEL_4	ADC channel 5.
SL_ADC_CHANNEL_5	ADC channel 6.
SL_ADC_CHANNEL_6	ADC channel 7.
SL_ADC_CHANNEL_7	ADC channel 8.
SL_ADC_CHANNEL_8	ADC channel 9.
SL_ADC_CHANNEL_9	ADC channel 10.
SL_ADC_CHANNEL_10	ADC channel 11.
SL_ADC_CHANNEL_11	ADC channel 12.
SL_ADC_CHANNEL_12	ADC channel 13.
SL_ADC_CHANNEL_13	ADC channel 14.
SL_ADC_CHANNEL_14	ADC channel 15.
SL_ADC_CHANNEL_15	ADC channel 16.

Definition at line `129` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

Typedef Documentation

sl_adc_channel_config_t

```
typedef adc_ch_config_t sl_adc_channel_config_t
```

Renamed ADC channel configuration structure.

Definition at line `50` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_adc_config_t

```
typedef adc_config_t sl_adc_config_t
```

Renamed ADC configuration structure.

Definition at line `51` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_adc_internal_config_t

```
typedef adc_inter_config_t sl_adc_internal_config_t
```

Renamed ADC internal configuration structure.

Definition at line `52` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_adc_external_config_t

```
typedef adc_extr_config_t sl_adc_external_config_t
```

Renamed ADC external trigger configuration structure.

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_callback_t

```
typedef void(* sl_adc_callback_t) (uint8_t channel, uint8_t event) )(uint8_t channel, uint8_t event)
```

Typedef for the function pointer of the callback function.

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

Function Documentation

sl_si91x_adc_configure_clock

```
sl_status_t sl_si91x_adc_configure_clock (sl_adc_clock_config_t *clock_configuration)
```

Configure the ADC clock.

Parameters

[in]	clock_configuration	: clock structure variables

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_FAIL (0x0001) - The function is failed
  - SL_STATUS_NOT_INITIALIZED (0x0011) - Clock is not initialized

Definition at line 191 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_init

```
sl_status_t sl_si91x_adc_init (sl_adc_channel_config_t adc_channel_config, sl_adc_config_t adc_config, float vref_value)
```

Initialize the ADC peripheral.

Parameters

[in]	adc_channel_config	: ADC channels configuration structure variable.
[in]	adc_config	: ADC operation configuration structure variable.
[in]	vref_value	: Reference voltage.

- Pre-conditions:
  - sl_si91x_adc_configure_clock - Only for FIFO mode on M4 state

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_BUSY (0x0004) - The function is already active
  - SL_STATUS_INVALID_COUNT (0x002B) - Mismatch count
  - SL_STATUS_INVALID_RANGE (0x0028) - Mismatch Range

Definition at line `210` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_set_channel_configuration

> sl_status_t sl_si91x_adc_set_channel_configuration (sl_adc_channel_config_t adc_channel_config, sl_adc_config_t adc_config)

Configure ADC channel parameters.

Parameters

[in]	adc_channel_config	: ADC channels configuration structure variable.
[in]	adc_config	: ADC operation configuration structure variable.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_INVALID_RANGE (0x0028) - Mismatch Range

Definition at line `228` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_register_event_callback

> sl_status_t sl_si91x_adc_register_event_callback (sl_adc_callback_t callback_event)

Register the user callback function.

Parameters

[in]	callback_event	Pointer to the function which needs to be called at the time of interrupt

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration

Returns

- status 0 if successful, else error code as follow

SL_STATUS_OK (0x0000) - Success
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
- SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 247 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_unregister_event_callback

> void sl_si91x_adc_unregister_event_callback (void)

Un-register the user callback function.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_adc_register_event_callback

Returns

- none

Definition at line 258 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_configure_external_trigger

> sl_status_t sl_si91x_adc_configure_external_trigger (sl_adc_external_config_t adc_external_trigger)

Configure the ADC external trigger.

Parameters

[in]	adc_external_trigger	: ADC external trigger configuration structure variable.

This API is used to mux select to choose between ulp_timer, ulp_gpio, M4_timer based on this detection edge and channel trigger will interrupt.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 280 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_configure_channel_sampling_rate

sl_status_t sl_si91x_adc_configure_channel_sampling_rate (sl_adc_internal_config_t adc_internal_config, uint8_t channel_num)

Configure the ADC sampling rate for ADC channels.

Parameters

[in]	adc_internal_config	: Channel offset and frequency for each channel to set sampling rate.
[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init

Returns

- sl_status_t : Returns 'SL_STATUS_OK' on successful execution.

Definition at line `294` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_get_external_trigger_status

sl_status_t sl_si91x_adc_get_external_trigger_status (sl_adc_external_config_t adc_external_trigger, uint8_t *ext_trigger)

Read the ADC external trigger status.

Parameters

[in]	adc_external_trigger	: ADC external trigger configuration structure variable.
[in]	ext_trigger	: The status of external trigger will be store in this.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line `316` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_clear_external_trigger

sl_status_t sl_si91x_adc_clear_external_trigger (sl_adc_external_config_t adc_external_trigger)

Clear the ADC external trigger status.

Parameters

[in]	adc_external_trigger	: ADC external trigger configuration structure variable.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 337 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_configure_ping_pong_memory_address

> sl_status_t sl_si91x_adc_configure_ping_pong_memory_address (sl_adc_internal_config_t adc_internal_config, uint8_t channel_num)

Configure the ADC ping and pong memory location and length.

Parameters

[in]	adc_internal_config	: ADC internal trigger configuration structure variable.
[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 353 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_enable_ping_pong

> sl_status_t sl_si91x_adc_enable_ping_pong (uint8_t channel_num)

Enable ping pong for corresponding ADC channels.

Parameters

[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:

sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_configure_ping_pong_memory_address

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 371 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_disable_ping_pong

```
sl_status_t sl_si91x_adc_disable_ping_pong (uint8_t channel_num)
```

Disable ping pong for corresponding ADC channels.

Parameters

[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_configure_ping_pong_memory_address
- Pre-conditions:
  - sl_si91x_adc_enable_ping_pong
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 392 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_internal_per_channel_dma_enable

```
sl_status_t sl_si91x_adc_internal_per_channel_dma_enable (uint8_t channel_num)
```

Enable internal DMA for corresponding ADC channels.

Parameters

[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_channel_enable

Returns

status 0 if successful, else error code as follow
- SL_STATUS_OK (0x0000) - Success
- SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 409 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_internal_per_channel_dma_disable

> sl_status_t sl_si91x_adc_internal_per_channel_dma_disable (uint8_t channel_num)

Disable internal dma channel for corresponding ADC channels.

Parameters

[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_channel_enable
- Pre-conditions:
  - sl_si91x_adc_internal_per_channel_dma_enable
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 430 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_configure_static_mode

> sl_status_t sl_si91x_adc_configure_static_mode (sl_adc_channel_config_t adc_channel_config, uint8_t channel_num)

Configure the ADC in Static Mode.

Parameters

[in]	adc_channel_config	: ADC channels configuration structure variable.
N/A	channel_num	

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_configure_ping_pong_memory_address
- Pre-conditions:
  - sl_si91x_adc_enable_ping_pong
- Pre-conditions:
  - sl_si91x_adc_channel_enable
- Pre-conditions:
  - sl_si91x_adc_internal_per_channel_dma_enable

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_INVALID_RANGE (0x0028) - Mismatch Range

Definition at line 454 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_configure_fifo_mode

> sl_status_t sl_si91x_adc_configure_fifo_mode (sl_adc_channel_config_t adc_channel_config, uint8_t channel_num)

Configure the ADC in FIFO Mode.

Parameters

[in]	adc_channel_config	: ADC channels configuration structure variable.
[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_configure_ping_pong_memory_address
- Pre-conditions:
  - sl_si91x_adc_enable_ping_pong
- Pre-conditions:
  - sl_si91x_adc_channel_enable
- Pre-conditions:
  - sl_si91x_adc_internal_per_channel_dma_enable

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_INVALID_RANGE (0x0028) - Mismatch Range

Definition at line 479 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_channel_enable

> sl_status_t sl_si91x_adc_channel_enable (uint8_t channel_num)

Enable the ADC channel.

Parameters

[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 494 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_channel_disable

```
sl_status_t sl_si91x_adc_channel_disable (uint8_t channel_num)
```

Disable the ADC channel.

Parameters

[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_channel_enable
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 513 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_set_power_mode

```
sl_status_t sl_si91x_adc_set_power_mode (POWER_STATE state)
```

Set to Power On and off for ADC.

Parameters

[in]	state	: **ADC_POWER_ON** - To powerup adc powergates, **ADC_POWER_OFF** - To powerdown adc powergates

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init

Returns

- status 0 if successful,
  - SL_STATUS_OK (0x0000) - Success

Definition at line 528 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_set_noise_average_mode

> sl_status_t sl_si91x_adc_set_noise_average_mode (boolean_t state)

Enable or Disable Noise averaging mode.

Parameters

[in]	state	: 1 - To enable noise averaging mode, 0 - To disable noise averaging mode

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_power_mode

Returns

- Returns 'SL_STATUS_OK' on successful execution.

Definition at line `544` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_temperature_sensor_enable

> sl_status_t sl_si91x_adc_temperature_sensor_enable (void)

Enable temp-sensor for ADC.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init

Returns

- Returns 'SL_STATUS_OK' on successful execution.

Definition at line `557` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_fifo_threshold_configuration

> sl_status_t sl_si91x_adc_fifo_threshold_configuration (sl_adc_config_t adc_config, sl_adc_fifo_thrld_config_t adc_fifo_threshold)

Configuring ADC fifo threshold.

Parameters

[in]	adc_config	: ADC operation configuration structure variable.
[in]	adc_fifo_threshold	: ADC fifo structure variable like aempty fifo, afull fifo threshold level.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init

Pre-conditions:
- sl_si91x_adc_set_channel_configuration

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_INVALID_RANGE (0x0028) - Mismatch Range

Definition at line 576 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_threshold_configuration

> sl_status_t sl_si91x_adc_threshold_configuration (sl_adc_threshold_config_t adc_threshold)

Configure the ADC threshold to compare threshold value with ADC data.

Parameters

[in]	adc_threshold	: ADC threshold configuration structure variables.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- Returns 'SL_STATUS_OK' on successful execution.

Definition at line 594 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_read_data

> sl_status_t sl_si91x_adc_read_data (sl_adc_channel_config_t adcchconfig, uint8_t channel_num)

Read the ADC samples when ulp memories are used.

Parameters

[in]	adcchconfig	: ADC channels configuration structure variable.
[in]	channel_num	channel_num : Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration
- Pre-conditions:
  - sl_si91x_adc_start

Returns

-

status 0 if successful, else error code as follow
- SL_STATUS_OK (0x0000) - Success
- SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
- SL_STATUS_INVALID_RANGE (0x0028) - Mismatch Range

Definition at line 615 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_read_data_static

```
sl_status_t sl_si91x_adc_read_data_static (sl_adc_channel_config_t adc_channel_config, sl_adc_config_t adc_config, uint16_t *adc_value)
```

Read the ADC samples when static mode is enabled.

Parameters

[in]	adc_channel_config	: ADC channels configuration structure variable.
[in]	adc_config	: Store the reading data on adc_value.
N/A	adc_value	

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_INVALID_RANGE (0x0028) - Mismatch Range

Definition at line 636 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_get_sampling_rate

```
uint32_t sl_si91x_adc_get_sampling_rate (uint8_t channel_num)
```

Read the ADC sampling rate when static mode is enabled.

Parameters

[in]	channel_num	: Channel number

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration
- Pre-conditions:
  - sl_si91x_adc_start

Returns

Returns sampling rate what it get from register

Definition at line `655` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_deinit

```
sl_status_t sl_si91x_adc_deinit (sl_adc_config_t adc_config)
```

De-initialize the ADC.

Parameters

[in]	adc_config	: ADC operation configuration structure variable.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line `670` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_start

```
sl_status_t sl_si91x_adc_start (sl_adc_config_t adc_config)
```

Start the ADC operation.

Parameters

[in]	adc_config	: ADC operation configuration structure variable.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line `687` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h`

sl_si91x_adc_stop

```
sl_status_t sl_si91x_adc_stop (sl_adc_config_t adc_config)
```

Stop the ADC operation.

Parameters

[in]	adc_config	: ADC operation configuration structure variable.

- Pre-conditions:
  - sl_si91x_adc_configure_clock
- Pre-conditions:
  - sl_si91x_adc_init
- Pre-conditions:
  - sl_si91x_adc_set_channel_configuration
- Pre-conditions:
  - sl_si91x_adc_start

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid

Definition at line 706 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_si91x_adc_get_version

```
sl_adc_version_t sl_si91x_adc_get_version (void)
```

Get the release, sqa and dev version of ADC.

Parameters

[in]		

Returns

- (sl_adc_version_t) type structure

Definition at line 714 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

Macro Definition Documentation

SL_INTERNAL_DMA

```
#define SL_INTERNAL_DMA
```

Value:

```
INTERNAL_DMA
```

Definition at line 62 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

SL_ADC_STATIC_MODE_EVENT

```
#define SL_ADC_STATIC_MODE_EVENT
```

Value:

```
ADC_STATIC_MODE_CALLBACK
```

Definition at line 63 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

SIGN_BIT

```
#define SIGN_BIT
```

Value:

```
BIT(11)
```

Definition at line 64 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_threshold_config_t

Public Attributes

uint16_t	threshold1 Threshold_1.
uint16_t	threshold2 Threshold_2.
uint8_t *	threshold1_cond Threshold_1 condition like Equal, Grater, Lesser.
uint8_t *	threshold2_cond Threshold_2 condition like Equal, Grater, Lesser.
uint8_t	range

Public Attribute Documentation

threshold1

```
uint16_t sl_adc_threshold_config_t::threshold1
```

Threshold_1.

Definition at line 149 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

threshold2

```
uint16_t sl_adc_threshold_config_t::threshold2
```

Threshold_2.

Definition at line 150 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

threshold1_cond

```
uint8_t* sl_adc_threshold_config_t::threshold1_cond
```

Threshold_1 condition like Equal, Grater, Lesser.

Definition at line 151 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

threshold2_cond

uint8_t* sl_adc_threshold_config_t::threshold2_cond

Threshold_2 condition like Equal, Grater, Lesser.

Definition at line 152 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

range

uint8_t sl_adc_threshold_config_t::range

Definition at line 153 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_fifo_thrld_config_t

Public Attributes

uint8_t	num_of_channel_en Number of channel enable.
uint8_t	a_empty_threshold AEMPTY Threshold.
uint8_t	a_full_threshold AFULL Threshold.
uint8_t	dma_type Internal or External DMA.

Public Attribute Documentation

num_of_channel_en

```
uint8_t sl_adc_fifo_thrld_config_t::num_of_channel_en
```

Number of channel enable.

Definition at line 157 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

a_empty_threshold

```
uint8_t sl_adc_fifo_thrld_config_t::a_empty_threshold
```

AEMPTY Threshold.

Definition at line 158 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

a_full_threshold

```
uint8_t sl_adc_fifo_thrld_config_t::a_full_threshold
```

AFULL Threshold.

Definition at line 159 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

dma_type

```
uint8_t sl_adc_fifo_thrld_config_t::dma_type
```

Internal or External DMA.

Definition at line 160 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_clock_config_t

Structure to hold the clock configuration parameters.

Public Attributes

uint16_t	division_factor Division Factor.
uint32_t	soc_pll_clock SoC PLL clock frequency.
uint32_t	soc_pll_reference_clock SoC PLL reference clock frequency.

Public Attribute Documentation

division_factor

```
uint16_t sl_adc_clock_config_t::division_factor
```

Division Factor.

Definition at line 165 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

soc_pll_clock

```
uint32_t sl_adc_clock_config_t::soc_pll_clock
```

SoC PLL clock frequency.

Definition at line 166 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

soc_pll_reference_clock

```
uint32_t sl_adc_clock_config_t::soc_pll_reference_clock
```

SoC PLL reference clock frequency.

Definition at line 167 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

sl_adc_version_t

Structure to hold the versions number of peripheral API.

Public Attributes

uint8_t	release	Release version number.
uint8_t	major	sqa version number
uint8_t	minor	dev version number

Public Attribute Documentation

release

```
uint8_t sl_adc_version_t::release
```

Release version number.

Definition at line 172 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

major

```
uint8_t sl_adc_version_t::major
```

sqa version number

Definition at line 173 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

minor

```
uint8_t sl_adc_version_t::minor
```

dev version number

Definition at line 174 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_adc.h

Calendar

Modules

clock_calibration_config_t

sl_calendar_version_t

Typedefs

typedef AON_CLK_T — sl_calendar_clock_t
Renaming clock type enum.

typedef RTC_TIME_CONFIG_T — sl_calendar_datetime_config_t
Renaming datetime structure.

typedef RTC_MONTH_T — sl_calendar_month_t
Renaming month structure.

typedef RTC_DAY_OF_WEEK_T — sl_calendar_days_of_week_t
Renaming days of week structure.

typedef void(* — calendar_callback_t)(void)
Typedef for the function pointer of the callback function.

Functions

TIME_CONVERSION_ENUM(time_conversion_enum)
Enumeration to represent time conversion format.

RC_CLOCK_CALIBRATION_ENUM(rc_clock_calibration_enum)
Enumeration to represent different rc clock calibration configurations.

RO_CLOCK_CALIBRATION_ENUM(ro_clock_calibration_enum)
Enumeration to represent different ro clock calibration configurations.

sl_status_t — sl_si91x_calendar_set_configuration(sl_calendar_clock_t clock_type)
Configuration and initialization of Calendar i.e., RTC clock.

sl_status_t — sl_si91x_calendar_set_date_time(sl_calendar_datetime_config_t *config)
Set the date and time of the Calendar RTC.

sl_status_t — sl_si91x_calendar_get_date_time(sl_calendar_datetime_config_t *config)
Get the date and time of an existing Calendar RTC.

sl_status_t — sl_si91x_calendar_rcclk_calibration(clock_calibration_config_t *clock_calibration_config)
Calibrate the RC Clock.

sl_status_t — sl_si91x_calendar_roclk_calibration(clock_calibration_config_t *clock_calibration_config)
Calibrate the RO Clock.

sl_status_t	sl_si91x_calendar_register_msec_trigger_callback(calendar_callback_t callback) Register the callback of the msec trigger and enable it.
sl_status_t	sl_si91x_calendar_register_sec_trigger_callback(calendar_callback_t callback) Register the callback of the sec trigger and enable it.
sl_status_t	sl_si91x_calendar_register_alarm_trigger_callback(calendar_callback_t callback) Register the callback of the alarm trigger and enable it.
sl_status_t	sl_si91x_calendar_unregister_msec_trigger_callback(void) Unregister the callback of the msec trigger and disable it.
sl_status_t	sl_si91x_calendar_unregister_sec_trigger_callback(void) Unregister the callback of the sec trigger and disable it.
sl_status_t	sl_si91x_calendar_unregister_alarm_trigger_callback(void) Unregister the callback of the alarm trigger and disable it.
sl_status_t	sl_si91x_calendar_set_alarm(sl_calendar_datetime_config_t *alarm) Set the date and time of new alarm in RTC.
sl_status_t	sl_si91x_calendar_get_alarm(sl_calendar_datetime_config_t *alarm) Get the date and time of an existing alarm in RTC.
sl_status_t	sl_si91x_calendar_build_datetime_struct(sl_calendar_datetime_config_t *date, uint8_t Century, uint8_t Year, sl_calendar_month_t Month, sl_calendar_days_of_week_t DayOfWeek, uint8_t Day, uint8_t Hour, uint8_t Minute, uint8_t Second, uint16_t Milliseconds) Build the structure for date-time configuration.
sl_status_t	sl_si91x_calendar_convert_unix_time_to_ntp_time(uint32_t time, uint32_t *ntp_time) Convert Unix timestamp to NTP timestamp.
sl_status_t	sl_si91x_calendar_convert_ntp_time_to_unix_time(uint32_t ntp_time, uint32_t *time) Convert NTP timestamp to Unix timestamp.
boolean_t	sl_si91x_calendar_is_msec_trigger_enabled(void) Return the state of msec trigger of RTC (enabled or disabled).
boolean_t	sl_si91x_calendar_is_sec_trigger_enabled(void) Return the state of sec trigger of RTC (enabled or disabled).
boolean_t	sl_si91x_calendar_is_alarm_trigger_enabled(void) Return the state of alarm trigger of RTC (enabled or disabled).
__STATIC_INLINE void	sl_si91x_calendar_rtc_start(void) Start the Calendar RTC.
__STATIC_INLINE void	sl_si91x_calendar_rtc_stop(void) Stop the Calendar RTC.
__STATIC_INLINE void	sl_si91x_calendar_calibration_init(void) Initialize the calibration for Calendar clocks.
__STATIC_INLINE void	sl_si91x_calendar_clear_msec_trigger(void) Clear the msec trigger.
__STATIC_INLINE void	sl_si91x_calendar_clear_sec_trigger(void) Clear the sec trigger.
__STATIC_INLINE void	sl_si91x_calendar_clear_alarm_trigger(void) Clear the alarm trigger.

__STATIC_INLINE void	sl_si91x_calendar_init(void) De-initialize calendar operation.
__STATIC_INLINE void	sl_si91x_calendar_deinit(void) De-initialize calendar operation.
sl_calendar_version_t	sl_si91x_calendar_get_version(void) Get the calendar version.

Macros

#define	SLI_ALARM_IRQHandler IRQ028_Handler Alarm IRQ Handler.
#define	SLI_MSEC_SEC_IRQHandler IRQ029_Handler RTC IRQ Handler.
#define	SLI_NVIC_ALARM MCU_CAL_ALARM_IRQn Alarm NVIQ enable.
#define	SLI_NVIC_MSEC_SEC MCU_CAL_RTC_IRQn RTC NVIQ enable.
#define	TIME_CONVERSION_ENUM (name) Time Conversion format enum declaration.
#define	RC_CLOCK_CALIBRATION_ENUM (name) Time Conversion format enum declaration.
#define	RO_CLOCK_CALIBRATION_ENUM (name) Time Conversion format enum declaration.

Typedef Documentation

sl_calendar_clock_t

```
typedef AON_CLK_T sl_calendar_clock_t
```

Renaming clock type enum.

Definition at line `71` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_calendar_datetime_config_t

```
typedef RTC_TIME_CONFIG_T sl_calendar_datetime_config_t
```

Renaming datetime structure.

Definition at line `72` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_calendar_month_t

```
typedef RTC_MONTH_T sl_calendar_month_t
```

Renaming month structure.

Definition at line 73 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_calendar_days_of_week_t

```
typedef RTC_DAY_OF_WEEK_T sl_calendar_days_of_week_t
```

Renaming days of week structure.

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

calendar_callback_t

```
typedef void(* calendar_callback_t) (void) )(void)
```

Typedef for the function pointer of the callback function.

Definition at line 79 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

Function Documentation

TIME_CONVERSION_ENUM

```
TIME_CONVERSION_ENUM (time_conversion_enum)
```

Enumeration to represent time conversion format.

Parameters

N/A		

Definition at line 92 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

RC_CLOCK_CALIBRATION_ENUM

```
RC_CLOCK_CALIBRATION_ENUM (rc_clock_calibration_enum)
```

Enumeration to represent different rc clock calibration configurations.

Parameters

N/A		

Definition at line 99 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

RO_CLOCK_CALIBRATION_ENUM

```
RO_CLOCK_CALIBRATION_ENUM (ro_clock_calibration_enum)
```

Enumeration to represent different ro clock calibration configurations.

Parameters

N/A		

Definition at line 110 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_set_configuration

```
sl_status_t sl_si91x_calendar_set_configuration (sl_calendar_clock_t clock_type)
```

Configuration and initialization of Calendar i.e., RTC clock.

Parameters

[in]	clock_type	(sl_calendar_clock_t) Enum for RTC Clock Type (RO, RC or XTAL)

Returns

- status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_FAIL (0x0001) - The function is failed

Definition at line 138 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_set_date_time

```
sl_status_t sl_si91x_calendar_set_date_time (sl_calendar_datetime_config_t *config)
```

Set the date and time of the Calendar RTC.

Parameters

[in]	config	(sl_calendar_datetime_config_t) Pointer to the Date Configuration Structure

- Pre-conditions:
  - sl_si91x_calendar_set_configuration
  - sl_si91x_calendar_init

Returns

- status 0 if successful, else error code as follow SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success

Definition at line 153 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_get_date_time

```
sl_status_t sl_si91x_calendar_get_date_time (sl_calendar_datetime_config_t *config)
```

Get the date and time of an existing Calendar RTC.

Parameters

[in]	config	(sl_calendar_datetime_config_t) Pointer to the Date Configuration Structure

- Pre-conditions:
  - sl_si91x_calendar_set_configurationsl_si91x_calendar_init

    \ref sl_si91x_calendar_set_date_time

Returns

- status 0 if successful, else error code as follow SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success

Definition at line 170 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_rcclk_calibration

```
sl_status_t sl_si91x_calendar_rcclk_calibration (clock_calibration_config_t *clock_calibration_config)
```

Calibrate the RC Clock.

Parameters

[in]	clock_calibration_config	(clock_calibration_config_t) pointer to the clock calibration structure

- Pre-conditions:
  - sl_si91x_calendar_calibration_init

Returns

- status 0 if successful, else error code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success

Definition at line 188 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_roclk_calibration

```
sl_status_t sl_si91x_calendar_roclk_calibration (clock_calibration_config_t *clock_calibration_config)
```

Calibrate the RO Clock.

Parameters

[in]	clock_calibration_config	(clock_calibration_config_t) pointer to the clock calibration structure

- Pre-conditions:
  - sl_si91x_calendar_calibration_init

Returns

- status 0 if successful, else error code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success

Definition at line 206 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_register_msec_trigger_callback

```
sl_status_t sl_si91x_calendar_register_msec_trigger_callback (calendar_callback_t callback)
```

Register the callback of the msec trigger and enable it.

Parameters

[in]	callback	(function pointer) Callback function pointer to be called when msec interrupt is triggered

Returns

- status 0 if successful, else error code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_BUSY (0x0004) - The callback is already registered, unregister previous callback before registering new one

Definition at line 221 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_register_sec_trigger_callback

```
sl_status_t sl_si91x_calendar_register_sec_trigger_callback (calendar_callback_t callback)
```

Register the callback of the sec trigger and enable it.

Parameters

[in]	callback	(function pointer) Callback function pointer to be called when sec interrupt is triggered

Returns

- status 0 if successful, else error code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_BUSY (0x0004) - The callback is already registered, unregister previous callback before registering new one

Definition at line 236 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_register_alarm_trigger_callback

```
sl_status_t sl_si91x_calendar_register_alarm_trigger_callback (calendar_callback_t callback)
```

Register the callback of the alarm trigger and enable it.

Parameters

[in]	callback	(function pointer) Callback function pointer to be called when alarm interrupt is triggered

Returns

- status 0 if successful, else error code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_BUSY (0x0004) - The callback is already registered, unregister previous callback before registering new one

Definition at line 251 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_unregister_msec_trigger_callback

```
sl_status_t sl_si91x_calendar_unregister_msec_trigger_callback (void)
```

Unregister the callback of the msec trigger and disable it.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_register_msec_trigger_callback

Returns

- status 0 if successful, else error code as follow SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_FAIL (0x0001) - The function is failed

Definition at line 266 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_unregister_sec_trigger_callback

```
sl_status_t sl_si91x_calendar_unregister_sec_trigger_callback (void)
```

Unregister the callback of the sec trigger and disable it.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_register_sec_trigger_callback

Returns

- status 0 if successful, else error code as follow SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_FAIL (0x0001) - The function is failed

Definition at line 281 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_unregister_alarm_trigger_callback

```
sl_status_t sl_si91x_calendar_unregister_alarm_trigger_callback (void)
```

Unregister the callback of the alarm trigger and disable it.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_register_alarm_trigger_callback

Returns

- status 0 if successful, else error code as follow SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_FAIL (0x0001) - The function is failed

Definition at line 296 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_set_alarm

```
sl_status_t sl_si91x_calendar_set_alarm (sl_calendar_datetime_config_t *alarm)
```

Set the date and time of new alarm in RTC.

Parameters

[in]	alarm	Pointer to the Date Configuration Structure

- Pre-conditions:
  - sl_si91x_calendar_set_configuration

sl_si91x_calendar_init

Returns

- status 0 if successful, else error code as follow SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success

Definition at line 313 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_get_alarm

```
sl_status_t sl_si91x_calendar_get_alarm (sl_calendar_datetime_config_t *alarm)
```

Get the date and time of an existing alarm in RTC.

Parameters

[in]	alarm	Pointer to the Date Configuration Structure

- Pre-conditions:
  - sl_si91x_calendar_set_configuration
- sl_si91x_calendar_init
- sl_si91x_calendar_set_alarm

Returns

- status 0 if successful, else error code as follow SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success

Definition at line 332 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_build_datetime_struct

```
sl_status_t sl_si91x_calendar_build_datetime_struct (sl_calendar_datetime_config_t *date, uint8_t Century, uint8_t Year,
sl_calendar_month_t Month, sl_calendar_days_of_week_t DayOfWeek, uint8_t Day, uint8_t Hour, uint8_t Minute, uint8_t
Second, uint16_t Milliseconds)
```

Build the structure for date-time configuration.

Parameters

[in]	date	Pointer to the Date Configuration Structure
[in]	Century	(uint8_t) Century (0-4)
[in]	Year	(uint8_t) Year (1-99) + (Century * 1000)
[in]	Month	(enum) Month from the RTC_MONTH_T enum
[in]	DayOfWeek	(enum) Day of Week from the RTC_DAY_OF_WEEK_T enum
[in]	Day	Day (uint8_t) (1-31)
[in]	Hour	Hour (uint8_t) (0-23)
[in]	Minute	Minutes (uint8_t) (0-59)
[in]	Second	Seconds (uint8_t) (0-59)
[in]	Milliseconds	Milliseconds (uint16_t) (0-999)

Returns

- status 0 if successful, else error code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid SL_STATUS_OK (0x0000) - Success

Definition at line 352 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_convert_unix_time_to_ntp_time

```
sl_status_t sl_si91x_calendar_convert_unix_time_to_ntp_time (uint32_t time, uint32_t *ntp_time)
```

Convert Unix timestamp to NTP timestamp.

Parameters

[in]	time	(uint32_t) Unix timestamp
[in]	ntp_time	(uint32_t) variable to store NTP timestamp

Returns

- status 0 if successful, else error code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success

Definition at line 375 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_convert_ntp_time_to_unix_time

```
sl_status_t sl_si91x_calendar_convert_ntp_time_to_unix_time (uint32_t ntp_time, uint32_t *time)
```

Convert NTP timestamp to Unix timestamp.

Parameters

[in]	ntp_time	(uint32_t) NTP timestamp
[in]	time	(uint32_t) variable to store Unix timestamp

Returns

- status 0 if successful, else error code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_OK (0x0000) - Success

Definition at line 389 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_is_msec_trigger_enabled

```
boolean_t sl_si91x_calendar_is_msec_trigger_enabled (void)
```

Return the state of msec trigger of RTC (enabled or disabled).

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_register_msec_trigger_callback

Returns

- (boolean)true if trigger is enabled, false otherwise

Definition at line 401 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_is_sec_trigger_enabled

```
boolean_t sl_si91x_calendar_is_sec_trigger_enabled (void)
```

Return the state of sec trigger of RTC (enabled or disabled).

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_register_sec_trigger_callback

Returns

- (boolean)true if trigger is enabled, false otherwise

Definition at line `413` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_is_alarm_trigger_enabled

```
boolean_t sl_si91x_calendar_is_alarm_trigger_enabled (void)
```

Return the state of alarm trigger of RTC (enabled or disabled).

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_register_alarm_trigger_callback

Returns

- (boolean)true if trigger is enabled, false otherwise

Definition at line `425` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_rtc_start

```
__STATIC_INLINE void sl_si91x_calendar_rtc_start (void)
```

Start the Calendar RTC.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_set_configuration
- sl_si91x_calendar_init

Returns

- none

Definition at line `439` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_rtc_stop

```
__STATIC_INLINE void sl_si91x_calendar_rtc_stop (void)
```

Stop the Calendar RTC.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_set_configuration
- sl_si91x_calendar_init
- sl_si91x_calendar_rtc_start

Returns

- none

Definition at line `458` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_calibration_init

```
__STATIC_INLINE void sl_si91x_calendar_calibration_init (void)
```

Initialize the calibration for Calendar clocks.

Parameters

N/A		

Returns

- none

Definition at line `469` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_clear_msec_trigger

```
__STATIC_INLINE void sl_si91x_calendar_clear_msec_trigger (void)
```

Clear the msec trigger.

Parameters

N/A		

Returns

- none

Definition at line `480` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_clear_sec_trigger

```
__STATIC_INLINE void sl_si91x_calendar_clear_sec_trigger (void)
```

Clear the sec trigger.

Parameters

N/A		

Returns

- none

Definition at line `491` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_clear_alarm_trigger

```
__STATIC_INLINE void sl_si91x_calendar_clear_alarm_trigger (void)
```

Clear the alarm trigger.

Parameters

N/A		

Returns

- none

Definition at line `502` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_init

```
__STATIC_INLINE void sl_si91x_calendar_init (void)
```

De-initialize calendar operation.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_set_configuration

Returns

- none

Definition at line `516` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

sl_si91x_calendar_deinit

```
__STATIC_INLINE void sl_si91x_calendar_deinit (void)
```

De-initialize calendar operation.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_calendar_init

Returns

- none

Definition at line 530 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_si91x_calendar_get_version

```
sl_calendar_version_t sl_si91x_calendar_get_version (void)
```

Get the calendar version.

Parameters

[in]		

This function is used to know the calendar version.

Returns

- sl_calendar_version_t type version

Definition at line 545 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

Macro Definition Documentation

SLI_ALARM_IRQHandler

```
#define SLI_ALARM_IRQHandler
```

Value:

```
IRQ028_Handler
```

Alarm IRQ Handler.

Definition at line 48 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

SLI_MSEC_SEC_IRQHandler

```
#define SLI_MSEC_SEC_IRQHandler
```

Value:

```
IRQ029_Handler
```

RTC IRQ Handler.

Definition at line 49 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

SLI_NVIC_ALARM

```
#define SLI_NVIC_ALARM
```

Value:

```
MCU_CAL_ALARM_IRQn
```

Alarm NVIQ enable.

Definition at line 50 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

SLI_NVIC_MSEC_SEC

```
#define SLI_NVIC_MSEC_SEC
```

Value:

```
MCU_CAL_RTC_IRQn
```

RTC NVIQ enable.

Definition at line 51 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

TIME_CONVERSION_ENUM

```
#define TIME_CONVERSION_ENUM
```

Value:

```
0 | typedef uint8_t name;       \
0 | enum name##_enum
```

Time Conversion format enum declaration.

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

RC_CLOCK_CALIBRATION_ENUM

```
#define RC_CLOCK_CALIBRATION_ENUM
```

Value:

```
0 | typedef uint8_t name;          \
0 | enum name##_enum
```

Time Conversion format enum declaration.

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

RO_CLOCK_CALIBRATION_ENUM

```
#define RO_CLOCK_CALIBRATION_ENUM
```

Value:

```
0 | typedef uint8_t name;          \
```

```
0 | enum name##_enum
```

Time Conversion format enum declaration.

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

clock_calibration_config_t

Structure to hold the parameters of clock calibration, trigger time macros can be used to fill trigger time.

Public Attributes

boolean_t	rc_enable_calibration true to enable and false to disable rc calibration
boolean_t	rc_enable_periodic_calibration true to enable and false to disable rc periodic calibration
uint8_t	rc_trigger_time rc trigger time, 5 sec, 10 sec, 15 sec, 30 sec, 1 min, 2 min
boolean_t	ro_enable_calibration true to enable and false to disable ro calibration
boolean_t	ro_enable_periodic_calibration true to enable and false to disable periodic calibration
uint8_t	ro_trigger_time ro trigger time, 1 sec, 2 sec, 4 sec, 8 sec

Public Attribute Documentation

rc_enable_calibration

```
boolean_t clock_calibration_config_t::rc_enable_calibration
```

true to enable and false to disable rc calibration

Definition at line `83` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

rc_enable_periodic_calibration

```
boolean_t clock_calibration_config_t::rc_enable_periodic_calibration
```

true to enable and false to disable rc periodic calibration

Definition at line `84` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

rc_trigger_time

```
uint8_t clock_calibration_config_t::rc_trigger_time
```

rc trigger time, 5 sec, 10 sec, 15 sec, 30 sec, 1 min, 2 min

Definition at line `85` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h`

ro_enable_calibration

> boolean_t clock_calibration_config_t::ro_enable_calibration

true to enable and false to disable ro calibration

Definition at line 86 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

ro_enable_periodic_calibration

> boolean_t clock_calibration_config_t::ro_enable_periodic_calibration

true to enable and false to disable periodic calibration

Definition at line 87 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

ro_trigger_time

> uint8_t clock_calibration_config_t::ro_trigger_time

ro trigger time, 1 sec, 2 sec, 4 sec, 8 sec

Definition at line 88 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

sl_calendar_version_t

Structure to hold the different versions of peripheral API.

Public Attributes

uint8_t	release Release version number.
uint8_t	major sqa version number
uint8_t	minor dev version number

Public Attribute Documentation

release

```
uint8_t sl_calendar_version_t::release
```

Release version number.

Definition at line 120 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

major

```
uint8_t sl_calendar_version_t::major
```

sqa version number

Definition at line 121 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

minor

```
uint8_t sl_calendar_version_t::minor
```

dev version number

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_calendar.h

Config Timer

Modules

- sl_config_timer_config_t
- sl_config_timer_ocu_config_t
- sl_config_timer_ocu_control_t
- sl_config_action_event_t
- sl_config_timer_interrupt_flags_t
- sl_config_timer_version_t

Enumerations

enum sl_config_timer_mode_t {

SL_COUNTER_16BIT
SL_COUNTER_32BIT
SL_COUNTER_MODE_LAST

}

Enumeration to represent timer modes.

enum sl_counter_number_t {

SL_COUNTER_0
SL_COUNTER_1
SL_COUNTER_NUMBER_LAST

}

Enumeration to represent timer counter numbers.

enum sl_counter0_direction_t {

SL_COUNTER0_UP
SL_COUNTER0_DOWN
SL_COUNTER0_UP_DOWN
SL_COUNTER0_DIRECTION_LAST

}

Enumeration to represent timer counter directions.

enum sl_counter1_direction_t {

SL_COUNTER1_UP
SL_COUNTER1_DOWN
SL_COUNTER1_UP_DOWN
SL_COUNTER1_DIRECTION_LAST

}

enum sl_config_timer_config_values_t {

SL_COUNTER_MODE_32 = COUNTER32_BITMODE
SL_COUNTER0_SOFT_RESET_ENABLE = SOFTRESET_COUNTER_0
SL_COUNTER0_PERIODIC_ENABLE = PERIODIC_ENCOUNTER_0
SL_COUNTER0_TRIGGER_ENABLE = COUNTER0_TRIG
SL_COUNTER0_SYNC_TRIGGER_ENABLE = COUNTER0_SYNC_TRIG
SL_COUNTER0_BUFFER_ENABLE = BUF_REG0EN
SL_COUNTER0_UP_DIRECTION = COUNTER0_UP
SL_COUNTER0_DOWN_DIRECTION = COUNTER0_DOWN
SL_COUNTER0_UP_DOWN_DIRECTION = COUNTER0_UP_DOWN
SL_COUNTER1_SOFT_RESET_ENABLE = SOFTRESET_COUNTER_1
SL_COUNTER1_PERIODIC_ENABLE = PERIODIC_ENCOUNTER_1
SL_COUNTER1_TRIGGER_ENABLE = COUNTER1_TRIG
SL_COUNTER1_SYNC_TRIGGER_ENABLE = COUNTER1_SYNC_TRIG
SL_COUNTER1_BUFFER_ENABLE = BUF_REG1EN
SL_COUNTER1_UP_DIRECTION = COUNTER1_UP
SL_COUNTER1_DOWN_DIRECTION = COUNTER1_DOWN
SL_COUNTER1_UP_DOWN_DIRECTION = COUNTER1_UP_DOWN
SL_COUNTER_PARAM_LAST

}

Enumeration to represent timer-config parameters values.

enum sl_config_timer_ocu_config_values_t {

SL_COUNTER0_OCU_OUTPUT_ENABLE = OUTPUT_OCU_0
SL_COUNTER0_OCU_DMA_ENABLE = OCU_DMA_0
SL_COUNTER0_OCU_8BIT_ENABLE = OCU_8_MODE_0
SL_COUNTER0_OCU_SYNC_ENABLE = OCU_SYNC_WITH_0
SL_COUNTER1_OCU_OUTPUT_ENABLE = OUTPUT_OCU_1
SL_COUNTER1_OCU_DMA_ENABLE = OCU_DMA_1
SL_COUNTER1_OCU_8BIT_ENABLE = OCU_8_MODE_1
SL_COUNTER1_OCU_SYNC_ENABLE = OCU_SYNC_WITH_1
SL_OCU_OUTPUT0_TOGGLE_HIGH = MAKE_OUTPUT_0_HIGH_SEL_0
SL_OCU_OUTPUT0_TOGGLE_LOW = MAKE_OUTPUT_0_LOW_SEL_0
SL_OCU_OUTPUT1_TOGGLE_HIGH = MAKE_OUTPUT_1_HIGH_SEL_1
SL_OCU_OUTPUT1_TOGGLE_LOW = MAKE_OUTPUT_1_LOW_SEL_1
SL_OCU_PARAM_LAST

}

Enumeration to represent timer OCU-config parameters values.

enum sl_config_timer_event_t {

SL_NO_EVENT
SL_EVENT0_RISING_EDGE
SL_EVENT1_RISING_EDGE
SL_EVENT2_RISING_EDGE
SL_EVENT3_RISING_EDGE
SL_EVENT0_FALLING_EDGE
SL_EVENT1_FALLING_EDGE
SL_EVENT2_FALLING_EDGE
SL_EVENT3_FALLING_EDGE
SL_EVENT0_RISING_FALLING_EDGE
SL_EVENT1_RISING_FALLING_EDGE
SL_EVENT2_RISING_FALLING_EDGE
SL_EVENT3_RISING_FALLING_EDGE
SL_EVENT0_LEVEL0
SL_EVENT1_LEVEL0
SL_EVENT2_LEVEL0
SL_EVENT3_LEVEL0
SL_EVENT0_LEVEL1
SL_EVENT1_LEVEL1
SL_EVENT2_LEVEL1
SL_EVENT3_LEVEL1
SL_AND_EVENT
SL_OR_EVENT
SL_EVENT0_RISING_EDGE_AND_EVENT
SL_EVENT0_RISING_EDGE_OR_EVENT
SL_EVENT1_RISING_EDGE_AND_EVENT
SL_EVENT1_RISING_EDGE_OR_EVENT
SL_EVENT2_RISING_EDGE_AND_EVENT
SL_EVENT2_RISING_EDGE_OR_EVENT
SL_EVENT3_RISING_EDGE_AND_EVENT
SL_EVENT3_RISING_EDGE_OR_EVENT
SL_EVENT0_RISING_EDGE_REGISTERED_AND_EVENT
SL_EVENT0_RISING_EDGE_REGISTERED_OR_EVENT
SL_EVENT1_RISING_EDGE_REGISTERED_AND_EVENT
SL_EVENT1_RISING_EDGE_REGISTERED_OR_EVENT
SL_EVENT2_RISING_EDGE_REGISTERED_AND_EVENT
SL_EVENT2_RISING_EDGE_REGISTERED_OR_EVENT
SL_EVENT3_RISING_EDGE_REGISTERED_AND_EVENT
SL_EVENT3_RISING_EDGE_REGISTERED_OR_EVENT
SL_EVENT_LAST

}

Enumeration to represent various timer input events.

enum sl_config_timer_action_t {

START
STOP
CONTINUE
HALT
INCREMENT
CAPTURE
INTERRUPT
OUTPUT
ACTION_LAST

}

Enumeration to represent various timer actions.

enum sl_config_timer_interrupt_flags_values_t {

SL_CT_EVENT_INTR_0_FLAG = RSI_CT_EVENT_INTR_0_I
SL_CT_FIFO_0_FULL_FLAG = RSI_CT_EVENT_FIFO_0_FULL_I
SL_CT_COUNTER_0_IS_ZERO_FLAG = RSI_CT_EVENT_COUNTER_0_IS_ZERO_I
SL_CT_COUNTER_0_IS_PEAK_FLAG = RSI_CT_EVENT_COUNTER_0_IS_PEAK_I
SL_CT_EVENT_INTR_1_FLAG = RSI_CT_EVENT_INTR_1_I
SL_CT_FIFO_1_FULL_FLAG = RSI_CT_EVENT_FIFO_1_FULL_I
SL_CT_COUNTER_1_IS_ZERO_FLAG = RSI_CT_EVENT_COUNTER_1_IS_ZERO_I
SL_CT_COUNTER_1_IS_PEAK_FLAG = RSI_CT_EVENT_COUNTER_1_IS_PEAK_I

}

Enumeration to represent various timer interrupt flag values.

Typedefs

typedef OCU_PARAMS_T — sl_config_timer_ocu_params_t
Renaming OCU parameters structure type.

typedef WFG_PARAMS_T — sl_config_timer_wfg_config_t
Renaming WFG parameters structure type.

typedef RSI_CT_CALLBACK_T — sl_config_timer_pwm_callback_t
Renaming MCPWM callback structure.

typedef void(* sl_config_timer_callback_t)(void *callback_flag)
Typedef for the function pointer of the interrupt callback function.

Functions

void sl_si91x_config_timer_init(void)
Initialize config-timer output GPIO pins and configures clock as 16 MHz.

sl_status_t sl_si91x_config_timer_set_mode(sl_config_timer_mode_t mode)
Set Config-timer mode as 32-bit or 16-bit counters.

sl_status_t sl_si91x_config_timer_set_configuration(sl_config_timer_config_t *timer_config_ptr)
Set Config-timer configurations such as 32-bit or 16-bit mode, periodic mode, software trigger enable, soft reset enable, buffer enable, sync trigger enable and direction.

void sl_si91x_config_timer_reset_configuration(void)
Reset config-timer parameters such as sets 16-bit mode, sets up-counter direction and disables periodic mode, soft reset, buffer, sync & software trigger of counters.

sl_status_t sl_si91x_config_timer_set_ocu_configuration(sl_config_timer_ocu_config_t *ocu_config_ptr)
Set Config-timer OCU configurations such as enables outputs in OCU mode, OCU-DMA mode, channel sync with OCU outputs, 8-bit mode for OCU outputs for both counters.

void sl_si91x_config_timer_reset_ocu_configuration(void)
Reset config-timer OCU parameters such as sets 16-bit mode, sets up-counter direction and disables DMA mode, channel sync and 8-bit mode for OCU outputs.

sl_status_t sl_si91x_config_timer_set_ocu_control(sl_config_timer_ocu_control_t *ocu_params)
Set Config-timer OCU mode first and next threshold values for counter-0 & counter-1 outputs , register PWM callback, Enable DMA support for counters.

sl_status_t sl_si91x_config_timer_set_wfg_configuration(sl_config_timer_wfg_config_t *wfg_config_ptr)
Set Config-timer WFG mode configurations such as select toggle high, low and peak for counter-0 & counter-1 outputs.

sl_status_t	sl_si91x_config_timer_set_initial_count(sl_config_timer_mode_t mode, uint32_t counter0_initial_value, uint32_t counter1_initial_value) Set Config-timer initial count as per timer mode.
sl_status_t	sl_si91x_config_timer_set_match_count(sl_config_timer_mode_t mode, sl_counter_number_t counter_number, uint32_t match_value) Set Config-timer match-count as per timer mode and counter-number.
sl_status_t	sl_si91x_config_timer_get_count(sl_config_timer_mode_t mode, sl_counter_number_t counter_number, uint32_t *count_value) Get Config-timer current count as per timer mode and counter-number.
sl_status_t	sl_si91x_config_timer_reset_counter(sl_counter_number_t counter_number) Soft reset Config-timer counter.
sl_status_t	sl_si91x_config_timer_start_on_software_trigger(sl_counter_number_t counter_number) Start config-timer counter by software trigger.
sl_status_t	sl_si91x_config_timer_select_action_event(sl_config_timer_action_t action, sl_config_timer_event_t select_event_counter0, sl_config_timer_event_t select_event_counter1) Select external input event for triggering selected timer-action such as start, stop, continue, halt, increment, capture, interrupt and output.
sl_status_t	sl_si91x_config_timer_configure_action_event(sl_config_action_event_t *event_config_handle) Configure external input-event's AND-event and OR-event for triggering selected timer-action such as start, stop, continue, halt, increment, capture, interrupt and output.
sl_status_t	sl_si91x_config_timer_register_callback(sl_config_timer_callback_t on_config_timer_callback, void *callback_flag_value, sl_config_timer_interrupt_flags_t *interrupt_flags) Register callback of timer interrupt and enabling respective interrupts as per selected interrupt flag.
sl_status_t	sl_si91x_config_timer_unregister_callback(sl_config_timer_interrupt_flags_t *interrupt_flags)
sl_status_t	sl_si91x_config_timer_resume_halt_event(sl_counter_number_t counter_number) Resume halt operation of Config-timer counter.
sl_status_t	sl_si91x_config_timer_read_capture(sl_counter_number_t counter_number, uint16_t *capture_value) Get Config-timer counter count value on occurrence of capture event occurrence.
sl_status_t	sl_si91x_config_timer_set_counter_sync(sl_counter_number_t counter_number, uint8_t sync_counter_value) Sync counter output with other channels, as per sync_counter_value.
sl_status_t	sl_si91x_config_timer_set_output_adc_pin(uint8_t pin1, uint8_t pin2) Sync counter output with other channels.
sl_status_t	sl_si91x_config_timer_set_wfg_compare_values(sl_counter_number_t counter_number, sl_config_timer_ocu_params_t *ocu_params)
void	sl_si91x_config_timer_deinit(void) De-initialize config-timer by disabling its clock.
sl_config_timer_version_t	sl_si91x_config_timer_get_version(void) Get the release version of the Config-timer.

Macros

#define	CT CT0 pointer to CT base address

Enumeration Documentation

sl_config_timer_mode_t

> sl_config_timer_mode_t

Enumeration to represent timer modes.

Enumerator	
SL_COUNTER_16BIT	enum for CT 16-bit mode
SL_COUNTER_32BIT	enum for CT 32-bit mode
SL_COUNTER_MODE_LAST	Last member of enum for validation.

Definition at line 67 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_counter_number_t

> sl_counter_number_t

Enumeration to represent timer counter numbers.

Enumerator	
SL_COUNTER_0	enum for CT counter-0
SL_COUNTER_1	enum for CT counter-1
SL_COUNTER_NUMBER_LAST	Last member of enum for validation.

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_counter0_direction_t

> sl_counter0_direction_t

Enumeration to represent timer counter directions.

Enumerator	
SL_COUNTER0_UP	CT counter-0 up-counting operation.
SL_COUNTER0_DOWN	CT counter-0 down-counting operation.
SL_COUNTER0_UP_DOWN	CT counter-0 up-down counting operation.
SL_COUNTER0_DIRECTION_LAST	Last member of enum for validation.

Definition at line 81 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_counter1_direction_t

> sl_counter1_direction_t

Enumerator	
SL_COUNTER1_UP	CT counter-1 up-counting operation.
SL_COUNTER1_DOWN	CT counter-1 down-counting operation.
SL_COUNTER1_UP_DOWN	CT counter-1 up-down counting operation.
SL_COUNTER1_DIRECTION_LAST	Last member of enum for validation.

Definition at line 88 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_config_values_t

> sl_config_timer_config_values_t

Enumeration to represent timer-config parameters values.

Enumerator	
SL_COUNTER_MODE_32	enum for CT 32-bit mode value
SL_COUNTER0_SOFT_RESET_ENABLE	enum for counter-0 soft reset enable value
SL_COUNTER0_PERIODIC_ENABLE	enum for counter-0 periodic mode enable value
SL_COUNTER0_TRIGGER_ENABLE	enum for counter-0 software trigger enable value
SL_COUNTER0_SYNC_TRIGGER_ENABLE	enum for counter-0 sync trigger enable value
SL_COUNTER0_BUFFER_ENABLE	enum for counter-0 buffer enable value
SL_COUNTER0_UP_DIRECTION	enum for counter-0 up-direction enable value
SL_COUNTER0_DOWN_DIRECTION	enum for counter-0 down-direction enable value
SL_COUNTER0_UP_DOWN_DIRECTION	enum for counter-0 up-down direction enable value
SL_COUNTER1_SOFT_RESET_ENABLE	enum for counter-1 soft reset enable value
SL_COUNTER1_PERIODIC_ENABLE	enum for counter-1 periodic mode enable value
SL_COUNTER1_TRIGGER_ENABLE	enum for counter-1 software trigger enable value
SL_COUNTER1_SYNC_TRIGGER_ENABLE	enum for counter-1 sync trigger enable value
SL_COUNTER1_BUFFER_ENABLE	enum for counter-1 buffer enable value
SL_COUNTER1_UP_DIRECTION	enum for counter-1 up-direction enable value
SL_COUNTER1_DOWN_DIRECTION	enum for counter-1 up-direction enable value
SL_COUNTER1_UP_DOWN_DIRECTION	enum for counter-1 up-direction enable value
SL_COUNTER_PARAM_LAST	Last member of enum for validation.

Definition at line 96 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_ocu_config_values_t

> sl_config_timer_ocu_config_values_t

Enumeration to represent timer OCU-config parameters values.

Enumerator	
SL_COUNTER0_OCU_OUTPUT_ENABLE	enum for counter-0 output enable value
SL_COUNTER0_OCU_DMA_ENABLE	enum for counter-0 OCU-DMA mode enable value
SL_COUNTER0_OCU_8BIT_ENABLE	enum for counter-0 OCU-8bit mode enable value
SL_COUNTER0_OCU_SYNC_ENABLE	enum for counter-0 OCU-sync mode enable value
SL_COUNTER1_OCU_OUTPUT_ENABLE	enum for counter-1 output enable value
SL_COUNTER1_OCU_DMA_ENABLE	enum for counter-1 OCU-DMA mode enable value
SL_COUNTER1_OCU_8BIT_ENABLE	enum for counter-1 OCU-8bit mode enable value
SL_COUNTER1_OCU_SYNC_ENABLE	enum for counter-1 OCU-sync mode enable value
SL_OCU_OUTPUT0_TOGGLE_HIGH	enum for counter-0 output toggle high

SL_OCU_OUTPUT0_TOGGLE_LOW	enum for counter-0 output toggle low
SL_OCU_OUTPUT1_TOGGLE_HIGH	enum for counter-1 output toggle high select value
SL_OCU_OUTPUT1_TOGGLE_LOW	enum for counter-1 output toggle low select value
SL_OCU_PARAM_LAST	Last member of enum for validation.

Definition at line 118 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_event_t

> sl_config_timer_event_t

Enumeration to represent various timer input events.

Enumerator	
SL_NO_EVENT	enum for no input event
SL_EVENT0_RISING_EDGE	enum for input-0 rising edge event
SL_EVENT1_RISING_EDGE	enum for input-1 rising edge event
SL_EVENT2_RISING_EDGE	enum for input-2 rising edge event
SL_EVENT3_RISING_EDGE	enum for input-3 rising edge event
SL_EVENT0_FALLING_EDGE	enum for input-0 falling edge event
SL_EVENT1_FALLING_EDGE	enum for input-1 falling edge event
SL_EVENT2_FALLING_EDGE	enum for input-2 falling edge event
SL_EVENT3_FALLING_EDGE	enum for input-3 falling edge event
SL_EVENT0_RISING_FALLING_EDGE	enum for input-0 rising-falling edge event
SL_EVENT1_RISING_FALLING_EDGE	enum for input-1 rising-falling edge event
SL_EVENT2_RISING_FALLING_EDGE	enum for input-2 rising-falling edge event
SL_EVENT3_RISING_FALLING_EDGE	enum for input-3 rising-falling edge event
SL_EVENT0_LEVEL0	enum for input-0 Level-0 event
SL_EVENT1_LEVEL0	enum for input-0 Level-0 event
SL_EVENT2_LEVEL0	enum for input-0 Level-0 event
SL_EVENT3_LEVEL0	enum for input-0 Level-0 event
SL_EVENT0_LEVEL1	enum for input-0 Level-0 event
SL_EVENT1_LEVEL1	enum for input-0 Level-0 event
SL_EVENT2_LEVEL1	enum for input-0 Level-0 event
SL_EVENT3_LEVEL1	enum for input-0 Level-0 event
SL_AND_EVENT	enum for and-event
SL_OR_EVENT	enum for or-event
SL_EVENT0_RISING_EDGE_AND_EVENT	enum for input-0 rising edge and-event
SL_EVENT0_RISING_EDGE_OR_EVENT	enum for input-0 rising edge or-event
SL_EVENT1_RISING_EDGE_AND_EVENT	enum for input-1 rising edge and-event
SL_EVENT1_RISING_EDGE_OR_EVENT	enum for input-1 rising edge or-event
SL_EVENT2_RISING_EDGE_AND_EVENT	enum for input-2 rising edge and-event
SL_EVENT2_RISING_EDGE_OR_EVENT	enum for input-2 rising edge or-event
SL_EVENT3_RISING_EDGE_AND_EVENT	enum for input-3 rising edge and-event

SL_EVENT3_RISING_EDGE_OR_EVENT	enum for input-3 rising edge or-event
SL_EVENT0_RISING_EDGE_REGISTERED_AND_EVENT	enum for input-0 rising edge registered and-event
SL_EVENT0_RISING_EDGE_REGISTERED_OR_EVENT	enum for input-0 rising edge registered or-event
SL_EVENT1_RISING_EDGE_REGISTERED_AND_EVENT	enum for input-1 rising edge registered and-event
SL_EVENT1_RISING_EDGE_REGISTERED_OR_EVENT	enum for input-1 rising edge registered or-event
SL_EVENT2_RISING_EDGE_REGISTERED_AND_EVENT	enum for input-2 rising edge registered and-event
SL_EVENT2_RISING_EDGE_REGISTERED_OR_EVENT	enum for input-2 rising edge registered or-event
SL_EVENT3_RISING_EDGE_REGISTERED_AND_EVENT	enum for input-3 rising edge registered and-event
SL_EVENT3_RISING_EDGE_REGISTERED_OR_EVENT	enum for input-3 rising edge registered or-event
SL_EVENT_LAST	Last member of enum for validation.

Definition at line 135 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_action_t

> sl_config_timer_action_t

Enumeration to represent various timer actions.

Enumerator	
START	enum for timer-start action
STOP	enum for timer-stop action
CONTINUE	enum for timer-continue action
HALT	enum for timer-halt action
INCREMENT	enum for timer-increment action
CAPTURE	enum for timer-capture action
INTERRUPT	enum for timer-interrupt action
OUTPUT	enum for timer-output action
ACTION_LAST	Last member of enum for validation.

Definition at line 179 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_interrupt_flags_values_t

> sl_config_timer_interrupt_flags_values_t

Enumeration to represent various timer interrupt flag values.

Enumerator	
SL_CT_EVENT_INTR_0_FLAG	enum for counter-0 event interrupt enable value
SL_CT_FIFO_0_FULL_FLAG	enum for counter-0 FIFO full interrupt enable value
SL_CT_COUNTER_0_IS_ZERO_FLAG	enum for counter-0 zero-count-value interrupt enable value
SL_CT_COUNTER_0_IS_PEAK_FLAG	enum for counter-0 match-value interrupt enable value
SL_CT_EVENT_INTR_1_FLAG	enum for counter-1 event interrupt enable value
SL_CT_FIFO_1_FULL_FLAG	enum for counter-1 FIFO full interrupt enable value
SL_CT_COUNTER_1_IS_ZERO_FLAG	enum for counter-1 zero-count-value interrupt enable value
SL_CT_COUNTER_1_IS_PEAK_FLAG	enum for counter-1 match-value interrupt enable value

Definition at line 192 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

Typedef Documentation

sl_config_timer_ocu_params_t

```
typedef OCU_PARAMS_T sl_config_timer_ocu_params_t
```

Renaming OCU parameters structure type.

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_wfg_config_t

```
typedef WFG_PARAMS_T sl_config_timer_wfg_config_t
```

Renaming WFG parameters structure type.

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_pwm_callback_t

```
typedef RSI_CT_CALLBACK_T sl_config_timer_pwm_callback_t
```

Renaming MCPWM callback structure.

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_callback_t

```
typedef void(* sl_config_timer_callback_t) (void *callback_flag) )(void *callback_flag)
```

Typedef for the function pointer of the interrupt callback function.

Parameters

N/A	callback_flag	(void *) parameter for updating flag values

Definition at line 64 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

Function Documentation

sl_si91x_config_timer_init

```
void sl_si91x_config_timer_init (void)
```

Initialize config-timer output GPIO pins and configures clock as 16 MHz.

Parameters

[in]		

Returns

- none

Definition at line 301 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_set_mode

```
sl_status_t sl_si91x_config_timer_set_mode (sl_config_timer_mode_t mode)
```

Set Config-timer mode as 32-bit or 16-bit counters.

Parameters

[in]	mode	sl_config_timer_mode_t for possible values

- Pre-conditions:
  - sl_si91x_config_timer_init();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_INVALID_MODE (0x0024) - 'mode' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, timer-mode is set properly

Definition at line 317 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_set_configuration

```
sl_status_t sl_si91x_config_timer_set_configuration (sl_config_timer_config_t *timer_config_ptr)
```

Set Config-timer configurations such as 32-bit or 16-bit mode, periodic mode, software trigger enable, soft reset enable, buffer enable, sync trigger enable and direction.

Parameters

[in]	timer_config_ptr	Pointer to CT config structure sl_config_timer_config_t

- Pre-conditions:
  - sl_si91x_config_timer_init();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Counter direction parameter has invalid value.
  - SL_STATUS_OK (0x0000) - Success, timer-mode is set properly

Definition at line 337 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_reset_configuration

```
void sl_si91x_config_timer_reset_configuration (void)
```

Reset config-timer parameters such as sets 16-bit mode, sets up-counter direction and disables periodic mode, soft reset, buffer, sync & software trigger of counters.

Parameters

[in]		

Returns

- none

Definition at line 346 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_set_ocu_configuration

```
sl_status_t sl_si91x_config_timer_set_ocu_configuration (sl_config_timer_ocu_config_t *ocu_config_ptr)
```

Set Config-timer OCU configurations such as enables outputs in OCU mode, OCU-DMA mode, channel sync with OCU outputs, 8-bit mode for OCU outputs for both counters.

Parameters

[in]	ocu_config_ptr	Pointer to CT OCU-config structure sl_config_timer_ocu_config_t

- Pre-conditions:
  - sl_si91x_config_timer_init();
- sl_si91x_config_timer_set_configuration();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer.
  - SL_STATUS_OK (0x0000) - Success, timer-mode is set properly.

Definition at line 366 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_reset_ocu_configuration

```
void sl_si91x_config_timer_reset_ocu_configuration (void)
```

Reset config-timer OCU parameters such as sets 16-bit mode, sets up-counter direction and disables DMA mode, channel sync and 8-bit mode for OCU outputs.

Parameters

[in]		

Returns

- none

Definition at line 375 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_set_ocu_control

```
sl_status_t sl_si91x_config_timer_set_ocu_control (sl_config_timer_ocu_control_t *ocu_params)
```

Set Config-timer OCU mode first and next threshold values for counter-0 & counter-1 outputs , register PWM callback, Enable DMA support for counters.

Parameters

[in]	ocu_params	Pointer to CT OCU-config structure sl_config_timer_ocu_control_t

- Pre-conditions:
  - sl_si91x_config_timer_init();

sl_si91x_config_timer_set_ocu_configuration();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer.
  - SL_STATUS_OK (0x0000) - Success, timer-mode is set properly.

Definition at line 395 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_set_wfg_configuration

```
sl_status_t sl_si91x_config_timer_set_wfg_configuration (sl_config_timer_wfg_config_t *wfg_config_ptr)
```

Set Config-timer WFG mode configurations such as select toggle high, low and peak for counter-0 & counter-1 outputs.

Parameters

[in]	wfg_config_ptr	Pointer to CT wfg-config structure sl_config_timer_wfg_config_t

- Pre-conditions:
  - sl_si91x_config_timer_init();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer.
  - SL_STATUS_OK (0x0000) - Success, timer-mode is set properly.

Definition at line 413 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_set_initial_count

```
sl_status_t sl_si91x_config_timer_set_initial_count (sl_config_timer_mode_t mode, uint32_t counter0_initial_value, uint32_t counter1_initial_value)
```

Set Config-timer initial count as per timer mode.

Parameters

[in]	mode	sl_config_timer_mode_t for possible values
[in]	counter0_initial_value	(uint16_t)
[in]	counter1_initial_value	(uint16_t)

- Pre-conditions:
  - sl_si91x_config_timer_init();
- l_si91x_config_timer_set_configuration();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_INVALID_MODE (0x0024) - 'mode' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, initial-count is set properly

Definition at line 434 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_set_match_count

```
sl_status_t sl_si91x_config_timer_set_match_count (sl_config_timer_mode_t mode, sl_counter_number_t counter_number, uint32_t match_value)
```

Set Config-timer match-count as per timer mode and counter-number.

Parameters

[in]	mode	sl_counter_number_t for possible values
[in]	counter_number	sl_config_timer_mode_t for possible values
[in]	match_value	(uint32_t)

- Pre-conditions:
  - sl_si91x_config_timer_init();
- l_si91x_config_timer_set_configuration();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_INVALID_MODE (0x0024) - 'mode' parameter value is invalid.
  - SL_STATUS_INVALID_PARAMETER 0x0021) - 'counter_number' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, match-value is set properly

Definition at line `459` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h`

sl_si91x_config_timer_get_count

```
sl_status_t sl_si91x_config_timer_get_count (sl_config_timer_mode_t mode, sl_counter_number_t counter_number, uint32_t *count_value)
```

Get Config-timer current count as per timer mode and counter-number.

Parameters

[in]	mode	sl_counter_number_t for possible values
[in]	counter_number	sl_config_timer_mode_t for possible values
[in]	count_value	*(uint32_t), pointer to the variable to store count value

- Pre-conditions:
  - sl_si91x_config_timer_init();
- l_si91x_config_timer_set_configuration();

Returns

- status 0 if successful, else errorcode SL_STATUS_INVALID_MODE (0x0024) - 'mode' parameter value is invalid.
  - SL_STATUS_INVALID_PARAMETER 0x0021) - 'counter_number' parameter value is invalid.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter 'count_value' is null pointer.
  - SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line `486` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h`

sl_si91x_config_timer_reset_counter

```
sl_status_t sl_si91x_config_timer_reset_counter (sl_counter_number_t counter_number)
```

Soft reset Config-timer counter.

Parameters

[in]	counter_number	sl_counter_number_t for possible values

- Pre-conditions:

sl_si91x_config_timer_init();

- l_si91x_config_timer_set_configuration();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_INVALID_PARAMETER 0x0021) - 'counter_number' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line 507 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_start_on_software_trigger

```
sl_status_t sl_si91x_config_timer_start_on_software_trigger (sl_counter_number_t counter_number)
```

Start config-timer counter by software trigger.

Parameters

[in]	counter_number	sl_counter_number_t for possible values

- Pre-conditions:
  - sl_si91x_config_timer_init();
- l_si91x_config_timer_set_configuration(),keep software trigger disable here

Returns

- status 0 if successful, else error-code as follow SL_STATUS_INVALID_PARAMETER 0x0021) - 'counter_number' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line 526 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_select_action_event

```
sl_status_t sl_si91x_config_timer_select_action_event (sl_config_timer_action_t action, sl_config_timer_event_t
select_event_counter0, sl_config_timer_event_t select_event_counter1)
```

Select external input event for triggering selected timer-action such as start, stop, continue, halt, increment, capture, interrupt and output.

Parameters

[in]	action	sl_config_timer_action_t for possible values
[in]	select_event_counter0	sl_config_timer_event_t for possible values, (selects input event for triggering counter-0 action)
[in]	select_event_counter1	sl_config_timer_event_t for possible values, (selects input event for triggering counter-1 action)

- Pre-conditions:
  - sl_si91x_config_timer_init();
- sl_si91x_config_timer_set_configuration(), keep software trigger disable here
- sl_si91x_config_timer_register_callback(), keep event interrupt flag enable for respective counter

Returns

-

status 0 if successful, else error-code as follow SL_STATUS_INVALID_PARAMETER 0x0021) - Selected input event or action parameter value is invalid.
- SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line 552 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_configure_action_event

```
sl_status_t sl_si91x_config_timer_configure_action_event (sl_config_action_event_t *event_config_handle)
```

Configure external input-event's AND-event and OR-event for triggering selected timer-action such as start, stop, continue, halt, increment, capture, interrupt and output.

Parameters

[in]	event_config_handle	Pointer to CT configure action events structure sl_config_action_event_t

- Pre-conditions:
  - sl_si91x_config_timer_init();
- sl_si91x_config_timer_set_configuration(), keep software trigger disable here sl_si91x_config_timer_register_callback(), keep event interrupt flag enable for respective counter sl_si91x_config_timer_select_action_event(), first selects the input event for respective action for respective counter

Returns

- status 0 if successful, else error-code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer.
  - SL_STATUS_INVALID_PARAMETER 0x0021) - and-event or or-event or event-valid-bits value is invalid.
  - SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line 582 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_register_callback

```
sl_status_t sl_si91x_config_timer_register_callback (sl_config_timer_callback_t on_config_timer_callback, void *callback_flag_value, sl_config_timer_interrupt_flags_t *interrupt_flags)
```

Register callback of timer interrupt and enabling respective interrupts as per selected interrupt flag.

Parameters

[in]	on_config_timer_callback	(function pointer) Callback function pointer to be called when timer interrupt occurred sl_config_timer_callback_t
[in]	callback_flag_value	(void *) pointer to interrupt flag value variable sl_config_timer_callback_t
[in]	interrupt_flags	pointer to interrupt flags structure sl_config_timer_interrupt_flags_t

- Pre-conditions:
  - sl_si91x_config_timer_init();
- sl_si91x_config_timer_set_configuration(), keep software trigger disable here sl_si91x_config_timer_unregister_timeout_callback(), if already registered for any interrupt

Returns

- status 0 if successful, else error-code as follow SL_STATUS_NULL_POINTER (0x0022) - parameter is a null pointer.
  - SL_STATUS_BUSY (0x0004) - The callback is already registered, unregister previous callback before registering new one.
  - SL_STATUS_OK (0x0000) - Successfully registered timer timer-out callback.

Definition at line 609 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_unregister_callback

```
sl_status_t sl_si91x_config_timer_unregister_callback (sl_config_timer_interrupt_flags_t *interrupt_flags)
```

Parameters

N/A	interrupt_flags	

Definition at line 628 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_resume_halt_event

```
sl_status_t sl_si91x_config_timer_resume_halt_event (sl_counter_number_t counter_number)
```

Resume halt operation of Config-timer counter.

Parameters

[in]	counter_number	sl_counter_number_t for possible values

- Pre-conditions:
  - sl_si91x_config_timer_init();
- l_si91x_config_timer_set_configuration();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_INVALID_PARAMETER 0x0021) - 'counter_number' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line 647 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_read_capture

```
sl_status_t sl_si91x_config_timer_read_capture (sl_counter_number_t counter_number, uint16_t *capture_value)
```

Get Config-timer counter count value on occurrence of capture event occurrence.

Parameters

[in]	counter_number	sl_counter_number_t for possible values
[in]	capture_value	(uint16_t*), pointer to the variable to store count value at capture event.

- Pre-conditions:
  - sl_si91x_config_timer_init();
- l_si91x_config_timer_set_configuration();
- sl_si91x_config_timer_select_action_event(), first select events for capture action

Returns

- status 0 if successful, else error-code as follow SL_STATUS_NULL_POINTER (0x0022) - The parameter 'capture_value' is null pointer.
  - SL_STATUS_INVALID_PARAMETER 0x0021) - 'counter_number' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line 671 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_set_counter_sync

```
sl_status_t sl_si91x_config_timer_set_counter_sync (sl_counter_number_t counter_number, uint8_t sync_counter_value)
```

Sync counter output with other channels, as per sync_counter_value.

Parameters

[in]	counter_number	sl_counter_number_t for possible values
[in]	sync_counter_value	(uint8_t)

- Pre-conditions:
  - sl_si91x_config_timer_init();
- l_si91x_config_timer_set_configuration();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_INVALID_PARAMETER 0x0021) - 'counter_number' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line `691` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h`

sl_si91x_config_timer_set_output_adc_pin

```
sl_status_t sl_si91x_config_timer_set_output_adc_pin (uint8_t pin1, uint8_t pin2)
```

Sync counter output with other channels.

Parameters

[in]	pin1	sl_counter_number_t for possible values
[in]	pin2	(uint8_t)

- Pre-conditions:
  - sl_si91x_config_timer_init();
- l_si91x_config_timer_set_configuration();

Returns

- status 0 if successful, else error-code as follow SL_STATUS_INVALID_PARAMETER 0x0021) - 'counter_number' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Success, count-value is set properly

Definition at line `711` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h`

sl_si91x_config_timer_set_wfg_compare_values

```
sl_status_t sl_si91x_config_timer_set_wfg_compare_values (sl_counter_number_t counter_number,
sl_config_timer_ocu_params_t *ocu_params)
```

Parameters

N/A	counter_number	
N/A	ocu_params	

Definition at line 714 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_deinit

```
void sl_si91x_config_timer_deinit (void)
```

De-initialize config-timer by disabling its clock.

Parameters

[in]		

Returns

- none

Definition at line 723 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_si91x_config_timer_get_version

```
sl_config_timer_version_t sl_si91x_config_timer_get_version (void)
```

Get the release version of the Config-timer.

Parameters

[in]		

Returns

- (sl_config_version_t) type structure

Definition at line 732 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

Macro Definition Documentation

CT

```
#define CT
```

Value:

```
CT0
```

pointer to CT base address

Definition at line 51 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_config_t

Structure to hold the parameters of timer configurations.

Public Attributes

boolean_t	is_counter_mode_32bit_enabled CT mode, sl_config_timer_mode_t for its values.
boolean_t	is_counter0_soft_reset_enabled counter-0 soft reset, true to enable & false to disable it
boolean_t	is_counter0_periodic_enabled counter-0 periodic mode, true to enable & false to disable it
boolean_t	is_counter0_trigger_enabled counter-0 software trigger, true to enable & false to disable it
boolean_t	is_counter0_sync_trigger_enabled counter-0 sync trigger, true to enable & false to disable it
boolean_t	is_counter0_buffer_enabled counter-0 buffer, true to enable & false to disable it
boolean_t	is_counter1_soft_reset_enabled counter-1 soft reset, true to enable & false to disable it
boolean_t	is_counter1_periodic_enabled counter-1 periodic mode, true to enable & false to disable it
boolean_t	is_counter1_trigger_enabled counter-1 software trigger, true to enable & false to disable it
boolean_t	is_counter1_sync_trigger_enabled counter-1 sync trigger, true to enable & false to disable it
boolean_t	is_counter1_buffer_enabled counter-1 buffer, true to enable & false to disable it
uint8_t	counter0_direction counter-0 direction sl_config_timer_counter0_direction_t for possible values
uint8_t	counter1_direction counter-1 direction sl_config_timer_counter1_direction_t for possible values

Public Attribute Documentation

is_counter_mode_32bit_enabled

```
boolean_t sl_config_timer_config_t::is_counter_mode_32bit_enabled
```

CT mode, sl_config_timer_mode_t for its values.

Definition at line 209 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_soft_reset_enabled

```
boolean_t sl_config_timer_config_t::is_counter0_soft_reset_enabled
```

counter-0 soft reset, true to enable & false to disable it

Definition at line 210 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_periodic_enabled

```
boolean_t sl_config_timer_config_t::is_counter0_periodic_enabled
```

counter-0 periodic mode, true to enable & false to disable it

Definition at line 211 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_trigger_enabled

```
boolean_t sl_config_timer_config_t::is_counter0_trigger_enabled
```

counter-0 software trigger, true to enable & false to disable it

Definition at line 212 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_sync_trigger_enabled

```
boolean_t sl_config_timer_config_t::is_counter0_sync_trigger_enabled
```

counter-0 sync trigger, true to enable & false to disable it

Definition at line 213 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_buffer_enabled

```
boolean_t sl_config_timer_config_t::is_counter0_buffer_enabled
```

counter-0 buffer, true to enable & false to disable it

Definition at line 214 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_soft_reset_enabled

```
boolean_t sl_config_timer_config_t::is_counter1_soft_reset_enabled
```

counter-1 soft reset, true to enable & false to disable it

Definition at line 215 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_periodic_enabled

```
boolean_t sl_config_timer_config_t::is_counter1_periodic_enabled
```

counter-1 periodic mode, true to enable & false to disable it

Definition at line 216 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_trigger_enabled

```
boolean_t sl_config_timer_config_t::is_counter1_trigger_enabled
```

counter-1 software trigger, true to enable & false to disable it

Definition at line 217 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_sync_trigger_enabled

```
boolean_t sl_config_timer_config_t::is_counter1_sync_trigger_enabled
```

counter-1 sync trigger, true to enable & false to disable it

Definition at line 218 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_buffer_enabled

```
boolean_t sl_config_timer_config_t::is_counter1_buffer_enabled
```

counter-1 buffer, true to enable & false to disable it

Definition at line 219 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

counter0_direction

```
uint8_t sl_config_timer_config_t::counter0_direction
```

counter-0 direction sl_config_timer_counter0_direction_t for possible values

Definition at line 220 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

counter1_direction

```
uint8_t sl_config_timer_config_t::counter1_direction
```

counter-1 direction sl_config_timer_counter1_direction_t for possible values

Definition at line 221 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_ocu_config_t

Structure to hold the parameters of timer output compare unit (OCU) configurations.

Public Attributes

boolean_t	is_counter0_ocu_output_enabled true to enable OCU output of counter-0, false to disable it
boolean_t	is_counter0_ocu_dma_enabled true to enable OCU DMA support of counter-0, false to disable it
boolean_t	is_counter0_ocu_8bit_mode_enabled true to enable OCU 8-bit mode of counter-0, false to disable it
boolean_t	is_counter0_ocu_sync_enabled true to enable counter-0 output sync with other channel, false to disable it
boolean_t	is_counter1_ocu_output_enabled true to enable OCU output of counter-1, false to disable it
boolean_t	is_counter1_ocu_dma_enabled true to enable OCU DMA support of counter-1, false to disable it
boolean_t	is_counter1_ocu_mode_enabled true to enable OCU 8-bit mode of counter-1, false to disable it
boolean_t	is_counter1_ocu_sync_enabled true to enable counter-1 output sync with other channel, false to disable it
boolean_t	is_counter0_toggle_output_high_enabled true to enable counter-0 output toggle high, false to disable it
boolean_t	is_counter0_toggle_output_low_enabled true to enable counter-0 output toggle high, false to disable it
boolean_t	is_counter1_toggle_output_high_enabled true to enable counter-1 output toggle high, false to disable it
boolean_t	is_counter1_toggle_output_low_enabled true to enable counter-1 output toggle high, false to disable it

Public Attribute Documentation

is_counter0_ocu_output_enabled

> boolean_t sl_config_timer_ocu_config_t::is_counter0_ocu_output_enabled

true to enable OCU output of counter-0, false to disable it

Definition at line 226 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_ocu_dma_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter0_ocu_dma_enabled

true to enable OCU DMA support of counter-0, false to disable it

Definition at line 227 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_ocu_8bit_mode_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter0_ocu_8bit_mode_enabled

true to enable OCU 8-bit mode of counter-0, false to disable it

Definition at line 228 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_ocu_sync_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter0_ocu_sync_enabled

true to enable counter-0 output sync with other channel, false to disable it

Definition at line 230 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_ocu_output_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter1_ocu_output_enabled

true to enable OCU output of counter-1, false to disable it

Definition at line 231 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_ocu_dma_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter1_ocu_dma_enabled

true to enable OCU DMA support of counter-1, false to disable it

Definition at line 232 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_ocu_mode_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter1_ocu_mode_enabled

true to enable OCU 8-bit mode of counter-1, false to disable it

Definition at line 233 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_ocu_sync_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter1_ocu_sync_enabled

true to enable counter-1 output sync with other channel, false to disable it

Definition at line 235 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_toggle_output_high_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter0_toggle_output_high_enabled

true to enable counter-0 output toggle high, false to disable it

Definition at line 237 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_toggle_output_low_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter0_toggle_output_low_enabled

true to enable counter-0 output toggle high, false to disable it

Definition at line 238 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_toggle_output_high_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter1_toggle_output_high_enabled

true to enable counter-1 output toggle high, false to disable it

Definition at line 240 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_toggle_output_low_enabled

boolean_t sl_config_timer_ocu_config_t::is_counter1_toggle_output_low_enabled

true to enable counter-1 output toggle high, false to disable it

Definition at line 241 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_ocu_control_t

Structure to hold the parameters of timer OCU control parameters like compare values, DMA state.

Public Attributes

boolean_t	is_counter_number_1 CT counter number sl_counter_number_t.
boolean_t	is_dma_state_enabled DMA state for counter, true to enable and false to disable it.
sl_config_timer_ocu_params_t *	params pointer to OCU control parameters structure
sl_config_timer_pwm_callback_t *	callback pointer to MCPWM callback structure

Public Attribute Documentation

is_counter_number_1

```
boolean_t sl_config_timer_ocu_control_t::is_counter_number_1
```

CT counter number sl_counter_number_t.

Definition at line 246 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_dma_state_enabled

```
boolean_t sl_config_timer_ocu_control_t::is_dma_state_enabled
```

DMA state for counter, true to enable and false to disable it.

Definition at line 247 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

params

```
sl_config_timer_ocu_params_t* sl_config_timer_ocu_control_t::params
```

pointer to OCU control parameters structure

Definition at line 248 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

callback

> sl_config_timer_pwm_callback_t* sl_config_timer_ocu_control_t::callback

pointer to MCPWM callback structure

Definition at line 249 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_action_event_t

Structure to hold the parameters of timer action's AND-event & OR-event configurations.

Public Attributes

uint8_t	action CT action sl_config_timer_action_t.
uint8_t	and_event_counter0 AND-event for counter-0 actionsl_config_timer_event_t.
uint8_t	or_event_counter0 OR-event for counter-0 actionsl_config_timer_event_t.
uint8_t	and_event_valid_bits_counter0 valid bits for counter-0 action AND-event, possible values 0 to 16
uint8_t	or_event_valid_bits_counter0 valid bits for counter-0 action OR-event, possible values 0 to 16
uint8_t	and_event_counter1 AND-event for counter-1 action sl_config_timer_event_t.
uint8_t	or_event_counter1 OR-event for counter-0 action sl_config_timer_event_t.
uint8_t	and_event_valid_bits_counter1 valid bits for counter-0 action AND-event, possible values 0 to 16
uint8_t	or_event_valid_bits_counter1 valid bits for counter-0 action OR-event, possible values 0 to 16

Public Attribute Documentation

action

```
uint8_t sl_config_action_event_t::action
```

CT action sl_config_timer_action_t.

Definition at line 254 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

and_event_counter0

```
uint8_t sl_config_action_event_t::and_event_counter0
```

AND-event for counter-0 actionsl_config_timer_event_t.

Definition at line 255 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

or_event_counter0

```
uint8_t sl_config_action_event_t::or_event_counter0
```

OR-event for counter-0 actionsl_config_timer_event_t.

Definition at line 256 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

and_event_valid_bits_counter0

```
uint8_t sl_config_action_event_t::and_event_valid_bits_counter0
```

valid bits for counter-0 action AND-event, possible values 0 to 16

Definition at line 257 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

or_event_valid_bits_counter0

```
uint8_t sl_config_action_event_t::or_event_valid_bits_counter0
```

valid bits for counter-0 action OR-event, possible values 0 to 16

Definition at line 258 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

and_event_counter1

```
uint8_t sl_config_action_event_t::and_event_counter1
```

AND-event for counter-1 action sl_config_timer_event_t.

Definition at line 259 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

or_event_counter1

```
uint8_t sl_config_action_event_t::or_event_counter1
```

OR-event for counter-0 action sl_config_timer_event_t.

Definition at line 260 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

and_event_valid_bits_counter1

```
uint8_t sl_config_action_event_t::and_event_valid_bits_counter1
```

valid bits for counter-0 action AND-event, possible values 0 to 16

Definition at line 261 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

or_event_valid_bits_counter1

```
uint8_t sl_config_action_event_t::or_event_valid_bits_counter1
```

valid bits for counter-0 action OR-event, possible values 0 to 16

Definition at line `262` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h`

sl_config_timer_interrupt_flags_t

Structure to hold various interrupt flags of config-timer.

Public Attributes

boolean_t	is_counter0_event_interrupt_enabled true to enable counter-0 interrupt on event occurrence, false to disable it
boolean_t	is_counter0_fifo_full_interrupt_enabled true to enable counter-0 interrupt on FIFO full, false to disable it
boolean_t	is_counter0_hit_zero_interrupt_enabled true to enable counter-0 interrupt on zero count value, false to disable it
boolean_t	is_counter0_hit_peak_interrupt_enabled true to enable counter-0 interrupt on match-value, false to disable it
boolean_t	is_counter1_event_interrupt_enabled true to enable counter-1 interrupt on event occurrence, false to disable it
boolean_t	is_counter1_fifo_full_interrupt_enabled true to enable counter-1 interrupt on FIFO full, false to disable it
boolean_t	is_counter1_hit_zero_interrupt_enabled true to enable counter-1 interrupt on zero count value, false to disable it
boolean_t	is_counter1_hit_peak_interrupt_enabled true to enable counter-1 interrupt on match-value, false to disable it

Public Attribute Documentation

is_counter0_event_interrupt_enabled

> boolean_t sl_config_timer_interrupt_flags_t::is_counter0_event_interrupt_enabled

true to enable counter-0 interrupt on event occurrence, false to disable it

Definition at line 268 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_fifo_full_interrupt_enabled

> boolean_t sl_config_timer_interrupt_flags_t::is_counter0_fifo_full_interrupt_enabled

true to enable counter-0 interrupt on FIFO full, false to disable it

Definition at line 270 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_hit_zero_interrupt_enabled

boolean_t sl_config_timer_interrupt_flags_t::is_counter0_hit_zero_interrupt_enabled

true to enable counter-0 interrupt on zero count value, false to disable it

Definition at line 272 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter0_hit_peak_interrupt_enabled

boolean_t sl_config_timer_interrupt_flags_t::is_counter0_hit_peak_interrupt_enabled

true to enable counter-0 interrupt on match-value, false to disable it

Definition at line 274 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_event_interrupt_enabled

boolean_t sl_config_timer_interrupt_flags_t::is_counter1_event_interrupt_enabled

true to enable counter-1 interrupt on event occurrence, false to disable it

Definition at line 276 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_fifo_full_interrupt_enabled

boolean_t sl_config_timer_interrupt_flags_t::is_counter1_fifo_full_interrupt_enabled

true to enable counter-1 interrupt on FIFO full, false to disable it

Definition at line 278 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_hit_zero_interrupt_enabled

boolean_t sl_config_timer_interrupt_flags_t::is_counter1_hit_zero_interrupt_enabled

true to enable counter-1 interrupt on zero count value, false to disable it

Definition at line 280 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

is_counter1_hit_peak_interrupt_enabled

boolean_t sl_config_timer_interrupt_flags_t::is_counter1_hit_peak_interrupt_enabled

true to enable counter-1 interrupt on match-value, false to disable it

Definition at line 282 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

sl_config_timer_version_t

Structure to hold the versions of peripheral API.

Public Attributes

uint8_t	release Release version number.
uint8_t	major sqa-version number
uint8_t	minor dev-version number

Public Attribute Documentation

release

```
uint8_t sl_config_timer_version_t::release
```

Release version number.

Definition at line 287 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

major

```
uint8_t sl_config_timer_version_t::major
```

sqa-version number

Definition at line 288 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

minor

```
uint8_t sl_config_timer_version_t::minor
```

dev-version number

Definition at line 289 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_config_timer.h

Direct Memory Access

Direct Memory Access

Modules

- sl_dma_callback_t
- sl_dma_init_t
- sl_channel_data_t
- sl_dma_xfer_t

Enumerations

enum sl_dma_transfer_type_t {

SL_DMA_MEMORY_TO_MEMORY
SL_DMA_MEMORY_TO_PERIPHERAL
SL_DMA_PERIPHERAL_TO_MEMORY

}

Enumeration holds transfer types of DMA.

enum sl_dma_transfer_mode_t {

SL_DMA_BASIC_MODE = UDMA_MODE_BASIC
SL_DMA_PINGPONG_MODE = UDMA_MODE_PINGPONG

}

Enumeration holds transfer modes of DMA.

enum sl_dma_peripheral_ack_t {

SL_USART0_ACK = 0×01
SL_UART1_ACK = 0×02
SL_UART3_ACK = 0×03
SL_SSI_SLAVE_ACK = 0×04
SL_SSI_MASTER_ACK = 0×05
SL_SSI1_SLAVE_ACK = 0×06
SL_I2C_ACK = 0×07

}

Enumeration holds peripheral ACK signals to DMA.

enum sl_dma_transfer_size_t {

SL_TRANSFER_SIZE_32 = SRC_SIZE_32
SL_TRANSFER_SIZE_16 = SRC_SIZE_16
SL_TRANSFER_SIZE_8 = SRC_SIZE_8

}

Enumeration holds DMA transfer sizes.

enum sl_dma_transfer_inc_t {

SL_TRANSFER_SRC_INC_32 = SRC_INC_32
SL_TRANSFER_SRC_INC_16 = SRC_INC_16
SL_TRANSFER_SRC_INC_8 = SRC_INC_8
SL_TRANSFER_SRC_INC_NONE = SRC_INC_NONE
SL_TRANSFER_DST_INC_32 = DST_INC_32
SL_TRANSFER_DST_INC_16 = DST_INC_16
SL_TRANSFER_DST_INC_8 = DST_INC_8
SL_TRANSFER_DST_INC_NONE = DST_INC_NONE

}

Enumeration holds DMA transfer address increment for source and destination.

enum sl_dma_callback_code_t {

SL_DMA_TRANSFER_DONE_CB = 1
SL_DMA_ERROR_CB = 2

}

Enumeration holds 8-bit codes which are used by callback_type in sl_si91x_dma_unregister_callbacks function.

Typedefs

typedef void(* sl_dma_transfer_complete)(uint32_t channel, void *data)

Typedef for user supplied callback function which is called when a DMA transfer completes.

typedef void(* sl_dma_error)(uint32_t channel, void *data)

Typedef for user supplied callback function which is called when a DMA error occurs.

Variables

sl_channel_data_t sl_channel_allocation_data_t

DMA driver channel allocator.

Functions

sl_status_t sl_si91x_dma_init(sl_dma_init_t *dma_init)

This function initializes DMA peripheral by enabling DMA clock and clearing DMA interrupts.

sl_status_t sl_si91x_dma_deinit(uint32_t dma_number)

This function deinitialize DMA module by disabling peripheral clock and interrupts, if there is no ongoing DMA transfer.

sl_status_t sl_si91x_dma_allocate_channel(uint32_t dma_number, uint32_t *channel_no, uint32_t priority)

This function check the available DMA channel and allocate the channel.

sl_status_t sl_si91x_dma_deallocate_channel(uint32_t dma_number, uint32_t channel_no)

sl_status_t sl_si91x_dma_register_callbacks(uint32_t dma_number, uint32_t channel_no, sl_dma_callback_t *callback_t)

sl_status_t sl_si91x_dma_unregister_callbacks(uint32_t dma_number, uint32_t channel_no, uint8_t callback_type)

sl_status_t sl_si91x_dma_transfer(uint32_t dma_number, uint32_t channel_no, sl_dma_xfer_t *dma_transfer_t)

sl_status_t sl_si91x_dma_simple_transfer(uint32_t dma_number, uint32_t channel_no, void *src_addr, void *dst_addr, uint32_t data_size)

sl_status_t sl_si91x_dma_stop_transfer(uint32_t dma_number, uint32_t channel_no)

sl_status_t sl_si91x_dma_channel_status_get(uint32_t dma_number, uint32_t channel_no)

This function returns the channel status (i.e) whether channel is allocated/busy/idle.

sl_status_t sl_si91x_dma_channel_enable(uint32_t dma_number, uint32_t channel_no)
This function enable DMA channel.

sl_status_t sl_si91x_dma_channel_disable(uint32_t dma_number, uint32_t channel_no)
This function disable DMA channel.

sl_status_t sl_si91x_dma_enable(uint32_t dma_number)
This function enable DMA peripheral.

Macros

#define SL_STATUS_DMA_CHANNEL_ALLOCATED (sl_status_t)0×45

#define SL_STATUS_DMA_NO_CHANNEL_AVAILABLE (sl_status_t)0×46

#define SL_STATUS_DMA_CHANNEL_ALREADY_UNALLOCATED (sl_status_t)0X47

#define SL_STATUS_DMA_CHANNEL_UNALLOCATED (sl_status_t)0X48

#define SL_CHANNEL_COUNT 32

#define ALTERNATE_DESCRIPTOR_DISABLE 0

#define ALTERNATE_DESCRIPTOR_ENABLE 1

#define BURST_REQUEST_ENABLE 1

#define BURST_REQUEST_DISABLE 0

#define CHANNEL_PRIO_DISABLE 0

#define CHANNEL_PRIO_ENABLE 1

#define PERIPHERAL_ACK_DISABLE 0

#define PERIPHERAL_REQUEST_DISABLE 0

#define PERIPHERAL_REQUEST_ENABLE 1

#define REQUEST_MASK_DISABLE 0

#define NEXT_BURST_ENABLE undefined

#define NEXT_BURST_DISABLE 0

#define SOURCE_PROTECT_CONTROL_DISABLE 0×000

#define DESTINATION_PROTECT_CONTROL_DISABLE 0×000

Enumeration Documentation

sl_dma_transfer_type_t

> sl_dma_transfer_type_t

Enumeration holds transfer types of DMA.

Enumerator	
SL_DMA_MEMORY_TO_MEMORY	Memory to memory transfer.

SL_DMA_MEMORY_TO_PERIPHERAL	Memory to peripheral transfer.
SL_DMA_PERIPHERAL_TO_MEMORY	Peripheral to memory transfer.

Definition at line 101 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_dma_transfer_mode_t

> sl_dma_transfer_mode_t

Enumeration holds transfer modes of DMA.

Enumerator	
SL_DMA_BASIC_MODE	Basic DMA mode.
SL_DMA_PINGPONG_MODE	Ping pong mode.

Definition at line 108 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_dma_peripheral_ack_t

> sl_dma_peripheral_ack_t

Enumeration holds peripheral ACK signals to DMA.

Enumerator	
SL_USART0_ACK	ACK code for USART0.
SL_UART1_ACK	ACK code for UART1.
SL_UART3_ACK	ACK code for UART3.
SL_SSI_SLAVE_ACK	ACK code for SSI slave.
SL_SSI_MASTER_ACK	ACK code for SSI master.
SL_SSI1_SLAVE_ACK	ACK code for SSI1.
SL_I2C_ACK	ACK code for I2C.

Definition at line 114 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_dma_transfer_size_t

> sl_dma_transfer_size_t

Enumeration holds DMA transfer sizes.

Enumerator	
SL_TRANSFER_SIZE_32	4 bytes transfer size
SL_TRANSFER_SIZE_16	2 bytes transfer size
SL_TRANSFER_SIZE_8	1 bytes transfer size

Definition at line 125 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_dma_transfer_inc_t

> sl_dma_transfer_inc_t

Enumeration holds DMA transfer address increment for source and destination.

Enumerator	
SL_TRANSFER_SRC_INC_32	4 bytes source address increment
SL_TRANSFER_SRC_INC_16	2 bytes source address increment
SL_TRANSFER_SRC_INC_8	1 byte source address increment
SL_TRANSFER_SRC_INC_NONE	No source address increment.
SL_TRANSFER_DST_INC_32	4 bytes destination address increment
SL_TRANSFER_DST_INC_16	2 bytes destination address increment
SL_TRANSFER_DST_INC_8	1 byte destination address increment
SL_TRANSFER_DST_INC_NONE	No destination address increment.

Definition at line `132` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h`

sl_dma_callback_code_t

```
sl_dma_callback_code_t
```

Enumeration holds 8-bit codes which are used by callback_type in sl_si91x_dma_unregister_callbacks function.

Enumerator	
SL_DMA_TRANSFER_DONE_CB	8-bit code for transfer complete callback
SL_DMA_ERROR_CB	8-bit code for error callback

Definition at line `145` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h`

Typedef Documentation

sl_dma_transfer_complete

```
typedef void(* sl_dma_transfer_complete) (uint32_t channel, void *data) )(uint32_t channel, void *data)
```

Typedef for user supplied callback function which is called when a DMA transfer completes.

Parameters

[in]	channel_no	DMA channel number
[in]	*data	An extra parameter for user application

Definition at line `89` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h`

sl_dma_error

```
typedef void(* sl_dma_error) (uint32_t channel, void *data) )(uint32_t channel, void *data)
```

Typedef for user supplied callback function which is called when a DMA error occurs.

Parameters

[in]	channel_no	DMA channel number
[in]	*data	An extra parameter for user application

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

Variable Documentation

sl_channel_allocation_data_t

```
sl_channel_data_t sl_channel_allocation_data_t[2][SL_CHANNEL_COUNT]
```

DMA driver channel allocator.

Definition at line 183 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

Function Documentation

sl_si91x_dma_init

```
sl_status_t sl_si91x_dma_init (sl_dma_init_t *dma_init)
```

This function initializes DMA peripheral by enabling DMA clock and clearing DMA interrupts.

Parameters

[in]	dma_init	dma initialization structure, dma_init->dma_number - 0->UDMA0, 1->UDMA1

- Pre-conditions:
  - none

Returns

- Initialization status SL_STATUS_OK - Initialization success SL_STATUS_NOT_INITIALIZED - Initialization fail

Definition at line 206 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_deinit

```
sl_status_t sl_si91x_dma_deinit (uint32_t dma_number)
```

This function deinitialize DMA module by disabling peripheral clock and interrupts, if there is no ongoing DMA transfer.

Parameters

[in]	dma_number	0->UDMA0, 1->UDMA1

- Pre-conditions:
  - sl_si91x_dma_init

Returns

- deinitialization status SL_STATUS_OK - Deinit success SL_STATUS_BUSY - Cannot deinit the peripheral due to an ongoing transfer SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized

Definition at line 222 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_allocate_channel

```
sl_status_t sl_si91x_dma_allocate_channel (uint32_t dma_number, uint32_t *channel_no, uint32_t priority)
```

This function check the available DMA channel and allocate the channel.

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1
[in]	channel_no	Address of channel number (1-32).
[in]	priority	channel priority.

This function also set the priority of allocated channel and assign the channel number to *channel_no variable. If no channel is available it will return SL_DMA_NO_CHANNEL_AVAILABLE. note: User can also initialize desired channel number and this API checks whether desired channel is available and allocates the channel if available. If user want driver to allocate the available channel, channel_no should be initialized to 0.

- Pre-conditions:
  - sl_si91x_dma_init

Returns

- channel allocation status SL_STATUS_OK - Channel allocated SL_STATUS_DMA_NO_CHANNEL_AVAILABLE - All DMA channels are allocated SL_STATUS_DMA_CHANNEL_ALLOCATED - Desired channel is already allocated SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized SL_STATUS_INVALID_PARAMETER - Channel no is invalid

Definition at line `244` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h`

sl_si91x_dma_deallocate_channel

> sl_status_t sl_si91x_dma_deallocate_channel (uint32_t dma_number, uint32_t channel_no)

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1
[in]	channel_no	channel number(1-32).

> This function deallocate DMA channel if there is no ongoing transfer on channel
>
> @pre Pre-conditions:

- sl_si91x_dma_init

sl_si91x_dma_allocate_channel

Returns

- channel deallocation status SL_STATUS_OK - Channel deallocated SL_STATUS_BUSY - Cannot deallocate channel due to an ongoing transfer SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized SL_STATUS_INVALID_PARAMETER - Channel no is invalid

Definition at line `264` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h`

sl_si91x_dma_register_callbacks

> sl_status_t sl_si91x_dma_register_callbacks (uint32_t dma_number, uint32_t channel_no, sl_dma_callback_t *callback_t)

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1
[in]	channel_no	channel number(1-32).

[in]	callback_t	structure containing callback functions.

This function register DMA callbacks (transfer complete & error)
User must update the sl_dma_callback_t structure and pass its address to this function.
User can have seperate callbacks for transfer complete and error for each channel

@pre Pre-conditions:

- sl_si91x_dma_init

sl_si91x_dma_allocate_channel

Returns

- channel deallocation status SL_STATUS_OK - Callback registered successfully SL_STATUS_INVALID_PARAMETER - Invalid channel number SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized

Definition at line 287 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_unregister_callbacks

sl_status_t sl_si91x_dma_unregister_callbacks (uint32_t dma_number, uint32_t channel_no, uint8_t callback_type)

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1
[in]	channel_no	channel number(1-32).
[in]	callback_type	Unregister the DMA callbacks based on the callback type (bit mapped to callbacks), (SL_DMA_TRANSFER_DONE_CB) - unregister transfer complete callback (SL_DMA_ERROR_CB) - unregister error callback (SL_DMA_TRANSFER_DONE_CB | SL_DMA_ERROR_CB) - unregister both transfer complete and error callback

This function unregister DMA callbacks (transfer complete & error)
User needs to update the 8-bit variable callback_type and pass to the function
Each bit is mapped to specific callback.

@pre Pre-conditions:

- sl_si91x_dma_init

sl_si91x_dma_allocate_channel

- sl_si91x_dma_register_callbacks

Returns

- channel deallocation status SL_STATUS_OK - Callback unregistered successfully SL_STATUS_INVALID_PARAMETER - Invalid channel number SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized

Definition at line 314 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_transfer

sl_status_t sl_si91x_dma_transfer (uint32_t dma_number, uint32_t channel_no, sl_dma_xfer_t *dma_transfer_t)

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1

[in]	channel_no	channel number(1-32).
[in]	dma_transfer_t	channel transfer data structure containing channel descriptor and other basic DMA parameters.

This function configure DMA channel descriptor and initiate DMA transfer
DMA primary descriptor is updated in this function and based on transfer mode
alternate descriptor is updated (only for ping pong mode). Also other DMA parameters
like peripheral ACK signal (for peripheral memory transfers), DMA priority etc. are
updated in this function.

@pre Pre-conditions:

- sl_si91x_dma_init

sl_si91x_dma_allocate_channel

- sl_si91x_dma_register_callbacks

Returns

- DMA transfer status SL_STATUS_OK - Transfer started successfully SL_STATUS_SUSPENDED - Transfer initialization fail SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized SL_STATUS_INVALID_PARAMETER - Channel no is invalid

Definition at line 341 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_simple_transfer

sl_status_t sl_si91x_dma_simple_transfer (uint32_t dma_number, uint32_t channel_no, void *src_addr, void *dst_addr, uint32_t data_size)

Parameters

N/A	dma_number	
N/A	channel_no	
N/A	src_addr	
N/A	dst_addr	
N/A	data_size	

This function configure DMA channel descriptor and initiate simple memory to memory
DMA transfer. User need to pass source address and destination address of transfer
along with transfer length in bytes.

@pre Pre-conditions:

- sl_si91x_dma_init

sl_si91x_dma_allocate_channel

- sl_si91x_dma_register_callbacks

```
@param[in] dma_number  dma_number  0->UDMA0, 1->UDMA1
@param[in] sl_channel_t  channel no(1-32)
@param[in] *src_addr     source address.
@param[in] *dst_addr    destination address.
@param[in] data_size     transfer size in bytes

@return DMA transfer status
      SL_STATUS_OK  - Transfer success
      SL_STATUS_SUSPENDED - Transfer initialization fail
      SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized
      SL_STATUS_INVALID_PARAMETER - Channel no is invalid
```

Definition at line 369 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_stop_transfer

> sl_status_t sl_si91x_dma_stop_transfer (uint32_t dma_number, uint32_t channel_no)

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1
[in]	channel_no	channel number(1-32).

> This function stop any active transfer on channel. If there is no active transfer on channel
> this function returns SL_DMA_CHANNEL_IDLE
>
> @pre Pre-conditions:

- sl_si91x_dma_init

sl_si91x_dma_allocate_channel

- sl_si91x_dma_simple_transfer/sl_si91x_dma_transfer

Returns

- DMA transfer status SL_STATUS_OK - Transfer stopped successfully SL_STATUS_IDLE - There is no active transfer on channel SL_STATUS_INVALID_PARAMETER - Invalid channel number SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized

Definition at line 396 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_channel_status_get

> sl_status_t sl_si91x_dma_channel_status_get (uint32_t dma_number, uint32_t channel_no)

This function returns the channel status (i.e) whether channel is allocated/busy/idle.

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1
[in]	channel_no	channel number(1-32).

- Pre-conditions:
  - sl_si91x_dma_init

Returns

- channel status SL_STATUS_IDLE - Channel is not allocated SL_STATUS_DMA_CHANNEL_ALREADY_ALLOCATED - Channel is already allocated and idle SL_STATUS_BUSY - Channel is allocated and busy SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized SL_STATUS_INVALID_PARAMETER - Channel no is invalid

Definition at line 415 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_channel_enable

> sl_status_t sl_si91x_dma_channel_enable (uint32_t dma_number, uint32_t channel_no)

This function enable DMA channel.

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1
[in]	channel_no	channel number(1-32).

- Pre-conditions:
  - sl_si91x_dma_init

Returns

- channel status SL_STATUS_OK - Channel enable success SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized SL_STATUS_INVALID_PARAMETER - Channel no is invalid

Definition at line 431 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_channel_disable

> sl_status_t sl_si91x_dma_channel_disable (uint32_t dma_number, uint32_t channel_no)

This function disable DMA channel.

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1
[in]	channel_no	channel number(1-32).

- Pre-conditions:
  - sl_si91x_dma_init

Returns

- channel status SL_STATUS_OK - Channel disable success SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized SL_STATUS_INVALID_PARAMETER - Channel no is invalid

Definition at line 447 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_si91x_dma_enable

> sl_status_t sl_si91x_dma_enable (uint32_t dma_number)

This function enable DMA peripheral.

Parameters

[in]	dma_number	dma_number 0->UDMA0, 1->UDMA1

- Pre-conditions:
  - sl_si91x_dma_init

Returns

- SL_STATUS_OK - Channel enable success SL_STATUS_NOT_INITIALIZED - DMA peripheral not initialized

Definition at line 460 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

Macro Definition Documentation

SL_STATUS_DMA_CHANNEL_ALLOCATED

```
#define SL_STATUS_DMA_CHANNEL_ALLOCATED
```

Value:

```
(sl_status_t)0x45
```

Definition at line 49 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

SL_STATUS_DMA_NO_CHANNEL_AVAILABLE

```
#define SL_STATUS_DMA_NO_CHANNEL_AVAILABLE
```

Value:

```
(sl_status_t)0x46
```

Definition at line 51 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

SL_STATUS_DMA_CHANNEL_ALREADY_UNALLOCATED

```
#define SL_STATUS_DMA_CHANNEL_ALREADY_UNALLOCATED
```

Value:

```
(sl_status_t)0X47
```

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

SL_STATUS_DMA_CHANNEL_UNALLOCATED

```
#define SL_STATUS_DMA_CHANNEL_UNALLOCATED
```

Value:

```
(sl_status_t)0X48
```

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

SL_CHANNEL_COUNT

```
#define SL_CHANNEL_COUNT
```

Value:

```
32
```

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

ALTERNATE_DESCRIPTOR_DISABLE

```
#define ALTERNATE_DESCRIPTOR_DISABLE
```

Value:

```
0
```

Definition at line 59 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

ALTERNATE_DESCRIPTOR_ENABLE

```
#define ALTERNATE_DESCRIPTOR_ENABLE
```

Value:

```
1
```

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

BURST_REQUEST_ENABLE

```
#define BURST_REQUEST_ENABLE
```

Value:

```
1
```

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

BURST_REQUEST_DISABLE

```
#define BURST_REQUEST_DISABLE
```

Value:

```
0
```

Definition at line 62 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

CHANNEL_PRIO_DISABLE

```
#define CHANNEL_PRIO_DISABLE
```

Value:

```
0
```

Definition at line 63 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

CHANNEL_PRIO_ENABLE

```
#define CHANNEL_PRIO_ENABLE
```

Value:

```
1
```

Definition at line 64 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

PERIPHERAL_ACK_DISABLE

```
#define PERIPHERAL_ACK_DISABLE
```

Value:

```
0
```

Definition at line 65 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

PERIPHERAL_REQUEST_DISABLE

```
#define PERIPHERAL_REQUEST_DISABLE
```

Value:

```
0
```

Definition at line 66 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

PERIPHERAL_REQUEST_ENABLE

```
#define PERIPHERAL_REQUEST_ENABLE
```

Value:

```
1
```

Definition at line 67 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

REQUEST_MASK_DISABLE

```
#define REQUEST_MASK_DISABLE
```

Value:

```
0
```

Definition at line 68 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

NEXT_BURST_ENABLE

```
#define NEXT_BURST_ENABLE
```

Value:

```
1 /* Force the channel to only respond to burst requests at
     the tail end of a scatter-gather transfer */
```

Definition at line 70 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

NEXT_BURST_DISABLE

```
#define NEXT_BURST_DISABLE
```

Value:

```
0
```

Definition at line 72 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

SOURCE_PROTECT_CONTROL_DISABLE

```
#define SOURCE_PROTECT_CONTROL_DISABLE
```

Value:

```
0×000
```

Definition at line 73 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

DESTINATION_PROTECT_CONTROL_DISABLE

```
#define DESTINATION_PROTECT_CONTROL_DISABLE
```

Value:

```
0×000
```

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_dma_callback_t

Structure holds Channel Callbacks.

Public Attributes

sl_dma_transfer_complete	transfer_complete_cb data transfer complete callback
sl_dma_error	error_cb Error callback.

Public Attribute Documentation

transfer_complete_cb

> sl_dma_transfer_complete sl_dma_callback_t::transfer_complete_cb

data transfer complete callback

Definition at line 152 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

error_cb

> sl_dma_error sl_dma_callback_t::error_cb

Error callback.

Definition at line 153 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_dma_init_t

Channel Initialization structure.

Public Attributes

uint32_t	dma_number 0 - UDMA0, 1 - UDMA1

Public Attribute Documentation

dma_number

```
uint32_t sl_dma_init_t::dma_number
```

0 - UDMA0, 1 - UDMA1

Definition at line 158 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_channel_data_t

channel allocation data

Public Attributes

uint32_t	priority Channel priority.
bool	allocated Channel allocation status.
sl_dma_callback_t	dma_callback_t Channel callback structure.
uint32_t	transfer_type DMA Transfer type.
uint32_t	transfer_mode DMA transfer mode.

Public Attribute Documentation

priority

```
uint32_t sl_channel_data_t::priority
```

Channel priority.

Definition at line 163 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

allocated

```
bool sl_channel_data_t::allocated
```

Channel allocation status.

Definition at line 164 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

dma_callback_t

```
sl_dma_callback_t sl_channel_data_t::dma_callback_t
```

Channel callback structure.

Definition at line 165 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

transfer_type

> uint32_t sl_channel_data_t::transfer_type

DMA Transfer type.

Definition at line 166 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

transfer_mode

> uint32_t sl_channel_data_t::transfer_mode

DMA transfer mode.

Definition at line 167 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

sl_dma_xfer_t

DMA transfer structure.

Public Attributes

uint32_t *	src_addr Source transfer address.
uint32_t *	dest_addr Destination transfer address.
uint32_t	src_inc Source address increment size.
uint32_t	dst_inc Destination address increment size.
uint32_t	xfer_size Transfer data size.
uint32_t	transfer_count Total transfer length.
uint8_t	transfer_type DMA transfer type.
uint8_t	dma_mode DMA transfer mode.
uint8_t	signal Peripheral signal which triggers DMA transfer (if 0, consider as software trigger)

Public Attribute Documentation

src_addr

```
uint32_t* sl_dma_xfer_t::src_addr
```

Source transfer address.

Definition at line 172 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

dest_addr

```
uint32_t* sl_dma_xfer_t::dest_addr
```

Destination transfer address.

Definition at line 173 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

src_inc

```
uint32_t sl_dma_xfer_t::src_inc
```

Source address increment size.

Definition at line 174 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

dst_inc

```
uint32_t sl_dma_xfer_t::dst_inc
```

Destination address increment size.

Definition at line 175 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

xfer_size

```
uint32_t sl_dma_xfer_t::xfer_size
```

Transfer data size.

Definition at line 176 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

transfer_count

```
uint32_t sl_dma_xfer_t::transfer_count
```

Total transfer length.

Definition at line 177 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

transfer_type

```
uint8_t sl_dma_xfer_t::transfer_type
```

DMA transfer type.

Definition at line 178 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

dma_mode

```
uint8_t sl_dma_xfer_t::dma_mode
```

DMA transfer mode.

Definition at line 179 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

signal

uint8_t sl_dma_xfer_t::signal

Peripheral signal which triggers DMA transfer (if 0, consider as software trigger)

Definition at line 180 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_dma.h

Disable UC Config

Disable UC Config

For the below listed peripherals, peripheral configuration is by default taken from the Universal Configurator (UC).

For disabling this behavior, undefine the respective peripheral "PERIPHERAL_UC" macro mentioned below.

- Calendar: CALENDAR_UC
- Config Timer: CONFIG_TIMER_UC
- Generic SPIC: GSPI_UC
- Serial Input-Output: SIO_UC
- Synchronous Serial Interface: SSI_UC
- USART: USART_UC
- Watchdog Timer: WDT_TIMER_UC

E-Fuse

E-Fuse

Modules

sl_efuse_version_t

Functions

sl_efuse_version_t	sl_si91x_efuse_get_version(void) To get the release, sqa and dev version of EFUSE.
sl_status_t	sl_si91x_efuse_enable_clock(void) This API Is Used to enable efuse clock.
sl_status_t	sl_si91x_efuse_disable_clock(void) This API Is Used to disable efuse clock.
sl_status_t	sl_si91x_efuse_init(void) This API is used to Initialize the EFUSE.
sl_status_t	sl_si91x_efuse_deinit(void) This API is used to Un-initialize the EFUSE.
sl_status_t	sl_si91x_efuse_set_address(uint16_t address) This API is used to set the eFUSE address for read and write operations.
sl_status_t	sl_si91x_efuse_get_address(uint16_t *read_address) This API is used to get the eFUSE address for read and write operations.
sl_status_t	sl_si91x_efuse_write_bit(uint16_t address, uint8_t bit_pos, uint32_t hold_time) This API is used to write the eFUSE data in the specified address.
sl_status_t	sl_si91x_efuse_memory_mapped_read_word(uint16_t address, uint16_t *read_word, uint32_t soc_clk) This API is used to Read the 1 word(16 bits) of data to EFUSE macro in memory mapped mode.
sl_status_t	sl_si91x_efuse_memory_mapped_read_byte(uint16_t address, uint8_t *read_byte, uint32_t soc_clk) This API is used to read the data from 32x8 byte eFUSE memory(OTP) in memory mapped mode.
sl_status_t	sl_si91x_efuse_fsm_read_byte(uint16_t address, uint8_t *read_byte, uint32_t soc_clk) This API is used to read the data from 32x8 byte eFUSE memory(OTP) in fsm mode.
__STATIC_INLINE void	sl_si91x_efuse_enable(void) This API is used to enable the EFUSE.
__STATIC_INLINE void	sl_si91x_efuse_disable(void) This API is used to Disable the EFUSE.

Function Documentation

sl_si91x_efuse_get_version

> sl_efuse_version_t sl_si91x_efuse_get_version (void)

To get the release, sqa and dev version of EFUSE.

Parameters

[in]		

Returns

- (sl_efuse_version_t) type structure

Definition at line 85 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_enable_clock

> sl_status_t sl_si91x_efuse_enable_clock (void)

This API Is Used to enable efuse clock.

Parameters

[in]		

Returns

- returns status 0 if successful, else error code.

Definition at line 93 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_disable_clock

> sl_status_t sl_si91x_efuse_disable_clock (void)

This API Is Used to disable efuse clock.

Parameters

[in]		

- sl_si91x_efuse_init()
- sl_si91x_efuse_set_address()
- sl_si91x_efuse_write_bit()
- sl_si91x_efuse_memory_mapped_read_byte()

Returns

- returns status 0 if successful, else error code.

Definition at line 109 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_init

> sl_status_t sl_si91x_efuse_init (void)

This API is used to Initialize the EFUSE.

Parameters

[in]		

This API first enables the efuse clock and then efuse. **Returns**

- returns status 0 if successful, else error code.

Definition at line 118 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_deinit

```
sl_status_t sl_si91x_efuse_deinit (void)
```

This API is used to Un-initialize the EFUSE.

Parameters

[in]		

This API first disables the efuse and then efuse clock.

- sl_si91x_efuse_init()
- sl_si91x_efuse_set_address()
- sl_si91x_efuse_write_bit()
- sl_si91x_efuse_memory_mapped_read_byte()

Returns

- returns status 0 if successful, else error code.

Definition at line 135 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_set_address

```
sl_status_t sl_si91x_efuse_set_address (uint16_t address)
```

This API is used to set the eFUSE address for read and write operations.

Parameters

[in]	address	- Holds the address at which the data has to be written in the efuse

- sl_si91x_efuse_init()

Returns

- returns status 0 if successful, else error code.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument

Definition at line 150 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_get_address

```
sl_status_t sl_si91x_efuse_get_address (uint16_t *read_address)
```

This API is used to get the eFUSE address for read and write operations.

Parameters

[in]	read_address	- Pointer holds the address at which the data has to be written in the efuse

- sl_si91x_efuse_init()
- sl_si91x_efuse_set_address()

Returns

- returns status 0 if successful, else error code as follow:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument

Definition at line 167 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_write_bit

```
sl_status_t sl_si91x_efuse_write_bit (uint16_t address, uint8_t bit_pos, uint32_t hold_time)
```

This API is used to write the eFUSE data in the specified address.

Parameters

[in]	address	- Holds the address at which the data has to be written in the efuse
[in]	bit_pos	- Variable that holds the position of bit on which the data will be placed
[in]	hold_time	- hold time in write bit

- sl_si91x_efuse_init()
- sl_si91x_efuse_set_address()

Returns

- returns status 0 if successful, else error code as follow.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line 185 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_memory_mapped_read_word

```
sl_status_t sl_si91x_efuse_memory_mapped_read_word (uint16_t address, uint16_t *read_word, uint32_t soc_clk)
```

This API is used to Read the 1 word(16 bits) of data to EFUSE macro in memory mapped mode.

Parameters

[in]	address	- Holds the address from where we are reading the 1 word of data
[in]	read_word	- Pointer that points to the data which is stored in the efuse
[in]	soc_clk	- It is input clock frequency

- sl_si91x_efuse_init()
- sl_si91x_efuse_set_address()
- sl_si91x_efuse_write_bit()

Returns

- returns status 0 if successful, else error code as follow.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line 209 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_memory_mapped_read_byte

```
sl_status_t sl_si91x_efuse_memory_mapped_read_byte (uint16_t address, uint8_t *read_byte, uint32_t soc_clk)
```

This API is used to read the data from 32x8 byte eFUSE memory(OTP) in memory mapped mode.

Parameters

[in]	address	- Holds the address from where we are reading the 1 byte of data
[in]	read_byte	- Pointer that points to the 8 bit data which is stored in the efuse
[in]	soc_clk	- It is input clock frequency

- sl_si91x_efuse_init()
- sl_si91x_efuse_set_address()
- sl_si91x_efuse_write_bit()

Returns

- returns status 0 if successful, else error code as follow.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
-
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line `233` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h`

sl_si91x_efuse_fsm_read_byte

```
sl_status_t sl_si91x_efuse_fsm_read_byte (uint16_t address, uint8_t *read_byte, uint32_t soc_clk)
```

This API is used to read the data from 32x8 byte eFUSE memory(OTP) in fsm mode.

Parameters

[in]	address	- Holds the address from where we are reading the 1 byte of data
[in]	read_byte	- Pointer that points to the 8 bit data which is stored in the efuse
[in]	soc_clk	- It is input clock frequency

- sl_si91x_efuse_init()
- sl_si91x_efuse_set_address()
- sl_si91x_efuse_write_bit()

Returns

- returns status 0 if successful, else error code as follow.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
-
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line `257` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h`

sl_si91x_efuse_enable

```
__STATIC_INLINE void sl_si91x_efuse_enable (void)
```

This API is used to enable the EFUSE.

Parameters

[in]		

- sl_si91x_efuse_enable_clock()

Returns

- none

Definition at line 266 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_si91x_efuse_disable

```
__STATIC_INLINE void sl_si91x_efuse_disable (void)
```

This API is used to Disable the EFUSE.

Parameters

[in]		

- sl_si91x_efuse_init()
- sl_si91x_efuse_set_address()
- sl_si91x_efuse_write_bit()
- sl_si91x_efuse_memory_mapped_read_byte()

Returns

- none

Definition at line 284 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

sl_efuse_version_t

Structure to hold the versions number of peripheral API.

Public Attributes

uint8_t	release
uint8_t	major
uint8_t	minor

Public Attribute Documentation

release

```
uint8_t sl_efuse_version_t::release
```

Definition at line 70 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

major

```
uint8_t sl_efuse_version_t::major
```

Definition at line 71 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

minor

```
uint8_t sl_efuse_version_t::minor
```

Definition at line 72 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_efuse.h

General-Purpose Input-Output

General-Purpose Input-Output

Modules

sl_gpio_t

Typedefs

typedef void(* sl_gpio_irq_callback_t)(uint32_t flag)
GPIO interrupt callback function pointer.

Functions

STATIC __INLINE sl_status_t sl_gpio_driver_clear_interrupts(uint32_t flags)

sl_status_t sl_gpio_driver_configure_interrupt(sl_gpio_t *gpio, uint32_t int_no, sl_gpio_interrupt_flag_t flags, sl_gpio_irq_callback_t gpio_callback, uint32_t *avl_intr_no)
Configure the GPIO pin interrupt.

sl_status_t sl_gpio_driver_set_pin_mode(sl_gpio_t *gpio, sl_gpio_mode_t mode, uint32_t output_value)
Set the pin mode for a GPIO pin.

sl_status_t sl_gpio_driver_get_pin_mode(sl_gpio_t *gpio, sl_gpio_mode_t *mode)
Get the GPIO pin status.

sl_status_t sl_gpio_driver_init(void)

sl_status_t sl_gpio_driver_deinit(void)

STATIC __INLINE sl_status_t sl_gpio_driver_set_pin(sl_gpio_t *gpio)
Set a single pin in GPIO configuration register to 1.

STATIC __INLINE sl_status_t sl_gpio_driver_clear_pin(sl_gpio_t *gpio)
Set a single pin in GPIO configuration register to 0.

STATIC __INLINE sl_status_t sl_gpio_driver_toggle_pin(sl_gpio_t *gpio)
Toggle a single pin in GPIO port register.

STATIC __INLINE sl_status_t sl_gpio_driver_get_pin(sl_gpio_t *gpio, uint8_t *pin_value)
Read the pad value for a single pin in a GPIO port.

STATIC __INLINE sl_status_t sl_gpio_driver_set_port(sl_gpio_port_t port, uint32_t pins)
Set bits GPIO data out register to 1.

STATIC __INLINE sl_status_t sl_gpio_driver_clear_port(sl_gpio_port_t port, uint32_t pins)
Set bits in configuration register for a port to 0.

STATIC __INLINE sl_status_t sl_gpio_driver_get_port_output(sl_gpio_port_t port, uint32_t *port_value)
Get the current setting for a GPIO configuration register.

STATIC __INLINE uint8_t	sl_gpio_driver_get_pin_output(sl_gpio_t *gpio) Get the current setting for a pin in a GPIO configuration register.
STATIC __INLINE sl_status_t	sl_gpio_driver_set_port_output_value(sl_gpio_port_t port, uint32_t val, uint32_t mask)
STATIC __INLINE sl_status_t	sl_gpio_driver_set_slew_rate(sl_gpio_port_t port, uint32_t slewrate, uint32_t slewrate_alt)
STATIC __INLINE uint32_t	sl_gpio_driver_get_port_input(sl_gpio_port_t port)
STATIC __INLINE sl_status_t	sl_gpio_driver_toggle_port_output(sl_gpio_port_t port, uint32_t pins)
STATIC __INLINE sl_status_t	sl_gpio_driver_enable_interrupts(uint32_t flags)
STATIC __INLINE sl_status_t	sl_gpio_driver_disable_interrupts(uint32_t flags)
STATIC __INLINE sl_status_t	sl_gpio_driver_set_interrupts(uint32_t flags)
STATIC __INLINE uint32_t	sl_gpio_driver_get_pending_interrupts(void)
STATIC __INLINE uint32_t	sl_gpio_driver_get_enabled_interrupts(void)
STATIC __INLINE uint32_t	sl_gpio_driver_get_enabled_pending_interrupts(void)
sl_status_t	sl_si91x_gpio_driver_set_pin_direction(uint8_t port, uint8_t pin, sl_si91x_gpio_direction_t direction) Set the direction for a GPIO pin.
uint8_t	sl_si91x_gpio_driver_get_pin_direction(uint8_t port, uint8_t pin) Get the direction GPIO.
sl_status_t	sl_si91x_gpio_driver_enable_pad_receiver(uint8_t gpio_num) Enable the receiver bit in the PAD configuration register.
sl_status_t	sl_si91x_gpio_driver_disable_pad_receiver(uint8_t gpio_num) Disable the receiver bit in the PAD configuration register.
sl_status_t	sl_si91x_gpio_driver_enable_pad_selection(uint8_t gpio_padnum) Select the pad(0 to 21).
sl_status_t	sl_si91x_gpio_driver_select_pad_driver_strength(uint8_t gpio_num, sl_si91x_gpio_driver_strength_select_t strength) Select drive strength of a GPIO pin.
sl_status_t	sl_si91x_gpio_driver_select_pad_driver_disable_state(uint8_t gpio_num, sl_si91x_gpio_driver_disable_state_t disable_state) Select the Driver disabled state control.
sl_status_t	sl_si91x_gpio_driver_select_group_interrupt_and_or(uint8_t port, sl_si91x_group_interrupt_t group_interrupt, sl_si91x_gpio_and_or_t and_or) Select AND/OR of the group interrupt.
sl_status_t	sl_si91x_gpio_driver_clear_group_interrupt(sl_si91x_group_interrupt_t group_interrupt) Clear the group interrupt status.

uint32_t	sl_si91x_gpio_driver_get_group_interrupt_status(uint8_t port, sl_si91x_group_interrupt_t group_interrupt) Get the group interrupt status.
sl_status_t	sl_si91x_gpio_driver_select_group_interrupt_wakeup(uint8_t port, sl_si91x_group_interrupt_t group_interrupt, sl_si91x_gpio_wakeup_t flags) Configure the group interrupt wake up the interrupt.
sl_status_t	sl_si91x_gpio_driver_configure_group_interrupt(sl_si91x_gpio_group_interrupt_config_t *configuration, sl_gpio_irq_callback_t gpio_callback) Configure the MCU HP group interrupts.
uint8_t	sl_si91x_gpio_driver_get_group_interrupt_polarity(sl_si91x_group_interrupt_t group_interrupt, uint8_t port, uint8_t pin) Get the polarity of group interrupt.
sl_status_t	sl_si91x_gpio_driver_set_group_interrupt_polarity(sl_si91x_group_interrupt_t group_interrupt, uint8_t port, uint8_t pin, sl_si91x_gpio_polarity_t polarity) Configure the polarity of group interrupt.
uint8_t	sl_si91x_gpio_driver_get_group_interrupt_level_edge(uint8_t port, sl_si91x_group_interrupt_t group_interrupt) Get the level/edge event of group interrupt.
sl_status_t	sl_si91x_gpio_driver_set_group_interrupt_level_edge(uint8_t port, sl_si91x_group_interrupt_t group_interrupt, sl_si91x_gpio_level_edge_t level_edge) Set the level/edge event of group interrupt.
sl_status_t	sl_si91x_gpio_driver_unmask_group_interrupt(uint8_t port, sl_si91x_group_interrupt_t group_interrupt) Unmask the group interrupts.
sl_status_t	sl_si91x_gpio_driver_mask_group_interrupt(uint8_t port, sl_si91x_group_interrupt_t group_interrupt) Mask the group interrupts.
sl_status_t	sl_si91x_gpio_driver_disable_clock(sl_si91x_gpio_select_clock_t clock) Disable HP/ULP GPIO clock.
sl_status_t	sl_si91x_gpio_driver_enable_clock(sl_si91x_gpio_select_clock_t clock) Enable HP/ULP GPIO clock.
sl_status_t	sl_si91x_gpio_driver_enable_group_interrupt(sl_si91x_group_interrupt_t group_interrupt, uint8_t port, uint8_t pin) Enable the group interrupts.
sl_status_t	sl_si91x_gpio_driver_disable_group_interrupt(sl_si91x_group_interrupt_t group_interrupt, uint8_t port, uint8_t pin) Disable the group interrupts.
sl_status_t	sl_si91x_gpio_driver_select_ulp_pad_slew_rate(uint8_t gpio_num, sl_si91x_gpio_slew_rate_t slew_rate) Select the slew rate.
sl_status_t	sl_si91x_gpio_driver_select_ulp_pad_driver_strength(uint8_t gpio_num, sl_si91x_gpio_driver_strength_select_t strength) Select the drive strength.
sl_status_t	sl_si91x_gpio_driver_select_ulp_pad_driver_disable_state(uint8_t gpio_num, sl_si91x_gpio_driver_disable_state_t disable_state) Select the driver-disabled state control.
sl_status_t	sl_si91x_gpio_driver_disable_ulp_pad_receiver(uint8_t gpio_num) Disable the receiver bit for ULP.
sl_status_t	sl_si91x_gpio_driver_enable_ulp_pad_receiver(uint8_t gpio_num) Enable the receiver bit for ULP.

sl_status_t	sl_si91x_gpio_driver_configure_ulp_pin_interrupt(uint8_t int_no, sl_si91x_gpio_interrupt_config_flag_t flags, sl_si91x_gpio_pin_ulp_t pin, sl_gpio_irq_callback_t gpio_callback) Configure the MCU ULP GPIO pin interrupt.
sl_status_t	sl_si91x_gpio_driver_set_uulp_npss_pin_mux(uint8_t pin, sl_si91x_uulp_npss_mode_t mode) Set the NPSS GPIO pin MUX(mode).
sl_status_t	sl_si91x_gpio_driver_select_uulp_npss_receiver(uint8_t pin, sl_si91x_gpio_receiver_t receiver) Enable/disable NPSS GPIO receiver.
sl_status_t	sl_si91x_gpio_driver_set_uulp_npss_direction(uint8_t pin, sl_si91x_gpio_direction_t direction) Set the direction of the NPSS GPIO.
uint8_t	sl_si91x_gpio_driver_get_uulp_npss_direction(uint8_t pin) Get the direction of the NPSS GPIO.
sl_status_t	sl_si91x_gpio_driver_set_uulp_npss_pin_value(uint8_t pin, sl_si91x_gpio_pin_value_t pin_value) Control the NPSS GPIO pin value.
uint8_t	sl_si91x_gpio_driver_get_uulp_npss_pin(uint8_t pin) Get the NPSS GPIO pin value.
sl_status_t	sl_si91x_gpio_driver_select_uulp_npss_polarity(uint8_t pin, sl_si91x_gpio_polarity_t polarity) Select the NPSS GPIO polarity.
sl_status_t	sl_si91x_gpio_driver_set_uulp_npss_wakeup_interrupt(uint8_t npssgpio_interrupt) Set the UULP NPSS GPIO to wake up interrupt.
sl_status_t	sl_si91x_gpio_driver_clear_uulp_npss_wakeup_interrupt(uint8_t npssgpio_interrupt) Clear the UULP NPSS GPIO to wake up interrupt.
sl_status_t	sl_si91x_gpio_driver_mask_uulp_npss_interrupt(uint8_t npssgpio_interrupt) Mask the NPSS GPIO interrupt.
sl_status_t	sl_si91x_gpio_driver_unmask_uulp_npss_interrupt(uint8_t npssgpio_interrupt) Unmask the NPSS GPIO interrupt.
sl_status_t	sl_si91x_gpio_driver_clear_uulp_interrupt(uint8_t npssgpio_interrupt) Clear the NPSS GPIO interrupt.
uint8_t	sl_si91x_gpio_driver_get_uulp_interrupt_status(void) Get the NPSS GPIO interrupt status.
uint32_t	sl_si91x_gpio_driver_get_ulp_interrupt_status(uint32_t flags) Get the ULP GPIO interrupt status.
sl_status_t	sl_si91x_gpio_driver_clear_ulp_interrupt(uint32_t flags) Clear one or more pending ULP GPIO interrupts.
sl_status_t	sl_si91x_gpio_driver_clear_ulp_group_interrupt(sl_si91x_group_interrupt_t group_interrupt) Clear the ULP group interrupt.
sl_status_t	sl_si91x_gpio_driver_configure_uulp_interrupt(sl_si91x_gpio_interrupt_config_flag_t flags, uint8_t npssgpio_interrupt, sl_gpio_irq_callback_t gpio_callback) Configure the UULP GPIO pin interrupt.
sl_status_t	sl_si91x_gpio_driver_configure_ulp_group_interrupt(sl_si91x_gpio_group_interrupt_config_t *configuration, sl_gpio_irq_callback_t gpio_callback) Configure ULP GPIO group interrupts.
sl_status_t	sl_si91x_gpio_driver_toggle_uulp_npss_pin(uint8_t pin) Toggle the UULP pin.

sl_status_t	sl_si91x_gpio_driver_set_uulp_pad_configuration(uulp_pad_config_t *pad_config) Indicate UULP GPIO PAD configuration.
sl_si91x_gpio_version_t	sl_si91x_gpio_driver_get_version(void) Get the release, SQA, and development version numbers of the GPIO peripheral.

Macros

#define	GPIO_MAX_OUTPUT_VALUE 1
#define	MAX_GROUP_INT 2
#define	GPIO_PORT_MAX_VALUE 4
#define	MAX_UULP_INT 5
#define	ULP_MAX_MODE 10
#define	GPIO_MAX_INTR_VALUE 8
#define	PORTD_PIN_MAX_VALUE 8
#define	PORTE_PIN_MAX_VALUE 11
#define	MAX_ULP_INTR 8
#define	MAX_MODE 15
#define	PORT_PIN_MAX_VALUE 15
#define	GPIO_FLAGS_MAX_VALUE 0×0F
#define	PORTA 0
#define	PORTB 1
#define	PORTC 2
#define	PORTD 3
#define	PORTE 4

Typedef Documentation

sl_gpio_irq_callback_t

```
typedef void(* sl_gpio_irq_callback_t) (uint32_t flag) )(uint32_t flag)
```

GPIO interrupt callback function pointer.

Definition at line 83 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

Function Documentation

sl_gpio_driver_clear_interrupts

> STATIC __INLINE sl_status_t sl_gpio_driver_clear_interrupts (uint32_t flags)

Parameters

N/A	flags	

Definition at line 107 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_configure_interrupt

> sl_status_t sl_gpio_driver_configure_interrupt (sl_gpio_t *gpio, uint32_t int_no, sl_gpio_interrupt_flag_t flags, sl_gpio_irq_callback_t gpio_callback, uint32_t *avl_intr_no)

Configure the GPIO pin interrupt.

Parameters

[in]	gpio	- Pointer to the structure of type sl_gpio_t
[in]	int_no	- The interrupt number to trigger.
[in]	flags	- Interrupt configuration flags
[in]	gpio_callback	- IRQ function pointer
[out]	avl_intr_no	- Pointer to the available interrupt number. SL_GPIO_INTERRUPT_UNAVAILABLE (0xFF) no available interrupt

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()

Returns

- The following values are returned:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
- SL_STATUS _OK (0X000) - Success

Definition at line 140 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_set_pin_mode

> sl_status_t sl_gpio_driver_set_pin_mode (sl_gpio_t *gpio, sl_gpio_mode_t mode, uint32_t output_value)

Set the pin mode for a GPIO pin.

Parameters

[in]	gpio	- Pointer to the structure of type sl_gpio_t
[in]	mode	- The desired pin mode.

[in]	output_value	- A value to set for the pin in the GPIO register. The GPIO setting is important for some input mode configurations.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

Returns

- The following values are returned:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
- SL_STATUS _OK (0X000) - Success

Definition at line 170 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_get_pin_mode

```
sl_status_t sl_gpio_driver_get_pin_mode (sl_gpio_t *gpio, sl_gpio_mode_t *mode)
```

Get the GPIO pin status.

Parameters

[in]	gpio	- Pointer to the structure of type sl_gpio_t
[in]	mode	- The desired pin mode.

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance. sl_gpio_driver_set_pin_mode();

Returns

- The following values are returned:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
- SL_STATUS _OK (0X000) - Success

Definition at line 199 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_init

```
sl_status_t sl_gpio_driver_init (void)
```

Parameters

N/A		

Definition at line 208 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_deinit

```
sl_status_t sl_gpio_driver_deinit (void)
```

Parameters

N/A		

Definition at line 217 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_set_pin

```
STATIC __INLINE sl_status_t sl_gpio_driver_set_pin (sl_gpio_t *gpio)
```

Set a single pin in GPIO configuration register to 1.

Parameters

[in]	gpio	- Pointer to the structure of type sl_gpio_t

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

- sl_gpio_driver_set_pin_mode();
- sl_si91x_gpio_driver_set_pin_direction();

Returns

- The following values are returned:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
- SL_STATUS _OK (0X000) - Success

Definition at line 249 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_clear_pin

> STATIC __INLINE sl_status_t sl_gpio_driver_clear_pin (sl_gpio_t *gpio)

Set a single pin in GPIO configuration register to 0.

Parameters

[in]	gpio	- Pointer to the structure of type sl_gpio_t

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(); for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

- sl_gpio_driver_set_pin_mode();
- sl_si91x_gpio_driver_set_pin_direction();

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS _OK (0X000) - Success

Definition at line 312 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_toggle_pin

> STATIC __INLINE sl_status_t sl_gpio_driver_toggle_pin (sl_gpio_t *gpio)

Toggle a single pin in GPIO port register.

Parameters

[in]	gpio	- Pointer to the structure of type sl_gpio_t

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

- sl_gpio_driver_set_pin_mode()

sl_si91x_gpio_driver_set_pin_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS _OK (0X000) - Success

Definition at line 375 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_get_pin

> STATIC __INLINE sl_status_t sl_gpio_driver_get_pin (sl_gpio_t *gpio, uint8_t *pin_value)

Read the pad value for a single pin in a GPIO port.

Parameters

[in]	gpio	- Pointer to the structure of type sl_gpio_t
N/A	pin_value	

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- sl_gpio_driver_set_pin_mode()
- sl_si91x_gpio_driver_set_pin_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS _OK (0X000) - Success

Definition at line 438 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_set_port

> STATIC __INLINE sl_status_t sl_gpio_driver_set_port (sl_gpio_port_t port, uint32_t pins)

Set bits GPIO data out register to 1.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	pins	- The GPIO pins in a port are set to 1's.

Pre-conditions:
- Pre-conditions:
- sl_si91x_gpio_driver_enable_clock()

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

- sl_gpio_driver_set_pin_mode();
- sl_si91x_gpio_driver_set_pin_direction();

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 502 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_clear_port

> STATIC __INLINE sl_status_t sl_gpio_driver_clear_port (sl_gpio_port_t port, uint32_t pins)

Set bits in configuration register for a port to 0.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	pins	- The GPIO pins in a port to clear.

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

- sl_gpio_driver_set_pin_mode()
- sl_si91x_gpio_driver_set_pin_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 544 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_get_port_output

```
STATIC __INLINE sl_status_t sl_gpio_driver_get_port_output (sl_gpio_port_t port, uint32_t *port_value)
```

Get the current setting for a GPIO configuration register.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
N/A	port_value	

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

- sl_gpio_driver_set_pin_mode()
- sl_si91x_gpio_driver_set_pin_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `585` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h`

sl_gpio_driver_get_pin_output

```
STATIC __INLINE uint8_t sl_gpio_driver_get_pin_output (sl_gpio_t *gpio)
```

Get the current setting for a pin in a GPIO configuration register.

Parameters

[in]	gpio	- Pointer to the structure of type sl_gpio_t

- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

- sl_gpio_driver_set_pin_mode()
- sl_si91x_gpio_driver_set_pin_direction()

Returns

- The GPIO pin value '0' - Output
  - '1' - Input

Definition at line 623 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_set_port_output_value

> STATIC __INLINE sl_status_t sl_gpio_driver_set_port_output_value (sl_gpio_port_t port, uint32_t val, uint32_t mask)

Parameters

N/A	port	
N/A	val	
N/A	mask	

Definition at line 687 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_set_slew_rate

> STATIC __INLINE sl_status_t sl_gpio_driver_set_slew_rate (sl_gpio_port_t port, uint32_t slewrate, uint32_t slewrate_alt)

Parameters

N/A	port	
N/A	slewrate	
N/A	slewrate_alt	

Definition at line 718 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_get_port_input

> STATIC __INLINE uint32_t sl_gpio_driver_get_port_input (sl_gpio_port_t port)

Parameters

N/A	port	

Definition at line 752 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_toggle_port_output

> STATIC __INLINE sl_status_t sl_gpio_driver_toggle_port_output (sl_gpio_port_t port, uint32_t pins)

Parameters

N/A	port	
N/A	pins	

Definition at line 793 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_enable_interrupts

```
STATIC __INLINE sl_status_t sl_gpio_driver_enable_interrupts (uint32_t flags)
```

Parameters

N/A	flags	

Definition at line 823 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_disable_interrupts

```
STATIC __INLINE sl_status_t sl_gpio_driver_disable_interrupts (uint32_t flags)
```

Parameters

N/A	flags	

Definition at line 848 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_set_interrupts

```
STATIC __INLINE sl_status_t sl_gpio_driver_set_interrupts (uint32_t flags)
```

Parameters

N/A	flags	

Definition at line 873 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_get_pending_interrupts

```
STATIC __INLINE uint32_t sl_gpio_driver_get_pending_interrupts (void)
```

Parameters

N/A		

Definition at line 895 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_get_enabled_interrupts

```
STATIC __INLINE uint32_t sl_gpio_driver_get_enabled_interrupts (void)
```

Parameters

N/A		

Definition at line 918 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_driver_get_enabled_pending_interrupts

> STATIC __INLINE uint32_t sl_gpio_driver_get_enabled_pending_interrupts (void)

Parameters

N/A		

Definition at line 941 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_si91x_gpio_driver_set_pin_direction

> sl_status_t sl_si91x_gpio_driver_set_pin_direction (uint8_t port, uint8_t pin, sl_si91x_gpio_direction_t direction)

Set the direction for a GPIO pin.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	pin	- The pin number on the port. HP instance has total 57 GPIO pins. Port 0, 1, 2 has 16 pins each. Port 3 has 9 pins. ULP instance has total 12 pins.
[in]	direction	- pin direction of type sl_si91x_gpio_direction_t '0' - Output • '1' - Input

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 81 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_get_pin_direction

> uint8_t sl_si91x_gpio_driver_get_pin_direction (uint8_t port, uint8_t pin)

Get the direction GPIO.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	pin	- The pin number on the port. HP instance has total 57 GPIO pins. Port 0, 1, 2 has 16 pins each. Port 3 has 9 pins. ULP instance has total 12 pins.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
-

Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_selection(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.

- sl_si91x_gpio_driver_set_pin_direction()
- sl_si91x_gpio_driver_get_pin_direction()

Returns

- Returns the direction of the pin. '0' - Output
  - '1' - Input

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_enable_pad_receiver

```
sl_status_t sl_si91x_gpio_driver_enable_pad_receiver (uint8_t gpio_num)
```

Enable the receiver bit in the PAD configuration register.

Parameters

[in]	gpio_num	- GPIO number to be use.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- sl_si91x_gpio_driver_enable_pad_selection(), for HP instance

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 129 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_disable_pad_receiver

```
sl_status_t sl_si91x_gpio_driver_disable_pad_receiver (uint8_t gpio_num)
```

Disable the receiver bit in the PAD configuration register.

Parameters

[in]	gpio_num	- GPIO number to be use.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- sl_si91x_gpio_driver_enable_pad_selection(), for HP instance

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 146 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_enable_pad_selection

> sl_status_t sl_si91x_gpio_driver_enable_pad_selection (uint8_t gpio_padnum)

Select the pad(0 to 21).

Parameters

[in]	gpio_padnum	- PAD number to be use

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 161 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_select_pad_driver_strength

> sl_status_t sl_si91x_gpio_driver_select_pad_driver_strength (uint8_t gpio_num, sl_si91x_gpio_driver_strength_select_t strength)

Select drive strength of a GPIO pin.

Parameters

[in]	gpio_num	- GPIO number to be use
[in]	strength	- Drive strength selector(E1,E2) of type sl_si91x_gpio_driver_strength_select_t possible values are 0, for two_milli_amps (E1=0,E2=0) • 1, for four_milli_amps (E1=0,E2=1) • 2, for eight_milli_amps (E1=1,E2=0) • 3, for twelve_milli_amps(E1=1,E2=1)

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 190 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_select_pad_driver_disable_state

> sl_status_t sl_si91x_gpio_driver_select_pad_driver_disable_state (uint8_t gpio_num, sl_si91x_gpio_driver_disable_state_t disable_state)

Select the Driver disabled state control.

Parameters

[in]	gpio_num	- GPIO number to be use
[in]	disable_state	- driver disable state of type sl_si91x_gpio_driver_disable_state_t possible values are 0, for HiZ (P1=0,P2=0) • 1, for Pull-up (P1=0,P2=1) • 2, for Pull-down (P1=1,P2=0) • 3, for Repeater (P1=1,P2=1)

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `220` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_select_group_interrupt_and_or

> sl_status_t sl_si91x_gpio_driver_select_group_interrupt_and_or (uint8_t port, sl_si91x_group_interrupt_t group_interrupt, sl_si91x_gpio_and_or_t and_or)

Select AND/OR of the group interrupt.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	group_interrupt	- Group interrupt number of type sl_si91x_group_interrupt_t
[in]	and_or	- AND/OR of GPIO group interrupts of type sl_si91x_gpio_and_or_t '0' - AND • '1' - OR

If multiple interrupts on same port (or) different are to be generated, then use this API. Example: Consider port 0: pin 2,3 and port 3: pin 1,2 for interrupt generation. Choose OR, any of the selected pin is fine for group interrupt generation Choose AND, all the selected pins are necessary for group interrupt generation

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance

Use corresponding pad receiver API for corresponding GPIO instance.

- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance
    Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `273` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_clear_group_interrupt

```
sl_status_t sl_si91x_gpio_driver_clear_group_interrupt (sl_si91x_group_interrupt_t group_interrupt)
```

Clear the group interrupt status.

Parameters

[in]	group_interrupt	- Group interrupt number of type sl_si91x_group_interrupt_t

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver()
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `300` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_get_group_interrupt_status

```
uint32_t sl_si91x_gpio_driver_get_group_interrupt_status (uint8_t port, sl_si91x_group_interrupt_t group_interrupt)
```

Get the group interrupt status.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4

[in]	group_interrupt	- Group interrupt number of type sl_si91x_group_interrupt_t

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode(),
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns the group interrupt status register 1, when interrupt is enabled
  - 0, when interrupt is disabled

Definition at line `333` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_select_group_interrupt_wakeup

> sl_status_t sl_si91x_gpio_driver_select_group_interrupt_wakeup (uint8_t port, sl_si91x_group_interrupt_t group_interrupt, sl_si91x_gpio_wakeup_t flags)

Configure the group interrupt wake up the interrupt.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	group_interrupt	- Group interrupt number of type sl_si91x_group_interrupt_t
[in]	flags	- GPIO group interrupt wake up flag of type sl_si91x_gpio_wakeup_t '1' - enable • '0' - disable

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `355` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_configure_group_interrupt

> sl_status_t sl_si91x_gpio_driver_configure_group_interrupt (sl_si91x_gpio_group_interrupt_config_t *configuration, sl_gpio_irq_callback_t gpio_callback)

Configure the MCU HP group interrupts.

Parameters

[in]	configuration	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	gpio_callback	- configuration pointer to sl_si91x_gpio_group_interrupt_config_t structure
[in]	gpio_callback	- IRQ function pointer

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver()
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS _OK (0X000) - Success

Definition at line `384` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_get_group_interrupt_polarity

> uint8_t sl_si91x_gpio_driver_get_group_interrupt_polarity (sl_si91x_group_interrupt_t group_interrupt, uint8_t port, uint8_t pin)

Get the polarity of group interrupt.

Parameters

[in]	group_interrupt	- GPIO group interrupt number of type sl_si91x_group_interrupt_t
[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	pin	- The pin number on the port. HP instance has total 57 GPIO pins. Port 0, 1, 2 has 16 pins each. Port 3 has 9 pins. ULP instance has total 12 pins.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns group interrupt polarity 1, when GPIO pin status is '1'

0, when GPIO pin status is '0'

Definition at line 422 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_set_group_interrupt_polarity

> sl_status_t sl_si91x_gpio_driver_set_group_interrupt_polarity (sl_si91x_group_interrupt_t group_interrupt, uint8_t port, uint8_t pin, sl_si91x_gpio_polarity_t polarity)

Configure the polarity of group interrupt.

Parameters

[in]	group_interrupt	- GPIO group interrupt number of type sl_si91x_group_interrupt_t
[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	pin	- The pin number on the port. HP instance has total 57 GPIO pins. Port 0, 1, 2 has 16 pins each. Port 3 has 9 pins. ULP instance has total 12 pins.
[in]	polarity	- polarity of GPIO group interrupt of type sl_si91x_gpio_polarity_t 1, group interrupt gets generated when GPIO pin status is '1' • 0, group interrupt gets generated when GPIO pin status is '0'

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 468 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_get_group_interrupt_level_edge

> uint8_t sl_si91x_gpio_driver_get_group_interrupt_level_edge (uint8_t port, sl_si91x_group_interrupt_t group_interrupt)

Get the level/edge event of group interrupt.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	group_interrupt	- GPIO group interrupt number of type sl_si91x_group_interrupt_t

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode(),
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction(),
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns group interrupt level_edge 1, for Edge
  - 0, for Level

Definition at line 504 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_set_group_interrupt_level_edge

> sl_status_t sl_si91x_gpio_driver_set_group_interrupt_level_edge (uint8_t port, sl_si91x_group_interrupt_t group_interrupt, sl_si91x_gpio_level_edge_t level_edge)

Set the level/edge event of group interrupt.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	group_interrupt	- GPIO group interrupt number of type sl_si91x_group_interrupt_t
[in]	level_edge	- GPIO level edge group interrupt of type sl_si91x_gpio_level_edge_t 1, for Edge • 0, for Level

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:

sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `544` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_unmask_group_interrupt

```
sl_status_t sl_si91x_gpio_driver_unmask_group_interrupt (uint8_t port, sl_si91x_group_interrupt_t group_interrupt)
```

Unmask the group interrupts.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	group_interrupt	- GPIO group interrupt number of type sl_si91x_group_interrupt_t

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `580` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_mask_group_interrupt

```
sl_status_t sl_si91x_gpio_driver_mask_group_interrupt (uint8_t port, sl_si91x_group_interrupt_t group_interrupt)
```

Mask the group interrupts.

Parameters

[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	group_interrupt	- GPIO group interrupt number of type sl_si91x_group_interrupt_t

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `609` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_disable_clock

```
sl_status_t sl_si91x_gpio_driver_disable_clock (sl_si91x_gpio_select_clock_t clock)
```

Disable HP/ULP GPIO clock.

Parameters

[in]	clock	- Selects M4 clock or ULP clock of type sl_si91x_gpio_select_clock_t 0, for M4 GPIO CLK • 1, for ULP GPIO CLK

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `626` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_enable_clock

```
sl_status_t sl_si91x_gpio_driver_enable_clock (sl_si91x_gpio_select_clock_t clock)
```

Enable HP/ULP GPIO clock.

Parameters

[in]	clock	- Selects M4 clock or ULP clock of type sl_si91x_gpio_select_clock_t 0, for M4 GPIO CLK • 1, for ULP GPIO CLK

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 643 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_enable_group_interrupt

> sl_status_t sl_si91x_gpio_driver_enable_group_interrupt (sl_si91x_group_interrupt_t group_interrupt, uint8_t port, uint8_t pin)

Enable the group interrupts.

Parameters

[in]	group_interrupt	- GPIO group interrupt number of type sl_si91x_group_interrupt_t
[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4
[in]	pin	- The pin number on the port. HP instance has total 57 GPIO pins. Port 0, 1, 2 has 16 pins each. Port 3 has 9 pins. ULP instance has total 12 pins.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 681 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_disable_group_interrupt

> sl_status_t sl_si91x_gpio_driver_disable_group_interrupt (sl_si91x_group_interrupt_t group_interrupt, uint8_t port, uint8_t pin)

Disable the group interrupts.

Parameters

[in]	group_interrupt	- GPIO group interrupt number of type sl_si91x_group_interrupt_t
[in]	port	- The port to associate with the pin. HP instance - PORT 0,1,2,3 ULP instance - PORT 4

[in]	pin	- The pin number on the port. HP instance has total 57 GPIO pins. Port 0, 1, 2 has 16 pins each. Port 3 has 9 pins. ULP instance has total 12 pins.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock, for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_pad_receiver(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver(), for ULP instance
    Use corresponding pad receiver API for corresponding GPIO instance.
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt(), for HP instance
- Pre-conditions:
  - sl_si91x_gpio_driver_configure_ulp_group_interrupt, for ULP instance
    Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 733 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_select_ulp_pad_slew_rate

```
sl_status_t sl_si91x_gpio_driver_select_ulp_pad_slew_rate (uint8_t gpio_num, sl_si91x_gpio_slew_rate_t slew_rate)
```

Select the slew rate.

Parameters

[in]	gpio_num	- GPIO number to be use
[in]	slew_rate	- slew rate of type sl_si91x_gpio_slew_rate_t '0' - Slow • '1' - Fast

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- sl_si91x_gpio_driver_enable_ulp_pad_receiver()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 757 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_select_ulp_pad_driver_strength

sl_status_t sl_si91x_gpio_driver_select_ulp_pad_driver_strength (uint8_t gpio_num, sl_si91x_gpio_driver_strength_select_t strength)

Select the drive strength.

Parameters

[in]	gpio_num	- GPIO number to be use
[in]	strength	- Drive strength selector(E1,E2) of type sl_si91x_gpio_driver_strength_select_t 0, for two_milli_amps (E1=0,E2=0) • 1, for four_milli_amps (E1=0,E2=1) • 2, for eight_milli_amps (E1=1,E2=0) • 3, for twelve_milli_amps(E1=1,E2=1)

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- sl_si91x_gpio_driver_enable_ulp_pad_receiver()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `784` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_select_ulp_pad_driver_disable_state

sl_status_t sl_si91x_gpio_driver_select_ulp_pad_driver_disable_state (uint8_t gpio_num, sl_si91x_gpio_driver_disable_state_t disable_state)

Select the driver-disabled state control.

Parameters

[in]	gpio_num	- GPIO number to be use
[in]	disable_state	- driver disable state of type sl_si91x_gpio_driver_disable_state_t 0, for HiZ (P1=0,P2=0) • 1, for Pull up (P1=0,P2=1) • 2, for Pull down (P1=1,P2=0) • 3, for Repeater (P1=1,P2=1)

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- sl_si91x_gpio_driver_enable_ulp_pad_receiver()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `812` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_disable_ulp_pad_receiver

> sl_status_t sl_si91x_gpio_driver_disable_ulp_pad_receiver (uint8_t gpio_num)

Disable the receiver bit for ULP.

Parameters

[in]	gpio_num	- GPIO number to be used

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `829` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_enable_ulp_pad_receiver

> sl_status_t sl_si91x_gpio_driver_enable_ulp_pad_receiver (uint8_t gpio_num)

Enable the receiver bit for ULP.

Parameters

[in]	gpio_num	- GPIO number to be used

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `844` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_configure_ulp_pin_interrupt

> sl_status_t sl_si91x_gpio_driver_configure_ulp_pin_interrupt (uint8_t int_no, sl_si91x_gpio_interrupt_config_flag_t flags, sl_si91x_gpio_pin_ulp_t pin, sl_gpio_irq_callback_t gpio_callback)

Configure the MCU ULP GPIO pin interrupt.

Parameters

[in]	int_no	- The interrupt number to trigger.
[in]	flags	- Interrupt configuration flags of type sl_si91x_gpio_interrupt_config_flag_t
[in]	pin	- GPIO pin number
[in]	gpio_callback	- IRQ function pointer

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:

sl_si91x_gpio_driver_enable_ulp_pad_receiver()
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS _OK (0X000) - Success

Definition at line `874` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_set_uulp_npss_pin_mux

```
sl_status_t sl_si91x_gpio_driver_set_uulp_npss_pin_mux (uint8_t pin, sl_si91x_uulp_npss_mode_t mode)
```

Set the NPSS GPIO pin MUX(mode).

Parameters

[in]	pin	- NPSS GPIO pin number(0...4) of type sl_si91x_uulp_npss_mode_t
[in]	mode	- NPSS GPIO MUX value

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- sl_si91x_gpio_driver_select_uulp_npss_receiver()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `896` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_select_uulp_npss_receiver

```
sl_status_t sl_si91x_gpio_driver_select_uulp_npss_receiver (uint8_t pin, sl_si91x_gpio_receiver_t receiver)
```

Enable/disable NPSS GPIO receiver.

Parameters

[in]	pin	- is NPSS GPIO pin number (0...4)
[in]	receiver	- is enable/disable NPSS GPIO receiver of type sl_si91x_gpio_receiver_t '1' - Enable • '0' - Disable

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument

SL_STATUS _OK (0X000) - Success

Definition at line 917 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_set_uulp_npss_direction

```
sl_status_t sl_si91x_gpio_driver_set_uulp_npss_direction (uint8_t pin, sl_si91x_gpio_direction_t direction)
```

Set the direction of the NPSS GPIO.

Parameters

[in]	pin	- is NPSS GPIO pin number (0...4)
[in]	direction	- is direction value (Input / Output) of type sl_si91x_gpio_direction_t '1' - Input Direction • '0' - Output Direction

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 944 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_get_uulp_npss_direction

```
uint8_t sl_si91x_gpio_driver_get_uulp_npss_direction (uint8_t pin)
```

Get the direction of the NPSS GPIO.

Parameters

[in]	pin	- is NPSS GPIO pin number(0...4)

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_direction()

Returns

- returns the GPIO pin direction
  - 1, Input Direction
  - 0, Output Direction

Definition at line 965 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_set_uulp_npss_pin_value

```
sl_status_t sl_si91x_gpio_driver_set_uulp_npss_pin_value (uint8_t pin, sl_si91x_gpio_pin_value_t pin_value)
```

Control the NPSS GPIO pin value.

Parameters

[in]	pin	- is NPSS GPIO pin number (0...4) of type sl_si91x_gpio_pin_value_t
[in]	pin_value	- is NPSS GPIO pin value '0' - Output • '1' - Input

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `995` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_get_uulp_npss_pin

```
uint8_t sl_si91x_gpio_driver_get_uulp_npss_pin (uint8_t pin)
```

Get the NPSS GPIO pin value.

Parameters

[in]	pin	- is NPSS GPIO pin number (0...4)

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_direction()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_value()

Returns

- returns the pin logical state of pin '0' - Output
  - '1' - Input

Definition at line `1021` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_select_uulp_npss_polarity

```
sl_status_t sl_si91x_gpio_driver_select_uulp_npss_polarity (uint8_t pin, sl_si91x_gpio_polarity_t polarity)
```

Select the NPSS GPIO polarity.

Parameters

[in]	pin	- is NPSS GPIO pin number (0...4)
[in]	polarity	- GPIO polarity sl_si91x_gpio_polarity_t '1' - High • '0' - Low

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `1042` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_set_uulp_npss_wakeup_interrupt

```
sl_status_t sl_si91x_gpio_driver_set_uulp_npss_wakeup_interrupt (uint8_t npssgpio_interrupt)
```

Set the UULP NPSS GPIO to wake up interrupt.

Parameters

[in]	npssgpio_interrupt	- OR'ed values of the NPSS GPIO interrupts

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line `1057` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h`

sl_si91x_gpio_driver_clear_uulp_npss_wakeup_interrupt

```
sl_status_t sl_si91x_gpio_driver_clear_uulp_npss_wakeup_interrupt (uint8_t npssgpio_interrupt)
```

Clear the UULP NPSS GPIO to wake up interrupt.

Parameters

[in]	npssgpio_interrupt	- OR'ed values of the NPSS GPIO interrupts

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 1072 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_mask_uulp_npss_interrupt

```
sl_status_t sl_si91x_gpio_driver_mask_uulp_npss_interrupt (uint8_t npssgpio_interrupt)
```

Mask the NPSS GPIO interrupt.

Parameters

[in]	npssgpio_interrupt	- OR'ed values of the NPSS GPIO interrupts

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_set_uulp_pad_configuration()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 1099 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_unmask_uulp_npss_interrupt

```
sl_status_t sl_si91x_gpio_driver_unmask_uulp_npss_interrupt (uint8_t npssgpio_interrupt)
```

Unmask the NPSS GPIO interrupt.

Parameters

[in]	npssgpio_interrupt	- OR'ed values of the NPSS GPIO interrupts

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_set_uulp_pad_configuration()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 1126 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_clear_uulp_interrupt

```
sl_status_t sl_si91x_gpio_driver_clear_uulp_interrupt (uint8_t npssgpio_interrupt)
```

Clear the NPSS GPIO interrupt.

Parameters

[in]	npssgpio_interrupt	- OR'ed values of the NPSS GPIO interrupts

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_set_uulp_pad_configuration()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_direction()
- Pre-conditions:
  - sl_si91x_gpio_configure_uulp_interrupt()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 1156 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_get_uulp_interrupt_status

```
uint8_t sl_si91x_gpio_driver_get_uulp_interrupt_status (void)
```

Get the NPSS GPIO interrupt status.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_set_uulp_pad_configuration()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:

sl_si91x_gpio_driver_set_uulp_npss_direction()
- Pre-conditions:
  - sl_si91x_gpio_configure_uulp_interrupt()

Returns

- returns the UULP INTR status. 1, interrupt has been raised
  - 0, interrupt is masked or not raised

Definition at line 1185 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_get_ulp_interrupt_status

> uint32_t sl_si91x_gpio_driver_get_ulp_interrupt_status (uint32_t flags)

Get the ULP GPIO interrupt status.

Parameters

[in]	flags	: ULP GPIO interrupt sources status.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver()
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_configure_ulp_pin_interrupt()

Returns

- returns the ULP INTR status. 1, interrupt has been raised
  - 0, interrupt is masked or not raised

Definition at line 1211 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_clear_ulp_interrupt

> sl_status_t sl_si91x_gpio_driver_clear_ulp_interrupt (uint32_t flags)

Clear one or more pending ULP GPIO interrupts.

Parameters

[in]	flags	: ULP GPIO interrupt sources to clear.

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver()
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
- Pre-conditions:
  - sl_si91x_gpio_configure_ulp_pin_interrupt()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 1238 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_clear_ulp_group_interrupt

```
sl_status_t sl_si91x_gpio_driver_clear_ulp_group_interrupt (sl_si91x_group_interrupt_t group_interrupt)
```

Clear the ULP group interrupt.

Parameters

[in]	group_interrupt	- Group interrupt number of type sl_si91x_group_interrupt_t

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver()
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()
    sl_si91x_gpio_driver_configure_ulp_group_interrupt()
    Use corresponding group interrupt configuration API for corresponding GPIO instance.

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 1266 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_configure_uulp_interrupt

```
sl_status_t sl_si91x_gpio_driver_configure_uulp_interrupt (sl_si91x_gpio_interrupt_config_flag_t flags, uint8_t
npssgpio_interrupt, sl_gpio_irq_callback_t gpio_callback)
```

Configure the UULP GPIO pin interrupt.

Parameters

[in]	flags	- Interrupt configuration flags of type sl_si91x_gpio_interrupt_config_flag_t
[in]	npssgpio_interrupt	- NPSS GPIO input number (0 to 4)
[in]	gpio_callback	- IRQ function pointer

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_set_uulp_pad_configuration()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  -

sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS _OK (0X000) - Success

Definition at line 1298 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_configure_ulp_group_interrupt

```
sl_status_t sl_si91x_gpio_driver_configure_ulp_group_interrupt (sl_si91x_gpio_group_interrupt_config_t *configuration,
sl_gpio_irq_callback_t gpio_callback)
```

Configure ULP GPIO group interrupts.

Parameters

[in]	configuration	- configuration pointer to sl_si91x_gpio_group_interrupt_config_t structure
[in]	gpio_callback	- IRQ function pointer

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_enable_ulp_pad_receiver()
- Pre-conditions:
  - sl_gpio_driver_set_pin_mode()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_pin_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS _OK (0X000) - Success

Definition at line 1326 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_toggle_uulp_npss_pin

```
sl_status_t sl_si91x_gpio_driver_toggle_uulp_npss_pin (uint8_t pin)
```

Toggle the UULP pin.

Parameters

[in]	pin	- UULP pin number to toggle

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()
- Pre-conditions:
  - sl_si91x_gpio_driver_select_uulp_npss_receiver()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_pin_mux()
- Pre-conditions:
  - sl_si91x_gpio_driver_set_uulp_npss_direction()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
  - SL_STATUS _OK (0X000) - Success

Definition at line 1351 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_set_uulp_pad_configuration

```
sl_status_t sl_si91x_gpio_driver_set_uulp_pad_configuration (uulp_pad_config_t *pad_config)
```

Indicate UULP GPIO PAD configuration.

Parameters

[in]	pad_config	: PAD configuration pointer to uulp_pad_config_t structure

- Pre-conditions:
  - sl_si91x_gpio_driver_enable_clock()

Returns

- returns status 0 if successful, else error code as follow. SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS _OK (0X000) - Success

Definition at line 1366 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

sl_si91x_gpio_driver_get_version

```
sl_si91x_gpio_version_t sl_si91x_gpio_driver_get_version (void)
```

Get the release, SQA, and development version numbers of the GPIO peripheral.

Parameters

[in]		

Returns

- returns structure of type sl_si91x_gpio_version_t

Definition at line 1373 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_driver_gpio.h

Macro Definition Documentation

GPIO_MAX_OUTPUT_VALUE

```
#define GPIO_MAX_OUTPUT_VALUE
```

Value:

```
1
```

Definition at line 50 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

MAX_GROUP_INT

```
#define MAX_GROUP_INT
```

Value:

```
2
```

Definition at line 51 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

GPIO_PORT_MAX_VALUE

```
#define GPIO_PORT_MAX_VALUE
```

Value:

```
4
```

Definition at line 52 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

MAX_UULP_INT

```
#define MAX_UULP_INT
```

Value:

```
5
```

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

ULP_MAX_MODE

```
#define ULP_MAX_MODE
```

Value:

```
10
```

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

GPIO_MAX_INTR_VALUE

```
#define GPIO_MAX_INTR_VALUE
```

Value:

```
8
```

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

PORTD_PIN_MAX_VALUE

```
#define PORTD_PIN_MAX_VALUE
```

Value:

```
8
```

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

PORTE_PIN_MAX_VALUE

```
#define PORTE_PIN_MAX_VALUE
```

Value:

```
11
```

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

MAX_ULP_INTR

```
#define MAX_ULP_INTR
```

Value:

```
8
```

Definition at line 59 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

MAX_MODE

```
#define MAX_MODE
```

Value:

```
15
```

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

PORT_PIN_MAX_VALUE

```
#define PORT_PIN_MAX_VALUE
```

Value:

```
15
```

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

GPIO_FLAGS_MAX_VALUE

```
#define GPIO_FLAGS_MAX_VALUE
```

Value:

```
0x0F
```

Definition at line 62 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

PORTA

```
#define PORTA
```

Value:

```
0
```

Definition at line 64 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

PORTB

```
#define PORTB
```

Value:

```
1
```

Definition at line 65 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

PORTC

```
#define PORTC
```

Value:

```
2
```

Definition at line 66 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

PORTD

```
#define PORTD
```

Value:

```
3
```

Definition at line 67 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

PORTE

```
#define PORTE
```

Value:

```
4
```

Definition at line 68 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

sl_gpio_t

structure to hold parameters of GPIO port and pin numbers.

Public Attributes

sl_gpio_port_t	port
uint8_t	pin

Public Attribute Documentation

port

```
sl_gpio_port_t sl_gpio_t::port
```

Definition at line 75 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

pin

```
uint8_t sl_gpio_t::pin
```

Definition at line 76 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_driver_gpio.h

Generic SPI

Generic SPI

Modules

sl_gspi_control_config_t

sl_gspi_clock_config_t

sl_gspi_version_t

Enumerations

enum gspi_event_typedef_t {

SL_GSPI_TRANSFER_COMPLETE = ARM_SPI_EVENT_TRANSFER_COMPLETE
SL_GSPI_DATA_LOST = ARM_SPI_EVENT_DATA_LOST
SL_GSPI_MODE_FAULT = ARM_SPI_EVENT_MODE_FAULT

}

Enumeration for different GSPI callback events.

enum sl_gspi_power_state_t {

SL_GSPI_POWER_OFF = ARM_POWER_OFF
SL_GSPI_LOW_POWER = ARM_POWER_LOW
SL_GSPI_FULL_POWER = ARM_POWER_FULL
SL_GSPI_POWER_MODE_LAST

}

Enumeration for GSPI power state.

enum clock_mode_typedef_t {

SL_GSPI_MODE_0 = ARM_SPI_CPOL0_CPHA0
SL_GSPI_MODE_3 = ARM_SPI_CPOL1_CPHA1

}

Enumeration for GSPI clock modes.

enum master_mode_typedef_t {

SL_GSPI_MASTER_INACTIVE = ARM_SPI_MODE_INACTIVE
SL_GSPI_MASTER_ACTIVE = ARM_SPI_MODE_MASTER
SL_GSPI_MASTER_MODE_LAST

}

Enumeration for GSPI master modes.

enum slave_select_mode_typedef_t {

SL_GSPI_MASTER_UNUSED = ARM_SPI_SS_MASTER_UNUSED
SL_GSPI_MASTER_SW = ARM_SPI_SS_MASTER_SW
SL_GSPI_MASTER_HW_OUTPUT = ARM_SPI_SS_MASTER_HW_OUTPUT
SL_GSPI_SLAVE_SELECT_MODE_LAST

}

Enumeration for GSPI slave select mode.

enum sl_gspi_instance_t {

SL_GSPI_MASTER
SL_GSPI_INSTANCE_LAST_ENUM

}

Enumeration for GSPI Master instance, enum for one member, future provision is provided.

enum sl_gspi_slave_number_t {

GSPI_SLAVE_0
GSPI_SLAVE_1
GSPI_SLAVE_2
GSPI_SLAVE_LAST_ENUM

}

Enumeration for GSPI Slave numbers.

Typedefs

typedef ARM_SPI_SignalEvent_t — sl_gspi_signal_event_t
Renamed signal event structure.

typedef ARM_SPI_STATUS — sl_gspi_status_t
Renamed status structure.

typedef ARM_DRIVER_SPI — sl_gspi_driver_t
Renamed GSPI driver structure.

typedef const void * — sl_gspi_handle_t
Created GSPI handle type.

Functions

sl_status_t sl_si91x_gspi_configure_clock(sl_gspi_clock_config_t *clock_configuration)
Set the clock for the GSPI peripheral, Configures the PLL clock and SOC clock with the value set by the user in the clock configuration structure.

sl_status_t sl_si91x_gspi_init(sl_gspi_instance_t instance, sl_gspi_handle_t *gspi_handle)
Initialize the GSPI.

sl_status_t sl_si91x_gspi_deinit(sl_gspi_handle_t gspi_handle)
Uninitialize the GSPI.

sl_status_t sl_si91x_gspi_set_configuration(sl_gspi_handle_t gspi_handle, sl_gspi_control_config_t *control_configuration)
Control and configure the GSPI.

sl_status_t sl_si91x_gspi_receive_data(sl_gspi_handle_t gspi_handle, void *data, uint32_t data_length)
Receive data from the slave device.

sl_status_t sl_si91x_gspi_send_data(sl_gspi_handle_t gspi_handle, const void *data, uint32_t data_length)
Send data to the slave device.

sl_status_t sl_si91x_gspi_transfer_data(sl_gspi_handle_t gspi_handle, const void *data_out, void *data_in, uint32_t data_length)
Send and receive data to the slave device simultaneously.

sl_status_t sl_si91x_gspi_set_master_state(sl_gspi_handle_t gspi_handle, boolean_t value)
Set the main state, i.e., activate/de-activate the main.

sl_status_t	sl_si91x_gspi_register_event_callback(sl_gspi_handle_t gspi_handle, sl_gspi_signal_event_t callback_event) Register the user callback function.
void	sl_si91x_gspi_unregister_event_callback(void) Un-register the user callback function.
sl_gspi_version_t	sl_si91x_gspi_get_version(void) Get the release, sqa, and dev version of GSPI.
sl_gspi_status_t	sl_si91x_gspi_get_status(sl_gspi_handle_t gspi_handle) Get the transfer status of GSPI.
uint32_t	sl_si91x_gspi_get_rx_data_count(sl_gspi_handle_t gspi_handle) Get data receive count of GSPI.
uint32_t	sl_si91x_gspi_get_tx_data_count(void) Get the transmit data count of GSPI.
int32_t	sl_si91x_gspi_get_clock_division_factor(sl_gspi_handle_t gspi_handle) Fetch the clock division factor.
uint32_t	sl_si91x_gspi_get_frame_length(void) Fetch the frame length, i.e., bit width.
__STATIC_INLINE sl_status_t	sl_si91x_gspi_set_slave_number(uint8_t number) Set the slave number in multi-slave operation.

Enumeration Documentation

gspi_event_typedef_t

> gspi_event_typedef_t

Enumeration for different GSPI callback events.

Enumerator	
SL_GSPI_TRANSFER_COMPLETE	Transfer complete event.
SL_GSPI_DATA_LOST	Data lost event.
SL_GSPI_MODE_FAULT	Mode fault event.

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_gspi_power_state_t

> sl_gspi_power_state_t

Enumeration for GSPI power state.

Enumerator	
SL_GSPI_POWER_OFF	Power mode OFF.
SL_GSPI_LOW_POWER	Low power mode.
SL_GSPI_FULL_POWER	Full power mode.
SL_GSPI_POWER_MODE_LAST	Last member of enum for validation.

Definition at line 63 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

clock_mode_typedef_t

> clock_mode_typedef_t

Enumeration for GSPI clock modes.

Enumerator	
SL_GSPI_MODE_0	Mode 0 CPOL0_CPHA0.
SL_GSPI_MODE_3	Mode 1 CPOL1_CPHA1.

Definition at line 71 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

master_mode_typedef_t

> master_mode_typedef_t

Enumeration for GSPI master modes.

Enumerator	
SL_GSPI_MASTER_INACTIVE	Master mode inactive.
SL_GSPI_MASTER_ACTIVE	Master mode active.
SL_GSPI_MASTER_MODE_LAST	Last member of enum for validation.

Definition at line 77 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

slave_select_mode_typedef_t

> slave_select_mode_typedef_t

Enumeration for GSPI slave select mode.

Enumerator	
SL_GSPI_MASTER_UNUSED	Master unused mode.
SL_GSPI_MASTER_SW	Master software mode.
SL_GSPI_MASTER_HW_OUTPUT	Master hardware output mode.
SL_GSPI_SLAVE_SELECT_MODE_LAST	Last member of enum for validation.

Definition at line 84 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_gspi_instance_t

> sl_gspi_instance_t

Enumeration for GSPI Master instance, enum for one member, future provision is provided.

Enumerator	
SL_GSPI_MASTER	GSPI Master Instance.
SL_GSPI_INSTANCE_LAST_ENUM	Last member of enum for validation.

Definition at line 92 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_gspi_slave_number_t

```
sl_gspi_slave_number_t
```

Enumeration for GSPI Slave numbers.

Enumerator	
GSPI_SLAVE_0	Slave No. 1.
GSPI_SLAVE_1	Slave No. 2.
GSPI_SLAVE_2	Slave No. 2.
GSPI_SLAVE_LAST_ENUM	Last member of enum for validation.

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

Typedef Documentation

sl_gspi_signal_event_t

```
typedef ARM_SPI_SignalEvent_t sl_gspi_signal_event_t
```

Renamed signal event structure.

Definition at line 50 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_gspi_status_t

```
typedef ARM_SPI_STATUS sl_gspi_status_t
```

Renamed status structure.

Definition at line 51 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_gspi_driver_t

```
typedef ARM_DRIVER_SPI sl_gspi_driver_t
```

Renamed GSPI driver structure.

Definition at line 52 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_gspi_handle_t

```
typedef const void* sl_gspi_handle_t
```

Created GSPI handle type.

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

Function Documentation

sl_si91x_gspi_configure_clock

```
sl_status_t sl_si91x_gspi_configure_clock (sl_gspi_clock_config_t *clock_configuration)
```

Set the clock for the GSPI peripheral, Configures the PLL clock and SOC clock with the value set by the user in the clock configuration structure.

Parameters

N/A	clock_configuration	Pointer to the clock configuration structure sl_gspi_clock_config_t

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_FAIL (0x0001) - The function has failed
  - SL_STATUS_NOT_INITIALIZED (0x0011) - Clock is not initialized

Definition at line 148 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_init

```
sl_status_t sl_si91x_gspi_init (sl_gspi_instance_t instance, sl_gspi_handle_t *gspi_handle)
```

Initialize the GSPI.

Parameters

[in]	instance	GSPI Instance.
[in]	gspi_handle	Double Pointer to the GSPI driver handle

If the DMA is enabled, it also initializes the DMA. Pass the address of the pointer for storing the GSPI master handle, which can be used in the future for other function calls.

- Pre-conditions:
  - sl_si91x_gspi_configure_clock

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 166 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_deinit

```
sl_status_t sl_si91x_gspi_deinit (sl_gspi_handle_t gspi_handle)
```

Uninitialize the GSPI.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle

If the DMA is enabled, it also uninitializes the DMA.

- Pre-conditions:
  - sl_si91x_gspi_init

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer

Definition at line 180 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_set_configuration

> sl_status_t sl_si91x_gspi_set_configuration (sl_gspi_handle_t gspi_handle, sl_gspi_control_config_t *control_configuration)

Control and configure the GSPI.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle
[in]	control_configuration	pointer to the configuration structure sl_gspi_control_config_t

- swap_read (enable/disable)
- swap_write (enable/disable)
- bit_width (8_bit/16_bit)
- clock_mode (mode0/mode3)
- slave_select_mode (hw_output/sw)
- bitrate

- Pre-conditions:
  - sl_si91x_gspi_configure_clock
  - sl_si91x_gspi_init
  - sl_si91x_gspi_configure_power_mode

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_FAIL (0x0001) - The function has failed
  - SL_STATUS_NOT_SUPPORTED (0x000F) - Parameter is not supported
  - SL_STATUS_BUSY (0x0004) - Driver is busy
  - SL_STATUS_INVALID_MODE (0x0024) - Slave select Mode is invalid
  - SL_STATUS_INVALID_TYPE (0x0026) - SPI frame format is not valid
  - SL_STATUS_INVALID_RANGE (0x0028) - Data bits (frame length) are not in range

Definition at line 210 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_receive_data

> sl_status_t sl_si91x_gspi_receive_data (sl_gspi_handle_t gspi_handle, void *data, uint32_t data_length)

Receive data from the slave device.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle
[in]	data	pointer to the variable that will store the received data
[in]	data_length	(uint32_t) number of data items to receive

- Pre-conditions:
  - sl_si91x_gspi_configure_clock
  - sl_si91x_gspi_init
  - sl_si91x_gspi_configure_power_mode
  - sl_si91x_gspi_set_configuration
  - sl_si91x_gspi_set_slave_number

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_FAIL (0x0001) - The function has failed
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 234 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_send_data

> sl_status_t sl_si91x_gspi_send_data (sl_gspi_handle_t gspi_handle, const void *data, uint32_t data_length)

Send data to the slave device.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle
[in]	data	const pointer to the variable that has data which needs to be sent
[in]	data_length	(uint32_t) number of data items to send

- Pre-conditions:
  - sl_si91x_gspi_configure_clock
  - sl_si91x_gspi_init
  - sl_si91x_gspi_configure_power_mode
  - sl_si91x_gspi_set_configuration
  - sl_si91x_gspi_set_slave_number

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_FAIL (0x0001) - The function has failed
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 257 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_transfer_data

> sl_status_t sl_si91x_gspi_transfer_data (sl_gspi_handle_t gspi_handle, const void *data_out, void *data_in, uint32_t data_length)

Send and receive data to the slave device simultaneously.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle
[in]	data_out	const pointer to the variable that has data which needs to be sent
[in]	data_in	pointer to the variable that will store the received data
[in]	data_length	(uint32_t) number of data items to receive

- Pre-conditions:
  - sl_si91x_gspi_configure_clock
  - sl_si91x_gspi_init
  - sl_si91x_gspi_configure_power_mode
  - sl_si91x_gspi_set_configuration
  - sl_si91x_gspi_set_slave_number

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_FAIL (0x0001) - The function has failed
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 281 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_set_master_state

```
sl_status_t sl_si91x_gspi_set_master_state (sl_gspi_handle_t gspi_handle, boolean_t value)
```

Set the main state, i.e., activate/de-activate the main.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle
[in]	value	(boolean_t) Enable or Disable

- Pre-conditions:
  - sl_si91x_gspi_configure_clock
  - sl_si91x_gspi_init
  - sl_si91x_gspi_configure_power_mode

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_BUSY (0x0004) - Driver is busy
  - SL_STATUS_INVALID_MODE (0x0024) - Mode is invalid (Fails to activate master)

Definition at line 303 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_register_event_callback

```
sl_status_t sl_si91x_gspi_register_event_callback (sl_gspi_handle_t gspi_handle, sl_gspi_signal_event_t callback_event)
```

Register the user callback function.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle
[in]	callback_event	Pointer to the function that needs to be called at the time of interrupt

Returns

- status
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line `316` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h`

sl_si91x_gspi_unregister_event_callback

> void sl_si91x_gspi_unregister_event_callback (void)

Un-register the user callback function.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_gspi_register_event_callback

Returns

- none

Definition at line `328` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h`

sl_si91x_gspi_get_version

> sl_gspi_version_t sl_si91x_gspi_get_version (void)

Get the release, sqa, and dev version of GSPI.

Parameters

[in]		

Returns

- (sl_gspi_version_t) type structure

Definition at line `337` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h`

sl_si91x_gspi_get_status

> sl_gspi_status_t sl_si91x_gspi_get_status (sl_gspi_handle_t gspi_handle)

Get the transfer status of GSPI.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle

- Pre-conditions:
  - sl_si91x_gspi_configure_clock
  - sl_si91x_gspi_init
  - sl_si91x_gspi_configure_power_mode
  - sl_si91x_gspi_set_configuration

Returns

- sl_gspi_status_t type structure

Definition at line 352 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_get_rx_data_count

```
uint32_t sl_si91x_gspi_get_rx_data_count (sl_gspi_handle_t gspi_handle)
```

Get data receive count of GSPI.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle

- Pre-conditions:
  - sl_si91x_gspi_configure_clock
  - sl_si91x_gspi_init
  - sl_si91x_gspi_configure_power_mode
  - sl_si91x_gspi_set_configuration

Returns

- uint32_t value of the RX data count

Definition at line 367 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_get_tx_data_count

```
uint32_t sl_si91x_gspi_get_tx_data_count (void)
```

Get the transmit data count of GSPI.

Parameters

[in]		Pointer to the GSPI driver handle

- Pre-conditions:
  - sl_si91x_gspi_configure_clock
  - sl_si91x_gspi_init
  - sl_si91x_gspi_configure_power_mode
  - sl_si91x_gspi_set_configuration

Returns

- uint32_t value of the tx data count

Definition at line 382 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_si91x_gspi_get_clock_division_factor

```
int32_t sl_si91x_gspi_get_clock_division_factor (sl_gspi_handle_t gspi_handle)
```

Fetch the clock division factor.

Parameters

[in]	gspi_handle	Pointer to the GSPI driver handle

Returns

- factor (int32_t) The value of the clock division factor

Definition at line `391` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h`

sl_si91x_gspi_get_frame_length

```
uint32_t sl_si91x_gspi_get_frame_length (void)
```

Fetch the frame length, i.e., bit width.

Parameters

[in]		

Returns

- frame_length (uint32_t) The value of the frame length

Definition at line `400` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h`

sl_si91x_gspi_set_slave_number

```
__STATIC_INLINE sl_status_t sl_si91x_gspi_set_slave_number (uint8_t number)
```

Set the slave number in multi-slave operation.

Parameters

[in]	number	Slave number

For a single slave also, this API needs to be called before transferring the data

Returns

- none

Definition at line `411` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h`

sl_gspi_control_config_t

Structure to hold the parameters for the configuration of GSPI peripheral.

Public Attributes

boolean_t	swap_read true to enable and false to disable swap read
boolean_t	swap_write true to enable and false to disable swap write
uint8_t	bit_width Bit width either 8 bit or 16 bit.
uint32_t	clock_mode Mode 0 or Mode 3 of GSPI.
uint32_t	slave_select_mode Slave select mode either software of hardware output.
uint32_t	bitrate Bitrate for setting the clock division factor.

Public Attribute Documentation

swap_read

```
boolean_t sl_gspi_control_config_t::swap_read
```

true to enable and false to disable swap read

Definition at line 107 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

swap_write

```
boolean_t sl_gspi_control_config_t::swap_write
```

true to enable and false to disable swap write

Definition at line 108 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

bit_width

```
uint8_t sl_gspi_control_config_t::bit_width
```

Bit width either 8 bit or 16 bit.

Definition at line 109 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

clock_mode

> uint32_t sl_gspi_control_config_t::clock_mode

Mode 0 or Mode 3 of GSPI.

Definition at line 110 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

slave_select_mode

> uint32_t sl_gspi_control_config_t::slave_select_mode

Slave select mode either software of hardware output.

Definition at line 111 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

bitrate

> uint32_t sl_gspi_control_config_t::bitrate

Bitrate for setting the clock division factor.

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_gspi_clock_config_t

Structure to hold the clock configuration parameters.

Public Attributes

uint8_t	soc_pll_mm_count_value SoC PLL count value.
uint16_t	intf_pll_500_control_value Int PLL control value.
uint32_t	intf_pll_clock Intf PLL clock frequency.
uint32_t	intf_pll_reference_clock Intf PLL reference clock frequency.
uint32_t	soc_pll_clock SoC PLL clock frequency.
uint32_t	soc_pll_reference_clock SoC PLL reference clock frequency.
uint16_t	division_factor Clock Division Factor.

Public Attribute Documentation

soc_pll_mm_count_value

```
uint8_t sl_gspi_clock_config_t::soc_pll_mm_count_value
```

SoC PLL count value.

Definition at line 117 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

intf_pll_500_control_value

```
uint16_t sl_gspi_clock_config_t::intf_pll_500_control_value
```

Int PLL control value.

Definition at line 118 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

intf_pll_clock

```
uint32_t sl_gspi_clock_config_t::intf_pll_clock
```

Intf PLL clock frequency.

Definition at line 119 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

intf_pll_reference_clock

```
uint32_t sl_gspi_clock_config_t::intf_pll_reference_clock
```

Intf PLL reference clock frequency.

Definition at line 120 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

soc_pll_clock

```
uint32_t sl_gspi_clock_config_t::soc_pll_clock
```

SoC PLL clock frequency.

Definition at line 121 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

soc_pll_reference_clock

```
uint32_t sl_gspi_clock_config_t::soc_pll_reference_clock
```

SoC PLL reference clock frequency.

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

division_factor

```
uint16_t sl_gspi_clock_config_t::division_factor
```

Clock Division Factor.

Definition at line 123 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

sl_gspi_version_t

Structure to hold the versions number of peripheral API.

Public Attributes

uint8_t	release	Release version number.
uint8_t	major	sqa version number
uint8_t	minor	dev version number

Public Attribute Documentation

release

```
uint8_t sl_gspi_version_t::release
```

Release version number.

Definition at line 128 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

major

```
uint8_t sl_gspi_version_t::major
```

sqa version number

Definition at line 129 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

minor

```
uint8_t sl_gspi_version_t::minor
```

dev version number

Definition at line 130 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_gspi.h

I2C

Modules

sl_i2c_config_t

sl_i2c_dma_config_t

sl_i2c_transfer_config_t

sl_i2c_pin_init_t

Enumerations

enum sl_i2c_instance_t {

SL_I2C0
SL_I2C1
SL_I2C2
SL_I2C_LAST

}

Enumeration to represent i2c instances.

enum sl_i2c_status_t {

SL_I2C_SUCCESS
SL_I2C_IDLE
SL_I2C_7BIT_ADD
SL_I2C_10BIT_ADD
SL_I2C_10BIT_ADD_WITH_REP_START
SL_I2C_ACKNOWLEDGE
SL_I2C_NACK
SL_I2C_DATA_TRANSFER_COMPLETE
SL_I2C_ARIBITRATION_LOST
SL_I2C_BUS_ERROR
SL_I2C_BUS_HOLD
SL_I2C_SDA_ERROR
SL_I2C_SCL_ERROR
SL_I2C_CALLBACK_BUSY
SL_I2C_DMA_TRANSFER_ERROR
SL_I2C_INVALID_PARAMETER

}

Enumeration to represent i2c driver status values.

enum sl_i2c_transfer_type_t {

SL_I2C_USING_INTERRUPT
SL_I2C_USING_DMA
SL_I2C_TRANFER_TYPE_LAST

}

Enumeration to represent i2c transfer type.

enum sl_i2c_operating_mode_t {

SL_I2C_STANDARD_MODE = 1
SL_I2C_FAST_MODE
SL_I2C_HIGH_SPEED_MODE
SL_I2C_FAST_PLUS_MODE
SL_I2C_OPERATING_MODE_LAST

}

Enumeration to represent i2c operating mode.

Typedefs

typedef void(* sl_i2c_callback_t)(sl_i2c_instance_t i2c_instance, uint32_t status)

Callback for I2C Driver.

Functions

sl_i2c_status_t sl_i2c_driver_init(sl_i2c_instance_t i2c_instance, const sl_i2c_config_t *p_user_config)

Initialize the I2C Module and clock.

sl_i2c_status_t sl_i2c_driver_set_follower_address(sl_i2c_instance_t i2c_instance, uint16_t address)

This API configures the follower address of the I2C module.

sl_i2c_status_t sl_i2c_driver_configure_fifo_threshold(sl_i2c_instance_t i2c_instance, uint8_t tx_threshold_value, uint8_t rx_threshold_value)

This function is used to configure the TX and RX FIFO threshold values.

sl_i2c_status_t sl_i2c_driver_get_frequency(sl_i2c_instance_t i2c_instance, uint32_t *frequency)

This API gets the current frequency of I2C transfer in MHz, by reading the system core clock frequency.

sl_i2c_status_t sl_i2c_driver_send_data_blocking(sl_i2c_instance_t i2c_instance, uint16_t address, uint8_t *tx_buffer, uint32_t tx_len)

This function sends the data in blocking mode (using interrupt), also sets the follower address when used in Leader application.

sl_i2c_status_t sl_i2c_driver_send_data_non_blocking(sl_i2c_instance_t i2c_instance, uint16_t address, uint32_t *tx_buffer, uint32_t tx_len, sl_i2c_dma_config_t *p_dma_config)

This function sends the data in non-blocking mode (using DMA), also sets the follower address when used in Leader application.

sl_i2c_status_t sl_i2c_driver_receive_data_blocking(sl_i2c_instance_t i2c_instance, uint16_t address, uint8_t *rx_buffer, uint32_t rx_len)

This function receives the data in blocking mode (using interrupt), also sets the follower address when used in Leader application.

sl_i2c_status_t sl_i2c_driver_receive_data_non_blocking(sl_i2c_instance_t i2c_instance, uint16_t address, uint8_t *rx_buffer, uint32_t rx_len, sl_i2c_dma_config_t *p_dma_config)

This function receives the data in non-blocking mode (using DMA), also sets the follower address when used in Leader application.

sl_i2c_status_t sl_i2c_driver_transfer_data(sl_i2c_instance_t i2c_instance, sl_i2c_transfer_config_t const *p_transfer_config, uint16_t address)

This function transmits and receives the data in blocking mode also sets the follower address when used in Leader application.

sl_i2c_status_t sl_i2c_driver_deinit(sl_i2c_instance_t i2c_instance)

De-initializes the I2C peripheral, disables clock, and unregisters the callback.

sl_i2c_status_t sl_si91x_i2c_pin_init(sl_i2c_pin_init_t *pin_init)

Set the Pin configuration for I2C.

Macros

#define SL_I2C0_DMA_TX_CHANNEL 31
I2C Peripheral.

#define SL_I2C0_DMA_RX_CHANNEL 30

#define SL_I2C1_DMA_TX_CHANNEL 3

#define SL_I2C1_DMA_RX_CHANNEL 2

#define SL_I2C2_DMA_TX_CHANNEL 5

#define SL_I2C2_DMA_RX_CHANNEL 4

Enumeration Documentation

sl_i2c_instance_t

> sl_i2c_instance_t

Enumeration to represent i2c instances.

Enumerator	
SL_I2C0	I2C Instance 0.
SL_I2C1	I2C Instance 1.
SL_I2C2	I2C Instance 2.
SL_I2C_LAST	Last member of enum for validation.

Definition at line 82 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_status_t

> sl_i2c_status_t

Enumeration to represent i2c driver status values.

Enumerator	
SL_I2C_SUCCESS	Success.
SL_I2C_IDLE	I2C Idle.
SL_I2C_7BIT_ADD	I2C 7 Bit address transfer.
SL_I2C_10BIT_ADD	I2C 10 Bit address transfer.
SL_I2C_10BIT_ADD_WITH_REP_START	I2C 10 Bit address transfer with Repeated Start.
SL_I2C_ACKNOWLEDGE	I2C ACK.
SL_I2C_NACK	I2C NACK.
SL_I2C_DATA_TRANSFER_COMPLETE	I2C data transfer complete.
SL_I2C_ARIBITRATION_LOST	I2C Arbitration Lost.
SL_I2C_BUS_ERROR	I2C Bus Error.
SL_I2C_BUS_HOLD	I2C Bus held by I2C Bus.
SL_I2C_SDA_ERROR	I2C SDA at loopback path is not equal to SDA output.

SL_I2C_SCL_ERROR	I2C SCL at loopback path is not equal to SCL output.
SL_I2C_CALLBACK_BUSY	I2C instance callback already registered.
SL_I2C_DMA_TRANSFER_ERROR	I2C DMA transfer error.
SL_I2C_INVALID_PARAMETER	Invalid Parameter.

Definition at line 90 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_transfer_type_t

> sl_i2c_transfer_type_t

Enumeration to represent i2c transfer type.

Enumerator	
SL_I2C_USING_INTERRUPT	The driver will use interrupts to perform I2C transfer.
SL_I2C_USING_DMA	The driver will use DMA to perform I2C transfer.
SL_I2C_TRANFER_TYPE_LAST	For Validation.

Definition at line 110 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_operating_mode_t

> sl_i2c_operating_mode_t

Enumeration to represent i2c operating mode.

Enumerator	
SL_I2C_STANDARD_MODE	Standard-mode, bidirectional data transfers up to 100 kbit/s.
SL_I2C_FAST_MODE	Fast-mode, bidirectional data transfers up to 400 kbit/s.
SL_I2C_HIGH_SPEED_MODE	High speed-mode, bidirectional data transfers up to 3.4 Mbit/s.
SL_I2C_FAST_PLUS_MODE	Fast-mode Plus, bidirectional data transfers up to 1 Mbit/s.
SL_I2C_OPERATING_MODE_LAST	Last member of enum for validation.

Definition at line 117 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

Typedef Documentation

sl_i2c_callback_t

> typedef void(* sl_i2c_callback_t) (sl_i2c_instance_t i2c_instance, uint32_t status))(sl_i2c_instance_t i2c_instance, uint32_t status)

Callback for I2C Driver.

Definition at line 125 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

Function Documentation

sl_i2c_driver_init

> sl_i2c_status_t sl_i2c_driver_init (sl_i2c_instance_t i2c_instance, const sl_i2c_config_t *p_user_config)

Initialize the I2C Module and clock.

Parameters

[in]	i2c_instance	I2C Instance to be initialized sl_i2c_instance_t
[in]	p_user_config	A pointer to I2C configuration structure sl_i2c_config_t

Sets I2C instance mode, operating mode (bus-speed), frequency, and transfer type (using Interrupt or DMA). If the transfer type is DMA, it initializes DMA as well. Registers I2C instance callback and clears pending interrupts. Configures SDL and SCL pins as per the instance.

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid
  - SL_I2C_CALLBACK_BUSY (0x0D) - Driver is busy

Definition at line `178` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h`

sl_i2c_driver_set_follower_address

> sl_i2c_status_t sl_i2c_driver_set_follower_address (sl_i2c_instance_t i2c_instance, uint16_t address)

This API configures the follower address of the I2C module.

Parameters

[in]	i2c_instance	I2C Instance sl_i2c_instance_t
[in]	address	Follower own address

Should be used only in Follower mode after instance initialization.

- Pre-conditions:
  - sl_i2c_driver_init
  - Here I2C mode should be set as follower mode

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid

Definition at line `195` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h`

sl_i2c_driver_configure_fifo_threshold

> sl_i2c_status_t sl_i2c_driver_configure_fifo_threshold (sl_i2c_instance_t i2c_instance, uint8_t tx_threshold_value, uint8_t rx_threshold_value)

This function is used to configure the TX and RX FIFO threshold values.

Parameters

[in]	i2c_instance	I2C Instance sl_i2c_instance_t

[in]	tx_threshold_value	Transmit FIFO threshold value
[in]	rx_threshold_value	Receive FIFO threshold value

The valid range is 0-255, with the additional restriction that hardware does not allow this value to be set to a value larger than the depth of the buffer. A value of 0 sets the threshold for 1 entry, and a value of 255 sets the threshold for 256 entries.

- Pre-conditions:
  - sl_i2c_driver_init

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid

Definition at line 215 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_driver_get_frequency

> sl_i2c_status_t sl_i2c_driver_get_frequency (sl_i2c_instance_t i2c_instance, uint32_t *frequency)

This API gets the current frequency of I2C transfer in MHz, by reading the system core clock frequency.

Parameters

[in]	i2c_instance	I2C Instance sl_i2c_instance_t
[out]	frequency	Currently configured frequency.

- Pre-conditions:
  - sl_i2c_driver_init

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid

Definition at line 233 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_driver_send_data_blocking

> sl_i2c_status_t sl_i2c_driver_send_data_blocking (sl_i2c_instance_t i2c_instance, uint16_t address, uint8_t *tx_buffer, uint32_t tx_len)

This function sends the data in blocking mode (using interrupt), also sets the follower address when used in Leader application.

Parameters

[in]	i2c_instance	I2C Instance sl_i2c_instance_t
[in]	address	Follower address, in follower mode, this parameter will be ignored
[in]	tx_buffer	A pointer to transmit data buffer
[in]	tx_len	Data length in number of bytes

Sets transmit empty interrupt and enables I2C interrupts.

- Pre-conditions:

sl_i2c_driver_init
- Here transfer-type should be set as interrupt-type
- sl_i2c_driver_set_follower_address (if sending from slave)
- sl_i2c_driver_configure_fifo_threshold

Note

- Maximum tx_len used can be 80000 (receives in around 10 seconds)

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid

Definition at line 256 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_driver_send_data_non_blocking

> sl_i2c_status_t sl_i2c_driver_send_data_non_blocking (sl_i2c_instance_t i2c_instance, uint16_t address, uint32_t *tx_buffer, uint32_t tx_len, sl_i2c_dma_config_t *p_dma_config)

This function sends the data in non-blocking mode (using DMA), also sets the follower address when used in Leader application.

Parameters

[in]	i2c_instance	I2C Instance.
[in]	address	Follower address
[in]	tx_buffer	A pointer to transmit data buffer
[in]	tx_len	Data length in number of bytes
[in]	p_dma_config	A pointer to DMA configuration structure sl_i2c_dma_config_t

Configures DMA rx and tx channels.

- Pre-conditions:
  - sl_i2c_driver_init
  - Here transfer-type should be set as DMA-type
  - sl_i2c_driver_set_follower_address, if used in salve application

Note

- Maximum tx_len values can be 30000 (receives back in around 4 seconds)

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid
  - SL_DMA_TRANSFER_ERROR - DMA parameters are invalid

Definition at line 283 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_driver_receive_data_blocking

> sl_i2c_status_t sl_i2c_driver_receive_data_blocking (sl_i2c_instance_t i2c_instance, uint16_t address, uint8_t *rx_buffer, uint32_t rx_len)

This function receives the data in blocking mode (using interrupt), also sets the follower address when used in Leader application.

Parameters

[in]	i2c_instance	I2C Instance sl_i2c_instance_t
[in]	address	Follower address, in follower mode this parameter will be ignored
[in]	rx_buffer	A pointer to receive data buffer
[in]	rx_len	Data length in number of bytes

Sets receive full interrupt while receiving data. Sets transmit empty interrupt while sending data.

- Pre-conditions:
  - sl_i2c_driver_init
  - Here transfer-type should be set as interrupt-type
  - sl_i2c_driver_set_follower_address, if used in follower application
  - sl_i2c_driver_configure_fifo_threshold

Note

- Maximum rx_len used can be 80000 (receives in around 10 seconds)

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid

Definition at line `311` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h`

sl_i2c_driver_receive_data_non_blocking

> sl_i2c_status_t sl_i2c_driver_receive_data_non_blocking (sl_i2c_instance_t i2c_instance, uint16_t address, uint8_t *rx_buffer, uint32_t rx_len, sl_i2c_dma_config_t *p_dma_config)

This function receives the data in non-blocking mode (using DMA), also sets the follower address when used in Leader application.

Parameters

[in]	i2c_instance	I2C Instance sl_i2c_instance_t
[in]	address	Follower address
[in]	rx_buffer	A pointer to receive data buffer
[in]	rx_len	Data length in number of bytes
[in]	p_dma_config	A pointer to DMA configuration structure sl_i2c_dma_config_t

Configures DMA rx and tx channels.

- Pre-conditions:
  - sl_i2c_driver_init
  - Here transfer-type should be set as DMA-type
  - sl_i2c_driver_set_follower_address, if used in salve application

Note

- Maximum rx_len values can be 30000 (receives back in around 4 seconds)

Returns

- status 0 if successful, else error code as follow

SL_I2C_SUCCESS (0x0000) - Success
- SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid
- SL_DMA_TRANSFER_ERROR - DMA parameters are invalid

Definition at line 338 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_driver_transfer_data

> sl_i2c_status_t sl_i2c_driver_transfer_data (sl_i2c_instance_t i2c_instance, sl_i2c_transfer_config_t const *p_transfer_config, uint16_t address)

This function transmits and receives the data in blocking mode also sets the follower address when used in Leader application.

Parameters

[in]	i2c_instance	I2C Instance sl_i2c_instance_t
[in]	p_transfer_config	Follower address
[in]	address	A pointer to transfer configuration structure sl_i2c_transfer_config_t

Sets receive full interrupt and enables I2C interrupts.

- Pre-conditions:
  - sl_i2c_driver_init
  - Here transfer-type should be set as Interrupt-type
  - sl_i2c_driver_set_follower_address, if used in follower application

Note

- Maximum tx_len & rx_len values can be 80000 (sends & receives back in around 20 seconds)

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid

Definition at line 363 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_driver_deinit

> sl_i2c_status_t sl_i2c_driver_deinit (sl_i2c_instance_t i2c_instance)

De-initializes the I2C peripheral, disables clock, and unregisters the callback.

Parameters

[in]	i2c_instance	I2C Instance sl_i2c_instance_t

- Pre-conditions:
  - sl_i2c_driver_init

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid

Definition at line 379 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_si91x_i2c_pin_init

```
sl_i2c_status_t sl_si91x_i2c_pin_init (sl_i2c_pin_init_t *pin_init)
```

Set the Pin configuration for I2C.

Parameters

[in]	pin_init	Pointer to pin init structure sl_i2c_pin_init_t

It configures the SDA and SCL pins.

Returns

- status 0 if successful, else error code as follow
  - SL_I2C_SUCCESS (0x0000) - Success
  - SL_I2C_INVALID_PARAMETER (0x0F) - Parameters are invalid

Definition at line 390 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

Macro Definition Documentation

SL_I2C0_DMA_TX_CHANNEL

```
#define SL_I2C0_DMA_TX_CHANNEL
```

Value:

```
31
```

I2C Peripheral.

Overview

I2C standard compliant bus interface with open-drain pins Configurable as Master or Slave Four speed modes: Standard Mode (100 kbps), Fast Mode (400 kbps), Fast Mode Plus (1Mbps) and High-Speed Mode (3.4 Mbps) 7 or 10-bit addressing 7 or 10-bit combined format transfers Support for Clock synchronization and Bus Clear Programmable SDA Hold time Integrated transmit and receive buffers with support for DMA Bulk transmit mode in I2C Slave mode Interrupt based operation (polled mode also available)

Initialization

Call init API with the init parameters. Call set follower address API, in follower mode Call Set FIFO threshold API

Definition at line 72 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

SL_I2C0_DMA_RX_CHANNEL

```
#define SL_I2C0_DMA_RX_CHANNEL
```

Value:

```
30
```

Definition at line 73 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

SL_I2C1_DMA_TX_CHANNEL

```
#define SL_I2C1_DMA_TX_CHANNEL
```

Value:

```
3
```

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

SL_I2C1_DMA_RX_CHANNEL

```
#define SL_I2C1_DMA_RX_CHANNEL
```

Value:

```
2
```

Definition at line 75 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

SL_I2C2_DMA_TX_CHANNEL

```
#define SL_I2C2_DMA_TX_CHANNEL
```

Value:

```
5
```

Definition at line 76 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

SL_I2C2_DMA_RX_CHANNEL

```
#define SL_I2C2_DMA_RX_CHANNEL
```

Value:

```
4
```

Definition at line 77 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_config_t

Structure to hold the parameters of i2c instance configurations.

Public Attributes

sl_i2c_mode_t	mode Leader/Follower Mode, 0 for leader mode and 1 for follower mode.
sl_i2c_operating_mode_t	operating_mode Speed mode sl_i2c_operating_mode_t for possible values.
sl_i2c_transfer_type_t	transfer_type Transfer type sl_i2c_transfer_type_t for possible values.
sl_i2c_callback_t	i2c_callback I2C callback sl_i2c_callback_t.

Public Attribute Documentation

mode

```
sl_i2c_mode_t sl_i2c_config_t::mode
```

Leader/Follower Mode, 0 for leader mode and 1 for follower mode.

Definition at line 129 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

operating_mode

```
sl_i2c_operating_mode_t sl_i2c_config_t::operating_mode
```

Speed mode sl_i2c_operating_mode_t for possible values.

Definition at line 130 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

transfer_type

```
sl_i2c_transfer_type_t sl_i2c_config_t::transfer_type
```

Transfer type sl_i2c_transfer_type_t for possible values.

Definition at line 131 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

i2c_callback

sl_i2c_callback_t sl_i2c_config_t::i2c_callback

I2C callback sl_i2c_callback_t.

Definition at line 132 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_dma_config_t

Structure to hold the parameters of DMA configuration.

Public Attributes

uint32_t	dma_tx_channel DMA transmit channel number.
uint32_t	dma_rx_channel DMA receive channel number.

Public Attribute Documentation

dma_tx_channel

```
uint32_t sl_i2c_dma_config_t::dma_tx_channel
```

DMA transmit channel number.

Definition at line 137 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

dma_rx_channel

```
uint32_t sl_i2c_dma_config_t::dma_rx_channel
```

DMA receive channel number.

Definition at line 138 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_transfer_config_t

Structure to hold the parameters of I2C transfer configuration.

Public Attributes

uint8_t *	tx_buffer Pointer to Tx Data buffer.
uint32_t	tx_len Number of bytes to transmit.
uint8_t *	rx_buffer Pointer to Rx Data buffer.
uint32_t	rx_len Number of bytes to receive.

Public Attribute Documentation

tx_buffer

```
uint8_t* sl_i2c_transfer_config_t::tx_buffer
```

Pointer to Tx Data buffer.

Definition at line 143 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

tx_len

```
uint32_t sl_i2c_transfer_config_t::tx_len
```

Number of bytes to transmit.

Definition at line 144 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

rx_buffer

```
uint8_t* sl_i2c_transfer_config_t::rx_buffer
```

Pointer to Rx Data buffer.

Definition at line 145 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

rx_len

uint32_t sl_i2c_transfer_config_t::rx_len

Number of bytes to receive.

Definition at line 146 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sl_i2c_pin_init_t

Structure to hold port and pin of i2c.

Public Attributes

uint8_t	sda_port PWM GPIO port.
uint8_t	sda_pin PWM GPIO pin.
uint8_t	sda_mux PWM GPIO mux.
uint8_t	sda_pad PWM GPIO pad.
uint8_t	scl_port PWM GPIO port.
uint8_t	scl_pin PWM GPIO pin.
uint8_t	scl_mux PWM GPIO mux.
uint8_t	scl_pad PWM GPIO pad.

Public Attribute Documentation

sda_port

```
uint8_t sl_i2c_pin_init_t::sda_port
```

PWM GPIO port.

Definition at line 151 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sda_pin

```
uint8_t sl_i2c_pin_init_t::sda_pin
```

PWM GPIO pin.

Definition at line 152 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sda_mux

```
uint8_t sl_i2c_pin_init_t::sda_mux
```

PWM GPIO mux.

Definition at line 153 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

sda_pad

```
uint8_t sl_i2c_pin_init_t::sda_pad
```

PWM GPIO pad.

Definition at line 154 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

scl_port

```
uint8_t sl_i2c_pin_init_t::scl_port
```

PWM GPIO port.

Definition at line 155 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

scl_pin

```
uint8_t sl_i2c_pin_init_t::scl_pin
```

PWM GPIO pin.

Definition at line 156 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

scl_mux

```
uint8_t sl_i2c_pin_init_t::scl_mux
```

PWM GPIO mux.

Definition at line 157 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

scl_pad

```
uint8_t sl_i2c_pin_init_t::scl_pad
```

PWM GPIO pad.

Definition at line 158 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2c.h

I2S

I2S

Modules

sl_i2s_version_t

sl_i2s_xfer_config_t

Enumerations

enum i2s_event_typedef_t {

SL_I2S_SEND_COMPLETE = ARM_SAI_EVENT_SEND_COMPLETE
SL_I2S_RECEIVE_COMPLETE = ARM_SAI_EVENT_RECEIVE_COMPLETE
SL_I2S_TX_UNDERFLOW = ARM_SAI_EVENT_TX_UNDERFLOW
SL_I2S_RX_OVERFLOW = ARM_SAI_EVENT_RX_OVERFLOW
SL_I2S_FRAME_ERROR = ARM_SAI_EVENT_FRAME_ERROR

}

Enumeration for different I2S callback events.

enum sl_i2s_power_state_t {

SL_I2S_POWER_OFF = ARM_POWER_OFF
SL_I2S_LOW_POWER = ARM_POWER_LOW
SL_I2S_FULL_POWER = ARM_POWER_FULL

}

Enumeration for I2S power state.

enum sl_i2s_mode_t {

SL_I2S_MASTER = ARM_SAI_MODE_MASTER
SL_I2S_SLAVE = ARM_SAI_MODE_SLAVE

}

Enumeration for I2S master and slave modes.

enum sl_sai_protocol_t {

SL_I2S_PROTOCOL = ARM_SAI_PROTOCOL_I2S
SL_PCM_PROTOCOL = 0

}

Enumeration for SAI protocol type.

enum sl_i2s_sync_t {

SL_I2S_SYNC = ARM_SAI_SYNCHRONOUS
SL_I2S_ASYNC = ARM_SAI_ASYNCHRONOUS

}

Enumeration for I2S SYNC/ASYNC modes.

enum sl_i2s_xfer_type_t {

SL_I2S_TRANSMIT = ARM_SAI_CONFIGURE_TX
SL_I2S_RECEIVE = ARM_SAI_CONFIGURE_RX
SL_I2S_TRANSMIT_CONTROL = ARM_SAI_CONTROL_TX
SL_I2S_RECEIVE_CONTROL = ARM_SAI_CONTROL_RX
SL_I2S_SEND_ABORT = ARM_SAI_ABORT_SEND
SL_I2S_RECEIVE_ABORT = ARM_SAI_ABORT_RECEIVE

}

Enumeration for I2S transfer type.

enum sl_i2s_xfer_size_t {

SL_I2S_DATA_SIZE16 = 16
SL_I2S_DATA_SIZE32 = 32

}

Enumeration for I2S transfer data size.

enum sl_i2s_data_resolution_t {

SL_I2S_RESOLUTION_12 = 12
SL_I2S_RESOLUTION_16 = 16
SL_I2S_RESOLUTION_20 = 20
SL_I2S_RESOLUTION_24 = 24
SL_I2S_RESOLUTION_32 = 32

}

Enumeration for I2S audio data resolutions.

enum sl_i2s_sampling_rate_t {

SL_I2S_SAMPLING_RATE_8000 = 8000
SL_I2S_SAMPLING_RATE_11025 = 11025
SL_I2S_SAMPLING_RATE_16000 = 16000
SL_I2S_SAMPLING_RATE_22050 = 22050
SL_I2S_SAMPLING_RATE_24000 = 24000
SL_I2S_SAMPLING_RATE_32000 = 32000
SL_I2S_SAMPLING_RATE_44100 = 44100
SL_I2S_SAMPLING_RATE_48000 = 48000
SL_I2S_SAMPLING_RATE_88200 = 88200
SL_I2S_SAMPLING_RATE_96000 = 96000
SL_I2S_SAMPLING_RATE_192000 = 192000

}

Enumeration for I2S sample rates.

Typedefs

typedef ARM_SAI_SignalEvent_t sl_i2s_signal_event_t

Renamed signal event structure.

typedef ARM_SAI_STATUS sl_i2s_status_t

Renamed status structure.

typedef ARM_DRIVER_SAI sl_i2s_driver_t

Renamed I2S driver structure.

typedef const void * sl_i2s_handle_t

Created I2S handle type.

Functions

sl_status_t	sl_si91x_i2s_init(uint32_t i2s_instance, sl_i2s_handle_t *i2s_handle) Initialize the I2S.
sl_status_t	sl_si91x_i2s_deinit(sl_i2s_handle_t *i2s_handle) Uninitialize I2S peripheral.
sl_status_t	sl_si91x_i2s_configure_power_mode(sl_i2s_handle_t i2s_handle, sl_i2s_power_state_t state) Change the power mode of I2S, the supported modes are.
sl_status_t	sl_si91x_i2s_config_transmit_receive(sl_i2s_handle_t i2s_handle, sl_i2s_xfer_config_t *xfer_config) Configure transmitter/Receiver parameters for I2S transfer.
sl_status_t	sl_si91x_i2s_transmit_data(sl_i2s_handle_t i2s_handle, const void *data, uint32_t size) Configure I2S peripheral DMA channel to send data.
sl_status_t	sl_si91x_i2s_receive_data(sl_i2s_handle_t i2s_handle, const void *data, uint32_t size) Configure I2S peripheral DMA channel to receive data.
sl_status_t	sl_si91x_i2s_register_event_callback(sl_i2s_handle_t i2s_handle, sl_i2s_signal_event_t callback_event) Register the user callback function.
sl_status_t	sl_si91x_i2s_unregister_event_callback(sl_i2s_handle_t i2s_handle) Un-register the user callback function.
uint32_t	sl_si91x_i2s_get_transmit_data_count(sl_i2s_handle_t i2s_handle) Get the transmit data count of I2S.
uint32_t	sl_si91x_i2s_get_receive_data_count(sl_i2s_handle_t i2s_handle) Get the receive data count of I2S.
sl_i2s_version_t	sl_si91x_i2s_get_version(void) Get the release, sqa, and dev version of I2S.
sl_i2s_status_t	sl_si91x_i2s_get_status(sl_i2s_handle_t i2s_handle) Get the transfer status I2S.
sl_status_t	sl_si91x_i2s_end_transfer(sl_i2s_handle_t i2s_handle, sl_i2s_xfer_type_t abort_type) Abort I2S Tx/Rx operations.

Enumeration Documentation

i2s_event_typedef_t

```
i2s_event_typedef_t
```

Enumeration for different I2S callback events.

Enumerator	
SL_I2S_SEND_COMPLETE	Send complete event.
SL_I2S_RECEIVE_COMPLETE	Receive lost event.
SL_I2S_TX_UNDERFLOW	Tx underflow event.
SL_I2S_RX_OVERFLOW	Rx Overflow event.
SL_I2S_FRAME_ERROR	Frame error event.

Definition at line `56` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h`

sl_i2s_power_state_t

```
sl_i2s_power_state_t
```

Enumeration for I2S power state.

Enumerator	
SL_I2S_POWER_OFF	Power mode OFF.
SL_I2S_LOW_POWER	Low power mode.
SL_I2S_FULL_POWER	Full power mode.

Definition at line 65 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_mode_t

```
sl_i2s_mode_t
```

Enumeration for I2S master and slave modes.

Enumerator	
SL_I2S_MASTER	I2S master mode.
SL_I2S_SLAVE	I2S slave mode.

Definition at line 72 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_sai_protocol_t

```
sl_sai_protocol_t
```

Enumeration for SAI protocol type.

Enumerator	
SL_I2S_PROTOCOL	I2S protocol.
SL_PCM_PROTOCOL	PCM protocol, currently not supported in the driver.

Definition at line 78 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_sync_t

```
sl_i2s_sync_t
```

Enumeration for I2S SYNC/ASYNC modes.

Enumerator	
SL_I2S_SYNC	I2S synchronous mode.
SL_I2S_ASYNC	I2S asynchronous mode.

Definition at line 84 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_xfer_type_t

```
sl_i2s_xfer_type_t
```

Enumeration for I2S transfer type.

Enumerator	
SL_I2S_TRANSMIT	I2S transmit.
SL_I2S_RECEIVE	I2S receive.
SL_I2S_TRANSMIT_CONTROL	I2S receive.
SL_I2S_RECEIVE_CONTROL	I2S receive.
SL_I2S_SEND_ABORT	I2S abort transmit.
SL_I2S_RECEIVE_ABORT	I2S abort receive.

Definition at line 90 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_xfer_size_t

> sl_i2s_xfer_size_t

Enumeration for I2S transfer data size.

Enumerator	
SL_I2S_DATA_SIZE16	16 bits
SL_I2S_DATA_SIZE32	32 bits

Definition at line 100 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_data_resolution_t

> sl_i2s_data_resolution_t

Enumeration for I2S audio data resolutions.

Enumerator	
SL_I2S_RESOLUTION_12	12 bit resolution
SL_I2S_RESOLUTION_16	16 bit resolution
SL_I2S_RESOLUTION_20	20 bit resolution
SL_I2S_RESOLUTION_24	24 bit resolution
SL_I2S_RESOLUTION_32	32 bit resolution

Definition at line 106 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_sampling_rate_t

> sl_i2s_sampling_rate_t

Enumeration for I2S sample rates.

Enumerator	
SL_I2S_SAMPLING_RATE_8000	8kHz
SL_I2S_SAMPLING_RATE_11025	11.025kHz
SL_I2S_SAMPLING_RATE_16000	16kHz
SL_I2S_SAMPLING_RATE_22050	22.05kHz

SL_I2S_SAMPLING_RATE_24000	24kHz
SL_I2S_SAMPLING_RATE_32000	32kHz
SL_I2S_SAMPLING_RATE_44100	44.1kHz
SL_I2S_SAMPLING_RATE_48000	48kHz
SL_I2S_SAMPLING_RATE_88200	88.2kHz
SL_I2S_SAMPLING_RATE_96000	96kHz
SL_I2S_SAMPLING_RATE_192000	192kHz

Definition at line 115 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

Typedef Documentation

sl_i2s_signal_event_t

```
typedef ARM_SAI_SignalEvent_t sl_i2s_signal_event_t
```

Renamed signal event structure.

Definition at line 50 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_status_t

```
typedef ARM_SAI_STATUS sl_i2s_status_t
```

Renamed status structure.

Definition at line 51 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_driver_t

```
typedef ARM_DRIVER_SAI sl_i2s_driver_t
```

Renamed I2S driver structure.

Definition at line 52 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_handle_t

```
typedef const void* sl_i2s_handle_t
```

Created I2S handle type.

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

Function Documentation

sl_si91x_i2s_init

> sl_status_t sl_si91x_i2s_init (uint32_t i2s_instance, sl_i2s_handle_t *i2s_handle)

Initialize the I2S.

Parameters

[in]	i2s_instance	instance
[in]	i2s_handle	Double Pointer to the I2S driver handle

Pass the address of the pointer for storing the I2S handle, which can be used in future for other function calls.

- Pre-conditions:
  - none

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_NULL_POINTER (0x0022) - Invalid null pointer received as argument
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line `165` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h`

sl_si91x_i2s_deinit

> sl_status_t sl_si91x_i2s_deinit (sl_i2s_handle_t *i2s_handle)

Uninitialize I2S peripheral.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle

- Pre-conditions:
  - sl_si91x_i2s_init

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_NULL_POINTER (0x0022) - Invalid null pointer received as argument

Definition at line `180` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h`

sl_si91x_i2s_configure_power_mode

> sl_status_t sl_si91x_i2s_configure_power_mode (sl_i2s_handle_t i2s_handle, sl_i2s_power_state_t state)

Change the power mode of I2S, the supported modes are.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle
[in]	state	state

- POWER_OFF

FULL_POWER

- Pre-conditions:
  - sl_si91x_i2s_init

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_NULL_POINTER (0x0022) - Invalid null pointer received as argument

Definition at line 198 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_config_transmit_receive

> sl_status_t sl_si91x_i2s_config_transmit_receive (sl_i2s_handle_t i2s_handle, sl_i2s_xfer_config_t *xfer_config)

Configure transmitter/Receiver parameters for I2S transfer.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle
[in]	xfer_config	of structure which stores transfer parameters

- Pre-conditions:
  - sl_si91x_i2s_init
  - sl_si91x_i2s_configure_power_mode

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_NULL_POINTER (0x0022) - Invalid null pointer received as argument

Definition at line 215 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_transmit_data

> sl_status_t sl_si91x_i2s_transmit_data (sl_i2s_handle_t i2s_handle, const void *data, uint32_t size)

Configure I2S peripheral DMA channel to send data.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle
[in]	data	data
[in]	size	size

- Pre-conditions:
  - sl_si91x_i2s_init
  - sl_si91x_i2s_configure_power_mode
  - sl_si91x_i2s_config_transmit_receive

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success

SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
- SL_STATUS_NULL_POINTER (0x0022) - Invalid null pointer received as argument

Definition at line 234 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_receive_data

```
sl_status_t sl_si91x_i2s_receive_data (sl_i2s_handle_t i2s_handle, const void *data, uint32_t size)
```

Configure I2S peripheral DMA channel to receive data.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle
[in]	data	data
[in]	size	size

- Pre-conditions:
  - sl_si91x_i2s_init
  - sl_si91x_i2s_configure_power_mode
  - sl_si91x_i2s_config_transmit_receive

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
  - SL_STATUS_NULL_POINTER (0x0022) - Invalid null pointer received as argument

Definition at line 253 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_register_event_callback

```
sl_status_t sl_si91x_i2s_register_event_callback (sl_i2s_handle_t i2s_handle, sl_i2s_signal_event_t callback_event)
```

Register the user callback function.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle
[in]	callback_event	Pointer to the function which needs to be called at the time of interrupt

- Pre-conditions:
  - none

Returns

- status 0 if successful, else error code as follow
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 269 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_unregister_event_callback

```
sl_status_t sl_si91x_i2s_unregister_event_callback (sl_i2s_handle_t i2s_handle)
```

Un-register the user callback function.

Parameters

[in]	i2s_handle	

- Pre-conditions:
  - sl_si91x_i2s_register_event_callback

Returns

- none

Definition at line 281 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_get_transmit_data_count

```
uint32_t sl_si91x_i2s_get_transmit_data_count (sl_i2s_handle_t i2s_handle)
```

Get the transmit data count of I2S.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle

- Pre-conditions:
  - sl_si91x_i2s_init
  - sl_si91x_i2s_configure_power_mode
  - sl_si91x_i2s_config_transmit_receive
  - sl_si91x_i2s_transmit_data

Returns

- uint32_t value of the tx data count

Definition at line 296 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_get_receive_data_count

```
uint32_t sl_si91x_i2s_get_receive_data_count (sl_i2s_handle_t i2s_handle)
```

Get the receive data count of I2S.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle

- Pre-conditions:
  - sl_si91x_i2s_init
  - sl_si91x_i2s_configure_power_mode
  - sl_si91x_i2s_config_transmit_receive
  - sl_si91x_i2s_receive_data

Returns

- uint32_t value of the rx data count

Definition at line 311 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_get_version

```
sl_i2s_version_t sl_si91x_i2s_get_version (void)
```

Get the release, sqa, and dev version of I2S.

Parameters

[in]		

- Pre-conditions:
  - none

Returns

- (sl_i2s_version_t) type structure

Definition at line 323 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_get_status

```
sl_i2s_status_t sl_si91x_i2s_get_status (sl_i2s_handle_t i2s_handle)
```

Get the transfer status I2S.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle

- Pre-conditions:
  - sl_si91x_i2s_init
  - sl_si91x_i2s_configure_power_mode
  - sl_si91x_i2s_config_transmit_receive

Returns

- (sl_i2s_status_t) type structure

Definition at line 337 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_si91x_i2s_end_transfer

```
sl_status_t sl_si91x_i2s_end_transfer (sl_i2s_handle_t i2s_handle, sl_i2s_xfer_type_t abort_type)
```

Abort I2S Tx/Rx operations.

Parameters

[in]	i2s_handle	Pointer to the I2S driver handle
[in]	abort_type	type, ARM_SAI_ABORT_SEND/ARM_SAI_ABORT_RECEIVE

- Pre-conditions:
  - sl_si91x_i2s_init
  - sl_si91x_i2s_configure_power_mode
  - sl_si91x_i2s_config_transmit_receive
  - sl_si91x_i2s_transmit_data/sl_si91x_i2s_receive_data

Returns

status 0 if successful, else error code as follow

- SL_STATUS_OK (0x0000) - Success
- SL_STATUS_INVALID_PARAMETER (0x0021) - Parameters are invalid
- SL_STATUS_NULL_POINTER (0x0022) - Invalid null pointer received as argument

Definition at line `356` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h`

sl_i2s_version_t

Structure to hold the versions number of peripheral API.

Public Attributes

uint8_t	release	Release version number.
uint8_t	major	sqa version number
uint8_t	minor	dev version number

Public Attribute Documentation

release

```
uint8_t sl_i2s_version_t::release
```

Release version number.

Definition at line 131 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

major

```
uint8_t sl_i2s_version_t::major
```

sqa version number

Definition at line 132 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

minor

```
uint8_t sl_i2s_version_t::minor
```

dev version number

Definition at line 133 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sl_i2s_xfer_config_t

Structure to hold transmit and receive config parameters.

Public Attributes

uint16_t	mode Master/Slave mode.
uint16_t	sync SYNC/ASYNC mode.
uint16_t	protocol I2S/PCM (currently only I2S is supported)
uint16_t	resolution Audio data resolutions.
uint32_t	data_size data size
uint32_t	sampling_rate Audio sampling rate.
uint32_t	transfer_type Tx/Rx.

Public Attribute Documentation

mode

```
uint16_t sl_i2s_xfer_config_t::mode
```

Master/Slave mode.

Definition at line 138 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sync

```
uint16_t sl_i2s_xfer_config_t::sync
```

SYNC/ASYNC mode.

Definition at line 139 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

protocol

```
uint16_t sl_i2s_xfer_config_t::protocol
```

I2S/PCM (currently only I2S is supported)

Definition at line 140 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

resolution

```
uint16_t sl_i2s_xfer_config_t::resolution
```

Audio data resolutions.

Definition at line 141 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

data_size

```
uint32_t sl_i2s_xfer_config_t::data_size
```

data size

Definition at line 142 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

sampling_rate

```
uint32_t sl_i2s_xfer_config_t::sampling_rate
```

Audio sampling rate.

Definition at line 143 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

transfer_type

```
uint32_t sl_i2s_xfer_config_t::transfer_type
```

Tx/Rx.

Definition at line 144 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_i2s.h

PSRAM Driver

PSRAM Driver

PSRAM Memory Management driver.

Introduction

PSRAM (Pseudo Static Random Access Memory) is a random-access memory whose internal structure is based on dynamic memory with refresh control signals generated internally, in the standby mode, so that it can mimic the functionality of a static memory. It combines the high density of DRAM with the ease-of-use of true SRAM. The M4 core communicates with the PSRAM via Quad SPI interface.

PSRAM Device Configuration

The PSRAM Driver offers configuring the following:

- Read-Write type
  - Normal: This supported only in SPI interface mode. Supports maximum frequency of 33MHz. Uses normal read and normal write command.
  - Fast: Supported in SPI and QPI mode. Uses fast read and normal write.
  - Quad IO: Supported in SPI and QPI mode. Uses fast quad read and write.
- Interface mode
  - SPI Mode (Serial IO)
  - QPI Mode (Quad IO)
- Operation frequency Source
  - Interface PLL Clock
  - ULP Reference clock
  - Modem PLL clock
  - SoC PLL clock

Linker configurations

The text segment, data segment, bss, heap and stack can be placed in PSRAM by installing the respective components present under "PSRAM Linker Configurations" from "SOFTWARE COMPONENTS" GUI. Since PSRAM is already initiliazed in bootloader, these components can be installed and the respective segments can be placed in PSRAM without installing "PSRAM Core" component and without initializing psram from application.

Usage

PSRAM Driver and QSPI are initiliazed by bootloader with Quad IO read-write type and QPI interface. The application is not required to reinitialize PSRAM device and QSPI unless the configurations required are different from the default set by the bootloader.

The PSRAM device handle "PSRAM_Device" of type sl_psram_info_type_t is defined "in sl_si91x_psram_handle.c".

sl_si91x_psram_uninit assumes that PSRAM was initialized with QPI mode and exits QPI mode within definition. If the PSRAM configuration in bootcode has SPI mode enabled, user is expected to comment the exit QPI mode function call in sl_si91x_psram_uninit.

To reconfigure and initliaze PSRAM, set required configurations from PSRAM Core component and call sl_si91x_psram_init within application.

Modules

sl_psram_id_type_t

sl_psram_info_type_t

Enumerations

enum sl_psram_return_type_t {

PSRAM_SUCCESS
PSRAM_FAILURE
PSRAM_UNKNOWN
PSRAM_UNKNOWN_DEVICE
PSRAM_CLOCK_INIT_FAILURE
PSRAM_NOT_INITIALIZED
PSRAM_SUPPORTED_DEVICE
PSRAM_DEVICE_MISMATCH
PSRAM_INVALID_HSIZE
PSRAM_NULL_ADDRESS
PSRAM_INVALID_ADDRESS_LENGTH
PSRAM_AUTO_MODE
PSRAM_MANUAL_MODE
PSRAM_UNSUPPORTED_SECURITY
PSRAM_MAX_SEC_SEGMENT_REACH

}

PSRAM return error code.

enum sl_psram_burst_size_type_t {

_WRAP16
_WRAP32
_WRAP64
_WRAP512

}

Wrap burst size enum.

enum sl_psram_dma_status_type_t {

DMA_NONE
DMA_DONE
DMA_FAIL

}

PSRAM DMA status enum.

Functions

sl_psram_return_type_t sl_si91x_psram_init(void)
Initialize the PSRAM Device

sl_psram_return_type_t sl_si91x_psram_uninit(void)
Uninitialize the PSRAM Device

sl_psram_return_type_t sl_si91x_psram_reset(void)
Reset the PSRAM Device

sl_psram_return_type_t sl_si91x_psram_manual_write_in_blocking_mode(uint32_t addr, void *SourceBuf, uint8_t hSize, uint32_t length)
Write data to PSRAM in manual mode

sl_psram_return_type_t sl_si91x_psram_manual_read_in_blocking_mode(uint32_t addr, void *DestBuf, uint8_t hSize, uint32_t length)
Read data from PSRAM in manual mode

sl_psram_return_t-

ype_t	sl_si91x_psram_manual_write_in_dma_mode(uint32_t addr, void *SourceBuf, uint8_t hSize, uint32_t length, sl_psram_dma_status_type_t *dmastatus) Write data to PSRAM in manual mode using DMA
sl_psram_return_type_t	sl_si91x_psram_manual_read_in_dma_mode(uint32_t addr, void *DestBuf, uint8_t hSize, uint32_t length, sl_psram_dma_status_type_t *dmaStatus) Read data from PSRAM in manual mode using DMA
sl_psram_return_type_t	sl_si91x_psram_enable_encry_decry(uint16_t keySize) Enable CTR encryption-decryption on PSRAM

Macros

#define	PSRAM_READ_ID (0x9F) Read ID command.
#define	PSRAM_ENTER_QPI (0x35) Enter QPI interface mode command.
#define	PSRAM_EXIT_QPI (0xF5) Exit QPI interface mode command.
#define	PSRAM_RESET_EN (0x66) Reset Enable command.
#define	PSRAM_RESET (0x99) Reset command.
#define	PSRAM_BURST_LEN (0xC0) Burst Length Toggle command.
#define	PSRAM_MODE_REG_READ (0xB5) Mode Register Read command.
#define	PSRAM_MODE_REG_WRITE (0xB1) Mode Register Write command.
#define	PSRAM_HALF_SLEEP (0xC0) Sleep Entry command.
#define	tXPHS_US (12) Sleep Exit chip select low pulse width in us.
#define	tXHS_US (160) Sleep Exit chip select low to CLK setup in us.
#define	tHS_US (8) Minimum sleep duration in us.

Enumeration Documentation

sl_psram_return_type_t

```
sl_psram_return_type_t
```

PSRAM return error code.

Enumerator	
PSRAM_SUCCESS	
PSRAM_FAILURE	

PSRAM_UNKNOWN	
PSRAM_UNKNOWN_DEVICE	
PSRAM_CLOCK_INIT_FAILURE	
PSRAM_NOT_INITIALIZED	
PSRAM_SUPPORTED_DEVICE	
PSRAM_DEVICE_MISMATCH	
PSRAM_INVALID_HSIZE	
PSRAM_NULL_ADDRESS	
PSRAM_INVALID_ADDRESS_LENGTH	
PSRAM_AUTO_MODE	
PSRAM_MANUAL_MODE	
PSRAM_UNSUPPORTED_SECURITY	
PSRAM_MAX_SEC_SEGMENT_REACH	

Definition at line 79 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_psram_burst_size_type_t

> sl_psram_burst_size_type_t

Wrap burst size enum.

Enumerator	
_WRAP16	
_WRAP32	
_WRAP64	
_WRAP512	

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_psram_dma_status_type_t

> sl_psram_dma_status_type_t

PSRAM DMA status enum.

Enumerator	
DMA_NONE	
DMA_DONE	
DMA_FAIL	

Definition at line 101 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

Function Documentation

sl_si91x_psram_init

> sl_psram_return_type_t sl_si91x_psram_init (void)

Initialize the PSRAM Device

Parameters

N/A		

Returns

- Status Code of the operation

Definition at line 192 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_si91x_psram_uninit

```
sl_psram_return_type_t sl_si91x_psram_uninit (void)
```

Uninitialize the PSRAM Device

Parameters

N/A		

Returns

- Status Code of the operation

Definition at line 201 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_si91x_psram_reset

```
sl_psram_return_type_t sl_si91x_psram_reset (void)
```

Reset the PSRAM Device

Parameters

N/A		

Returns

- Status Code of the operation

Definition at line 210 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_si91x_psram_manual_write_in_blocking_mode

```
sl_psram_return_type_t sl_si91x_psram_manual_write_in_blocking_mode (uint32_t addr, void *SourceBuf, uint8_t hSize,
uint32_t length)
```

Write data to PSRAM in manual mode

Parameters

[in]	addr	PSRAM address for write operation
[in]	SourceBuf	Reference of the Source buffer
[in]	hSize	Size of each element
[in]	length	Number of elements for write operation

Returns

- Status Code of the operation

Definition at line 231 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_si91x_psram_manual_read_in_blocking_mode

> sl_psram_return_type_t sl_si91x_psram_manual_read_in_blocking_mode (uint32_t addr, void *DestBuf, uint8_t hSize, uint32_t length)

Read data from PSRAM in manual mode

Parameters

[in]	addr	PSRAM address for read operation
[in]	DestBuf	Size of each element
[in]	hSize	Number of elements for read operation
[out]	length	Reference of the Destination buffer

Returns

- Status Code of the operation

Definition at line 255 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_si91x_psram_manual_write_in_dma_mode

> sl_psram_return_type_t sl_si91x_psram_manual_write_in_dma_mode (uint32_t addr, void *SourceBuf, uint8_t hSize, uint32_t length, sl_psram_dma_status_type_t *dmastatus)

Write data to PSRAM in manual mode using DMA

Parameters

[in]	addr	PSRAM address for write operation
[in]	SourceBuf	Reference of the Source buffer
[in]	hSize	Size of each element
[in]	length	Number of elements for write operation
[out]	dmastatus	DMA operation completion status

Returns

- Status Code of the operation

Definition at line 282 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_si91x_psram_manual_read_in_dma_mode

> sl_psram_return_type_t sl_si91x_psram_manual_read_in_dma_mode (uint32_t addr, void *DestBuf, uint8_t hSize, uint32_t length, sl_psram_dma_status_type_t *dmaStatus)

Read data from PSRAM in manual mode using DMA

Parameters

[in]	addr	PSRAM address for read operation
[in]	DestBuf	Size of each element
[in]	hSize	Number of elements for read operation

[out]	length	Reference of the Destination buffer
[out]	dmaStatus	DMA operation completion status

Returns

- Status Code of the operation

Definition at line 310 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_si91x_psram_enable_encry_decry

```
sl_psram_return_type_t sl_si91x_psram_enable_encry_decry (uint16_t keySize)
```

Enable CTR encryption-decryption on PSRAM

Parameters

[in]	keySize	Pass 128/256-Bit size

Returns

- Status Code of the operation

Definition at line 346 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

Macro Definition Documentation

PSRAM_READ_ID

```
#define PSRAM_READ_ID
```

Value:

```
(0x9F)
```

Read ID command.

Definition at line 36 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PSRAM_ENTER_QPI

```
#define PSRAM_ENTER_QPI
```

Value:

```
(0x35)
```

Enter QPI interface mode command.

Definition at line 37 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PSRAM_EXIT_QPI

```
#define PSRAM_EXIT_QPI
```

Value:

```
(0xF5)
```

Exit QPI interface mode command.

Definition at line 38 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PSRAM_RESET_EN

```
#define PSRAM_RESET_EN
```

Value:

```
(0x66)
```

Reset Enable command.

Definition at line 39 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PSRAM_RESET

```
#define PSRAM_RESET
```

Value:

```
(0x99)
```

Reset command.

Definition at line 40 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PSRAM_BURST_LEN

```
#define PSRAM_BURST_LEN
```

Value:

```
(0xC0)
```

Burst Length Toggle command.

Definition at line 41 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PSRAM_MODE_REG_READ

```
#define PSRAM_MODE_REG_READ
```

Value:

```
(0xB5)
```

Mode Register Read command.

Definition at line 42 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PSRAM_MODE_REG_WRITE

```
#define PSRAM_MODE_REG_WRITE
```

Value:

```
(0xB1)
```

Mode Register Write command.

Definition at line 43 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PSRAM_HALF_SLEEP

```
#define PSRAM_HALF_SLEEP
```

Value:

```
(0xC0)
```

Sleep Entry command.

Definition at line 44 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

tXPHS_US

```
#define tXPHS_US
```

Value:

```
(12)
```

Sleep Exit chip select low pulse width in us.

Definition at line 70 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

tXHS_US

```
#define tXHS_US
```

Value:

```
(160)
```

Sleep Exit chip select low to CLK setup in us.

Definition at line 71 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

tHS_US

```
#define tHS_US
```

Value:

```
(8)
```

Minimum sleep duration in us.

Definition at line 72 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_psram_id_type_t

Read ID structure.

Public Attributes

uint8_t	MFID
uint8_t	KGD
uint8_t	EID

Public Attribute Documentation

MFID

```
uint8_t sl_psram_id_type_t::MFID
```

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

KGD

```
uint8_t sl_psram_id_type_t::KGD
```

Definition at line 123 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

EID

```
uint8_t sl_psram_id_type_t::EID[6]
```

Definition at line 124 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

sl_psram_info_type_t

PSRAM configuration handle structure.

Public Attributes

PSRAMDevType	deviceName PSRAM Device enum value.
sl_psram_id_type_t	deviceID Device ID struct.
uint32_t	devDensity Device Density in bits.
uint32_t	normalReadMAXFrequency Max frequency for normal read.
uint32_t	fastReadMAXFrequency Max frequency for fast read.
PSRAMRWType	rwType Read-Write type configuration.
spi_config_t	spi_config SPI Config for QSPI interface.
uint16_t	defaultBurstWrapSize Default burst wrap size.
uint16_t	toggleBurstWrapSize Toggle Burst Wrap size.

Public Attribute Documentation

deviceName

> PSRAMDevType sl_psram_info_type_t::deviceName

PSRAM Device enum value.

Definition at line 129 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

deviceID

> sl_psram_id_type_t sl_psram_info_type_t::deviceID

Device ID struct.

Definition at line 130 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

devDensity

uint32_t sl_psram_info_type_t::devDensity

Device Density in bits.

Definition at line 131 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

normalReadMAXFrequency

uint32_t sl_psram_info_type_t::normalReadMAXFrequency

Max frequency for normal read.

Definition at line 132 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

fastReadMAXFrequency

uint32_t sl_psram_info_type_t::fastReadMAXFrequency

Max frequency for fast read.

Definition at line 133 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

rwType

PSRAMRWType sl_psram_info_type_t::rwType

Read-Write type configuration.

Definition at line 134 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

spi_config

spi_config_t sl_psram_info_type_t::spi_config

SPI Config for QSPI interface.

Definition at line 135 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

defaultBurstWrapSize

uint16_t sl_psram_info_type_t::defaultBurstWrapSize

Default burst wrap size.

Definition at line 136 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

toggleBurstWrapSize

```
uint16_t sl_psram_info_type_t::toggleBurstWrapSize
```

Toggle Burst Wrap size.

Definition at line 137 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_psram.h

PWM

PWM

Modules

sl_pwm_version_t

sl_pwm_config_t

sl_pwm_init_t

sl_pwm_fault_init_t

Enumerations

enum sl_pwm_fault_t {

SL_FAULTA
SL_FAULTB
SL_FAULT_LAST

}

Enum to control fault a and fault b.

enum sl_pwm_dead_time_t {

SL_DEAD_TIME_DISABLE
SL_DEAD_TIME_ENABLE
SL_DEAD_TIME_LAST

}

Enum to control dead time.

enum sl_pwm_duty_cycle_t {

SL_DUTY_CYCLE_DISABLE
SL_DUTY_CYCLE_ENABLE
SL_DUTY_CYCLE_LAST

}

Enum to control duty cycle.

enum sl_pwm_override_t {

SL_OVERRIDE_RESET
SL_OVERRIDE_SET
SL_OVERRIDE_LAST

}

Enum to control override.

enum sl_pwm_output_fault_t {

SL_OUTPUT_FAULT_RESET
SL_OUTPUT_FAULT_SET
SL_OUTPUT_FAULT_LAST

}

Enum to control output fault.

enum sl_pwm_event_t {

SL_EVENT_DISABLE

SL_EVENT_ENABLE
SL_EVENT_LAST

}

Enum to control events.

enum sl_pwm_channel_t {

SL_CHANNEL_1
SL_CHANNEL_2
SL_CHANNEL_3
SL_CHANNEL_4
SL_CHANNEL_LAST

}

Enumeration for PWM channels(1-4)

enum sl_pwm_timer_t {

SL_BASE_TIMER_EACH_CHANNEL
SL_BASE_TIMER_ALL_CHANNEL
SL_BASE_TIMER_LAST

}

Enumeration for PWM timer.

enum sl_pwm_polarity_low_t {

SL_POLARITYL_LOW
SL_POLARITYL_HIGH
SL_POLARITYL_LAST

}

Enumeration for PWM polarity low.

enum sl_pwm_polarity_high_t {

SL_POLARITYH_LOW
SL_POLARITYH_HIGH
SL_POLARITYH_LAST

}

Enumeration for PWM polarity high.

enum sl_pwm_mode_t {

SL_MODE_INDEPENDENT
SL_MODE_COMPLEMENTARY
SL_MODE_LAST

}

Enumeration for PWM mode.

enum sl_pwm_base_timer_mode_t {

SL_FREE_RUN_MODE = 0
SL_SINGLE_EVENT_MODE = 1
SL_DOWN_COUNT_MODE = 2
SL_UP_DOWN_MODE = 4
SL_UP_DOWN_DOUBLE_UPDATE = 5
SL_BASE_TIMER_MODE_LAST = 6

}

Enumeration for PWM base timer modes.

enum sl_pwm_output_t {

SL_OUTPUT_LOW0
SL_OUTPUT_LOW1
SL_OUTPUT_LOW2
SL_OUTPUT_LOW3
SL_OUTPUT_HIGH0
SL_OUTPUT_HIGH1
SL_OUTPUT_HIGH2
SL_OUTPUT_HIGH3
SL_OUTPUT_LAST

}

Enumeration for PWM output complementary pairs.

enum sl_pwm_svt_t {

SL_SVT_COUNT_UP
SL_SVT_COUNT_DOWN
SL_SVT_COUNT_LAST

}

Enumeration for PWM special event trigger time base.

enum sl_pwm_post_t {

SL_TIME_PERIOD_POSTSCALE_1_1
SL_TIME_PERIOD_POSTSCALE_1_2
SL_TIME_PERIOD_POSTSCALE_1_3
SL_TIME_PERIOD_POSTSCALE_1_4
SL_TIME_PERIOD_POSTSCALE_1_5
SL_TIME_PERIOD_POSTSCALE_1_6
SL_TIME_PERIOD_POSTSCALE_1_7
SL_TIME_PERIOD_POSTSCALE_1_8
SL_TIME_PERIOD_POSTSCALE_1_9
SL_TIME_PERIOD_POSTSCALE_1_10
SL_TIME_PERIOD_POSTSCALE_1_11
SL_TIME_PERIOD_POSTSCALE_1_12
SL_TIME_PERIOD_POSTSCALE_1_13
SL_TIME_PERIOD_POSTSCALE_1_14
SL_TIME_PERIOD_POSTSCALE_1_15
SL_TIME_PERIOD_POSTSCALE_1_16
SL_TIME_PERIOD_POSTSCALE_1_LAST

}

Enumeration for PWM Time base output post scale bits.

enum sl_pwm_pre_t {

SL_TIME_PERIOD_PRESCALE_1
SL_TIME_PERIOD_PRESCALE_2
SL_TIME_PERIOD_PRESCALE_4
SL_TIME_PERIOD_PRESCALE_8
SL_TIME_PERIOD_PRESCALE_16
SL_TIME_PERIOD_PRESCALE_32
SL_TIME_PERIOD_PRESCALE_64
SL_TIME_PERIOD_PRESCALE_LAST

}

Enumeration for PWM input clock pre scale select value.

enum sl_pwm_fault_input_t {

SL_VALUE_INACTIVE
SL_VALUE_ACTIVE
SL_VALUE_LAST

}

Enumeration for fault input override value.

enum sl_pwm_output_override_t {

SL_OP_OVERRIDE_UNSYNC
SL_OP_OVERRIDE_SYNC
SL_OP_OVERRIDE_LAST

}
Enumeration for output override control.

enum sl_pwm_override_value_t {

SL_OVERRIDE_VALUE0
SL_OVERRIDE_VALUE1
SL_OVERRIDE_VALUE_LAST

}
Enumeration for override value.

enum sl_pwm_trigger_t {

SL_TRIGGER_DISABLE
SL_TRIGGER_ENABLE
SL_TRIGGER_LAST

}
PWM enable external triggering.

Typedefs

typedef RSI_MCPWM_T — sl_si91x_mcpwm_t
PWM structure.

typedef RSI_MCPWM_SVT_CONFIG_T — sl_si91x_pwm_svt_config_t
PWM Special Event trigger configuration parameters structure.

typedef RSI_MCPWM_DT_CONFIG_T — sl_si91x_pwm_dt_config_t
PWM DeadTime configuration parameters structure.

typedef RSI_MCPWM_CALLBACK_T — sl_si91x_pwm_callback_t
PWM Callback structure.

Functions

void sl_si91x_pwm_deinit(void)
This API is used to de-initialize the PWM peripheral.

sl_pwm_version_t sl_si91x_pwm_get_version(void)
This API is used to get the PWM version.

sl_status_t sl_si91x_pwm_set_configuration(sl_pwm_config_t *pwm_config)
This API is used to set the PWM configuration parameters.

sl_status_t sl_si91x_pwm_set_output_polarity(boolean_t polarity_low, boolean_t polarity_high)
This API is used to set output polarity for MCPWM.

sl_status_t sl_si91x_pwm_start(sl_pwm_channel_t channel)
This API is used to start the MCPWM operation for the required channel.

sl_status_t sl_si91x_pwm_stop(sl_pwm_channel_t channel)
This API is used to stop the MCPWM operation for the required channel.

sl_status_t sl_si91x_pwm_control_base_timer(sl_pwm_timer_t base_timer)
This API is used to select the number of base timers as four base timers for four channels or one base timer for all channels of MCPWM.

sl_status_t sl_si91x_pwm_set_time_period(sl_pwm_channel_t channel, uint32_t period, uint32_t init_val)
This API is used to set time period and counter initial value for the required MCPWM channel.

sl_status_t sl_si91x_pwm_trigger_special_event(sl_pwm_svt_t direction, sl_si91x_pwm_svt_config_t *pwm_config)
This API is used to configure special event trigger generation for the required MCPWM channel, which allows the A/D converter to be synchronized to the PWM time base.

sl_status_t sl_si91x_pwm_configure_dead_time(sl_si91x_pwm_dt_config_t *dead_time, sl_pwm_channel_t channel)
This API is used to configure Dead time insertion parameters for MCPWM.

sl_status_t sl_si91x_pwm_reset_channel(sl_pwm_channel_t channel)
This API is used to reset the required channel of MCPWM.

sl_status_t sl_si91x_pwm_reset_counter(sl_pwm_channel_t channel)
This API is used to reset the counter from the required channel of MCPWM.

sl_status_t sl_si91x_pwm_control_period(sl_pwm_post_t post_scale, sl_pwm_pre_t pre_scale, sl_pwm_channel_t channel)
This API is used to set base time period control for the required MCPWM channel.

sl_status_t sl_si91x_pwm_control_fault(sl_pwm_fault_t fault, sl_pwm_output_t pwm_output, sl_pwm_override_value_t value)
This API is used to control fault A/B pin output value to be overridden when a fault condition occurs.

sl_status_t sl_si91x_pwm_set_base_timer_mode(sl_pwm_base_timer_mode_t mode, sl_pwm_channel_t channel)
This API is used to set the mode of the base timer for the required channel.

sl_status_t sl_si91x_pwm_set_output_mode(sl_pwm_mode_t mode, sl_pwm_channel_t channel)
This API is used to set output mode for the MCPWM.

sl_status_t sl_si91x_pwm_register_callback(sl_si91x_pwm_callback_t *callback_event, uint16_t flag)
Handles all interrupt flags of MCPWM.

sl_status_t sl_si91x_pwm_unregister_callback(uint16_t flag)
Unregisters the PWM event.

sl_status_t sl_si91x_pwm_read_counter(uint16_t *counter_value, sl_pwm_channel_t channel)
This API is used to read the counter current value.

sl_status_t sl_si91x_pwm_get_counter_direction(uint8_t *counter_direction, sl_pwm_channel_t channel)
This API is used to get time period counter direction status of the required MCPWM channel.

sl_status_t sl_si91x_pwm_control_dead_time(sl_pwm_dead_time_t dead_time, uint32_t flag)
Controls dead time insertion at the rising edge or falling edge of any four channels.

sl_status_t sl_si91x_pwm_clear_interrupt(uint32_t flag)
This API is used to clear the interrupts of MCPWM.

sl_status_t sl_si91x_pwm_get_interrupt_status(uint32_t flag, uint16_t *intr_status)
This API is used to get the interrupt status of interrupt flags of MCPWM.

sl_status_t sl_si91x_pwm_configure_duty_cycle(sl_pwm_duty_cycle_t duty_cycle, uint32_t value, sl_pwm_channel_t channel)
This API is used to control duty cycle control parameters for the required MCPWM channel.

sl_status_t sl_si91x_pwm_output_override(sl_pwm_override_t override, sl_pwm_output_t pwm_output)
This API is used to control the output override operation of MCPWM.

sl_status_t	sl_si91x_pwm_control_override(sl_pwm_override_t override, sl_pwm_output_override_t value) This API is used to control the override control parameter, output is in sync with PWM time period depending on operating mode.
sl_status_t	sl_si91x_pwm_control_override_value(sl_pwm_override_t override, sl_pwm_output_t pwm_output, sl_pwm_override_value_t value) This API is used to control override value for the required output of MCPWM.
sl_status_t	sl_si91x_pwm_control_output_fault(sl_pwm_output_fault_t output_fault, uint32_t value) This API is used to control output fault override control parameters for the required PWM output.
sl_status_t	sl_si91x_pwm_control_special_event_trigger(sl_pwm_event_t event) This API is used to control the generation of a special event trigger for the required channel of MCPWM.
sl_status_t	sl_si91x_pwm_select_dead_time(sl_pwm_dead_time_t dead_time, uint32_t value) This API is used to control dead time control parameters for the required channel.
sl_status_t	sl_si91x_pwm_set_duty_cycle(uint32_t duty_cycle, sl_pwm_channel_t channel) This API is used to select duty cycle for the required MCPWM channel.
sl_status_t	sl_si91x_pwm_get_duty_cycle(sl_pwm_channel_t channel, uint32_t *duty_cycle) This API is used to get duty cycle for the required MCPWM channel.
sl_status_t	sl_si91x_pwm_enable_external_trigger(sl_pwm_trigger_t enable) This API is used to enable the use of an external trigger for base time counter increment or decrement of MCPWM.
sl_status_t	sl_si91x_pwm_get_time_period(sl_pwm_channel_t channel, uint16_t *time_period) Get time period for the required channel.
sl_status_t	sl_si91x_pwm_init(sl_pwm_init_t *pwm_init) This API is used to initialize PWM pins and clock.
sl_status_t	sl_si91x_pwm_fault_init(sl_pwm_fault_init_t *pwm_fault) This API is used to initialize PWM event pins.

Enumeration Documentation

sl_pwm_fault_t

> sl_pwm_fault_t

Enum to control fault a and fault b.

Enumerator	
SL_FAULTA	PWM fault A.
SL_FAULTB	PWM fault B.
SL_FAULT_LAST	Last member of enum for validation.

Definition at line `70` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

sl_pwm_dead_time_t

> sl_pwm_dead_time_t

Enum to control dead time.

Enumerator	
SL_DEAD_TIME_DISABLE	PWM dead time disable.

SL_DEAD_TIME_ENABLE	PWM dead time enable.
SL_DEAD_TIME_LAST	Last member of enum for validation.

Definition at line 77 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_duty_cycle_t

> sl_pwm_duty_cycle_t

Enum to control duty cycle.

Enumerator	
SL_DUTY_CYCLE_DISABLE	PWM duty cycle disable.
SL_DUTY_CYCLE_ENABLE	PWM duty cycle enable.
SL_DUTY_CYCLE_LAST	Last member of enum for validation.

Definition at line 84 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_override_t

> sl_pwm_override_t

Enum to control override.

Enumerator	
SL_OVERRIDE_RESET	PWM override reset.
SL_OVERRIDE_SET	PWM override set.
SL_OVERRIDE_LAST	Last member of enum for validation.

Definition at line 91 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_output_fault_t

> sl_pwm_output_fault_t

Enum to control output fault.

Enumerator	
SL_OUTPUT_FAULT_RESET	PWM output fault reset.
SL_OUTPUT_FAULT_SET	PWM output fault set.
SL_OUTPUT_FAULT_LAST	Last member of enum for validation.

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_event_t

> sl_pwm_event_t

Enum to control events.

Enumerator	
SL_EVENT_DISABLE	PWM event disable.
SL_EVENT_ENABLE	PWM event enable.
SL_EVENT_LAST	Last member of enum for validation.

Definition at line 105 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_channel_t

> sl_pwm_channel_t

Enumeration for PWM channels(1-4)

Enumerator	
SL_CHANNEL_1	PWM channel 1.
SL_CHANNEL_2	PWM channel 2.
SL_CHANNEL_3	PWM channel 3.
SL_CHANNEL_4	PWM channel 4.
SL_CHANNEL_LAST	Last member of enum for validation.

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_timer_t

> sl_pwm_timer_t

Enumeration for PWM timer.

Enumerator	
SL_BASE_TIMER_EACH_CHANNEL	PWM timer for each channel.
SL_BASE_TIMER_ALL_CHANNEL	PWM timer for all channels.
SL_BASE_TIMER_LAST	Last member of enum for validation.

Definition at line 121 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_polarity_low_t

> sl_pwm_polarity_low_t

Enumeration for PWM polarity low.

Enumerator	
SL_POLARITYL_LOW	PWM output polarity for low side(L0-L3)-low.
SL_POLARITYL_HIGH	PWM output polarity for low side(H0-H3)-high.
SL_POLARITYL_LAST	Last member of enum for validation.

Definition at line 128 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_polarity_high_t

sl_pwm_polarity_high_t

Enumeration for PWM polarity high.

Enumerator	
SL_POLARITYH_LOW	PWM output polarity for high side(L0-L3)-low.
SL_POLARITYH_HIGH	PWM output polarity for high side(H0-H3)-high.
SL_POLARITYH_LAST	Last member of enum for validation.

Definition at line 135 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_mode_t

sl_pwm_mode_t

Enumeration for PWM mode.

Enumerator	
SL_MODE_INDEPENDENT	PWM independent mode.
SL_MODE_COMPLEMENTARY	PWM complementary mode.
SL_MODE_LAST	Last member of enum for validation.

Definition at line 142 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_base_timer_mode_t

sl_pwm_base_timer_mode_t

Enumeration for PWM base timer modes.

Enumerator	
SL_FREE_RUN_MODE	PWM free run mode.
SL_SINGLE_EVENT_MODE	PWM single event mode.
SL_DOWN_COUNT_MODE	PWM down count mode.
SL_UP_DOWN_MODE	PWM up/down mode.
SL_UP_DOWN_DOUBLE_UPDATE	PWM up/down double update mode.
SL_BASE_TIMER_MODE_LAST	Last member of enum for validation.

Definition at line 149 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_output_t

sl_pwm_output_t

Enumeration for PWM output complementary pairs.

Enumerator	
SL_OUTPUT_LOW0	PWM output L0.

SL_OUTPUT_LOW1	PWM output L1.
SL_OUTPUT_LOW2	PWM output L2.
SL_OUTPUT_LOW3	PWM output L3.
SL_OUTPUT_HIGH0	PWM output H0.
SL_OUTPUT_HIGH1	PWM output H1.
SL_OUTPUT_HIGH2	PWM output H2.
SL_OUTPUT_HIGH3	PWM output H3.
SL_OUTPUT_LAST	Last member of enum for validation.

Definition at line 159 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_svt_t

```
sl_pwm_svt_t
```

Enumeration for PWM special event trigger time base.

Enumerator	
SL_SVT_COUNT_UP	Special event trigger occurs when time base is counting up.
SL_SVT_COUNT_DOWN	Special event trigger occurs when time base is counting down.
SL_SVT_COUNT_LAST	Last member of enum for validation.

Definition at line 172 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_post_t

```
sl_pwm_post_t
```

Enumeration for PWM Time base output post scale bits.

Enumerator	
SL_TIME_PERIOD_POSTSCALE_1_1	1:1 post scale
SL_TIME_PERIOD_POSTSCALE_1_2	1:2 post scale
SL_TIME_PERIOD_POSTSCALE_1_3	1:3 post scale
SL_TIME_PERIOD_POSTSCALE_1_4	1:4 post scale
SL_TIME_PERIOD_POSTSCALE_1_5	1:5 post scale
SL_TIME_PERIOD_POSTSCALE_1_6	1:6 post scale
SL_TIME_PERIOD_POSTSCALE_1_7	1:7 post scale
SL_TIME_PERIOD_POSTSCALE_1_8	1:8 post scale
SL_TIME_PERIOD_POSTSCALE_1_9	1:9 post scale
SL_TIME_PERIOD_POSTSCALE_1_10	1:10 post scale
SL_TIME_PERIOD_POSTSCALE_1_11	1:11 post scale
SL_TIME_PERIOD_POSTSCALE_1_12	1:12 post scale
SL_TIME_PERIOD_POSTSCALE_1_13	1:13 post scale
SL_TIME_PERIOD_POSTSCALE_1_14	1:14 post scale
SL_TIME_PERIOD_POSTSCALE_1_15	1:15 post scale

SL_TIME_PERIOD_POSTSCALE_1_16	1:16 post scale
SL_TIME_PERIOD_POSTSCALE_1_LAST	Last member of enum for validation.

Definition at line 179 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_pre_t

> sl_pwm_pre_t

Enumeration for PWM input clock pre scale select value.

Enumerator	
SL_TIME_PERIOD_PRESCALE_1	1x input clock period
SL_TIME_PERIOD_PRESCALE_2	2x input clock period
SL_TIME_PERIOD_PRESCALE_4	4x input clock period
SL_TIME_PERIOD_PRESCALE_8	8x input clock period
SL_TIME_PERIOD_PRESCALE_16	16x input clock period
SL_TIME_PERIOD_PRESCALE_32	32x input clock period
SL_TIME_PERIOD_PRESCALE_64	64x input clock period
SL_TIME_PERIOD_PRESCALE_LAST	Last member of enum for validation.

Definition at line 200 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_fault_input_t

> sl_pwm_fault_input_t

Enumeration for fault input override value.

Enumerator	
SL_VALUE_INACTIVE	PWM output pin is driven inactive on an external fault input A/B event.
SL_VALUE_ACTIVE	PWM output pin is driven active on an external fault input A/B event.
SL_VALUE_LAST	Last member of enum for validation.

Definition at line 212 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_output_override_t

> sl_pwm_output_override_t

Enumeration for output override control.

Enumerator	
SL_OP_OVERRIDE_UNSYNC	no effect.
SL_OP_OVERRIDE_SYNC	Output override is in sync with pwm time period depending on operating mode.
SL_OP_OVERRIDE_LAST	Last member of enum for validation.

Definition at line 219 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_override_value_t

```
sl_pwm_override_value_t
```

Enumeration for override value.

Enumerator	
SL_OVERRIDE_VALUE0	Override value 0.
SL_OVERRIDE_VALUE1	Override value 1.
SL_OVERRIDE_VALUE_LAST	Last member of enum for validation.

Definition at line 226 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_trigger_t

```
sl_pwm_trigger_t
```

PWM enable external triggering.

Enumerator	
SL_TRIGGER_DISABLE	Disabling external triggering.
SL_TRIGGER_ENABLE	Enabling external triggering.
SL_TRIGGER_LAST	Last member of enum for validation.

Definition at line 233 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

Typedef Documentation

sl_si91x_mcpwm_t

```
typedef RSI_MCPWM_T sl_si91x_mcpwm_t
```

PWM structure.

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_svt_config_t

```
typedef RSI_MCPWM_SVT_CONFIG_T sl_si91x_pwm_svt_config_t
```

PWM Special Event trigger configuration parameters structure.

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_dt_config_t

```
typedef RSI_MCPWM_DT_CONFIG_T sl_si91x_pwm_dt_config_t
```

PWM DeadTime configuration parameters structure.

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_callback_t

```
typedef RSI_MCPWM_CALLBACK_T sl_si91x_pwm_callback_t
```

PWM Callback structure.

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

Function Documentation

sl_si91x_pwm_deinit

```
void sl_si91x_pwm_deinit (void)
```

This API is used to de-initialize the PWM peripheral.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration

Returns

- none

Definition at line 294 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_get_version

```
sl_pwm_version_t sl_si91x_pwm_get_version (void)
```

This API is used to get the PWM version.

Parameters

[in]		

Returns

- returns structure of type sl_pwm_version_t

Definition at line 302 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_set_configuration

```
sl_status_t sl_si91x_pwm_set_configuration (sl_pwm_config_t *pwm_config)
```

This API is used to set the PWM configuration parameters.

Parameters

[in]	pwm_config	pointer to configuration parameters of type sl_pwm_config_t

- Pre-conditions:
  - sl_si91x_pwm_init

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 316 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_set_output_polarity

```
sl_status_t sl_si91x_pwm_set_output_polarity (boolean_t polarity_low, boolean_t polarity_high)
```

This API is used to set output polarity for MCPWM.

Parameters

[in]	polarity_low	Output polarity for low side (L3, L2, L1, L0)
[in]	polarity_high	Output polarity for high side (H3, H2, H1, H0)

- Pre-conditions:
  - sl_si91x_pwm_init

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 331 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_start

```
sl_status_t sl_si91x_pwm_start (sl_pwm_channel_t channel)
```

This API is used to start the MCPWM operation for the required channel.

Parameters

[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 349 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_stop

```
sl_status_t sl_si91x_pwm_stop (sl_pwm_channel_t channel)
```

This API is used to stop the MCPWM operation for the required channel.

Parameters

[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback
  - sl_si91x_pwm_start

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 368 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_control_base_timer

```
sl_status_t sl_si91x_pwm_control_base_timer (sl_pwm_timer_t base_timer)
```

This API is used to select the number of base timers as four base timers for four channels or one base timer for all channels of MCPWM.

Parameters

[in]	base_timer	PWM base timer of type sl_pwm_timer_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 384 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_set_time_period

```
sl_status_t sl_si91x_pwm_set_time_period (sl_pwm_channel_t channel, uint32_t period, uint32_t init_val)
```

This API is used to set time period and counter initial value for the required MCPWM channel.

Parameters

[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t
[in]	period	Time period value
[in]	init_val	Update the base time counter initial value

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 401 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_trigger_special_event

> sl_status_t sl_si91x_pwm_trigger_special_event (sl_pwm_svt_t direction, sl_si91x_pwm_svt_config_t *pwm_config)

This API is used to configure special event trigger generation for the required MCPWM channel, which allows the A/D converter to be synchronized to the PWM time base.

Parameters

[in]	direction	Special event trigger for time base direction of type sl_pwm_svt_t
[in]	pwm_config	Pointer to the structure of type sl_si91x_pwm_svt_config_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 421 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_configure_dead_time

> sl_status_t sl_si91x_pwm_configure_dead_time (sl_si91x_pwm_dt_config_t *dead_time, sl_pwm_channel_t channel)

This API is used to configure Dead time insertion parameters for MCPWM.

Parameters

[in]	dead_time	Pointer to the structure of type sl_si91x_pwm_dt_config_t
[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

Pre-conditions:
- sl_si91x_pwm_init
- sl_si91x_pwm_set_configuration
- sl_si91x_pwm_set_base_timer_mode
- sl_si91x_pwm_control_period
- sl_si91x_pwm_register_callback

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 440 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_reset_channel

```
sl_status_t sl_si91x_pwm_reset_channel (sl_pwm_channel_t channel)
```

This API is used to reset the required channel of MCPWM.

Parameters

[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 458 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_reset_counter

```
sl_status_t sl_si91x_pwm_reset_counter (sl_pwm_channel_t channel)
```

This API is used to reset the counter from the required channel of MCPWM.

Parameters

[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 476 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_control_period

> sl_status_t sl_si91x_pwm_control_period (sl_pwm_post_t post_scale, sl_pwm_pre_t pre_scale, sl_pwm_channel_t channel)

This API is used to set base time period control for the required MCPWM channel.

Parameters

[in]	post_scale	Time base output post scale bits of type sl_pwm_post_t
[in]	pre_scale	Base timer input clock pre scale select value of type sl_pwm_pre_t
[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 494 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_control_fault

> sl_status_t sl_si91x_pwm_control_fault (sl_pwm_fault_t fault, sl_pwm_output_t pwm_output, sl_pwm_override_value_t value)

This API is used to control fault A/B pin output value to be overridden when a fault condition occurs.

Parameters

[in]	fault	Enum of type sl_pwm_fault_t
[in]	pwm_output	Enum of type sl_pwm_output_t
[in]	value	Fault input A/B PWM output override value of type sl_pwm_override_value_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback

Returns

- returns status 0 if successful, else error code as follows:

SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
- SL_STATUS_OK - Success
- SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 514 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_set_base_timer_mode

```
sl_status_t sl_si91x_pwm_set_base_timer_mode (sl_pwm_base_timer_mode_t mode, sl_pwm_channel_t channel)
```

This API is used to set the mode of the base timer for the required channel.

Parameters

[in]	mode	Base timer operating mode of type sl_pwm_base_timer_mode_t
[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 530 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_set_output_mode

```
sl_status_t sl_si91x_pwm_set_output_mode (sl_pwm_mode_t mode, sl_pwm_channel_t channel)
```

This API is used to set output mode for the MCPWM.

Parameters

[in]	mode	PWM Output mode of type sl_pwm_mode_t
[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 546 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_register_callback

```
sl_status_t sl_si91x_pwm_register_callback (sl_si91x_pwm_callback_t *callback_event, uint16_t flag)
```

Handles all interrupt flags of MCPWM.

Parameters

[in]	callback_event	Structure of type sl_si91x_pwm_callback_t
[in]	flag	It is the logical OR of different interrupts generated on multiple channels

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is a null pointer

Definition at line `564` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

sl_si91x_pwm_unregister_callback

```
sl_status_t sl_si91x_pwm_unregister_callback (uint16_t flag)
```

Unregisters the PWM event.

Parameters

[in]	flag	It is the logical OR of different interrupts generated on multiple channels

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback
  - sl_si91x_pwm_start

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is a null pointer

Definition at line `582` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

sl_si91x_pwm_read_counter

```
sl_status_t sl_si91x_pwm_read_counter (uint16_t *counter_value, sl_pwm_channel_t channel)
```

This API is used to read the counter current value.

Parameters

[in]	counter_value	Counter value
[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback
  - sl_si91x_pwm_start

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 602 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_get_counter_direction

```
sl_status_t sl_si91x_pwm_get_counter_direction (uint8_t *counter_direction, sl_pwm_channel_t channel)
```

This API is used to get time period counter direction status of the required MCPWM channel.

Parameters

[out]	counter_direction	Pointer to the counter direction (up/down) of type uint8_t
[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback
  - sl_si91x_pwm_start

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 622 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_control_dead_time

```
sl_status_t sl_si91x_pwm_control_dead_time (sl_pwm_dead_time_t dead_time, uint32_t flag)
```

Controls dead time insertion at the rising edge or falling edge of any four channels.

Parameters

[in]	dead_time	Enum of type sl_pwm_dead_time_t

[in]	flag	ORing of the following values: • DT_EN_CH0 • DT_EN_CH1 • DT_EN_CH2 • DT_EN_CH3

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line `643` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

sl_si91x_pwm_clear_interrupt

```
sl_status_t sl_si91x_pwm_clear_interrupt (uint32_t flag)
```

This API is used to clear the interrupts of MCPWM.

Parameters

[in]	flag	The logical OR of different interrupts generated on multiple channels

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback
  - sl_si91x_pwm_start

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is a null pointer

Definition at line `661` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

sl_si91x_pwm_get_interrupt_status

```
sl_status_t sl_si91x_pwm_get_interrupt_status (uint32_t flag, uint16_t *intr_status)
```

This API is used to get the interrupt status of interrupt flags of MCPWM.

Parameters

[in]	flag	Flag value
[out]	intr_status	Pointer to interrupt status

Pre-conditions:
- sl_si91x_pwm_init
- sl_si91x_pwm_set_configuration
- sl_si91x_pwm_set_base_timer_mode
- sl_si91x_pwm_control_period
- sl_si91x_pwm_register_callback
- sl_si91x_pwm_start

Returns

- Interrupt status of the required interrupt flag

Definition at line 677 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_configure_duty_cycle

> sl_status_t sl_si91x_pwm_configure_duty_cycle (sl_pwm_duty_cycle_t duty_cycle, uint32_t value, sl_pwm_channel_t channel)

This API is used to control duty cycle control parameters for the required MCPWM channel.

Parameters

[in]	duty_cycle	Enum of type sl_pwm_duty_cycle_t
[in]	value	This can be a logical OR of the following parameters: • IMDT_DUTYCYCLE_UPDATE_EN: Enable to update the duty cycle immediately • DUTYCYCLE_UPDATE_DISABLE: Duty cycle register updation disable
[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 697 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_output_override

> sl_status_t sl_si91x_pwm_output_override (sl_pwm_override_t override, sl_pwm_output_t pwm_output)

This API is used to control the output override operation of MCPWM.

Parameters

[in]	override	Enum of type sl_pwm_override_t
[in]	pwm_output	PWM output override of type sl_pwm_output_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 714 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_control_override

> sl_status_t sl_si91x_pwm_control_override (sl_pwm_override_t override, sl_pwm_output_override_t value)

This API is used to control the override control parameter, output is in sync with PWM time period depending on operating mode.

Parameters

[in]	override	Enum of type sl_pwm_override_t
[in]	value	Output override to be in sync with PWM time period sl_pwm_output_override_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 732 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_control_override_value

> sl_status_t sl_si91x_pwm_control_override_value (sl_pwm_override_t override, sl_pwm_output_t pwm_output, sl_pwm_override_value_t value)

This API is used to control override value for the required output of MCPWM.

Parameters

[in]	override	Enum of type sl_pwm_override_t
[in]	pwm_output	PWM output override of type sl_pwm_output_t
[in]	value	Override value of type sl_pwm_override_value_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success

SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 750 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_control_output_fault

```
sl_status_t sl_si91x_pwm_control_output_fault (sl_pwm_output_fault_t output_fault, uint32_t value)
```

This API is used to control output fault override control parameters for the required PWM output.

Parameters

[in]	output_fault	Enum of type sl_pwm_output_fault_t
[in]	value	This can be a logical OR of the below parameters: • FLT_A_MODE: if bit one then cycle by cycle by mode and zero then latched mode • FLT_B_MODE: if bit one then cycle by cycle by mode and zero then latched mode • OP_POLARITY_H: Output polarity for high (H3, H2, H1, H0) side signals. If bit 0 then in active low mode and 1 then active high mode. • OP_POLARITY_L: Output polarity for low (L3, L2, L1, L0) side signals. If bit 0 then in active low mode and 1 then active high mode. • FLT_A_ENABLE: Enable fault A • FLT_B_ENABLE: Enable fault B • COMPLEMENT_MODE: PWM I/O pair mode If the bit is 1 then PWM I/O pin pair is in the complementary output mode If the bit is 0 then PWM I/O pin pair is in the independent output mode

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode
  - sl_si91x_pwm_control_period
  - sl_si91x_pwm_register_callback
  - sl_si91x_pwm_start

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 782 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_control_special_event_trigger

```
sl_status_t sl_si91x_pwm_control_special_event_trigger (sl_pwm_event_t event)
```

This API is used to control the generation of a special event trigger for the required channel of MCPWM.

Parameters

[in]	event	Enum of type sl_pwm_event_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 797 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_select_dead_time

```
sl_status_t sl_si91x_pwm_select_dead_time (sl_pwm_dead_time_t dead_time, uint32_t value)
```

This API is used to control dead time control parameters for the required channel.

Parameters

[in]	dead_time	Enum of type sl_pwm_dead_time_t
[in]	value	This can be a logical OR of the below parameters: • DEADTIME_SELECT_ACTIVE: Deadtime select bits for PWM going active Possible values are as below if bit zero then use counter A, if one then use counter B • DEADTIME_SELECT_INACTIVE: Deadtime select bits for PWM going inactive Possible values are as below if bit zero then use counter A, if one then use counter B

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 817 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_set_duty_cycle

```
sl_status_t sl_si91x_pwm_set_duty_cycle (uint32_t duty_cycle, sl_pwm_channel_t channel)
```

This API is used to select duty cycle for the required MCPWM channel.

Parameters

[in]	duty_cycle	Duty cycle value
[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 833 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_get_duty_cycle

```
sl_status_t sl_si91x_pwm_get_duty_cycle (sl_pwm_channel_t channel, uint32_t *duty_cycle)
```

This API is used to get duty cycle for the required MCPWM channel.

Parameters

[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t
[out]	duty_cycle	Pointer to the duty cycle value

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 849 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_enable_external_trigger

```
sl_status_t sl_si91x_pwm_enable_external_trigger (sl_pwm_trigger_t enable)
```

This API is used to enable the use of an external trigger for base time counter increment or decrement of MCPWM.

Parameters

[in]	enable	Controlling external trigger of type sl_pwm_trigger_t

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration
  - sl_si91x_pwm_set_base_timer_mode

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line 866 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_si91x_pwm_get_time_period

```
sl_status_t sl_si91x_pwm_get_time_period (sl_pwm_channel_t channel, uint16_t *time_period)
```

Get time period for the required channel.

Parameters

[in]	channel	Channel number (1 to 4) of type sl_pwm_channel_t
[out]	time_period	Pointer to read time period

- Pre-conditions:
  - sl_si91x_pwm_init
  - sl_si91x_pwm_set_configuration

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER - The parameter is an invalid argument
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line `882` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

sl_si91x_pwm_init

```
sl_status_t sl_si91x_pwm_init (sl_pwm_init_t *pwm_init)
```

This API is used to initialize PWM pins and clock.

Parameters

[in]	pwm_init	Pointer to the structure of type sl_pwm_init_t

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line `893` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

sl_si91x_pwm_fault_init

```
sl_status_t sl_si91x_pwm_fault_init (sl_pwm_fault_init_t *pwm_fault)
```

This API is used to initialize PWM event pins.

Parameters

[in]	pwm_fault	Pointer to the structure of type sl_pwm_fault_init_t

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is null pointer

Definition at line `904` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

sl_pwm_version_t

Structure to hold the different versions of peripheral API.

Public Attributes

uint8_t	release	Release version number.
uint8_t	major	major version number
uint8_t	minor	minor version number

Public Attribute Documentation

release

```
uint8_t sl_pwm_version_t::release
```

Release version number.

Definition at line 244 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

major

```
uint8_t sl_pwm_version_t::major
```

major version number

Definition at line 245 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

minor

```
uint8_t sl_pwm_version_t::minor
```

minor version number

Definition at line 246 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_config_t

Structure to hold the parameters of timer configurations.

Public Attributes

sl_pwm_channel_t	channel PWM channel selection(ch0 to ch4)
uint32_t	frequency PWM frequency.
boolean_t	is_polarity_low PWM output polarity for low side.
boolean_t	is_polarity_high PWM output polarity for high side.
boolean_t	is_mode PWM mode(independent/complementary)
uint32_t	base_time_counter_initial_value PWM base time counter initial value.
uint8_t	duty_cycle PWM duty cycle.
uint8_t	base_timer_mode PWM base timer mode.
uint8_t	channel_timer_selection PWM channel timer selection.

Public Attribute Documentation

channel

```
sl_pwm_channel_t sl_pwm_config_t::channel
```

PWM channel selection(ch0 to ch4)

Definition at line 251 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

frequency

```
uint32_t sl_pwm_config_t::frequency
```

PWM frequency.

Definition at line 252 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

is_polarity_low

```
boolean_t sl_pwm_config_t::is_polarity_low
```

PWM output polarity for low side.

Definition at line 253 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

is_polarity_high

```
boolean_t sl_pwm_config_t::is_polarity_high
```

PWM output polarity for high side.

Definition at line 254 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

is_mode

```
boolean_t sl_pwm_config_t::is_mode
```

PWM mode(independent/complementary)

Definition at line 255 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

base_time_counter_initial_value

```
uint32_t sl_pwm_config_t::base_time_counter_initial_value
```

PWM base time counter initial value.

Definition at line 256 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

duty_cycle

```
uint8_t sl_pwm_config_t::duty_cycle
```

PWM duty cycle.

Definition at line 257 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

base_timer_mode

```
uint8_t sl_pwm_config_t::base_timer_mode
```

PWM base timer mode.

Definition at line 258 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

channel_timer_selection

```
uint8_t sl_pwm_config_t::channel_timer_selection
```

PWM channel timer selection.

Definition at line 259 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_init_t

Structure to hold portl and pinh for channels.

Public Attributes

uint8_t	port_l PWM GPIO output low side port.
uint8_t	pin_l PWM GPIO output low side pin.
uint8_t	port_h PWM GPIO output high side port.
uint8_t	pin_h PWM GPIO output high side pin.
uint8_t	mux_l PWM GPIO output low side mux.
uint8_t	mux_h PWM GPIO output high side mux.
uint8_t	pad_l PWM GPIO output low side pad.
uint8_t	pad_h PWM GPIO output high side pad.

Public Attribute Documentation

port_l

```
uint8_t sl_pwm_init_t::port_l
```

PWM GPIO output low side port.

Definition at line `264` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

pin_l

```
uint8_t sl_pwm_init_t::pin_l
```

PWM GPIO output low side pin.

Definition at line `265` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h`

port_h

uint8_t sl_pwm_init_t::port_h

PWM GPIO output high side port.

Definition at line 266 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

pin_h

uint8_t sl_pwm_init_t::pin_h

PWM GPIO output high side pin.

Definition at line 267 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

mux_l

uint8_t sl_pwm_init_t::mux_l

PWM GPIO output low side mux.

Definition at line 268 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

mux_h

uint8_t sl_pwm_init_t::mux_h

PWM GPIO output high side mux.

Definition at line 269 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

pad_l

uint8_t sl_pwm_init_t::pad_l

PWM GPIO output low side pad.

Definition at line 270 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

pad_h

uint8_t sl_pwm_init_t::pad_h

PWM GPIO output high side pad.

Definition at line 271 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

sl_pwm_fault_init_t

Structure to hold port and pin of different events.

Public Attributes

uint8_t	port PWM GPIO port.
uint8_t	pin PWM GPIO pin.
uint8_t	mux PWM GPIO mux.
uint8_t	pad PWM GPIO pad.

Public Attribute Documentation

port

```
uint8_t sl_pwm_fault_init_t::port
```

PWM GPIO port.

Definition at line 276 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

pin

```
uint8_t sl_pwm_fault_init_t::pin
```

PWM GPIO pin.

Definition at line 277 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

mux

```
uint8_t sl_pwm_fault_init_t::mux
```

PWM GPIO mux.

Definition at line 278 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

pad

```
uint8_t sl_pwm_fault_init_t::pad
```

PWM GPIO pad.

Definition at line 279 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_pwm.h

SDIO Secondary

Modules

sl_sdio_secondary_version_t

Enumerations

enum sl_sdio_slave_rx_intr_status_t {

HOST_INTR_NOT_RECEIVED = 0
HOST_INTR_RECEIVED = 1
HOST_INTR_NOT_RECEIVED = 0
HOST_INTR_RECEIVED = 1

}

Typedefs

typedef void(* sl_sdio_secondary_callback_t)(uint8_t events)

typedef void(* sl_sdio_secondary_gpdma_callback_t)(uint8_t dma_ch)

Functions

sl_status_t sl_si91x_sdio_secondary_init(void)
Initializes SDIO secondary, by default SDIO pin access with TA this API gives the pin access to M4, selects the SDIO mode and enable the sdio primary interrupts

void sl_si91x_sdio_secondary_send(uint8_t num_of_blocks, uint8_t *data_buf)
Trigger sequence to send data from sdio secondary in non blocking mode to host/primary device.

void sl_si91x_sdio_secondary_receive(uint8_t *data_buf)
Receive data on sdio secondary in non blocking mode from host/primary device using DMA.

sl_status_t sl_si91x_sdio_secondary_register_event_callback(sl_sdio_secondary_callback_t callback_event, uint32_t flag)
Register the SDIO secondary user callback function.

void sl_si91x_sdio_secondary_unregister_event_callback(uint32_t flag)
Un-register the SDIO secondary user callback function.

sl_status_t sl_si91x_sdio_secondary_gpdma_register_event_callback(sl_sdio_secondary_gpdma_callback_t callback_event)
Register the SDIO secondary dma user callback function.

void sl_si91x_sdio_secondary_gpdma_unregister_event_callback(void)
Un-register the SDIO secondary dma user callback function.

sl_sdio_secondary_version_t sl_si91x_sdio_secondary_get_version(void)
Get the SDIO Secondary Driver version.

void	sl_si91x_sdio_secondary_peripheral_init(void) This API initializes the SDIO secondary, by default SDIO pin access with TA this API gives the pin access to M4, selects the SDIO mode and enable the sdio primary interrupts.
__STATIC_INLINE void	sl_si91x_sdio_secondary_enable_interrupts(uint32_t flags) Enable the SDIO interrupts.
__STATIC_INLINE void	sl_si91x_sdio_secondary_disable_interrupts(uint32_t flags) Disable the SDIO interrupts.
__INLINE void	sl_si91x_sdio_secondary_set_interrupts(uint32_t flags) Set the interrupts i.e., unmask according to the flags passed in the parameter.
__INLINE void	sl_si91x_sdio_secondary_clear_interrupts(uint32_t flags) Clear the interrupts i.e., mask according to the flags passed in the parameter.
__INLINE uint32_t	sl_si91x_sdio_secondary_get_pending_interrupts(void) Get the pending function interrupt.
__INLINE uint32_t	sl_si91x_sdio_secondary_get_enabled_interrupts(void) Get all the enabled interrupts in function1.
__INLINE uint32_t	sl_si91x_sdio_secondary_get_enabled_pending_interrupts(void) Get all the enabled interrupts in function1 and pending interrupts.
__INLINE uint16_t	sl_si91x_sdio_secondary_get_block_cnt(void) To get block count for the last received CMD53.
__INLINE uint16_t	sl_si91x_sdio_secondary_get_block_len(void) To get the length of each for the last received CMD53.
__INLINE void	sl_si91x_sdio_secondary_set_tx_blocks(uint8_t no_of_blocks) To set no of blocks to be transferred This API is used when transferring the data from secondary to primary in block mode.

Macros

#define	NUMGPDMADESC 10
#define	HOST_INTR_RECEIVE_EVENT BIT(0) Events for HIF irq handler.
#define	SDIO SDIO0
#define	SDIO_Handler HIF1_IRQHandler
#define	GPDMA_Handler IRQ031_Handler
#define	RX_SOURCE_ADDR 0×20200080 SDIO Write FIFO Data Register.
#define	TX_SOURCE_ADDR 0×20200040 SDIO Read FIFO Data Register.
#define	SDIO_MODE_SELECT 0×0705
#define	MASK_HOST_INTERRUPT 0xF0
#define	M4_MISC_CONFIG_BASE 0×46008000
#define	SDIO_BASE 0×20200000

```
#define RX_NUM_CHUNKS (*(volatile uint32_t *)(SDIO_BASE + 0x242))
#define M4_HOST_INTR_MASK_REG (*(volatile uint32_t *)(M4_MISC_CONFIG_BASE + 0x00))
#define M4_HOST_INTR_STATUS_REG (*(volatile uint32_t *)(M4_MISC_CONFIG_BASE + 0x04))
#define M4_HOST_INTR_CLEAR (*(volatile uint32_t *)(M4_MISC_CONFIG_BASE + 0x08))
#define MISC_CFG_HOST_CTRL (*(volatile uint32_t *)(M4_MISC_CONFIG_BASE + 0x0C))
#define SL_SDIO_WR_INT_EN BIT(0)
#define SL_SDIO_RD_INT_EN BIT(1)
#define SL_SDIO_CSA_INT_EN BIT(2)
#define SL_SDIO_CMD52_INT_EN BIT(3)
#define SL_SDIO_PWR_LEV_INT_EN BIT(4)
#define SL_SDIO_CRC_ERR_INT_EN BIT(5)
#define SL_SDIO_ABORT_INT_EN BIT(6)
#define SL_SDIO_TOUT_INT_EN BIT(7)
#define SL_SDIO_WR_INT_MSK BIT(0)
#define SL_SDIO_RD_INT_MSK BIT(1)
#define SL_SDIO_CSA_INT_MSK BIT(2)
#define SL_SDIO_CMD52_INT_MSK BIT(3)
#define SL_SDIO_PWR_LEV_INT_MSK BIT(4)
#define SL_SDIO_CRC_ERR_INT_MSK BIT(5)
#define SL_SDIO_ABORT_INT_MSK BIT(6)
#define SL_SDIO_TOUT_INT_MSK BIT(7)
#define SL_SDIO_WR_INT_UNMSK BIT(0)
#define SL_SDIO_RD_INT_UNMSK BIT(1)
#define SL_SDIO_CSA_INT_UNMSK BIT(2)
#define SL_SDIO_CMD52_INT_UNMSK BIT(3)
#define SL_SDIO_PWR_LEV_INT_UNMSK BIT(4)
#define SL_SDIO_CRC_ERR_INT_UNMSK BIT(5)
#define SL_SDIO_ABORT_INT_UNMSK BIT(6)
#define SL_SDIO_TOUT_INT_UNMSK BIT(7)
```

Enumeration Documentation

sl_sdio_slave_rx_intr_status_t

sl_sdio_slave_rx_intr_status_t

Enumerator	
HOST_INTR_NOT_RECEIVED	
HOST_INTR_RECEIVED	
HOST_INTR_NOT_RECEIVED	
HOST_INTR_RECEIVED	

Definition at line 45 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

Typedef Documentation

sl_sdio_secondary_callback_t

```
typedef void(* sl_sdio_secondary_callback_t) (uint8_t events) )(uint8_t events)
```

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_sdio_secondary_gpdma_callback_t

```
typedef void(* sl_sdio_secondary_gpdma_callback_t) (uint8_t dma_ch) )(uint8_t dma_ch)
```

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

Function Documentation

sl_si91x_sdio_secondary_init

```
sl_status_t sl_si91x_sdio_secondary_init (void)
```

Initializes SDIO secondary, by default SDIO pin access with TA this API gives the pin access to M4, selects the SDIO mode and enable the sdio primary interrupts

Parameters

[in]		

Returns

- none

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_si91x_sdio_secondary_send

```
void sl_si91x_sdio_secondary_send (uint8_t num_of_blocks, uint8_t *data_buf)
```

Trigger sequence to send data from sdio secondary in non blocking mode to host/primary device.

Parameters

[in]	num_of_blocks	Number of blocks to be sent
[in]	data_buf	Reference of the Source buffer

This API accepts the num of blocks i.e one block will have block length (1 to 1024) byptes and pointer to the data buffer to be transffered.

Returns

- none

Definition at line 91 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_si91x_sdio_secondary_receive

```
void sl_si91x_sdio_secondary_receive (uint8_t *data_buf)
```

Receive data on sdio secondary in non blocking mode from host/primary device using DMA.

Parameters

[in]	data_buf	Reference of the Destination buffer in whuch data will receive

Returns

- none

Definition at line 103 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_si91x_sdio_secondary_register_event_callback

```
sl_status_t sl_si91x_sdio_secondary_register_event_callback (sl_sdio_secondary_callback_t callback_event, uint32_t flag)
```

Register the SDIO secondary user callback function.

Parameters

[in]	callback_event	Pointer to the function which needs to be called at the time of interrupt
[in]	flag	Interrupt flag to be registered

- Pre-conditions:
  - sl_si91x_sdio_secondary_init();

Returns

- status 0 if successful, else error code as follow SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_si91x_sdio_secondary_unregister_event_callback

```
void sl_si91x_sdio_secondary_unregister_event_callback (uint32_t flag)
```

Un-register the SDIO secondary user callback function.

Parameters

[in]	flag	Interrupt flag to be unregistered

Returns

- none

Definition at line 131 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_si91x_sdio_secondary_gpdma_register_event_callback

```
sl_status_t sl_si91x_sdio_secondary_gpdma_register_event_callback (sl_sdio_secondary_gpdma_callback_t callback_event)
```

Register the SDIO secondary dma user callback function.

Parameters

[in]	callback_event	Pointer to the function which needs to be called at the time of interrupt

- Pre-conditions:
  - sl_si91x_sdio_secondary_init();

Returns

- status 0 if successful, else error code as follow SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 149 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_si91x_sdio_secondary_gpdma_unregister_event_callback

```
void sl_si91x_sdio_secondary_gpdma_unregister_event_callback (void)
```

Un-register the SDIO secondary dma user callback function.

Parameters

[in]		

Returns

- none

Definition at line 158 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_si91x_sdio_secondary_get_version

```
sl_sdio_secondary_version_t sl_si91x_sdio_secondary_get_version (void)
```

Get the SDIO Secondary Driver version.

Parameters

[in]		

This function is used to know the SDIO Secondary Driver version

Returns

sl_sdio_secondary_version_t type version

Definition at line 170 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

sl_si91x_sdio_secondary_peripheral_init

```
void sl_si91x_sdio_secondary_peripheral_init (void)
```

This API initializes the SDIO secondary, by default SDIO pin access with TA this API gives the pin access to M4, selects the SDIO mode and enable the sdio primary interrupts.

Parameters

[in]		

Returns

- none

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

sl_si91x_sdio_secondary_enable_interrupts

```
__STATIC_INLINE void sl_si91x_sdio_secondary_enable_interrupts (uint32_t flags)
```

Enable the SDIO interrupts.

Parameters

[in]	flags	Interrupt Flag which need to be enabled Different interrupts flag which can be enabled are, SL_SDIO_WR_INT_EN SL_SDIO_RD_INT_EN SL_SDIO_CSA_INT_EN SL_SDIO_CMD52_INT_EN SL_SDIO_PWR_LEV_INT_EN SL_SDIO_CRC_ERR_INT_EN SL_SDIO_ABORT_INT_EN SL_SDIO_TOUT_INT_EN

Returns

- none

Definition at line 116 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

sl_si91x_sdio_secondary_disable_interrupts

```
__STATIC_INLINE void sl_si91x_sdio_secondary_disable_interrupts (uint32_t flags)
```

Disable the SDIO interrupts.

Parameters

[in]	flags	Interrupt Flag which need to be disabled Different interrupts flags which can be disabled are, SL_SDIO_WR_INT_EN SL_SDIO_RD_INT_EN SL_SDIO_CSA_INT_EN SL_SDIO_CMD52_INT_EN SL_SDIO_PWR_LEV_INT_EN SL_SDIO_CRC_ERR_INT_EN SL_SDIO_ABORT_INT_EN SL_SDIO_TOUT_INT_EN

Returns

- none

Definition at line 138 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

sl_si91x_sdio_secondary_set_interrupts

> __INLINE void sl_si91x_sdio_secondary_set_interrupts (uint32_t flags)

Set the interrupts i.e., unmask according to the flags passed in the parameter.

Parameters

[in]	flags	Interrupt flags which needs to be set. Different interrupts flags which need to be set, SL_SDIO_WR_INT_UNMSK SL_SDIO_RD_INT_UNMSK SL_SDIO_CSA_INT_UNMSK SL_SDIO_CMD52_INT_UNMSK SL_SDIO_PWR_LEV_INT_UNMSK SL_SDIO_CRC_ERR_INT_UNMSK SL_SDIO_ABORT_INT_UNMSK SL_SDIO_TOUT_INT_UNMSK

- sl_si91x_sdio_secondary_enable_interrupts();

Returns

- none

Definition at line `161` of file `components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h`

sl_si91x_sdio_secondary_clear_interrupts

> __INLINE void sl_si91x_sdio_secondary_clear_interrupts (uint32_t flags)

Clear the interrupts i.e., mask according to the flags passed in the parameter.

Parameters

[in]	flags	Interrupt flags which needs to be masked. SL_SDIO_WR_INT_MSK SL_SDIO_RD_INT_MSK SL_SDIO_CSA_INT_MSK SL_SDIO_CMD52_INT_MSK SL_SDIO_PWR_LEV_INT_MSK SL_SDIO_CRC_ERR_INT_MSK SL_SDIO_ABORT_INT_MSK SL_SDIO_TOUT_INT_MSK

- sl_si91x_sdio_secondary_disable_interrupts();

Returns

- none

Definition at line `182` of file `components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h`

sl_si91x_sdio_secondary_get_pending_interrupts

> __INLINE uint32_t sl_si91x_sdio_secondary_get_pending_interrupts (void)

Get the pending function interrupt.

Parameters

[in]		

Returns

- Return the pending interrupt status

Definition at line `195` of file `components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h`

sl_si91x_sdio_secondary_get_enabled_interrupts

__INLINE uint32_t sl_si91x_sdio_secondary_get_enabled_interrupts (void)

Get all the enabled interrupts in function1.

Parameters

[in]		

Returns

- No of interrupts enabled

Definition at line 207 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

sl_si91x_sdio_secondary_get_enabled_pending_interrupts

__INLINE uint32_t sl_si91x_sdio_secondary_get_enabled_pending_interrupts (void)

Get all the enabled interrupts in function1 and pending interrupts.

Parameters

[in]		

Returns

- No of interrupts enabled

Definition at line 219 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

sl_si91x_sdio_secondary_get_block_cnt

__INLINE uint16_t sl_si91x_sdio_secondary_get_block_cnt (void)

To get block count for the last received CMD53.

Parameters

[in]		

Returns

- No of block counts

Definition at line 231 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

sl_si91x_sdio_secondary_get_block_len

__INLINE uint16_t sl_si91x_sdio_secondary_get_block_len (void)

To get the length of each for the last received CMD53.

Parameters

[in]		

Returns

Length of each block

Definition at line 244 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

sl_si91x_sdio_secondary_set_tx_blocks

```
__INLINE void sl_si91x_sdio_secondary_set_tx_blocks (uint8_t no_of_blocks)
```

To set no of blocks to be transferred This API is used when transferring the data from secondary to primary in block mode.

Parameters

[in]	no_of_blocks	no of blocks to be transfered

Returns

- none

Definition at line 259 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

Macro Definition Documentation

NUMGPDMADESC

```
#define NUMGPDMADESC
```

Value:

```
10
```

Definition at line 37 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

HOST_INTR_RECEIVE_EVENT

```
#define HOST_INTR_RECEIVE_EVENT
```

Value:

```
BIT(0)
```

Events for HIF irq handler.

Definition at line 40 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

SDIO

```
#define SDIO
```

Value:

```
SDIO0
```

Definition at line 36 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SDIO_Handler

```
#define SDIO_Handler
```

Value:

```
HIF1_IRQHandler
```

Definition at line 38 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

GPDMA_Handler

```
#define GPDMA_Handler
```

Value:

```
IRQ031_Handler
```

Definition at line 39 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

RX_SOURCE_ADDR

```
#define RX_SOURCE_ADDR
```

Value:

```
0x20200080
```

SDIO Write FIFO Data Register.

Definition at line 42 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

TX_SOURCE_ADDR

```
#define TX_SOURCE_ADDR
```

Value:

```
0x20200040
```

SDIO Read FIFO Data Register.

Definition at line 45 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SDIO_MODE_SELECT

```
#define SDIO_MODE_SELECT
```

Value:

```
0x0705
```

Definition at line 47 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

MASK_HOST_INTERRUPT

```
#define MASK_HOST_INTERRUPT
```

Value:

```
0xF0
```

Definition at line 48 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

M4_MISC_CONFIG_BASE

```
#define M4_MISC_CONFIG_BASE
```

Value:

```
0x46008000
```

Definition at line 50 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SDIO_BASE

```
#define SDIO_BASE
```

Value:

```
0x20200000
```

Definition at line 51 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

RX_NUM_CHUNKS

```
#define RX_NUM_CHUNKS
```

Value:

```
(*(volatile uint32_t *)(SDIO_BASE + 0x242))
```

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

M4_HOST_INTR_MASK_REG

```
#define M4_HOST_INTR_MASK_REG
```

Value:

```
(*(volatile uint32_t *)(M4_MISC_CONFIG_BASE + 0x00))
```

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

M4_HOST_INTR_STATUS_REG

```
#define M4_HOST_INTR_STATUS_REG
```

Value:

```
(*(volatile uint32_t *)(M4_MISC_CONFIG_BASE + 0x04))
```

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

M4_HOST_INTR_CLEAR

```
#define M4_HOST_INTR_CLEAR
```

Value:

```
(*(volatile uint32_t *)(M4_MISC_CONFIG_BASE + 0x08))
```

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

MISC_CFG_HOST_CTRL

```
#define MISC_CFG_HOST_CTRL
```

Value:

```
(*(volatile uint32_t *)(M4_MISC_CONFIG_BASE + 0x0C))
```

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_WR_INT_EN

```
#define SL_SDIO_WR_INT_EN
```

Value:

```
BIT(0)
```

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_RD_INT_EN

```
#define SL_SDIO_RD_INT_EN
```

Value:

```
BIT(1)
```

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CSA_INT_EN

```
#define SL_SDIO_CSA_INT_EN
```

Value:

```
BIT(2)
```

Definition at line 62 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CMD52_INT_EN

```
#define SL_SDIO_CMD52_INT_EN
```

Value:

```
BIT(3)
```

Definition at line 63 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_PWR_LEV_INT_EN

```
#define SL_SDIO_PWR_LEV_INT_EN
```

Value:

```
BIT(4)
```

Definition at line 64 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CRC_ERR_INT_EN

```
#define SL_SDIO_CRC_ERR_INT_EN
```

Value:

```
BIT(5)
```

Definition at line 65 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_ABORT_INT_EN

```
#define SL_SDIO_ABORT_INT_EN
```

Value:

```
BIT(6)
```

Definition at line 66 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_TOUT_INT_EN

```
#define SL_SDIO_TOUT_INT_EN
```

Value:

```
BIT(7)
```

Definition at line 67 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_WR_INT_MSK

```
#define SL_SDIO_WR_INT_MSK
```

Value:

```
BIT(0)
```

Definition at line 70 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_RD_INT_MSK

```
#define SL_SDIO_RD_INT_MSK
```

Value:

```
BIT(1)
```

Definition at line 71 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CSA_INT_MSK

```
#define SL_SDIO_CSA_INT_MSK
```

Value:

```
BIT(2)
```

Definition at line 72 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CMD52_INT_MSK

```
#define SL_SDIO_CMD52_INT_MSK
```

Value:

```
BIT(3)
```

Definition at line 73 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_PWR_LEV_INT_MSK

```
#define SL_SDIO_PWR_LEV_INT_MSK
```

Value:

```
BIT(4)
```

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CRC_ERR_INT_MSK

#define SL_SDIO_CRC_ERR_INT_MSK

Value:

```
BIT(5)
```

Definition at line 75 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_ABORT_INT_MSK

#define SL_SDIO_ABORT_INT_MSK

Value:

```
BIT(6)
```

Definition at line 76 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_TOUT_INT_MSK

#define SL_SDIO_TOUT_INT_MSK

Value:

```
BIT(7)
```

Definition at line 77 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_WR_INT_UNMSK

#define SL_SDIO_WR_INT_UNMSK

Value:

```
BIT(0)
```

Definition at line 80 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_RD_INT_UNMSK

#define SL_SDIO_RD_INT_UNMSK

Value:

```
BIT(1)
```

Definition at line 81 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CSA_INT_UNMSK

#define SL_SDIO_CSA_INT_UNMSK

Value:

```
BIT(2)
```

Definition at line 82 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CMD52_INT_UNMSK

```
#define SL_SDIO_CMD52_INT_UNMSK
```

Value:

```
BIT(3)
```

Definition at line 83 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_PWR_LEV_INT_UNMSK

```
#define SL_SDIO_PWR_LEV_INT_UNMSK
```

Value:

```
BIT(4)
```

Definition at line 84 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_CRC_ERR_INT_UNMSK

```
#define SL_SDIO_CRC_ERR_INT_UNMSK
```

Value:

```
BIT(5)
```

Definition at line 85 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_ABORT_INT_UNMSK

```
#define SL_SDIO_ABORT_INT_UNMSK
```

Value:

```
BIT(6)
```

Definition at line 86 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

SL_SDIO_TOUT_INT_UNMSK

```
#define SL_SDIO_TOUT_INT_UNMSK
```

Value:

```
BIT(7)
```

Definition at line 87 of file components/device/silabs/si91x/mcu/drivers/unified_peripheral_drivers/inc/sl_si91x_peripheral_sdio_secondary.h

sl_sdio_secondary_version_t

Structure to hold the versions number of API.

Public Attributes

uint8_t	release Release version number.
uint8_t	major Major version number.
uint8_t	minor Minor version number.

Public Attribute Documentation

release

```
uint8_t sl_sdio_secondary_version_t::release
```

Release version number.

Definition at line 52 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

major

```
uint8_t sl_sdio_secondary_version_t::major
```

Major version number.

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

minor

```
uint8_t sl_sdio_secondary_version_t::minor
```

Minor version number.

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sdio_secondary.h

Serial Input-Output

Serial Input-Output

Modules

- sl_sio_version_t
- sl_sio_spi_t
- sl_sio_uart_t
- sl_sio_i2c_t

Enumerations

enum sl_sio_spi_event_t {

  SL_SIO_SPI_TX_DONE
  SL_SIO_SPI_RX_DONE

}

SIO-SPI Events.

enum sl_sio_uart_event_t {

  SL_SIO_UART_TX_DONE
  SL_SIO_UART_RX_DONE
  SL_SIO_UART_PARITY_ERR
  SL_SIO_UART_FRAMING_ERR
  SL_SIO_UART_RECEIVE_CHAR

}

SIO-UART Events.

enum sl_sio_spi_mode_t {

  SL_SIO_SPI_MODE_0 = 0
  SL_SIO_SPI_MODE_3 = 3

}

SIO-SPI modes configurations.

enum sl_sio_spi_bit_width_t {

  SL_SIO_SPI_BIT_8 = 8
  SL_SIO_SPI_BIT_16 = 16

}

SIO-SPI bit width configurations.

enum sl_sio_spi_msb_lsb_t {

  SL_SIO_SPI_LSB_FIRST = 0
  SL_SIO_SPI_MSB_FIRST = 1

}

SIO-SPI MSB/LSB first configurations.

enum sl_sio_spi_bit_length_t {

  SL_SIO_UART_BIT_8 = 8
  SL_SIO_UART_BIT_16 = 16

}
SIO-UART bit length configurations.

enum sl_sio_spi_parity_t {
SL_SIO_UART_EVEN_PARITY = 0
SL_SIO_UART_ODD_PARITY = 1
}
SIO-UART parity configurations.

enum sl_sio_spi_stop_bit_t {
SL_SIO_UART_STOP_BIT_1 = 1
SL_SIO_UART_STOP_BIT_2 = 2
}
SIO-UART stop bit configurations.

Typedefs

typedef stc_sio_spi_cfg_t — sl_sio_spi_config_t
SIO-SPI configuration structure.

typedef stc_sio_spi_xfer_t — sl_sio_spi_xfer_config_t
SIO-SPI Transfer structure.

typedef stc_sio_i2s_config_t — sl_sio_i2s_config_t
SIO-I2S configuration structure.

typedef stc_sio_i2s_xfer_t — sl_sio_i2s_xfer_config_t
SIO-I2S Transfer structure.

typedef stc_sio_uart_config_t — sl_sio_uart_config_t
SIO-UART configuration structure.

typedef sio_i2s_func_ptr_t — sl_sio_i2s_callback_t
SIO-I2S callback function pointer.

typedef sio_Spi_func_ptr_t — sl_sio_spi_callback_t
SIO-SPI callback function pointer.

typedef sio_Uart_func_ptr_t — sl_sio_uart_callback_t
SIO_UART callback function pointer.

typedef stc_sio_i2c_config_t — sl_sio_i2c_config_t
SIO-I2C configuration structure.

Functions

sl_status_t sl_si91x_sio_init(void)
Initialize the SIO module.

sl_status_t sl_si91x_sio_spi_init(sl_sio_spi_config_t *configuration)
Initialize the SIO-SPI module.

sl_status_t sl_si91x_sio_spi_pin_initialization(sl_sio_spi_t *sio_spi_init)
Initialize SIO SPI pins and clock.

sl_status_t	sl_si91x_sio_uart_pin_initialization(sl_sio_uart_t *sio_uart_init) Initialize SIO UART pins and clock.
sl_status_t	sl_si91x_sio_i2c_pin_initialization(sl_sio_i2c_t *sio_i2c_init) Initialize SIO I2C pins and clock.
sl_status_t	sl_si91x_sio_spi_cs_assert(uint8_t chip_select_num) Assert the SIO SPI chip select.
sl_status_t	sl_si91x_sio_spi_cs_deassert(uint8_t chip_select_num) De-assert the SIO SPI chip select.
sl_status_t	sl_si91x_sio_spi_register_event_callback(sl_sio_spi_callback_t callback_event) Register the user callback function.
void	sl_si91x_sio_spi_unregister_event_callback(void) Un-register the user callback function.
sl_status_t	sl_si91x_sio_spi_transfer(sl_sio_spi_xfer_config_t *xfer_config) Transfer the SIO SPI data.
sl_sio_version_t	sl_si91x_sio_get_version(void) Get the SIO version.
sl_status_t	sl_si91x_sio_uart_init(sl_sio_uart_config_t *configuration) Initialize SIO-UART, i.e., set baud rate, parity, channel selection, stop bits, and data length.
sl_status_t	sl_si91x_sio_uart_send(const void *buffer, uint16_t length) Send the data over SIO-UART.
sl_status_t	sl_si91x_sio_uart_send_blocking(const void *buffer, uint16_t length) Send the data over SIO-UART in blocking mode.
sl_status_t	sl_si91x_sio_uart_read(void *data_buffer, uint16_t num_bytes) Read data from UART.
sl_status_t	sl_si91x_sio_uart_read_blocking(void *data_buffer, uint16_t num_bytes) Read data from UART in blocking mode.
sl_status_t	sl_si91x_sio_uart_register_event_callback(sl_sio_uart_callback_t callback_event) Register the user callback function.
sl_status_t	sl_si91x_sio_i2c_write(stc_sio_i2c_config_t *configuration, uint8_t address, uint8_t *data, uint16_t length) Write data using SIO-I2C.
sl_status_t	sl_si91x_sio_i2c_read(stc_sio_i2c_config_t *configuration, uint8_t address, uint8_t *data, uint16_t length) Read data using SIO-I2C.
sl_status_t	sl_si91x_sio_i2c_transfer(stc_sio_i2c_config_t *configuration, uint8_t address, uint8_t *tx_buffer, uint16_t tx_length, uint8_t *rx_buffer, uint16_t rx_length) Transfer data using SIO-I2C.
void	sl_si91x_sio_i2c_generate_start(void) Generate I2C start in SIO.
void	sl_si91x_sio_i2c_generate_stop(void) Generate I2C stop in SIO.
void	sl_si91x_sio_uart_unregister_event_callback(void) Un-register the user callback function.
void	sl_si91x_sio_uart_rx_done(void) Used when UART receive is done.

sl_status_t	sl_si91x_sio_configure_interrupt(en_sio_channels_t channel, interrupt_flag_t flag) Configure pin detection mode to be considered for GPIO interrupt.
sl_status_t	sl_si91x_sio_match_pattern(en_sio_channels_t channel, pattern_match_t pattern, uint8_t slice, uint32_t slice_pattern) Match the pattern with data to be detected.
sl_status_t	sl_si91x_sio_shift_clock(uint32_t divider, en_sio_channels_t channel) Generate the shift clock.
sl_status_t	sl_si91x_sio_select_clock(en_sio_channels_t channel, clock_type_t clock) Select clock.
sl_status_t	sl_si91x_sio_position_counter(en_sio_channels_t channel, uint32_t data_shift) Shift the number of bits.
sl_status_t	sl_si91x_sio_control_flow(en_sio_channels_t channel, flow_control_t flow_control) Enable/disable the flow control bit.
sl_status_t	sl_si91x_sio_reverse_load(en_sio_channels_t channel, reverse_load_t reverse) Load data to buffer in reverse order.
sl_status_t	sl_si91x_sio_set_interrupt(en_sio_channels_t channel) Enable the common swap interrupt.
sl_status_t	sl_si91x_sio_clear_interrupt(en_sio_channels_t channel) Disable the common swap interrupt.
sl_status_t	sl_si91x_sio_mask_interrupt(en_sio_channels_t channel) Mask the common swap interrupt.
sl_status_t	sl_si91x_sio_unmask_interrupt(en_sio_channels_t channel) Unmask the common swap interrupt.
uint32_t	sl_si91x_sio_get_interrupt_status(void) Read the common swap interrupt status.
sl_status_t	sl_si91x_sio_set_shift_interrupt(en_sio_channels_t channel) Enable the common shift interrupt.
sl_status_t	sl_si91x_sio_clear_shift_interrupt(en_sio_channels_t channel) Disable the common shift interrupt.
sl_status_t	sl_si91x_sio_mask_shift_interrupt(en_sio_channels_t channel) Mask the common shift interrupt.
sl_status_t	sl_si91x_sio_unmask_shift_interrupt(en_sio_channels_t channel) Unmask the common shift interrupt.
uint32_t	sl_si91x_sio_shift_interrupt_status(void) Read the common shift interrupt status.
sl_status_t	sl_si91x_sio_edge_select(en_sio_channels_t channel, edge_select_t edge_sel) Select edge of the clock cycle for sampling bits.
uint32_t	sl_si91x_sio_read_buffer(en_sio_channels_t channel) Read SIO buffer register.
sl_status_t	sl_si91x_sio_write_buffer(en_sio_channels_t channel, uint32_t data) Write into SIO buffer register.

Macros

#define SL_SIO_CH_0 0
SIO-SPI channel 0.

#define SL_SIO_CH_1 1
SIO-SPI channel 1.

#define SL_SIO_CH_2 2
SIO-SPI channel 2.

#define SL_SIO_CH_3 3
SIO-SPI channel 3.

#define SL_SIO_CH_4 4
SIO-SPI channel 4.

#define SL_SIO_CH_5 5
SIO-SPI channel 5.

#define SL_SIO_CH_6 6
SIO-SPI channel 6.

#define SL_SIO_CH_7 7
SIO-SPI channel 7.

Enumeration Documentation

sl_sio_spi_event_t

> sl_sio_spi_event_t

SIO-SPI Events.

Enumerator	
SL_SIO_SPI_TX_DONE	SIO-SPI Transfer done event.
SL_SIO_SPI_RX_DONE	SIO-SPI Receive done event.

Definition at line 83 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_uart_event_t

> sl_sio_uart_event_t

SIO-UART Events.

Enumerator	
SL_SIO_UART_TX_DONE	SIO-UART transfer done event.
SL_SIO_UART_RX_DONE	SIO-UART receive done event.
SL_SIO_UART_PARITY_ERR	SIO-UART parity error.
SL_SIO_UART_FRAMING_ERR	SIO-UART framing error.
SL_SIO_UART_RECEIVE_CHAR	SIO-UART receive character.

Definition at line 89 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_mode_t

> sl_sio_spi_mode_t

SIO-SPI modes configurations.

Enumerator	
SL_SIO_SPI_MODE_0	SIO-SPI mode 0.
SL_SIO_SPI_MODE_3	SIO-SPI mode 3.

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_bit_width_t

> sl_sio_spi_bit_width_t

SIO-SPI bit width configurations.

Enumerator	
SL_SIO_SPI_BIT_8	SIO-SPI bit width 8.
SL_SIO_SPI_BIT_16	SIO-SPI bit width 16.

Definition at line 104 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_msb_lsb_t

> sl_sio_spi_msb_lsb_t

SIO-SPI MSB/LSB first configurations.

Enumerator	
SL_SIO_SPI_LSB_FIRST	SIO-SPI LSB first.
SL_SIO_SPI_MSB_FIRST	SIO-SPI MSB first.

Definition at line 110 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_bit_length_t

> sl_sio_spi_bit_length_t

SIO-UART bit length configurations.

Enumerator	
SL_SIO_UART_BIT_8	SIO-SPI bit length 8.
SL_SIO_UART_BIT_16	SIO-SPI bit length 16.

Definition at line 116 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_parity_t

> sl_sio_spi_parity_t

SIO-UART parity configurations.

Enumerator	
SL_SIO_UART_EVEN_PARITY	SIO-SPI even parity.
SL_SIO_UART_ODD_PARITY	SIO-SPI odd parity.

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_stop_bit_t

```
sl_sio_spi_stop_bit_t
```

SIO-UART stop bit configurations.

Enumerator	
SL_SIO_UART_STOP_BIT_1	SIO-UART stop bit 1.
SL_SIO_UART_STOP_BIT_2	SIO-UART stop bit 2.

Definition at line 128 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

Typedef Documentation

sl_sio_spi_config_t

```
typedef stc_sio_spi_cfg_t sl_sio_spi_config_t
```

SIO-SPI configuration structure.

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_xfer_config_t

```
typedef stc_sio_spi_xfer_t sl_sio_spi_xfer_config_t
```

SIO-SPI Transfer structure.

Definition at line 62 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_i2s_config_t

```
typedef stc_sio_i2s_config_t sl_sio_i2s_config_t
```

SIO-I2S configuration structure.

Definition at line 63 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_i2s_xfer_config_t

```
typedef stc_sio_i2s_xfer_t sl_sio_i2s_xfer_config_t
```

SIO-I2S Transfer structure.

Definition at line 64 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_uart_config_t

```
typedef stc_sio_uart_config_t sl_sio_uart_config_t
```

SIO-UART configuration structure.

Definition at line 65 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_i2s_callback_t

```
typedef sio_i2s_func_ptr_t sl_sio_i2s_callback_t
```

SIO-I2S callback function pointer.

Definition at line 66 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_callback_t

```
typedef sio_Spi_func_ptr_t sl_sio_spi_callback_t
```

SIO-SPI callback function pointer.

Definition at line 67 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_uart_callback_t

```
typedef sio_Uart_func_ptr_t sl_sio_uart_callback_t
```

SIO_UART callback function pointer.

Definition at line 68 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_i2c_config_t

```
typedef stc_sio_i2c_config_t sl_sio_i2c_config_t
```

SIO-I2C configuration structure.

Definition at line 69 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

Function Documentation

sl_si91x_sio_init

```
sl_status_t sl_si91x_sio_init (void)
```

Initialize the SIO module.

Parameters

[in]		

It initializes the SIO GPIO's and enables the SIO module clock. **Returns**

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_FAIL (0x0001) - Fail, SIO initialization failed
  - SL_STATUS_OK (0x0000) - Success, SIO initialization successful

Definition at line `200` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

sl_si91x_sio_spi_init

```
sl_status_t sl_si91x_sio_spi_init (sl_sio_spi_config_t *configuration)
```

Initialize the SIO-SPI module.

Parameters

[in]	configuration	- Pointer to SIO-SPI configuration structure sl_sio_spi_config_t

It configures the SPI mode, bit length, bit order, SIO frequency, and the SIO channels for the SPI transfer lines.

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success, SPI initialization done properly

Definition at line `216` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

sl_si91x_sio_spi_pin_initialization

```
sl_status_t sl_si91x_sio_spi_pin_initialization (sl_sio_spi_t *sio_spi_init)
```

Initialize SIO SPI pins and clock.

Parameters

[in]	sio_spi_init	: Pointer to the structure of type sl_sio_spi_t

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is a null pointer

Definition at line `226` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

sl_si91x_sio_uart_pin_initialization

```
sl_status_t sl_si91x_sio_uart_pin_initialization (sl_sio_uart_t *sio_uart_init)
```

Initialize SIO UART pins and clock.

Parameters

[in]	sio_uart_init	: Pointer to the structure of type sl_sio_uart_t

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is a null pointer

Definition at line 236 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_i2c_pin_initialization

```
sl_status_t sl_si91x_sio_i2c_pin_initialization (sl_sio_i2c_t *sio_i2c_init)
```

Initialize SIO I2C pins and clock.

Parameters

[in]	sio_i2c_init	: Pointer to the structure of type sl_sio_i2c_t

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK - Success
  - SL_STATUS_NULL_POINTER - The parameter is a null pointer

Definition at line 246 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_spi_cs_assert

```
sl_status_t sl_si91x_sio_spi_cs_assert (uint8_t chip_select_num)
```

Assert the SIO SPI chip select.

Parameters

[in]	chip_select_num	- Chip select number (0 to 7)

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_spi_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success

Definition at line 260 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_spi_cs_deassert

```
sl_status_t sl_si91x_sio_spi_cs_deassert (uint8_t chip_select_num)
```

De-assert the SIO SPI chip select.

Parameters

[in]	chip_select_num	- Chip select number (0 to 7)

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_spi_init()
  - sl_si91x_sio_spi_cs_assert()
  - sl_si91x_sio_spi_transfer()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success

Definition at line `276` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

sl_si91x_sio_spi_register_event_callback

```
sl_status_t sl_si91x_sio_spi_register_event_callback (sl_sio_spi_callback_t callback_event)
```

Register the user callback function.

Parameters

[in]	callback_event	- Pointer to the function sl_sio_spi_callback_t which needs to be called at the time of interrupt.

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_spi_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line `291` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

sl_si91x_sio_spi_unregister_event_callback

```
void sl_si91x_sio_spi_unregister_event_callback (void)
```

Un-register the user callback function.

Parameters

[in]		

- Pre-conditions:

sl_si91x_sio_init()
- sl_si91x_sio_spi_init()
- sl_si91x_sio_spi_register_event_callback()

Returns

- none

Definition at line 302 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_spi_transfer

```
sl_status_t sl_si91x_sio_spi_transfer (sl_sio_spi_xfer_config_t *xfer_config)
```

Transfer the SIO SPI data.

Parameters

[in]	xfer_config	- Pointer to SIO-SPI transfer configuration structure sl_sio_spi_xfer_config_t

It is used to make the SIO-SPI transfer in non-blocking mode.

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_spi_init()
  - sl_si91x_sio_spi_cs_assert()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success

Definition at line 319 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_get_version

```
sl_sio_version_t sl_si91x_sio_get_version (void)
```

Get the SIO version.

Parameters

[in]		

Returns

- returns structure of type sl_sio_version_t

Definition at line 326 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_uart_init

```
sl_status_t sl_si91x_sio_uart_init (sl_sio_uart_config_t *configuration)
```

Initialize SIO-UART, i.e., set baud rate, parity, channel selection, stop bits, and data length.

Parameters

[in]	configuration	- Pointer to SIO-UART configuration structure sl_sio_uart_config_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success, UART initialization done properly

Definition at line 341 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_uart_send

```
sl_status_t sl_si91x_sio_uart_send (const void *buffer, uint16_t length)
```

Send the data over SIO-UART.

Parameters

[in]	buffer	- data pointer to send
[in]	length	- data length

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success

Definition at line 356 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_uart_send_blocking

```
sl_status_t sl_si91x_sio_uart_send_blocking (const void *buffer, uint16_t length)
```

Send the data over SIO-UART in blocking mode.

Parameters

[in]	buffer	- data pointer to send
[in]	length	- number of bytes to send

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()
  - sl_si91x_sio_uart_send() (or) sl_si91x_sio_uart_send_blocking()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument

SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
- SL_STATUS_OK (0x0000) - Success

Definition at line 372 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_uart_read

```
sl_status_t sl_si91x_sio_uart_read (void *data_buffer, uint16_t num_bytes)
```

Read data from UART.

Parameters

[in]	data_buffer	- data buffer pointer to read
[in]	num_bytes	- number of bytes read

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()
  - sl_si91x_sio_uart_send() (or) sl_si91x_sio_uart_send_blocking()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success

Definition at line 388 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_uart_read_blocking

```
sl_status_t sl_si91x_sio_uart_read_blocking (void *data_buffer, uint16_t num_bytes)
```

Read data from UART in blocking mode.

Parameters

[in]	data_buffer	- data buffer pointer to read
[in]	num_bytes	- number of bytes read

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()
  - sl_si91x_sio_uart_send() (or)
  - sl_si91x_sio_uart_send_blocking()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success

Definition at line 405 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_uart_register_event_callback

sl_status_t sl_si91x_sio_uart_register_event_callback (sl_sio_uart_callback_t callback_event)

Register the user callback function.

Parameters

[in]	callback_event	- Pointer to the function sl_sio_uart_callback_t which needs to be called at the time of interrupt.

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 420 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_i2c_write

sl_status_t sl_si91x_sio_i2c_write (stc_sio_i2c_config_t *configuration, uint8_t address, uint8_t *data, uint16_t length)

Write data using SIO-I2C.

Parameters

[in]	configuration	- pointer to the I2C configuration structure stc_sio_i2c_config_t in SIO module
[in]	address	- slave address
[in]	data	- pointer to the data
[in]	length	- data length

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_i2c_generate_start()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 438 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_i2c_read

sl_status_t sl_si91x_sio_i2c_read (stc_sio_i2c_config_t *configuration, uint8_t address, uint8_t *data, uint16_t length)

Read data using SIO-I2C.

Parameters

[in]	configuration	- pointer to the I2C configuration structure stc_sio_i2c_config_t in SIO module

[in]	address	- slave address
[in]	data	- pointer to the data
[in]	length	- data length

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_i2c_generate_start()
  - sl_si91x_sio_i2c_write() (or)
  - sl_si91x_sio_i2c_transfer()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 461 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_i2c_transfer

> sl_status_t sl_si91x_sio_i2c_transfer (stc_sio_i2c_config_t *configuration, uint8_t address, uint8_t *tx_buffer, uint16_t tx_length, uint8_t *rx_buffer, uint16_t rx_length)

Transfer data using SIO-I2C.

Parameters

[in]	configuration	- Pointer to the I2C configuration structure stc_sio_i2c_config_t in SIO module
[in]	address	- Slave address
[in]	tx_buffer	- Pointer to the data transmit buffer
[in]	tx_length	- TX data length
[in]	rx_buffer	- Pointer to the data receive buffer
[in]	rx_length	- RX data length

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_i2c_generate_start()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 482 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_i2c_generate_start

> void sl_si91x_sio_i2c_generate_start (void)

Generate I2C start in SIO.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- none

Definition at line 497 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_i2c_generate_stop

```
void sl_si91x_sio_i2c_generate_stop (void)
```

Generate I2C stop in SIO.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_i2c_generate_start()
  - sl_si91x_sio_i2c_write() (or)
  - sl_si91x_sio_i2c_transfer()

Returns

- none

Definition at line 510 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_uart_unregister_event_callback

```
void sl_si91x_sio_uart_unregister_event_callback (void)
```

Un-register the user callback function.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()
  - sl_si91x_sio_uart_register_event_callback()

Returns

- none

Definition at line 522 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_uart_rx_done

```
void sl_si91x_sio_uart_rx_done (void)
```

Used when UART receive is done.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()
  - sl_si91x_sio_uart_send()
  - sl_si91x_sio_uart_read()

Returns

- none

Definition at line 535 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_configure_interrupt

> sl_status_t sl_si91x_sio_configure_interrupt (en_sio_channels_t channel, interrupt_flag_t flag)

Configure pin detection mode to be considered for GPIO interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t
[in]	flag	- GPIO interrupt generated interrupt_flag_t • Rise edge: 0 • Fall edge: 1 • Level 0: 2 • Level 1: 3

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 554 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_match_pattern

> sl_status_t sl_si91x_sio_match_pattern (en_sio_channels_t channel, pattern_match_t pattern, uint8_t slice, uint32_t slice_pattern)

Match the pattern with data to be detected.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t
[in]	pattern	- Pattern match bit to be enabled for pattern match to take place pattern_match_t • Pattern match disable: 0 • Pattern match enable: 1

[in]	slice	- Slice number (0,1,2,8,9,10) to select
[in]	slice_pattern	- Pattern to match for selected slice

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 573 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_shift_clock

> sl_status_t sl_si91x_sio_shift_clock (uint32_t divider, en_sio_channels_t channel)

Generate the shift clock.

Parameters

[in]	divider	- Desired clock frequency configuration
[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 591 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_select_clock

> sl_status_t sl_si91x_sio_select_clock (en_sio_channels_t channel, clock_type_t clock)

Select clock.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t
[in]	clock	- Clock used for shift operations clock_type_t • Internal clock: 0 • External clock: 1

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()
  - sl_si91x_sio_uart_register_event_callback()
  - sl_si91x_sio_uart_send()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line `611` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

sl_si91x_sio_position_counter

```
sl_status_t sl_si91x_sio_position_counter (en_sio_channels_t channel, uint32_t data_shift)
```

Shift the number of bits.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t
[in]	data_shift	- Number of shifts to happen before reloading register with data. Value to be set = (total no. of valid bits in shift register/ no. of bits per shift) - 1

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line `627` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

sl_si91x_sio_control_flow

```
sl_status_t sl_si91x_sio_control_flow (en_sio_channels_t channel, flow_control_t flow_control)
```

Enable/disable the flow control bit.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t
[in]	flow_control	- It decides whether to continue data shifting based on data present in shift register validation flow_control_t • Flow control disable: 0 • Flow control enable: 1

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 645 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_reverse_load

```
sl_status_t sl_si91x_sio_reverse_load (en_sio_channels_t channel, reverse_load_t reverse)
```

Load data to buffer in reverse order.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t
[in]	reverse	- If data to be shifted out MSB first, it is to be set reverse_load_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 660 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_set_interrupt

```
sl_status_t sl_si91x_sio_set_interrupt (en_sio_channels_t channel)
```

Enable the common swap interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 674 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_clear_interrupt

```
sl_status_t sl_si91x_sio_clear_interrupt (en_sio_channels_t channel)
```

Disable the common swap interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  -

sl_si91x_sio_init()
- sl_si91x_sio_i2c_generate_start()
- sl_si91x_sio_i2c_write() (or)
- sl_si91x_sio_i2c_transfer()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 691 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_mask_interrupt

> sl_status_t sl_si91x_sio_mask_interrupt (en_sio_channels_t channel)

Mask the common swap interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 705 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_unmask_interrupt

> sl_status_t sl_si91x_sio_unmask_interrupt (en_sio_channels_t channel)

Unmask the common swap interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 719 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_get_interrupt_status

uint32_t sl_si91x_sio_get_interrupt_status (void)

Read the common swap interrupt status.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns interrupt status

Definition at line 729 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_set_shift_interrupt

sl_status_t sl_si91x_sio_set_shift_interrupt (en_sio_channels_t channel)

Enable the common shift interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 743 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_clear_shift_interrupt

sl_status_t sl_si91x_sio_clear_shift_interrupt (en_sio_channels_t channel)

Disable the common shift interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 757 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_mask_shift_interrupt

```
sl_status_t sl_si91x_sio_mask_shift_interrupt (en_sio_channels_t channel)
```

Mask the common shift interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 771 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_unmask_shift_interrupt

```
sl_status_t sl_si91x_sio_unmask_shift_interrupt (en_sio_channels_t channel)
```

Unmask the common shift interrupt.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 785 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_shift_interrupt_status

```
uint32_t sl_si91x_sio_shift_interrupt_status (void)
```

Read the common shift interrupt status.

Parameters

[in]		- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns shift interrupt status

Definition at line 795 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_edge_select

```
sl_status_t sl_si91x_sio_edge_select (en_sio_channels_t channel, edge_select_t edge_sel)
```

Select edge of the clock cycle for sampling bits.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t
[in]	edge_sel	- Select the edge for bit sample to start edge_select_t • Positive edge: 0 • Negative edge: 1

- Pre-conditions:
  - sl_si91x_sio_init()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line 812 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_read_buffer

```
uint32_t sl_si91x_sio_read_buffer (en_sio_channels_t channel)
```

Read SIO buffer register.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()
  - sl_si91x_sio_uart_register_event_callback()
  - sl_si91x_sio_uart_send()

Returns

- returns data from buffer

Definition at line 825 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_si91x_sio_write_buffer

```
sl_status_t sl_si91x_sio_write_buffer (en_sio_channels_t channel, uint32_t data)
```

Write into SIO buffer register.

Parameters

[in]	channel	- SIO channel to be selected en_sio_channels_t
[in]	data	- Data to be written to buffer

- Pre-conditions:
  - sl_si91x_sio_init()
  - sl_si91x_sio_uart_init()
  - sl_si91x_sio_uart_register_event_callback()
  - sl_si91x_sio_uart_send()

Returns

- returns status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_NULL_POINTER (0x0022n) - The parameter is null pointer
  - SL_STATUS_OK (0X000) - Success

Definition at line `843` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

Macro Definition Documentation

SL_SIO_CH_0

```
#define SL_SIO_CH_0
```

Value:

```
0
```

SIO-SPI channel 0.

Definition at line `52` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

SL_SIO_CH_1

```
#define SL_SIO_CH_1
```

Value:

```
1
```

SIO-SPI channel 1.

Definition at line `53` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h`

SL_SIO_CH_2

```
#define SL_SIO_CH_2
```

Value:

```
2
```

SIO-SPI channel 2.

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

SL_SIO_CH_3

```
#define SL_SIO_CH_3
```

Value:

```
3
```

SIO-SPI channel 3.

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

SL_SIO_CH_4

```
#define SL_SIO_CH_4
```

Value:

```
4
```

SIO-SPI channel 4.

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

SL_SIO_CH_5

```
#define SL_SIO_CH_5
```

Value:

```
5
```

SIO-SPI channel 5.

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

SL_SIO_CH_6

```
#define SL_SIO_CH_6
```

Value:

```
6
```

SIO-SPI channel 6.

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

SL_SIO_CH_7

```
#define SL_SIO_CH_7
```

Value:

```
7
```

SIO-SPI channel 7.

Definition at line 59 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_version_t

Structure to hold the different versions of peripheral API.

Public Attributes

uint8_t	release Release version number.
uint8_t	major major version number
uint8_t	minor minor version number

Public Attribute Documentation

release

```
uint8_t sl_sio_version_t::release
```

Release version number.

Definition at line 138 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

major

```
uint8_t sl_sio_version_t::major
```

major version number

Definition at line 139 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

minor

```
uint8_t sl_sio_version_t::minor
```

minor version number

Definition at line 140 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_spi_t

Structure to hold port and pin of sio spi.

Public Attributes

uint8_t	spi_cs_port SIO SPI CS port.
uint8_t	spi_cs_pin SIO SPI CS pin.
uint8_t	spi_cs_mux SIO SPI CS mux.
uint8_t	spi_cs_pad SIO SPI CS pad.
uint8_t	spi_clk_port SIO SPI CLK port.
uint8_t	spi_clk_pin SIO SPI CLK pin.
uint8_t	spi_clk_mux SIO SPI CLK mux.
uint8_t	spi_clk_pad SIO SPI CLK pad.
uint8_t	spi_mosi_port SIO SPI MOSI port.
uint8_t	spi_mosi_pin SIO SPI MOSI pin.
uint8_t	spi_mosi_mux SIO SPI MOSI mux.
uint8_t	spi_mosi_pad SIO SPI MOSI pad.
uint8_t	spi_miso_port SIO SPI MISO port.
uint8_t	spi_miso_pin SIO SPI MISO pin.
uint8_t	spi_miso_mux SIO SPI MISO mux.
uint8_t	spi_miso_pad SIO SPI MISO pad.

Public Attribute Documentation

spi_cs_port

uint8_t sl_sio_spi_t::spi_cs_port

SIO SPI CS port.

Definition at line 145 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_cs_pin

uint8_t sl_sio_spi_t::spi_cs_pin

SIO SPI CS pin.

Definition at line 146 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_cs_mux

uint8_t sl_sio_spi_t::spi_cs_mux

SIO SPI CS mux.

Definition at line 147 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_cs_pad

uint8_t sl_sio_spi_t::spi_cs_pad

SIO SPI CS pad.

Definition at line 148 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_clk_port

uint8_t sl_sio_spi_t::spi_clk_port

SIO SPI CLK port.

Definition at line 149 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_clk_pin

uint8_t sl_sio_spi_t::spi_clk_pin

SIO SPI CLK pin.

Definition at line 150 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_clk_mux

uint8_t sl_sio_spi_t::spi_clk_mux

SIO SPI CLK mux.

Definition at line 151 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_clk_pad

uint8_t sl_sio_spi_t::spi_clk_pad

SIO SPI CLK pad.

Definition at line 152 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_mosi_port

uint8_t sl_sio_spi_t::spi_mosi_port

SIO SPI MOSI port.

Definition at line 153 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_mosi_pin

uint8_t sl_sio_spi_t::spi_mosi_pin

SIO SPI MOSI pin.

Definition at line 154 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_mosi_mux

uint8_t sl_sio_spi_t::spi_mosi_mux

SIO SPI MOSI mux.

Definition at line 155 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_mosi_pad

uint8_t sl_sio_spi_t::spi_mosi_pad

SIO SPI MOSI pad.

Definition at line 156 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_miso_port

uint8_t sl_sio_spi_t::spi_miso_port

SIO SPI MISO port.

Definition at line 157 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_miso_pin

uint8_t sl_sio_spi_t::spi_miso_pin

SIO SPI MISO pin.

Definition at line 158 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_miso_mux

uint8_t sl_sio_spi_t::spi_miso_mux

SIO SPI MISO mux.

Definition at line 159 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

spi_miso_pad

uint8_t sl_sio_spi_t::spi_miso_pad

SIO SPI MISO pad.

Definition at line 160 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_uart_t

Structure to hold port and pin of sio uart.

Public Attributes

uint8_t	uart_tx_port SIO UART TX port.
uint8_t	uart_tx_pin SIO UART TX pin.
uint8_t	uart_tx_mux SIO UART TX mux.
uint8_t	uart_tx_pad SIO UART TX pad.
uint8_t	uart_rx_port SIO UART RX port.
uint8_t	uart_rx_pin SIO UART RX pin.
uint8_t	uart_rx_mux SIO UART RX mux.
uint8_t	uart_rx_pad SIO UART RX pad.

Public Attribute Documentation

uart_tx_port

```
uint8_t sl_sio_uart_t::uart_tx_port
```

SIO UART TX port.

Definition at line 165 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

uart_tx_pin

```
uint8_t sl_sio_uart_t::uart_tx_pin
```

SIO UART TX pin.

Definition at line 166 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

uart_tx_mux

uint8_t sl_sio_uart_t::uart_tx_mux

SIO UART TX mux.

Definition at line 167 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

uart_tx_pad

uint8_t sl_sio_uart_t::uart_tx_pad

SIO UART TX pad.

Definition at line 168 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

uart_rx_port

uint8_t sl_sio_uart_t::uart_rx_port

SIO UART RX port.

Definition at line 169 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

uart_rx_pin

uint8_t sl_sio_uart_t::uart_rx_pin

SIO UART RX pin.

Definition at line 170 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

uart_rx_mux

uint8_t sl_sio_uart_t::uart_rx_mux

SIO UART RX mux.

Definition at line 171 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

uart_rx_pad

uint8_t sl_sio_uart_t::uart_rx_pad

SIO UART RX pad.

Definition at line 172 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

sl_sio_i2c_t

Structure to hold port and pin of sio i2c.

Public Attributes

uint8_t	i2c_sda_port SIO I2C SDA port.
uint8_t	i2c_sda_pin SIO I2C SDA pin.
uint8_t	i2c_sda_mux SIO I2C SDA mux.
uint8_t	i2c_sda_pad SIO I2C SDA pad.
uint8_t	i2c_scl_port SIO I2C SCL port.
uint8_t	i2c_scl_pin SIO I2C SCL pin.
uint8_t	i2c_scl_mux SIO I2C SCL mux.
uint8_t	i2c_scl_pad SIO I2C SCL pad.

Public Attribute Documentation

i2c_sda_port

```
uint8_t sl_sio_i2c_t::i2c_sda_port
```

SIO I2C SDA port.

Definition at line 177 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

i2c_sda_pin

```
uint8_t sl_sio_i2c_t::i2c_sda_pin
```

SIO I2C SDA pin.

Definition at line 178 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

i2c_sda_mux

uint8_t sl_sio_i2c_t::i2c_sda_mux

SIO I2C SDA mux.

Definition at line 179 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

i2c_sda_pad

uint8_t sl_sio_i2c_t::i2c_sda_pad

SIO I2C SDA pad.

Definition at line 180 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

i2c_scl_port

uint8_t sl_sio_i2c_t::i2c_scl_port

SIO I2C SCL port.

Definition at line 181 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

i2c_scl_pin

uint8_t sl_sio_i2c_t::i2c_scl_pin

SIO I2C SCL pin.

Definition at line 182 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

i2c_scl_mux

uint8_t sl_sio_i2c_t::i2c_scl_mux

SIO I2C SCL mux.

Definition at line 183 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

i2c_scl_pad

uint8_t sl_sio_i2c_t::i2c_scl_pad

SIO I2C SCL pad.

Definition at line 184 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sio.h

Synchronous Serial Interface

Synchronous Serial Interface

Modules

sl_ssi_version_t

sl_ssi_control_config_t

sl_ssi_clock_config_t

Enumerations

enum ssi_event_typedef_t {

SSI_EVENT_TRANSFER_COMPLETE = ARM_SPI_EVENT_TRANSFER_COMPLETE
SSI_EVENT_DATA_LOST = ARM_SPI_EVENT_DATA_LOST
SSI_EVENT_MODE_FAULT = ARM_SPI_EVENT_MODE_FAULT

}

Enumeration for different SSI callback events.

enum ssi_peripheral_clock_mode_t {

SL_SSI_PERIPHERAL_CPOL0_CPHA0 = ARM_SPI_CPOL0_CPHA0
SL_SSI_PERIPHERAL_CPOL0_CPHA1 = ARM_SPI_CPOL0_CPHA1
SL_SSI_PERIPHERAL_CPOL1_CPHA0 = ARM_SPI_CPOL1_CPHA0
SL_SSI_PERIPHERAL_CPOL1_CPHA1 = ARM_SPI_CPOL1_CPHA1
SL_SSI_PERIPHERAL_TI_SSI = ARM_SPI_TI_SSI
SL_SSI_PERIPHERAL_MICROWIRE = ARM_SPI_MICROWIRE
SL_SSI_PERIPHERAL_MODE_LAST

}

Enumeration for different SSI peripheral clock modes.

enum sl_ssi_instance_t {

SL_SSI_MASTER_ACTIVE = ARM_SPI_MODE_MASTER
SL_SSI_SLAVE_ACTIVE = ARM_SPI_MODE_SLAVE
SL_SSI_ULP_MASTER_ACTIVE
SL_SSI_INSTANCE_LAST

}

Enumeration for different SSI peripheral device modes.

enum sl_ssi_slave_number_t {

SSI_SLAVE_0
SSI_SLAVE_1
SSI_SLAVE_2
SSI_SLAVE_3
SSI_SLAVE_NUMBER_LAST_ENUM

}

Enumeration for SSI Slave numbers.

Typedefs

typedef ARM_POWER_ST-

ATE	sl_ssi_power_state_t renaming arm power state structure
typedef ARM_SPI_STATUS	sl_ssi_status_t renaming arm SPI status
typedef ARM_DRIVER_SPI	sl_ssi_driver_t Renaming SSI driver structure.
typedef const void *	sl_ssi_handle_t SSI Handle to be passed into APIs.
typedef ARM_SPI_SignalEvent_t	sl_ssi_signal_event_t Callback typedef for SSI.

Variables

uint8_t	release Release version number.
uint8_t	major sqa version number
uint8_t	minor dev version number
uint8_t	bit_width bit width either 8 or 16 bit
uint32_t	device_mode mode such as Master or Slave mode
uint32_t	clock_mode clock mode such as CPOL0 CPHA1
uint32_t	baud_rate baud rate for SSI
uint32_t	receive_sample_delay Delay for receive input signal.
uint16_t	division_factor Clock Division Factor.
uint16_t	intf_pll_500_control_value intf PLL control value
uint32_t	intf_pll_clock intf PLL clock frequency
uint32_t	intf_pll_reference_clock intf PLL reference clock frequency
uint32_t	soc_pll_clock SoC PLL Clock frequency.
uint32_t	soc_pll_reference_clock SoC PLL reference clock frequency.

uint8_t	soc_pll_mm_count_value SoC PLL count value.

Functions

sl_status_t	sl_si91x_ssi_configure_clock(sl_ssi_clock_config_t *clock_config) Configure the SSI clock.
sl_status_t	sl_si91x_ssi_init(sl_ssi_instance_t instance, sl_ssi_handle_t *ssi_handle) Initialize the SSI master.
sl_status_t	sl_si91x_ssi_deinit(sl_ssi_handle_t ssi_handle) Uninitialize the SSI primary.
sl_status_t	sl_si91x_ssi_set_configuration(sl_ssi_handle_t ssi_handle, sl_ssi_control_config_t *control_configuration, sl_ssi_slave_number_t slave_number) Control the SPI interface.
sl_status_t	sl_si91x_ssi_receive_data(sl_ssi_handle_t ssi_handle, void *data, uint32_t data_length) Start receiving data from the SPI interface.
sl_status_t	sl_si91x_ssi_send_data(sl_ssi_handle_t ssi_handle, const void *data, uint32_t data_length) Start sending data from SPI interface.
sl_status_t	sl_si91x_ssi_transfer_data(sl_ssi_handle_t ssi_handle, const void *data_out, void *data_in, uint32_t data_length) Start sending/receiving bi-directional full duplex data to/from SPI secondary.
sl_ssi_status_t	sl_si91x_ssi_get_status(sl_ssi_handle_t ssi_handle) Get the SSI status.
sl_ssi_version_t	sl_si91x_ssi_get_version(void) Get the driver version.
uint32_t	sl_si91x_ssi_get_rx_data_count(sl_ssi_handle_t ssi_handle) Get RX transferred data count with connected device.
uint32_t	sl_si91x_ssi_get_tx_data_count(sl_ssi_handle_t ssi_handle) Get TX transferred data count with connected device.
sl_status_t	sl_si91x_ssi_register_event_callback(sl_ssi_handle_t ssi_handle, sl_ssi_signal_event_t callback_event) Register the user event callback.
void	sl_si91x_ssi_unregister_event_callback(void) Unregister the user event callback.
uint32_t	sl_si91x_ssi_get_clock_division_factor(sl_ssi_handle_t ssi_handle) Get the clock division factor.
uint32_t	sl_si91x_ssi_get_frame_length(sl_ssi_handle_t ssi_handle) Get the frame length (bit width).
uint32_t	sl_si91x_ssi_get_tx_fifo_threshold(sl_ssi_handle_t ssi_handle) To fetch the transfer fifo threshold value, this value controls the level of entries at which the transmit FIFO controller triggers an interrupt.
uint32_t	sl_si91x_ssi_get_rx_fifo_threshold(sl_ssi_handle_t ssi_handle) To fetch the receiver fifo threshold value, this value controls the level of entries at which the receive FIFO controller triggers an interrupt.
uint32_t	sl_si91x_ssi_get_receiver_sample_delay(sl_ssi_handle_t ssi_handle) To fetch the receiver sample delay value, it used to delay the sample of the rxd input signal.

__STATIC_INLINE sl_status_t sl_si91x_ssi_set_slave_number(uint8_t number)
Set the secondary number in multi-secondary operation.

Enumeration Documentation

ssi_event_typedef_t

> ssi_event_typedef_t

Enumeration for different SSI callback events.

Enumerator	
SSI_EVENT_TRANSFER_COMPLETE	Transfer complete event.
SSI_EVENT_DATA_LOST	Data lost event.
SSI_EVENT_MODE_FAULT	Mode fault event.

Definition at line 65 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

ssi_peripheral_clock_mode_t

> ssi_peripheral_clock_mode_t

Enumeration for different SSI peripheral clock modes.

Enumerator	
SL_SSI_PERIPHERAL_CPOL0_CPHA0	SSI_PERIPHERAL MODE 0 CPOL0_CPHA0.
SL_SSI_PERIPHERAL_CPOL0_CPHA1	SSI_PERIPHERAL MODE 1 CPOL0_CPHA1.
SL_SSI_PERIPHERAL_CPOL1_CPHA0	SSI_PERIPHERAL MODE 2 CPOL1_CPHA0.
SL_SSI_PERIPHERAL_CPOL1_CPHA1	SSI_PERIPHERAL MODE 3 CPOL1_CPHA1.
SL_SSI_PERIPHERAL_TI_SSI	SSI_PERIPHERAL MODE TI_SSI.
SL_SSI_PERIPHERAL_MICROWIRE	SSI_PERIPHERAL MODE MICROWIRE.
SL_SSI_PERIPHERAL_MODE_LAST	SSI_PERIPHERAL MODE_LAST.

Definition at line 75 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_instance_t

> sl_ssi_instance_t

Enumeration for different SSI peripheral device modes.

Enumerator	
SL_SSI_MASTER_ACTIVE	SSI DEVICE MODE MASTER.
SL_SSI_SLAVE_ACTIVE	SSI DEVICE MODE SLAVE.
SL_SSI_ULP_MASTER_ACTIVE	ULP SSI DEVICE MODE MASTER.
SL_SSI_INSTANCE_LAST	LSAT member of enum for validation.

Definition at line 86 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_slave_number_t

sl_ssi_slave_number_t

Enumeration for SSI Slave numbers.

Enumerator	
SSI_SLAVE_0	Slave No. 1.
SSI_SLAVE_1	Slave No. 2.
SSI_SLAVE_2	Slave No. 3.
SSI_SLAVE_3	Slave No. 4.
SSI_SLAVE_NUMBER_LAST_ENUM	Last member of enum for validation.

Definition at line 118 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

Typedef Documentation

sl_ssi_power_state_t

typedef ARM_POWER_STATE sl_ssi_power_state_t

renaming arm power state structure

Definition at line 52 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_status_t

typedef ARM_SPI_STATUS sl_ssi_status_t

renaming arm SPI status

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_driver_t

typedef ARM_DRIVER_SPI sl_ssi_driver_t

Renaming SSI driver structure.

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_handle_t

typedef const void* sl_ssi_handle_t

SSI Handle to be passed into APIs.

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_signal_event_t

> typedef ARM_SPI_SignalEvent_t sl_ssi_signal_event_t

Callback typedef for SSI.

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

Variable Documentation

release

> uint8_t sl_ssi_version_t::release

Release version number.

Definition at line 59 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

major

> uint8_t sl_ssi_version_t::major

sqa version number

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

minor

> uint8_t sl_ssi_version_t::minor

dev version number

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

bit_width

> uint8_t sl_ssi_control_config_t::bit_width

bit width either 8 or 16 bit

Definition at line 97 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

device_mode

> uint32_t sl_ssi_control_config_t::device_mode

mode such as Master or Slave mode

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

clock_mode

> uint32_t sl_ssi_control_config_t::clock_mode

clock mode such as CPOL0 CPHA1

Definition at line 99 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

baud_rate

> uint32_t sl_ssi_control_config_t::baud_rate

baud rate for SSI

Definition at line 100 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

receive_sample_delay

> uint32_t sl_ssi_control_config_t::receive_sample_delay

Delay for receive input signal.

Definition at line 101 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

division_factor

> uint16_t sl_ssi_clock_config_t::division_factor

Clock Division Factor.

Definition at line 108 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

intf_pll_500_control_value

> uint16_t sl_ssi_clock_config_t::intf_pll_500_control_value

intf PLL control value

Definition at line 109 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

intf_pll_clock

> uint32_t sl_ssi_clock_config_t::intf_pll_clock

intf PLL clock frequency

Definition at line 110 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

intf_pll_reference_clock

> uint32_t sl_ssi_clock_config_t::intf_pll_reference_clock

intf PLL reference clock frequency

Definition at line 111 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

soc_pll_clock

> uint32_t sl_ssi_clock_config_t::soc_pll_clock

SoC PLL Clock frequency.

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

soc_pll_reference_clock

> uint32_t sl_ssi_clock_config_t::soc_pll_reference_clock

SoC PLL reference clock frequency.

Definition at line 113 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

soc_pll_mm_count_value

> uint8_t sl_ssi_clock_config_t::soc_pll_mm_count_value

SoC PLL count value.

Definition at line 114 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

Function Documentation

sl_si91x_ssi_configure_clock

> sl_status_t sl_si91x_ssi_configure_clock (sl_ssi_clock_config_t *clock_config)

Configure the SSI clock.

Parameters

[in]	clock_config	Pointer to clock config structure sl_ssi_clock_config_t

Returns

- status 0 if successful, else error code as follow.
  - SL_STATUS_OK (0x0000) - Success, timer clock-source parameters configured properly

Definition at line 133 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_init

> sl_status_t sl_si91x_ssi_init (sl_ssi_instance_t instance, sl_ssi_handle_t *ssi_handle)

Initialize the SSI master.

Parameters

[in]	instance	(Master/Slave/ULP Master), ssi_handle
N/A	ssi_handle	

Returns

- status 0 if successful, else error code as follow.
  - SL_STATUS_OK (0x0000) - Success, otherwise fail error code as follow

Definition at line 142 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_deinit

> sl_status_t sl_si91x_ssi_deinit (sl_ssi_handle_t ssi_handle)

Uninitialize the SSI primary.

Parameters

[in]	ssi_handle	Handle

Returns

- status 0 if successful, else error code as follow.
  - SL_STATUS_OK (0x0000) - Success, otherwise fail error code as follow

Definition at line 151 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_set_configuration

> sl_status_t sl_si91x_ssi_set_configuration (sl_ssi_handle_t ssi_handle, sl_ssi_control_config_t *control_configuration, sl_ssi_slave_number_t slave_number)

Control the SPI interface.

Parameters

[in]	ssi_handle	handle, pointer to control config structure.
N/A	control_configuration	
N/A	slave_number	

Returns

- status 0 if successful, else error code as follow.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - invalid parameters
  - SL_STATUS_OK (0x0000) - Success

Definition at line 161 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_receive_data

> sl_status_t sl_si91x_ssi_receive_data (sl_ssi_handle_t ssi_handle, void *data, uint32_t data_length)

Start receiving data from the SPI interface.

Parameters

[in]	ssi_handle	handle, pointer to Rx data buffer, length of data
N/A	data	
N/A	data_length	

Returns

- status 0 if successful, else error code as follow.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_OK (0x0000) - Success

Definition at line `173` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h`

sl_si91x_ssi_send_data

> sl_status_t sl_si91x_ssi_send_data (sl_ssi_handle_t ssi_handle, const void *data, uint32_t data_length)

Start sending data from SPI interface.

Parameters

[in]	ssi_handle	handle, pointer to Tx data buffer, length of data
N/A	data	
N/A	data_length	

Returns

- status 0 if successful, else error code as follow.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer
  - SL_STATUS_OK (0x0000) - Success

Definition at line `183` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h`

sl_si91x_ssi_transfer_data

> sl_status_t sl_si91x_ssi_transfer_data (sl_ssi_handle_t ssi_handle, const void *data_out, void *data_in, uint32_t data_length)

Start sending/receiving bi-directional full duplex data to/from SPI secondary.

Parameters

[in]	ssi_handle	handle, pointer to Tx buffer, pointer to Rx buffer, and data length
N/A	data_out	
N/A	data_in	
N/A	data_length	

Returns

- status 0 if successful, else error code as follow.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success, otherwise error code as follow on failure.

Definition at line 194 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_status

```
sl_ssi_status_t sl_si91x_ssi_get_status (sl_ssi_handle_t ssi_handle)
```

Get the SSI status.

Parameters

[in]	ssi_handle	handle

Returns

- busy, data lost or mode fault returns as a bit field.

Definition at line 206 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_version

```
sl_ssi_version_t sl_si91x_ssi_get_version (void)
```

Get the driver version.

Parameters

[in]		

Returns

- driver version.

Definition at line 215 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_rx_data_count

```
uint32_t sl_si91x_ssi_get_rx_data_count (sl_ssi_handle_t ssi_handle)
```

Get RX transferred data count with connected device.

Parameters

[in]	ssi_handle	handle

Returns

- transferred data count

Definition at line 223 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_tx_data_count

```
uint32_t sl_si91x_ssi_get_tx_data_count (sl_ssi_handle_t ssi_handle)
```

Get TX transferred data count with connected device.

Parameters

[in]	ssi_handle	handle

Returns

- transferred data count

Definition at line 231 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_register_event_callback

```
sl_status_t sl_si91x_ssi_register_event_callback (sl_ssi_handle_t ssi_handle, sl_ssi_signal_event_t callback_event)
```

Register the user event callback.

Parameters

[in]	ssi_handle	handle
[in]	callback_event	pointer callback_event

It registers the callback, i.e., stores the callback function address and pass to the variable that is called in Interrupt Handler. If another callback is registered without unregistering previous callback then, it returns an error code as follow, so it is mandatory to unregister the callback before registering another callback.

Returns

- status 0 if successful, else error code as follow.
  - SL_STATUS_BUSY (0x0004) - The callback is already registered, unregister previous callback before registering new one.
  - SL_STATUS_OK (0x0000) - Success

Definition at line 247 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_unregister_event_callback

```
void sl_si91x_ssi_unregister_event_callback (void)
```

Unregister the user event callback.

Parameters

[in]		

It unregisters the callback, i.e., clears the callback function address and passes a NULL value to the variable.

Definition at line 256 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_clock_division_factor

```
uint32_t sl_si91x_ssi_get_clock_division_factor (sl_ssi_handle_t ssi_handle)
```

Get the clock division factor.

Parameters

[in]	ssi_handle	Pointer to the SSI driver handle

Returns

- factor(int32_t) Clock division factor

Definition at line 264 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_frame_length

```
uint32_t sl_si91x_ssi_get_frame_length (sl_ssi_handle_t ssi_handle)
```

Get the frame length (bit width).

Parameters

[in]	ssi_handle	Pointer to the SSI driver handle

Returns

- frame_length (uint32_t) Frame length

Definition at line 272 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_tx_fifo_threshold

```
uint32_t sl_si91x_ssi_get_tx_fifo_threshold (sl_ssi_handle_t ssi_handle)
```

To fetch the transfer fifo threshold value, this value controls the level of entries at which the transmit FIFO controller triggers an interrupt.

Parameters

[in]	ssi_handle	Pointer to the SSI driver handle

Returns

- Transfer fifo threshold (uint32_t) The value of transfer fifo threshold

Definition at line 281 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_rx_fifo_threshold

```
uint32_t sl_si91x_ssi_get_rx_fifo_threshold (sl_ssi_handle_t ssi_handle)
```

To fetch the receiver fifo threshold value, this value controls the level of entries at which the receive FIFO controller triggers an interrupt.

Parameters

[in]	ssi_handle	Pointer to the SSI driver handle

Returns

- Receiver fifo threshold (uint32_t) The value of receiver fifo threshold

Definition at line 290 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_get_receiver_sample_delay

```
uint32_t sl_si91x_ssi_get_receiver_sample_delay (sl_ssi_handle_t ssi_handle)
```

To fetch the receiver sample delay value, it used to delay the sample of the rxd input signal.

Parameters

[in]	ssi_handle	Pointer to the SSI driver handle

Each value represents a single ssi_clk delay on the sample of the rxd signal.

Returns

- Receiver sample delay (uint32_t) The value of receiver sample delay

Definition at line 300 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_si91x_ssi_set_slave_number

> __STATIC_INLINE sl_status_t sl_si91x_ssi_set_slave_number (uint8_t number)

Set the secondary number in multi-secondary operation.

Parameters

[in]	number	Slave number

For single secondary also, this API needs to be called before transferring the data.

Returns

- none

Definition at line 310 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_version_t

Public Attributes

uint8_t	release Release version number.
uint8_t	major sqa version number
uint8_t	minor dev version number

Public Attribute Documentation

release

```
uint8_t sl_ssi_version_t::release
```

Release version number.

Definition at line 59 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

major

```
uint8_t sl_ssi_version_t::major
```

sqa version number

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

minor

```
uint8_t sl_ssi_version_t::minor
```

dev version number

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_control_config_t

typedef for SSI control config struct

Public Attributes

uint8_t	bit_width bit width either 8 or 16 bit
uint32_t	device_mode mode such as Master or Slave mode
uint32_t	clock_mode clock mode such as CPOL0 CPHA1
uint32_t	baud_rate baud rate for SSI
uint32_t	receive_sample_delay Delay for receive input signal.

Public Attribute Documentation

bit_width

```
uint8_t sl_ssi_control_config_t::bit_width
```

bit width either 8 or 16 bit

Definition at line 97 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

device_mode

```
uint32_t sl_ssi_control_config_t::device_mode
```

mode such as Master or Slave mode

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

clock_mode

```
uint32_t sl_ssi_control_config_t::clock_mode
```

clock mode such as CPOL0 CPHA1

Definition at line 99 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

baud_rate

> uint32_t sl_ssi_control_config_t::baud_rate

baud rate for SSI

Definition at line 100 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

receive_sample_delay

> uint32_t sl_ssi_control_config_t::receive_sample_delay

Delay for receive input signal.

Definition at line 101 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

sl_ssi_clock_config_t

typedef for SSI clock config struct

Public Attributes

uint16_t	division_factor Clock Division Factor.
uint16_t	intf_pll_500_control_value intf PLL control value
uint32_t	intf_pll_clock intf PLL clock frequency
uint32_t	intf_pll_reference_clock intf PLL reference clock frequency
uint32_t	soc_pll_clock SoC PLL Clock frequency.
uint32_t	soc_pll_reference_clock SoC PLL reference clock frequency.
uint8_t	soc_pll_mm_count_value SoC PLL count value.

Public Attribute Documentation

division_factor

```
uint16_t sl_ssi_clock_config_t::division_factor
```

Clock Division Factor.

Definition at line 108 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

intf_pll_500_control_value

```
uint16_t sl_ssi_clock_config_t::intf_pll_500_control_value
```

intf PLL control value

Definition at line 109 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

intf_pll_clock

```
uint32_t sl_ssi_clock_config_t::intf_pll_clock
```

intf PLL clock frequency

Definition at line 110 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

intf_pll_reference_clock

```
uint32_t sl_ssi_clock_config_t::intf_pll_reference_clock
```

intf PLL reference clock frequency

Definition at line 111 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

soc_pll_clock

```
uint32_t sl_ssi_clock_config_t::soc_pll_clock
```

SoC PLL Clock frequency.

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

soc_pll_reference_clock

```
uint32_t sl_ssi_clock_config_t::soc_pll_reference_clock
```

SoC PLL reference clock frequency.

Definition at line 113 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

soc_pll_mm_count_value

```
uint8_t sl_ssi_clock_config_t::soc_pll_mm_count_value
```

SoC PLL count value.

Definition at line 114 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ssi.h

System RTC

System RTC

Modules

sl_sysrtc_clock_config_t

sl_sysrtc_interrupt_enables_t

sl_sysrtc_version_t

Enumerations

enum sl_sysrtc_group_number_t {

SL_SYSRTC_GROUP_0
SL_SYSRTC_GROUP_1
SL_SYSRTC_GROUP_LAST

}

Enumeration to represent sysrtc group numbers.

enum sl_sysrtc_channel_number_t {

SL_SYSRTC_CHANNEL_0
SL_SYSRTC_CHANNEL_1
SL_SYSRTC_CHANNEL_LAST

}

Enumeration to represent sysrtc channel numbers.

Typedefs

typedef rsi_sysrtc_clk_inp_t sl_clock_sources_t

typedef rsi_sysrtc_config_t sl_sysrtc_config_t

typedef rsi_sysrtc_group_config_t sl_sysrtc_group_config_t

typedef rsi_sysrtc_group_channel_compare_config_t sl_sysrtc_group_compare_channel_action_config_t

typedef rsi_sysrtc_group_channel_capture_config_t sl_sysrtc_group_capture_channel_input_edge_config_t

typedef void(* sl_sysrtc_callback_t)(void *callback_flag)
Typedef for the function pointer of the interrupt callback function.

Functions

sl_status_t sl_si91x_sysrtc_init(const sl_sysrtc_config_t *config_ptr)
Initializes SYSRTC and enables the peripheral.

sl_status_t sl_si91x_sysrtc_configure_clock(sl_sysrtc_clock_config_t *clk_ptr)
Configures SYSRTC input clock source.

sl_status_t sl_si91x_sysrtc_configure_group(sl_sysrtc_group_number_t group_number, sl_sysrtc_group_config_t const *config_ptr)
Configures SYSRTC compare and capture channel's groups, configures group number, and enables its compare & capture channels.

sl_status_t sl_si91x_sysrtc_register_callback(sl_sysrtc_callback_t on_sysrtc_callback, void *callback_flag_value, sl_sysrtc_group_number_t group_number, sl_sysrtc_interrupt_enables_t *interrupt_enable_ptr)
Registers callback of timer interrupt and enables respective interrupts as per selected interrupt flag.

sl_status_t sl_si91x_sysrtc_unregister_callback(sl_sysrtc_group_number_t group_number, sl_sysrtc_interrupt_enables_t *interrupt_enabled_handle)
Unregisters timer interrupt callback and disables interrupts as per selected interrupt flag.

sl_status_t sl_si91x_sysrtc_set_compare_value(sl_sysrtc_group_number_t group_number, sl_sysrtc_channel_number_t channel, uint32_t compare_value)
Sets SYSRTC compare value for the selected channel of the given group.

sl_status_t sl_si91x_sysrtc_get_compare_value(sl_sysrtc_group_number_t group_number, sl_sysrtc_channel_number_t channel, uint32_t *compare_value)
Gets SYSRTC compare register value for the selected channel of the given group.

sl_status_t sl_si91x_sysrtc_sets_register_capture_input(sl_sysrtc_group_number_t group_number)
Sets register input high for the capture channel of the passed group of SYSRTC.

sl_status_t sl_si91x_sysrtc_set_gpio_as_capture_input(sl_sysrtc_group_number_t group_number)
Configures SYSRTC input GPIO pins for SYSRTC capture channel input.

sl_status_t sl_si91x_sysrtc_set_compare_output_gpio(sl_sysrtc_group_number_t group_number, sl_sysrtc_channel_number_t channel)
Configures SYSRTC compare output GPIO pins of the given channel of the given group.

sl_status_t sl_si91x_sysrtc_get_count(uint32_t *count_value)
Gets SYSRTC current counter register value.

sl_status_t sl_si91x_sysrtc_get_capture_value(sl_sysrtc_group_number_t group_number, uint32_t *capture_value)
Gets SYSRTC capture register value of the given group.

sl_status_t sl_si91x_sysrtc_get_compare_output(sl_sysrtc_group_number_t group_number, sl_sysrtc_channel_number_t channel, uint32_t *compare_output_bit)
Gets SYSRTC compare register value of the given group and channel.

sl_status_t sl_si91x_sysrtc_is_running(boolean_t *running_status)
Gets SYSRTC running status, status is true if running else false if stopped.

sl_status_t sl_si91x_sysrtc_is_locked(boolean_t *lock_status)
Gets SYSRTC lock status, status is true if locked else false if unlocked.

void sl_si91x_sysrtc_enable_input_output_gpio(bool is_gpio_enabled)
Enables SYSRTC IO through GPIO if passed true else through register if passed false.

__STATIC_INLINE void	sl_si91x_sysrtc_start(void) Starts SYSRTC counter.
__STATIC_INLINE void	sl_si91x_sysrtc_stop(void) Stops the SYSRTC counter.
__STATIC_INLINE void	sl_si91x_sysrtc_reset(void) Restores SYSRTC to its reset state.
__STATIC_INLINE void	sl_si91x_sysrtc_lock(void) Locks SYSRTC registers.
__STATIC_INLINE void	sl_si91x_sysrtc_unlock(void) Unlocks SYSRTC registers.
__STATIC_INLINE void	sl_si91x_sysrtc_set_count(uint32_t value) Sets the SYSRTC counter register value of counter.
sl_sysrtc_version_t	sl_si91x_sysrtc_get_version(void) To get the release version of SYSRTC.
void	sl_si91x_sysrtc_deinit(void) De-initializes SYSRTC by disabling its clock.

Enumeration Documentation

sl_sysrtc_group_number_t

> sl_sysrtc_group_number_t

Enumeration to represent sysrtc group numbers.

	Enumerator
SL_SYSRTC_GROUP_0	enum for SYSRTC group-0
SL_SYSRTC_GROUP_1	enum for SYSRTC group-1
SL_SYSRTC_GROUP_LAST	Last member of enum for validation.

Definition at line 70 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_channel_number_t

> sl_sysrtc_channel_number_t

Enumeration to represent sysrtc channel numbers.

	Enumerator
SL_SYSRTC_CHANNEL_0	enum for SYSRTC channel-0
SL_SYSRTC_CHANNEL_1	enum for SYSRTC channel-1
SL_SYSRTC_CHANNEL_LAST	Last member of enum for validation.

Definition at line 77 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

Typedef Documentation

sl_clock_sources_t

```
typedef rsi_sysrtc_clk_inp_t sl_clock_sources_t
```

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_config_t

```
typedef rsi_sysrtc_config_t sl_sysrtc_config_t
```

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_group_config_t

```
typedef rsi_sysrtc_group_config_t sl_sysrtc_group_config_t
```

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_group_compare_channel_action_config_t

```
typedef rsi_sysrtc_group_channel_compare_config_t sl_sysrtc_group_compare_channel_action_config_t
```

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_group_capture_channel_input_edge_config_t

```
typedef rsi_sysrtc_group_channel_capture_config_t sl_sysrtc_group_capture_channel_input_edge_config_t
```

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_callback_t

```
typedef void(* sl_sysrtc_callback_t) (void *callback_flag) )(void *callback_flag)
```

Typedef for the function pointer of the interrupt callback function.

Parameters

N/A	callback_flag	(void *) parameter for updating flag values

Definition at line 67 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

Function Documentation

sl_si91x_sysrtc_init

```
sl_status_t sl_si91x_sysrtc_init (const sl_sysrtc_config_t *config_ptr)
```

Initializes SYSRTC and enables the peripheral.

Parameters

[in]	config_ptr	Pointer to SYSRTC config structure sl_sysrtc_config_t.

- Pre-conditions:
  - sl_si91x_sysrtc_configure_clock

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer.
  - SL_STATUS_OK (0x0000) - Success, SYSRTC initialized properly.

Definition at line 121 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_configure_clock

```
sl_status_t sl_si91x_sysrtc_configure_clock (sl_sysrtc_clock_config_t *clk_ptr)
```

Configures SYSRTC input clock source.

Parameters

[in]	clk_ptr	Pointer to clock configuration structure sl_sysrtc_clock_config_t.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Clock source parameter has an invalid value.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer.
  - SL_STATUS_OK (0x0000) - Success, timer clock-source parameters configured properly.

Definition at line 132 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_configure_group

```
sl_status_t sl_si91x_sysrtc_configure_group (sl_sysrtc_group_number_t group_number, sl_sysrtc_group_config_t const *config_ptr)
```

Configures SYSRTC compare and capture channel's groups, configures group number, and enables its compare & capture channels.

Parameters

[in]	group_number	SYSRTC group number to be used.
[in]	config_ptr	(const *) Pointer to group configuration structure sl_sysrtc_group_config_t.

Also configures match out actions for respective compare channel and input events for capture.

- Pre-conditions:
  - sl_si91x_sysrtc_configure_clock
  - sl_si91x_sysrtc_init

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Counter direction parameter has an invalid value.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer.

SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 150 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_register_callback

```
sl_status_t sl_si91x_sysrtc_register_callback (sl_sysrtc_callback_t on_sysrtc_callback, void *callback_flag_value,
sl_sysrtc_group_number_t group_number, sl_sysrtc_interrupt_enables_t *interrupt_enable_ptr)
```

Registers callback of timer interrupt and enables respective interrupts as per selected interrupt flag.

Parameters

[in]	on_sysrtc_callback	(function pointer) Callback function pointer to be called when timer interrupt occurred.
[in]	callback_flag_value	(void *) pointer to interrupt flag value variable sl_sysrtc_callback_t.
[in]	group_number	SYSRTC group number whose interrupts needs to be enabled.
[in]	interrupt_enable_ptr	pointer to interrupt enable structure sl_sysrtc_interrupt_enables_t.

- Pre-conditions:
  - sl_si91x_sysrtc_init
  - sl_si91x_sysrtc_configure_clock
  - sl_si91x_sysrtc_configure_group(), keep respective interrupt channel enable
  - sl_si91x_sysrtc_unregister_timeout_callback(), if already registered for any interrupt

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - interrupt_enable_ptr parameter is a null pointer.
  - SL_STATUS_BUSY (0x0004) - The callback is already registered, unregister previous callback before registering a new one.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number parameter has an invalid value.
  - SL_STATUS_OK (0x0000) - Successfully registered timer timeout callback.

Definition at line 175 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_unregister_callback

```
sl_status_t sl_si91x_sysrtc_unregister_callback (sl_sysrtc_group_number_t group_number, sl_sysrtc_interrupt_enables_t
*interrupt_enabled_handle)
```

Unregisters timer interrupt callback and disables interrupts as per selected interrupt flag.

Parameters

[in]	group_number	pointer to interrupts enable structure sl_sysrtc_interrupt_enables_t.
[in]	interrupt_enabled_handle	SYSRTC group number whose interrupts needs to be enabled.

- Pre-conditions:
  - sl_si91x_sysrtc_register_callback(), first register a particular interrupt flag.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number parameter has an invalid value.
  - SL_STATUS_NULL_POINTER (0x0022) - parameter is a null pointer.
  - SL_STATUS_OK (0x0000) - Successfully unregistered timer interrupt callback.

Definition at line 194 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_set_compare_value

> sl_status_t sl_si91x_sysrtc_set_compare_value (sl_sysrtc_group_number_t group_number, sl_sysrtc_channel_number_t channel, uint32_t compare_value)

Sets SYSRTC compare value for the selected channel of the given group.

Parameters

[in]	group_number	SYSRTC group number to use.
[in]	channel	Channel number to use.
[in]	compare_value	Compare register value.

- Pre-conditions:
  - First enable the compare channel of the respective group through sl_si91x_sysrtc_configure_group.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number or channel parameter has an invalid value.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line `211` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h`

sl_si91x_sysrtc_get_compare_value

> sl_status_t sl_si91x_sysrtc_get_compare_value (sl_sysrtc_group_number_t group_number, sl_sysrtc_channel_number_t channel, uint32_t *compare_value)

Gets SYSRTC compare register value for the selected channel of the given group.

Parameters

[in]	group_number	SYSRTC group number to use.
[in]	channel	Channel number to use.
[in]	compare_value	Pointer to the variable to store compare value read.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number or channel parameter has an invalid value.
  - SL_STATUS_NULL_POINTER (0x0022) - compare_value parameter is a null pointer.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line `226` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h`

sl_si91x_sysrtc_sets_register_capture_input

> sl_status_t sl_si91x_sysrtc_sets_register_capture_input (sl_sysrtc_group_number_t group_number)

Sets register input high for the capture channel of the passed group of SYSRTC.

Parameters

[in]	group_number	SYSRTC group number to use.

- Pre-conditions:

First enable & config capture channel of the respective group through sl_si91x_sysrtc_configure_group.
- Disable GPIO input through sl_si91x_sysrtc_enable_input_output_gpio, by passing false.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number parameter has an invalid value.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 244 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_set_gpio_as_capture_input

> sl_status_t sl_si91x_sysrtc_set_gpio_as_capture_input (sl_sysrtc_group_number_t group_number)

Configures SYSRTC input GPIO pins for SYSRTC capture channel input.

Parameters

[in]	group_number	SYSRTC group number to use.

- Pre-conditions:
  - First enable & config capture channel of the respective group through sl_si91x_sysrtc_configure_group.
  - Enable GPIO IO through sl_si91x_sysrtc_enable_input_output_gpio, by passing true.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number parameter has an invalid value.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 260 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_set_compare_output_gpio

> sl_status_t sl_si91x_sysrtc_set_compare_output_gpio (sl_sysrtc_group_number_t group_number, sl_sysrtc_channel_number_t channel)

Configures SYSRTC compare output GPIO pins of the given channel of the given group.

Parameters

[in]	group_number	SYSRTC group number to use.
[in]	channel	Channel number to use.

- Pre-conditions:
  - First enable & config output action of compare channel of the respective group through sl_si91x_sysrtc_configure_group.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number or channel parameter has an invalid value.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 275 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_get_count

sl_status_t sl_si91x_sysrtc_get_count (uint32_t *count_value)

Gets SYSRTC current counter register value.

Parameters

[in]	count_value	Pointer to the variable to store count value read.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - count_value parameter is a null pointer.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 286 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_get_capture_value

sl_status_t sl_si91x_sysrtc_get_capture_value (sl_sysrtc_group_number_t group_number, uint32_t *capture_value)

Gets SYSRTC capture register value of the given group.

Parameters

[in]	group_number	SYSRTC group number to use.
[in]	capture_value	Pointer to the variable to store capture value read.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - count_value parameter is a null pointer.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number parameter has an invalid value.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 298 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_get_compare_output

sl_status_t sl_si91x_sysrtc_get_compare_output (sl_sysrtc_group_number_t group_number, sl_sysrtc_channel_number_t channel, uint32_t *compare_output_bit)

Gets SYSRTC compare register value of the given group and channel.

Parameters

[in]	group_number	SYSRTC group number to use.
[in]	channel	Channel number to use.
[in]	compare_output_bit	Pointer to the variable to store compare output.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - compare_output parameter is a null pointer.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - group_number or channel parameter has an invalid value.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 311 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_is_running

```
sl_status_t sl_si91x_sysrtc_is_running (boolean_t *running_status)
```

Gets SYSRTC running status, status is true if running else false if stopped.

Parameters

[in]	running_status	Pointer to the variable to store SYSRTC running status.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - running_status parameter is a null pointer.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 323 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_is_locked

```
sl_status_t sl_si91x_sysrtc_is_locked (boolean_t *lock_status)
```

Gets SYSRTC lock status, status is true if locked else false if unlocked.

Parameters

[in]	lock_status	Pointer to the variable to store SYSRTC lock status.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - lock_status parameter is a null pointer.
  - SL_STATUS_OK (0x0000) - Success, parameters configured properly.

Definition at line 333 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_enable_input_output_gpio

```
void sl_si91x_sysrtc_enable_input_output_gpio (bool is_gpio_enabled)
```

Enables SYSRTC IO through GPIO if passed true else through register if passed false.

Parameters

[in]	is_gpio_enabled	Bool to enable or disable IO through GPIO.

Returns

- none

Definition at line 341 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_start

```
__STATIC_INLINE void sl_si91x_sysrtc_start (void)
```

Starts SYSRTC counter.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_sysrtc_init
  - sl_si91x_sysrtc_configure_clock(), keep software trigger disable here
  - sl_si91x_sysrtc_configure_group(), keep respective interrupt channel enable
  - sl_si91x_sysrtc_register_callback, keep respective interrupt enabled
  - sl_si91x_sysrtc_set_count

Note

- This function will send a start command to the SYSRTC peripheral. The SYSRTC peripheral will use some LF clock ticks before the command is executed. The rsi_sysrtc_wait_sync() function is used to wait for the start command to be executed.

Returns

- none

Definition at line `362` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h`

sl_si91x_sysrtc_stop

```
__STATIC_INLINE void sl_si91x_sysrtc_stop (void)
```

Stops the SYSRTC counter.

Parameters

N/A		

This function will send a stop command to the SYSRTC peripheral. The SYSRTC peripheral will use some LF clock ticks before the command is executed. The rsi_sysrtc_wait_sync() function can be used to wait for the stop command to be executed.

Note

- This function requires the SYSRTC to be enabled.

Returns

- none

Definition at line `380` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h`

sl_si91x_sysrtc_reset

```
__STATIC_INLINE void sl_si91x_sysrtc_reset (void)
```

Restores SYSRTC to its reset state.

Parameters

N/A		

Returns

- none

Definition at line 391 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_lock

```
__STATIC_INLINE void sl_si91x_sysrtc_lock (void)
```

Locks SYSRTC registers.

Parameters

N/A		

Note

- When SYSRTC registers are locked SYSRTC_EN, SYSRTC_CFG, SYSRTC_CMD, SYSRTC_SWRST, SYSRTC_CNT and SYSRTC_TOPCNT registers cannot be written to.

Returns

- none

Definition at line 405 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_unlock

```
__STATIC_INLINE void sl_si91x_sysrtc_unlock (void)
```

Unlocks SYSRTC registers.

Parameters

N/A		

- Pre-conditions:
  - sl_si91x_sysrtc_lock

Returns

- none

Definition at line 419 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_set_count

```
__STATIC_INLINE void sl_si91x_sysrtc_set_count (uint32_t value)
```

Sets the SYSRTC counter register value of counter.

Parameters

[in]	value	The new SYSRTC counter value.

- Pre-conditions:
  - sl_si91x_sysrtc_stop

Returns

- none

Definition at line 433 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_get_version

```
sl_sysrtc_version_t sl_si91x_sysrtc_get_version (void)
```

To get the release version of SYSRTC.

Parameters

[in]		

Returns

- (sl_sysrtc_version_t) type structure.

Definition at line 445 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_si91x_sysrtc_deinit

```
void sl_si91x_sysrtc_deinit (void)
```

De-initializes SYSRTC by disabling its clock.

Parameters

[in]		

Returns

- none

Definition at line 453 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_clock_config_t

Structure to hold the parameters of sysrtc clock-source configurations.

Public Attributes

uint8_t	clock_source input clock sources \refsl_clock_sources_t
uint32_t	division_factor clock division factor

Public Attribute Documentation

clock_source

```
uint8_t sl_sysrtc_clock_config_t::clock_source
```

input clock sources \refsl_clock_sources_t

Definition at line 85 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

division_factor

```
uint32_t sl_sysrtc_clock_config_t::division_factor
```

clock division factor

Definition at line 86 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_interrupt_enables_t

Structure to hold the parameters of sysrtc group interrupt enabling bits.

Public Attributes

boolean_t	group0_overflow_interrupt_is_enabled true to enable group0 overflow interrupt, false to disable it
boolean_t	group0_compare0_interrupt_is_enabled true to enable group0 overflow interrupt, false to disable it
boolean_t	group0_compare1_interrupt_is_enabled true to enable group0 overflow interrupt, false to disable it
boolean_t	group0_capture0_interrupt_is_enabled true to enable group0 overflow interrupt, false to disable it
boolean_t	group1_overflow_interrupt_is_enabled true to enable group1 overflow interrupt, false to disable it
boolean_t	group1_compare0_interrupt_is_enabled true to enable group1 overflow interrupt, false to disable it
boolean_t	group1_compare1_interrupt_is_enabled true to enable group1 overflow interrupt, false to disable it
boolean_t	group1_capture0_interrupt_is_enabled true to enable group1 overflow interrupt, false to disable it

Public Attribute Documentation

group0_overflow_interrupt_is_enabled

```
boolean_t sl_sysrtc_interrupt_enables_t::group0_overflow_interrupt_is_enabled
```

true to enable group0 overflow interrupt, false to disable it

Definition at line `91` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h`

group0_compare0_interrupt_is_enabled

```
boolean_t sl_sysrtc_interrupt_enables_t::group0_compare0_interrupt_is_enabled
```

true to enable group0 overflow interrupt, false to disable it

Definition at line `92` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h`

group0_compare1_interrupt_is_enabled

boolean_t sl_sysrtc_interrupt_enables_t::group0_compare1_interrupt_is_enabled

true to enable group0 overflow interrupt, false to disable it

Definition at line 93 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

group0_capture0_interrupt_is_enabled

boolean_t sl_sysrtc_interrupt_enables_t::group0_capture0_interrupt_is_enabled

true to enable group0 overflow interrupt, false to disable it

Definition at line 94 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

group1_overflow_interrupt_is_enabled

boolean_t sl_sysrtc_interrupt_enables_t::group1_overflow_interrupt_is_enabled

true to enable group1 overflow interrupt, false to disable it

Definition at line 95 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

group1_compare0_interrupt_is_enabled

boolean_t sl_sysrtc_interrupt_enables_t::group1_compare0_interrupt_is_enabled

true to enable group1 overflow interrupt, false to disable it

Definition at line 96 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

group1_compare1_interrupt_is_enabled

boolean_t sl_sysrtc_interrupt_enables_t::group1_compare1_interrupt_is_enabled

true to enable group1 overflow interrupt, false to disable it

Definition at line 97 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

group1_capture0_interrupt_is_enabled

boolean_t sl_sysrtc_interrupt_enables_t::group1_capture0_interrupt_is_enabled

true to enable group1 overflow interrupt, false to disable it

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

sl_sysrtc_version_t

Structure to hold the versions of peripheral API.

Public Attributes

uint8_t	release Release version number.
uint8_t	major sqa-version number
uint8_t	minor dev-version number

Public Attribute Documentation

release

```
uint8_t sl_sysrtc_version_t::release
```

Release version number.

Definition at line 103 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

major

```
uint8_t sl_sysrtc_version_t::major
```

sqa-version number

Definition at line 104 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

minor

```
uint8_t sl_sysrtc_version_t::minor
```

dev-version number

Definition at line 105 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_sysrtc.h

Ultra Low-Power Timer

Modules

ulp_timer_clk_src_config_t

ulp_timer_config_t

sl_ulp_timer_version_t

Enumerations

enum ulp_timer_instance_t {

ULP_TIMER_0
ULP_TIMER_1
ULP_TIMER_2
ULP_TIMER_3
ULP_TIMER_LAST

}

Enumeration to represent ulp-timer instances.

enum ulp_timer_mode_t {

ULP_TIMER_MODE_ONESHOT
ULP_TIMER_MODE_PERIODIC
ULP_TIMER_MODE_LAST

}

Enumeration to represent ULP-timer modes.

enum ulp_timer_type_t {

ULP_TIMER_TYP_DEFAULT
ULP_TIMER_TYP_1US
ULP_TIMER_TYP_256US
ULP_TIMER_TYP_LAST

}

Enumeration to represent ULP-timer types.

enum ulp_timer_clk_input_source_t {

ULP_TIMER_REF_CLK_SRC
ULP_TIMER_32KHZ_RO_CLK_SRC
ULP_TIMER_32KHZ_RC_CLK_SRC
ULP_TIMER_32KHZ_XTAL_CLK_SRC
ULP_TIMER_32MHZ_RC_CLK_SRC
ULP_TIMER_20MHZ_RO_CLK_SRC
ULP_TIMER_ULP_SOC_CLK_SRC
ULP_TIMER_ULP_CLK_SRC_LAST

}

Enumeration to represent values of clock sources to select as Timer clock.

Typedefs

typedef void(* ulp_timer_callback_t)(void)
Typedef for the function pointer of the callback function.

typedef CLK_ENABLE_T ulp_timer_clock_t
Renaming clock type enum.

typedef ulp_timer_dir_t ulp_timer_direction_t
Renaming clock type enum.

Variables

uint8_t ulp_timer_clk_type
true to enable static and false to enable dynamic clock type

boolean_t ulp_timer_sync_to_ulpss_pclk
true to enable and false to disable ULP timer in synchronous mode to ULPSS pclk

uint8_t ulp_timer_clk_input_src
timer input clock source, refer ulp_timer_clk_input_source_t for possible values

boolean_t ulp_timer_skip_switch_time
true to wait and false to Skip waiting for switching timer clk

uint8_t timer_num
timer number, SL_ULP_TIMER_NUMBER for default values

uint8_t timer_mode
timer mode, SL_ULP_TIMER_MODE for default values

uint8_t timer_type
timer type, SL_ULP_TIMER_TYPE for default values

uint32_t timer_match_value
timer match value SL_ULP_TIMER_MATCH_VALUE(delay time) in microseconds

uint8_t timer_direction
timer direction, SL_ULP_TIMER_DIRECTION for default values

uint8_t release
Release version number.

uint8_t major
sqa version number

uint8_t minor
dev version number

Functions

sl_status_t sl_si91x_ulp_timer_configure_clock(ulp_timer_clk_src_config_t *timer_clk_ptr)
Configure the ULP-Timer input clock source.

sl_status_t sl_si91x_ulp_timer_set_configuration(ulp_timer_config_t *timer_config_ptr)
Configure ULP-Timer parameters such as timer number, mode, type, match-value & direction.

sl_status_t sl_si91x_ulp_timer_start(ulp_timer_instance_t timer_num)
Start the ULP-Timer.

sl_status_t sl_si91x_ulp_timer_stop(ulp_timer_instance_t timer_num)
Stop the ULP-Timer.

sl_status_t	sl_si91x_ulp_timer_restart(ulp_timer_instance_t timer_num) Restart an already running ULP-Timer, means it will first stop the timer and then start it again.
sl_status_t	sl_si91x_ulp_timer_set_type(ulp_timer_instance_t timer_num, ulp_timer_type_t timer_type) Set the ULP-Timer type.
sl_status_t	sl_si91x_ulp_timer_set_direction(ulp_timer_instance_t timer_num, ulp_timer_direction_t timer_direction) Set the ULP-Timer direction as up-counting or down-counting.
sl_status_t	sl_si91x_ulp_timer_set_mode(ulp_timer_instance_t timer_num, ulp_timer_mode_t timer_mode) Set ULP-Timer Mode.
sl_status_t	sl_si91x_ulp_timer_set_count(ulp_timer_instance_t timer_num, uint32_t timer_match_value) Set ULP-Timer match value.
sl_status_t	sl_si91x_ulp_timer_get_count(ulp_timer_instance_t timer_num, uint32_t *count_value) Read the ULP-Timer count.
sl_status_t	sl_si91x_ulp_timer_get_type(ulp_timer_instance_t timer_num, uint32_t *timer_type) Read the ULP-Timer type.
sl_status_t	sl_si91x_ulp_timer_get_mode(ulp_timer_instance_t timer_num, uint32_t *timer_mode) Read the ULP-Timer mode.
sl_status_t	sl_si91x_ulp_timer_get_direction(ulp_timer_instance_t timer_num, uint32_t *timer_direction) Read the ULP-Timer direction (up-counter or down-counter).
sl_status_t	sl_si91x_ulp_timer_register_timeout_callback(ulp_timer_instance_t timer_num, ulp_timer_callback_t on_timeout_callback) Register callback of timer timeout interrupt and enabling it.
sl_status_t	sl_si91x_ulp_timer_unregister_timeout_callback(ulp_timer_instance_t timer_num) Unregister callback of timer timeout interrupt and disabling it.
sl_status_t	sl_si91x_ulp_timer_configure_soc_clock(boolean_t div_factor_type, uint16_t div_factor) Configure the Ulpss SoC clock from M4 SOC clock, to enable the SOC clock source.
sl_status_t	sl_si91x_ulp_timer_configure_xtal_clock(uint8_t xtal_pin) Configure the XTAL clock when the clock source is an external XTAL clock.
sl_status_t	sl_si91x_ulp_timer_init(ulp_timer_clk_src_config_t *timer_clk_ptr) Initialize ULP-Timer clock by configuring the clock source.
void	sl_si91x_ulp_timer_deinit(void) De-Initializes ULP-Timer clock by disabling the peripheral clock.
sl_ulp_timer_version_t	sl_si91x_ulp_timer_get_version(void) Get the ULP_TIMER version.

Macros

#define	SL_TIMER_MATCH_VALUE_DEFAULT 20000000 default 1-second timer match-value for down-counter timer-type with 20Mhz clock

Enumeration Documentation

ulp_timer_instance_t

```
ulp_timer_instance_t
```

Enumeration to represent ulp-timer instances.

Enumerator	
ULP_TIMER_0	ULP Timer0 Instance.
ULP_TIMER_1	ULP Timer1 Instance.
ULP_TIMER_2	ULP Timer2 Instance.
ULP_TIMER_3	ULP Timer3 Instance.
ULP_TIMER_LAST	Last member of enum for validation.

Definition at line `65` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h`

ulp_timer_mode_t

> ulp_timer_mode_t

Enumeration to represent ULP-timer modes.

Enumerator	
ULP_TIMER_MODE_ONESHOT	ULP Timer one-shot mode.
ULP_TIMER_MODE_PERIODIC	ULP Timer periodic mode.
ULP_TIMER_MODE_LAST	Last member of enum for validation.

Definition at line `74` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h`

ulp_timer_type_t

> ulp_timer_type_t

Enumeration to represent ULP-timer types.

Enumerator	
ULP_TIMER_TYP_DEFAULT	ULP Timer normal down counter type.
ULP_TIMER_TYP_1US	ULP Timer one microsecond type.
ULP_TIMER_TYP_256US	ULP Timer 256 microsecond type.
ULP_TIMER_TYP_LAST	Last member of enum for validation.

Definition at line `81` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h`

ulp_timer_clk_input_source_t

> ulp_timer_clk_input_source_t

Enumeration to represent values of clock sources to select as Timer clock.

Enumerator	
ULP_TIMER_REF_CLK_SRC	ref clock input source
ULP_TIMER_32KHZ_RO_CLK_SRC	32 kHz ro clock input source
ULP_TIMER_32KHZ_RC_CLK_SRC	32 kHz rc clock input source
ULP_TIMER_32KHZ_XTAL_CLK_SRC	32 kHz xtal clock input source

ULP_TIMER_32MHZ_RC_CLK_SRC	32 MHz rc clock input source
ULP_TIMER_20MHZ_RO_CLK_SRC	20 MHz ro clock input source
ULP_TIMER_ULP_SOC_CLK_SRC	SoC clock input source.
ULP_TIMER_ULP_CLK_SRC_LAST	Last member of enum for validation.

Definition at line 89 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

Typedef Documentation

ulp_timer_callback_t

```
typedef void(* ulp_timer_callback_t) (void) )(void)
```

Typedef for the function pointer of the callback function.

Definition at line 59 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_clock_t

```
typedef CLK_ENABLE_T ulp_timer_clock_t
```

Renaming clock type enum.

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_direction_t

```
typedef ulp_timer_dir_t ulp_timer_direction_t
```

Renaming clock type enum.

Definition at line 62 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

Variable Documentation

ulp_timer_clk_type

```
uint8_t ulp_timer_clk_src_config_t::ulp_timer_clk_type
```

true to enable static and false to enable dynamic clock type

Definition at line 102 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_sync_to_ulpss_pclk

```
boolean_t ulp_timer_clk_src_config_t::ulp_timer_sync_to_ulpss_pclk
```

true to enable and false to disable ULP timer in synchronous mode to ULPSS pclk

Definition at line 104 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_clk_input_src

> uint8_t ulp_timer_clk_src_config_t::ulp_timer_clk_input_src

timer input clock source, refer ulp_timer_clk_input_source_t for possible values

Definition at line 106 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_skip_switch_time

> boolean_t ulp_timer_clk_src_config_t::ulp_timer_skip_switch_time

true to wait and false to Skip waiting for switching timer clk

Definition at line 107 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_num

> uint8_t ulp_timer_config_t::timer_num

timer number, SL_ULP_TIMER_NUMBER for default values

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_mode

> uint8_t ulp_timer_config_t::timer_mode

timer mode, SL_ULP_TIMER_MODE for default values

Definition at line 113 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_type

> uint8_t ulp_timer_config_t::timer_type

timer type, SL_ULP_TIMER_TYPE for default values

Definition at line 114 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_match_value

> uint32_t ulp_timer_config_t::timer_match_value

timer match value SL_ULP_TIMER_MATCH_VALUE(delay time) in microseconds

Definition at line 115 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_direction

> uint8_t ulp_timer_config_t::timer_direction

timer direction, SL_ULP_TIMER_DIRECTION for default values

Definition at line 116 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

release

> uint8_t sl_ulp_timer_version_t::release

Release version number.

Definition at line 121 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

major

> uint8_t sl_ulp_timer_version_t::major

sqa version number

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

minor

> uint8_t sl_ulp_timer_version_t::minor

dev version number

Definition at line 123 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

Function Documentation

sl_si91x_ulp_timer_configure_clock

> sl_status_t sl_si91x_ulp_timer_configure_clock (ulp_timer_clk_src_config_t *timer_clk_ptr)

Configure the ULP-Timer input clock source.

Parameters

[in]	timer_clk_ptr	Pointer to timer_clk configuration structure

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Timer clock type or timer clock source values are invalid
  - SL_STATUS_INVALID_CONFIGURATION (0x0023) - Timer clock-source configuration structure members have invalid configurations.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer

SL_STATUS_OK (0x0000) - Success, timer clock-source parameters configured properly

Definition at line 138 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_set_configuration

```
sl_status_t sl_si91x_ulp_timer_set_configuration (ulp_timer_config_t *timer_config_ptr)
```

Configure ULP-Timer parameters such as timer number, mode, type, match-value & direction.

Parameters

[in]	timer_config_ptr	Pointer to the timer_configuration structure

Also configures integral and fractional values of clock cycles per microseconds or per 256 microseconds, as per the timer-type value.

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock();

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_INVALID_INDEX (0x0027) - Timer configuration structure member 'timer_num' has an invalid value.
  - SL_STATUS_INVALID_MODE (0x0024) - Timer configuration structure member 'timer_mode' has an invalid value.
  - SL_STATUS_INVALID_TYPE (0x0026) - Timer configuration structure member 'timer_type' has an invalid value.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Timer configuration structure member 'timer_direction' has an invalid value.
  - SL_STATUS_OK (0x0000) - Success, timer parameters are configured properly

Definition at line 157 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_start

```
sl_status_t sl_si91x_ulp_timer_start (ulp_timer_instance_t timer_num)
```

Start the ULP-Timer.

Parameters

[in]	timer_num	enum for ULP-timer Number (0 to 3), ulp_timer_instance_t for possible values

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_set_configuration
  - sl_si91x_ulp_timer_register_timeout_callback

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully started the timer instance

Definition at line 172 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_stop

```
sl_status_t sl_si91x_ulp_timer_stop (ulp_timer_instance_t timer_num)
```

Stop the ULP-Timer.

Parameters

[in]	timer_num	enum for ULP-timer Number (0 to 3), ulp_timer_instance_t for possible values

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_set_configuration
  - sl_si91x_ulp_timer_start

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully stopped the timer instance

Definition at line `187` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h`

sl_si91x_ulp_timer_restart

```
sl_status_t sl_si91x_ulp_timer_restart (ulp_timer_instance_t timer_num)
```

Restart an already running ULP-Timer, means it will first stop the timer and then start it again.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_set_configuration
  - sl_si91x_ulp_timer_start

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully restarted the timer instance

Definition at line `202` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h`

sl_si91x_ulp_timer_set_type

```
sl_status_t sl_si91x_ulp_timer_set_type (ulp_timer_instance_t timer_num, ulp_timer_type_t timer_type)
```

Set the ULP-Timer type.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values
[in]	timer_type	enum for ULP-timer Type, ulp_timer_type_t for possible values

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_stop

Returns

- status 0 if successful, else error code as follows:

SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
- SL_STATUS_INVALID_TYPE (0x0026) - 'timer_type' parameter value is invalid.
- SL_STATUS_OK (0x0000) - Successfully set timer-type

Definition at line 218 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_set_direction

```
sl_status_t sl_si91x_ulp_timer_set_direction (ulp_timer_instance_t timer_num, ulp_timer_direction_t timer_direction)
```

Set the ULP-Timer direction as up-counting or down-counting.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values
[in]	timer_direction	enum for ULP-timer direction, ulp_timer_direction_t for possible values

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_stop

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_INVALID_PARAMETER (0x0021) - 'timer_direction' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully set timer-direction

Definition at line 234 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_set_mode

```
sl_status_t sl_si91x_ulp_timer_set_mode (ulp_timer_instance_t timer_num, ulp_timer_mode_t timer_mode)
```

Set ULP-Timer Mode.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values
[in]	timer_mode	enum for ULP-timer mode, ulp_timer_mode_t for possible values

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_stop

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_INVALID_MODE (0x0024) - 'timer_mode' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully set timer-mode

Definition at line 250 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_set_count

```
sl_status_t sl_si91x_ulp_timer_set_count (ulp_timer_instance_t timer_num, uint32_t timer_match_value)
```

Set ULP-Timer match value.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values.
[in]	timer_match_value	for ULP-timer timeout value

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_stop
  - sl_si91x_ulp_timer_set_type
  - sl_si91x_ulp_timer_set_direction

Returns

- status 0 if successful, else error-code
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully set timer match value.

Definition at line `267` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h`

sl_si91x_ulp_timer_get_count

```
sl_status_t sl_si91x_ulp_timer_get_count (ulp_timer_instance_t timer_num, uint32_t *count_value)
```

Read the ULP-Timer count.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values.
[in]	count_value	Pointer to variable which will store the current count of the timer

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_start
  - sl_si91x_ulp_timer_set_count

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - 'Pointer to count_value' parameter is a null pointer.
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully read the timer's current count value

Definition at line `284` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h`

sl_si91x_ulp_timer_get_type

```
sl_status_t sl_si91x_ulp_timer_get_type (ulp_timer_instance_t timer_num, uint32_t *timer_type)
```

Read the ULP-Timer type.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values.
[in]	timer_type	Pointer to variable which will store the current count of the timer

- Pre-conditions:

sl_si91x_ulp_timer_configure_clock
- sl_si91x_ulp_timer_set_type

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - 'Pointer to timer_type' parameter is a null pointer.
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully read the timer's current count value

Definition at line 300 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_get_mode

```
sl_status_t sl_si91x_ulp_timer_get_mode (ulp_timer_instance_t timer_num, uint32_t *timer_mode)
```

Read the ULP-Timer mode.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values.
[in]	timer_mode	Pointer to variable which will store the current count of the timer

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_set_mode

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - 'Pointer to timer_mode' parameter is a null pointer.
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully read the timer's current count value

Definition at line 316 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_get_direction

```
sl_status_t sl_si91x_ulp_timer_get_direction (ulp_timer_instance_t timer_num, uint32_t *timer_direction)
```

Read the ULP-Timer direction (up-counter or down-counter).

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values.
[in]	timer_direction	Pointer to variable which will store the current count of the timer

- Pre-conditions:
  - sl_si91x_ulp_timer_configure_clock
  - sl_si91x_ulp_timer_set_direction

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - 'Pointer to timer_direction' parameter is a null pointer.
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_OK (0x0000) - Successfully read the timer's current count value

Definition at line 332 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_register_timeout_callback

```
sl_status_t sl_si91x_ulp_timer_register_timeout_callback (ulp_timer_instance_t timer_num, ulp_timer_callback_t
on_timeout_callback)
```

Register callback of timer timeout interrupt and enabling it.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values.
[in]	on_timeout_callback	(function pointer) Callback function pointer to be called when the timer timeout interrupt occurred.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - 'pointer to callback_data_input' parameter is a null pointer.
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_ALLOCATION_FAILED (0x0019) - Timer interrupt enabling failed.
  - SL_STATUS_BUSY (0x0004) - The callback is already registered, unregister the previous callback before registering a new one.
  - SL_STATUS_OK (0x0000) - Successfully registered the timer timeout callback

Definition at line 348 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_unregister_timeout_callback

```
sl_status_t sl_si91x_ulp_timer_unregister_timeout_callback (ulp_timer_instance_t timer_num)
```

Unregister callback of timer timeout interrupt and disabling it.

Parameters

[in]	timer_num	enum for ULP-timer Number, ulp_timer_instance_t for possible values.

- Pre-conditions:
  - sl_si91x_ulp_timer_register_timeout_callback

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_INDEX (0x0027) - 'timer_num' parameter value is invalid.
  - SL_STATUS_ALLOCATION_FAILED (0x0019) - Timer interrupt disabling failed.
  - SL_STATUS_OK (0x0000) - Successfully unregistered the timer timeout callback

Definition at line 363 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_configure_soc_clock

```
sl_status_t sl_si91x_ulp_timer_configure_soc_clock (boolean_t div_factor_type, uint16_t div_factor)
```

Configure the Ulpss SoC clock from M4 SOC clock, to enable the SOC clock source.

Parameters

[in]	div_factor_type	value to divide the clock, ensure that it should be an odd number if div_factor_type is 1 & vice versa

[in]	div_factor	selects the type of divider for m4_soc_clk_2ulpss • 0 => Even Divider is selected • 1 => Odd Divider is selected

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER(0x0021) - 'div_factor' is not according to div_factor_type

Definition at line 376 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_configure_xtal_clock

```
sl_status_t sl_si91x_ulp_timer_configure_xtal_clock (uint8_t xtal_pin)
```

Configure the XTAL clock when the clock source is an external XTAL clock.

Parameters

[in]	xtal_pin	: Pin number of NPSS_GPIO. Possible values are 0,1,2,3,4

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER(0x0021) - 'xtal_pin' parameter value is invalid.

Definition at line 385 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_init

```
sl_status_t sl_si91x_ulp_timer_init (ulp_timer_clk_src_config_t *timer_clk_ptr)
```

Initialize ULP-Timer clock by configuring the clock source.

Parameters

[in]	timer_clk_ptr	Pointer to timer_clk configuration structure

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Timer clock type or timer clock source values are invalid
  - SL_STATUS_INVALID_CONFIGURATION (0x0023) - Timer clock-source configuration structure members have invalid configurations.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success, timer clock-source parameters configured properly

Definition at line 397 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_deinit

```
void sl_si91x_ulp_timer_deinit (void)
```

De-Initializes ULP-Timer clock by disabling the peripheral clock.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_ulp_timer_init
  - sl_si91x_ulp_timer_unregister_timeout_callback

Returns

- none

Definition at line 409 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_si91x_ulp_timer_get_version

```
sl_ulp_timer_version_t sl_si91x_ulp_timer_get_version (void)
```

Get the ULP_TIMER version.

Parameters

[in]		

This function is used to know the ULP_TIMER version

Returns

- sl_ulp_timer_version_t type version

Definition at line 421 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

Macro Definition Documentation

SL_TIMER_MATCH_VALUE_DEFAULT

```
#define SL_TIMER_MATCH_VALUE_DEFAULT
```

Value:

```
20000000
```

default 1-second timer match-value for down-counter timer-type with 20Mhz clock

Definition at line 52 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_clk_src_config_t

Structure to hold the parameters of timer clock-source configurations.

Public Attributes

uint8_t	ulp_timer_clk_type true to enable static and false to enable dynamic clock type
boolean_t	ulp_timer_sync_to_ulpss_pclk true to enable and false to disable ULP timer in synchronous mode to ULPSS pclk
uint8_t	ulp_timer_clk_input_src timer input clock source, refer ulp_timer_clk_input_source_t for possible values
boolean_t	ulp_timer_skip_switch_time true to wait and false to Skip waiting for switching timer clk

Public Attribute Documentation

ulp_timer_clk_type

```
uint8_t ulp_timer_clk_src_config_t::ulp_timer_clk_type
```

true to enable static and false to enable dynamic clock type

Definition at line 102 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_sync_to_ulpss_pclk

```
boolean_t ulp_timer_clk_src_config_t::ulp_timer_sync_to_ulpss_pclk
```

true to enable and false to disable ULP timer in synchronous mode to ULPSS pclk

Definition at line 104 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_clk_input_src

```
uint8_t ulp_timer_clk_src_config_t::ulp_timer_clk_input_src
```

timer input clock source, refer ulp_timer_clk_input_source_t for possible values

Definition at line 106 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_skip_switch_time

boolean_t ulp_timer_clk_src_config_t::ulp_timer_skip_switch_time

true to wait and false to Skip waiting for switching timer clk

Definition at line 107 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

ulp_timer_config_t

Structure to hold the parameters of timer configurations.

Public Attributes

uint8_t	timer_num timer number, SL_ULP_TIMER_NUMBER for default values
uint8_t	timer_mode timer mode, SL_ULP_TIMER_MODE for default values
uint8_t	timer_type timer type, SL_ULP_TIMER_TYPE for default values
uint32_t	timer_match_value timer match value SL_ULP_TIMER_MATCH_VALUE(delay time) in microseconds
uint8_t	timer_direction timer direction, SL_ULP_TIMER_DIRECTION for default values

Public Attribute Documentation

timer_num

```
uint8_t ulp_timer_config_t::timer_num
```

timer number, SL_ULP_TIMER_NUMBER for default values

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_mode

```
uint8_t ulp_timer_config_t::timer_mode
```

timer mode, SL_ULP_TIMER_MODE for default values

Definition at line 113 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_type

```
uint8_t ulp_timer_config_t::timer_type
```

timer type, SL_ULP_TIMER_TYPE for default values

Definition at line 114 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_match_value

uint32_t ulp_timer_config_t::timer_match_value

timer match value SL_ULP_TIMER_MATCH_VALUE(delay time) in microseconds

Definition at line 115 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

timer_direction

uint8_t ulp_timer_config_t::timer_direction

timer direction, SL_ULP_TIMER_DIRECTION for default values

Definition at line 116 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

sl_ulp_timer_version_t

Structure to hold the different versions of peripheral API.

Public Attributes

uint8_t	release Release version number.
uint8_t	major sqa version number
uint8_t	minor dev version number

Public Attribute Documentation

release

```
uint8_t sl_ulp_timer_version_t::release
```

Release version number.

Definition at line 121 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

major

```
uint8_t sl_ulp_timer_version_t::major
```

sqa version number

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

minor

```
uint8_t sl_ulp_timer_version_t::minor
```

dev version number

Definition at line 123 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_ulp_timer.h

USART

USART

Modules

sl_si91x_usart_control_config_t

sl_usart_version_t

Enumerations

enum usart_event_typedef_t {

SL_USART_EVENT_SEND_COMPLETE = ARM_USART_EVENT_SEND_COMPLETE
SL_USART_EVENT_RECEIVE_COMPLETE = ARM_USART_EVENT_RECEIVE_COMPLETE
SL_USART_EVENT_TRANSFER_COMPLETE = ARM_USART_EVENT_TRANSFER_COMPLETE
SL_USART_EVENT_TX_COMPLETE = ARM_USART_EVENT_TX_COMPLETE
SL_USART_EVENT_TX_UNDERFLOW = ARM_USART_EVENT_TX_UNDERFLOW
SL_USART_EVENT_RX_OVERFLOW = ARM_USART_EVENT_RX_OVERFLOW
SL_USART_EVENT_RX_TIMEOUT = ARM_USART_EVENT_RX_TIMEOUT
SL_USART_EVENT_RX_BREAK = ARM_USART_EVENT_RX_BREAK
SL_USART_EVENT_RX_FRAMING_ERROR = ARM_USART_EVENT_RX_FRAMING_ERROR
SL_USART_EVENT_RX_PARITY_ERROR = ARM_USART_EVENT_RX_PARITY_ERROR
SL_USART_EVENT_CTS = ARM_USART_EVENT_CTS
SL_USART_EVENT_DSR = ARM_USART_EVENT_DSR
SL_USART_EVENT_DCD = ARM_USART_EVENT_DCD
SL_USART_EVENT_RI = ARM_USART_EVENT_RI

}

USART/UART Events.

enum power_mode_typedef_t {

SL_POWER_OFF = ARM_POWER_OFF
SL_POWER_LOW = ARM_POWER_LOW
SL_POWER_FULL = ARM_POWER_FULL
SL_POWER_MODE_LAST

}

General power states.

enum usart_databits_typedef_t {

SL_USART_DATA_BITS_5 = ARM_USART_DATA_BITS_5
SL_USART_DATA_BITS_6 = ARM_USART_DATA_BITS_6
SL_USART_DATA_BITS_7 = ARM_USART_DATA_BITS_7
SL_USART_DATA_BITS_8 = ARM_USART_DATA_BITS_8
SL_USART_DATA_BITS_9 = ARM_USART_DATA_BITS_9

}

Databit selection.

enum usart_parity_typedef_t {

SL_USART_NO_PARITY = ARM_USART_PARITY_NONE
SL_USART_EVEN_PARITY = ARM_USART_PARITY_EVEN
SL_USART_ODD_PARITY = ARM_USART_PARITY_ODD

}

Parity selection.

enum usart_modem_control_typedef_t {

SL_USART_RTS_CLEAR = ARM_USART_RTS_CLEAR
SL_USART_RTS_SET = ARM_USART_RTS_SET
SL_USART_DTR_CLEAR = ARM_USART_DTR_CLEAR
SL_USART_DTR_SET = ARM_USART_DTR_SET
SL_USART_MODEM_CONTROL_LAST

}

USART Modem control selection.

enum usart_stopbit_typedef_t {

SL_USART_STOP_BITS_1 = ARM_USART_STOP_BITS_1
SL_USART_STOP_BITS_1_5 = ARM_USART_STOP_BITS_1_5
SL_USART_STOP_BITS_2 = ARM_USART_STOP_BITS_2

}

Stop bits selection, used for asynchronous operation.

enum usart_hwflowcontol_typedef_t {

SL_USART_FLOW_CONTROL_NONE = ARM_USART_FLOW_CONTROL_NONE
SL_USART_FLOW_CONTROL_CTS = ARM_USART_FLOW_CONTROL_CTS
SL_USART_FLOW_CONTROL_RTS = ARM_USART_FLOW_CONTROL_RTS
SL_USART_FLOW_CONTROL_RTS_CTS = ARM_USART_FLOW_CONTROL_RTS_CTS

}

Hardware Flow Control Selection.

enum usart_mode_typedef_t {

SL_USART_MODE_ASYNCHRONOUS = ARM_USART_MODE_ASYNCHRONOUS
SL_USART_MODE_SYNCHRONOUS_MASTER = ARM_USART_MODE_SYNCHRONOUS_MASTER
SL_USART_MODE_SYNCHRONOUS_SLAVE = ARM_USART_MODE_SYNCHRONOUS_SLAVE
SL_USART_MODE_SINGLE_WIRE = ARM_USART_MODE_SINGLE_WIRE
SL_USART_MODE_IRDA = ARM_USART_MODE_IRDA

}

USART Mode selection.

enum usart_misc_control_typedef_t {

SL_USART_MISC_CONTROL_NONE
SL_USART_SET_DEFAULT_TX_VALUE = ARM_USART_SET_DEFAULT_TX_VALUE
SL_USART_CONTROL_TX = ARM_USART_CONTROL_TX
SL_USART_CONTROL_RX = ARM_USART_CONTROL_RX
SL_USART_CONTROL_BREAK = ARM_USART_CONTROL_BREAK
SL_USART_ABORT_SEND = ARM_USART_ABORT_SEND
SL_USART_ABORT_RECEIVE = ARM_USART_ABORT_RECEIVE
SL_USART_ABORT_TRANSFER = ARM_USART_ABORT_TRANSFER

}

USART misc control selection.

Typedefs

typedef ARM_USART_SignalEvent_t sl_usart_signal_event_t

typedef ARM_USART_STATUS sl_usart_status_t

typedef ARM_POWER_STATE sl_usart_power_state_t

typedef ARM_USART_MODEM_CONTROL	sl_usart_modem_control_t
typedef ARM_USART_MODEM_STATUS	sl_usart_modem_status_t
typedef ARM_USART_CAPABILITIES	sl_usart_capabilities_t
typedef ARM_DRIVER_USART	sl_usart_driver_t
typedef USART_RESOURCES	usart_resources_t
typedef const void *	sl_usart_handle_t

Functions

sl_status_t	sl_si91x_usart_init(usart_peripheral_t usart_instance, sl_usart_handle_t *usart_handle) Initialize USART/UART interface.
sl_status_t	sl_si91x_usart_deinit(sl_usart_handle_t usart_handle) Deinit USART/UART interface.
sl_status_t	sl_si91x_usart_register_event_callback(sl_usart_signal_event_t callback_event) Register the user callback function.
void	sl_si91x_usart_unregister_event_callback(void) Un-register the user callback function.
sl_status_t	sl_si91x_usart_send_data(sl_usart_handle_t usart_handle, const void *data, uint32_t data_length) Start sending data to USART transmitter.
sl_status_t	sl_si91x_usart_receive_data(sl_usart_handle_t usart_handle, void *data, uint32_t data_length) Start receiving data from USART receiver.
sl_status_t	sl_si91x_usart_transfer_data(sl_usart_handle_t usart_handle, const void *data_out, void *data_in, uint32_t data_length) Start sending/receiving data to/from USART transmitter/receiver.
uint32_t	sl_si91x_usart_get_tx_data_count(sl_usart_handle_t usart_handle) Get the USART/UART transfer data count.
uint32_t	sl_si91x_usart_get_rx_data_count(sl_usart_handle_t usart_handle) Get the USART/UART received data count.
sl_status_t	sl_si91x_usart_set_configuration(sl_usart_handle_t usart_handle, sl_si91x_usart_control_config_t *control_configuration) Configure the different configurations of USART Interface.
sl_status_t	sli_si91x_usart_set_non_uc_configuration(sl_usart_handle_t usart_handle, sl_si91x_usart_control_config_t *control_configuration) This is an internal function used to configure the different configurations of USART Interface, this API will not pick the configurations from USART UC.

sl_usart_status_t	sl_si91x_usart_get_status(sl_usart_handle_t usart_handle) Get USART status.
sl_status_t	sl_si91x_usart_set_modem_control(sl_usart_handle_t usart_handle, sl_usart_modem_control_t control) Set USART Modem Control line state.
sl_usart_modem_status_t	sl_si91x_usart_get_modem_status(sl_usart_handle_t usart_handle) Get USART Modem Control line state.
sl_usart_version_t	sl_si91x_usart_get_version(void) Get the USART version.
sl_status_t	sl_si91x_usart_get_configurations(uint8_t usart_module, sl_si91x_usart_control_config_t *usart_config) Get the USART configurations set.

Enumeration Documentation

usart_event_typedef_t

> usart_event_typedef_t

USART/UART Events.

Enumerator	
SL_USART_EVENT_SEND_COMPLETE	Send complete event.
SL_USART_EVENT_RECEIVE_COMPLETE	Receive complete event.
SL_USART_EVENT_TRANSFER_COMPLETE	Transfer complete event.
SL_USART_EVENT_TX_COMPLETE	Tx complete event event.
SL_USART_EVENT_TX_UNDERFLOW	TX underflow event.
SL_USART_EVENT_RX_OVERFLOW	RX overflow event.
SL_USART_EVENT_RX_TIMEOUT	RX timeout event.
SL_USART_EVENT_RX_BREAK	RX break event.
SL_USART_EVENT_RX_FRAMING_ERROR	RX Framing error event.
SL_USART_EVENT_RX_PARITY_ERROR	RX parity error event.
SL_USART_EVENT_CTS	CTS event.
SL_USART_EVENT_DSR	DSR EVENT.
SL_USART_EVENT_DCD	DCD event.
SL_USART_EVENT_RI	RI Event.

Definition at line `67` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h`

power_mode_typedef_t

> power_mode_typedef_t

General power states.

Enumerator	
SL_POWER_OFF	Power Off.
SL_POWER_LOW	Power low.
SL_POWER_FULL	Power Full.
SL_POWER_MODE_LAST	

Definition at line 85 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_databits_typedef_t

> usart_databits_typedef_t

Databit selection.

Enumerator	
SL_USART_DATA_BITS_5	5 data bits
SL_USART_DATA_BITS_6	6 data bits
SL_USART_DATA_BITS_7	7 data bits
SL_USART_DATA_BITS_8	8 data bits
SL_USART_DATA_BITS_9	9 data bits

Definition at line 93 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_parity_typedef_t

> usart_parity_typedef_t

Parity selection.

Enumerator	
SL_USART_NO_PARITY	No parity.
SL_USART_EVEN_PARITY	Even parity.
SL_USART_ODD_PARITY	Odd parity.

Definition at line 102 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_modem_control_typedef_t

> usart_modem_control_typedef_t

USART Modem control selection.

Enumerator	
SL_USART_RTS_CLEAR	RTS clear.
SL_USART_RTS_SET	RTS Set.
SL_USART_DTR_CLEAR	DTR Clear.
SL_USART_DTR_SET	Activate DTR.
SL_USART_MODEM_CONTROL_LAST	

Definition at line 109 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_stopbit_typedef_t

> usart_stopbit_typedef_t

Stop bits selection, used for asynchronous operation.

Enumerator	
SL_USART_STOP_BITS_1	1 stop bits.
SL_USART_STOP_BITS_1_5	1.5 stop bits.
SL_USART_STOP_BITS_2	2 stop bits

Definition at line 118 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_hwflowcontol_typedef_t

> usart_hwflowcontol_typedef_t

Hardware Flow Control Selection.

Enumerator	
SL_USART_FLOW_CONTROL_NONE	No hardware flow control.
SL_USART_FLOW_CONTROL_CTS	CTS signal is enabled for TX flow control.
SL_USART_FLOW_CONTROL_RTS	RTS signal is enabled for RX flow control.
SL_USART_FLOW_CONTROL_RTS_CTS	CTS and RTS signals are enabled for TX and RX flow control.

Definition at line 125 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_mode_typedef_t

> usart_mode_typedef_t

USART Mode selection.

Enumerator	
SL_USART_MODE_ASYNCHRONOUS	Asynchronous mode.
SL_USART_MODE_SYNCHRONOUS_MASTER	Synchronous mode master.
SL_USART_MODE_SYNCHRONOUS_SLAVE	Synchronous mode slave.
SL_USART_MODE_SINGLE_WIRE	UART Single-wire (half-duplex); arg = Baudrate.
SL_USART_MODE_IRDA	UART IrDA; arg = Baudrate.

Definition at line 134 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_misc_control_typedef_t

> usart_misc_control_typedef_t

USART misc control selection.

Enumerator	
SL_USART_MISC_CONTROL_NONE	
SL_USART_SET_DEFAULT_TX_VALUE	Set default TX value.
SL_USART_CONTROL_TX	Set transfer line.
SL_USART_CONTROL_RX	Set receive line.
SL_USART_CONTROL_BREAK	Set USART control break.
SL_USART_ABORT_SEND	Abort send.

SL_USART_ABORT_RECEIVE	Abort receive.
SL_USART_ABORT_TRANSFER	Abort transfer.

Definition at line 143 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

Typedef Documentation

sl_usart_signal_event_t

```
typedef ARM_USART_SignalEvent_t sl_usart_signal_event_t
```

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_usart_status_t

```
typedef ARM_USART_STATUS sl_usart_status_t
```

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_usart_power_state_t

```
typedef ARM_POWER_STATE sl_usart_power_state_t
```

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_usart_modem_control_t

```
typedef ARM_USART_MODEM_CONTROL sl_usart_modem_control_t
```

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_usart_modem_status_t

```
typedef ARM_USART_MODEM_STATUS sl_usart_modem_status_t
```

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_usart_capabilities_t

```
typedef ARM_USART_CAPABILITIES sl_usart_capabilities_t
```

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_usart_driver_t

```
typedef ARM_DRIVER_USART sl_usart_driver_t
```

Definition at line 59 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_resources_t

```
typedef USART_RESOURCES usart_resources_t
```

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_usart_handle_t

```
typedef const void* sl_usart_handle_t
```

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

Function Documentation

sl_si91x_usart_init

```
sl_status_t sl_si91x_usart_init (usart_peripheral_t usart_instance, sl_usart_handle_t *usart_handle)
```

Initialize USART/UART interface.

Parameters

[in]	usart_instance	Pointer to the USART/UART driver
[in]	usart_handle	Callback function which needs to be called on data transfer

This function will configure the clocks for USART/UART module and also initialize the DMA for UART/USART if DMA is enabled for data transfers.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_FAIL (0x0001) - Fail, UART/USART initialization failed
  - SL_STATUS_OK (0x0000) - Success, UART/USART initialization done properly

Definition at line 193 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_deinit

```
sl_status_t sl_si91x_usart_deinit (sl_usart_handle_t usart_handle)
```

Deinit USART/UART interface.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver

This function will disable the clocks for USART/UART module and also deinitialize the DMA for UART/USART if DMA is enabled for data transfers.

Returns

- status 0 if successful, else error code as follows:

SL_STATUS_FAIL (0x0001) - Fail, UART/USART Deinitialization failed
- SL_STATUS_OK (0x0000) - Success, UART/USART Deinitialization done properly

Definition at line 206 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_register_event_callback

```
sl_status_t sl_si91x_usart_register_event_callback (sl_usart_signal_event_t callback_event)
```

Register the user callback function.

Parameters

[in]	callback_event	Pointer to the function which needs to be called at the time of interrupt

- Pre-conditions:
  - sl_si91x_usart_set_configuration

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_BUSY (0x0004) - Driver is busy

Definition at line 220 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_unregister_event_callback

```
void sl_si91x_usart_unregister_event_callback (void)
```

Un-register the user callback function.

Parameters

[in]		

Returns

- none

Definition at line 228 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_send_data

```
sl_status_t sl_si91x_usart_send_data (sl_usart_handle_t usart_handle, const void *data, uint32_t data_length)
```

Start sending data to USART transmitter.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver
[in]	data	Pointer to the variable which contains transfer data
[in]	data_length	Data_length to Send

If DMA mode is set, this function will configure the DMA channel and enable the DMA channel, then transfer the data pointed to it. If DMA mode is not set, it fills the data to the transfer FIFO and transfers the data.

Pre-conditions:
- sl_si91x_usart_init
- sl_si91x_usart_set_configuration

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_FAIL (0x0001) - Fail, Data transfer failed
  - SL_STATUS_BUSY (0x0004) - Busy, already data transfer is going on
  - SL_STATUS_OK (0x0000) - Success, UART/USART initialization done properly

Definition at line `249` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h`

sl_si91x_usart_receive_data

> sl_status_t sl_si91x_usart_receive_data (sl_usart_handle_t usart_handle, void *data, uint32_t data_length)

Start receiving data from USART receiver.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver
[in]	data	Pointer to the variable which will store the received data
[in]	data_length	Data_length to receive

If DMA mode is set, it configures the DMA channel, enables the DMA channel, and receives data via DMA. If DMA mode is not set, it receives the data from the FIFO.

- Pre-conditions:
  - sl_si91x_usart_init
  - sl_si91x_usart_set_configuration

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_FAIL (0x0001) - Fail, Data transfer failed
  - SL_STATUS_BUSY (0x0004) - Busy, already data transfer is going on
  - SL_STATUS_OK (0x0000) - Success, UART/USART initialization done properly

Definition at line `270` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h`

sl_si91x_usart_transfer_data

> sl_status_t sl_si91x_usart_transfer_data (sl_usart_handle_t usart_handle, const void *data_out, void *data_in, uint32_t data_length)

Start sending/receiving data to/from USART transmitter/receiver.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver
[in]	data_out	Pointer to the variable which will store the data to be transferred
[in]	data_in	Pointer to the variable which will store the received data
[in]	data_length	Data_length to receive

This function will configure the DMA channel and enable the DMA channel, DMA if DMA mode is set and transfer's the data pointed to it.

- Pre-conditions:
  - sl_si91x_usart_init
  - sl_si91x_usart_set_configuration

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_FAIL (0x0001) - Fail, Data transfer failed
  - SL_STATUS_BUSY (0x0004) - Busy, already data transfer is going on
  - SL_STATUS_OK (0x0000) - Success, UART/USART initialization done properly

Note

- This function to be used in USART mode only, i.e., synchronous mode only in asynchronous mode use sl_si91x_usart_receive_data() and sl_si91x_usart_send_data()

Definition at line 295 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_get_tx_data_count

```
uint32_t sl_si91x_usart_get_tx_data_count (sl_usart_handle_t usart_handle)
```

Get the USART/UART transfer data count.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver

This function will return the USART data transferred count.

Returns

- return the no of bytes transferred

Definition at line 308 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_get_rx_data_count

```
uint32_t sl_si91x_usart_get_rx_data_count (sl_usart_handle_t usart_handle)
```

Get the USART/UART received data count.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver

This function will return the USART/UART data received count.

Returns

- return the no of bytes received

Definition at line 318 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_set_configuration

> sl_status_t sl_si91x_usart_set_configuration (sl_usart_handle_t usart_handle, sl_si91x_usart_control_config_t *control_configuration)

Configure the different configurations of USART Interface.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver
[in]	control_configuration	pointer to the USART configurations

This function configure the USART in different configurations such as USART mode, Data Bits, Parity, stop bits, flow control and baud rate.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_BUSY (0x0004) - Busy, already data transfer is going on
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_INVALID_MODE (0x0024) - USART Invalid mode of operation
  - SL_STATUS_NOT_SUPPORTED (0x000F) - Feature not supported
  - SL_STATUS_OK (0x0000) - Success, UART/USART initialization done properly

Definition at line `335` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h`

sli_si91x_usart_set_non_uc_configuration

> sl_status_t sli_si91x_usart_set_non_uc_configuration (sl_usart_handle_t usart_handle, sl_si91x_usart_control_config_t *control_configuration)

This is an internal function used to configure the different configurations of USART Interface, this API will not pick the configurations from USART UC.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver
[in]	control_configuration	pointer to the USART configurations

This function configure the USART in different configurations such as USART mode, Data Bits, Parity, stop bits, flow control and baud rate.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_BUSY (0x0004) - Busy, already data transfer is going on
  - SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is an invalid argument
  - SL_STATUS_INVALID_MODE (0x0024) - USART Invalid mode of operation
  - SL_STATUS_NOT_SUPPORTED (0x000F) - Feature not supported
  - SL_STATUS_OK (0x0000) - Success, UART/USART initialization done properly

Definition at line `354` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h`

sl_si91x_usart_get_status

> sl_usart_status_t sl_si91x_usart_get_status (sl_usart_handle_t usart_handle)

Get USART status.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver

This function will return USART/UART transfer and receive status.

Returns

- USART line status: TX busy, RX busy, TX underflow, RX overflow, RX break, RX framing error, RX parity error

Definition at line 366 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_set_modem_control

```
sl_status_t sl_si91x_usart_set_modem_control (sl_usart_handle_t usart_handle, sl_usart_modem_control_t control)
```

Set USART Modem Control line state.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver
[in]	control	USART modem control

This function will set the USART modem control line.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NOT_SUPPORTED (0x000F) - Feature not supported
  - SL_STATUS_OK (0x0000) - Success, UART/USART initialization done properly

Definition at line 379 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_get_modem_status

```
sl_usart_modem_status_t sl_si91x_usart_get_modem_status (sl_usart_handle_t usart_handle)
```

Get USART Modem Control line state.

Parameters

[in]	usart_handle	Pointer to the USART/UART driver

This function returns USART modem control status.

Returns

- USART modem status states are active or not

Definition at line 389 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_get_version

```
sl_usart_version_t sl_si91x_usart_get_version (void)
```

Get the USART version.

Parameters

[in]		

This function is used to know the USART version.

Returns

- sl_usart_version_t type version

Definition at line 399 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_get_configurations

> sl_status_t sl_si91x_usart_get_configurations (uint8_t usart_module, sl_si91x_usart_control_config_t *usart_config)

Get the USART configurations set.

Parameters

[in]	usart_module	USART peripheral type • 0 - USART0 • 1 - UART1 • 2 - ULP_UART
[in]	usart_config	Pointer to the USART configurations structure

Get the USART configurations set in the module such as baud rate, parity bit, stop bits etc.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - Invalid NULL pointer received as an argument
  - SL_STATUS_OK (0x0000) - Success, UART/USART configurations retrieved successfully

Definition at line 416 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_si91x_usart_control_config_t

USART configuration control structure.

Public Attributes

uint32_t	baudrate baud rate
usart_mode_typedef_t	mode USART mode of operation.
usart_parity_typedef_t	parity USART parity bit.
usart_stopbit_typedef_t	stopbits USART stop bits.
usart_hwflowcontrol_typedef_t	hwflowcontrol USART h/w flow control.
usart_databits_typedef_t	databits USART databits.
usart_misc_control_typedef_t	misc_control USART MISC_CONTROL.
uint32_t	usart_module USART module.
boolean_t	config_enable USART TX and RX Config enable.
boolean_t	synch_mode Synchronous mode.

Public Attribute Documentation

baudrate

```
uint32_t sl_si91x_usart_control_config_t::baudrate
```

baud rate

Definition at line 159 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

mode

```
usart_mode_typedef_t sl_si91x_usart_control_config_t::mode
```

USART mode of operation.

Definition at line 160 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

parity

> usart_parity_typedef_t sl_si91x_usart_control_config_t::parity

USART parity bit.

Definition at line 161 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

stopbits

> usart_stopbit_typedef_t sl_si91x_usart_control_config_t::stopbits

USART stop bits.

Definition at line 162 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

hwflowcontrol

> usart_hwflowcontol_typedef_t sl_si91x_usart_control_config_t::hwflowcontrol

USART h/w flow control.

Definition at line 163 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

databits

> usart_databits_typedef_t sl_si91x_usart_control_config_t::databits

USART databits.

Definition at line 164 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

misc_control

> usart_misc_control_typedef_t sl_si91x_usart_control_config_t::misc_control

USART MISC_CONTROL.

Definition at line 165 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

usart_module

> uint32_t sl_si91x_usart_control_config_t::usart_module

USART module.

Definition at line 166 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

config_enable

> boolean_t sl_si91x_usart_control_config_t::config_enable

USART TX and RX Config enable.

Definition at line 167 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

synch_mode

> boolean_t sl_si91x_usart_control_config_t::synch_mode

Synchronous mode.

Definition at line 168 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

sl_usart_version_t

Structure to hold the different versions of peripheral API.

Public Attributes

uint8_t	release Release version number.
uint8_t	major sqa version number
uint8_t	minor dev version number

Public Attribute Documentation

release

```
uint8_t sl_usart_version_t::release
```

Release version number.

Definition at line 173 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

major

```
uint8_t sl_usart_version_t::major
```

sqa version number

Definition at line 174 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

minor

```
uint8_t sl_usart_version_t::minor
```

dev version number

Definition at line 175 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_usart.h

Watchdog Timer

Modules

watchdog_timer_clock_config_t

watchdog_timer_config_t

sl_watchdog_timer_version_t

Enumerations

enum bg_pmu_clock_t {

RO_32KHZ_CLOCK = 1
MCU_FSM__CLOCK = 2

}

Enumeration to represent bg-pmu clock sources.

enum time_delays_t {

TIME_DELAY_0
TIME_DELAY_1
TIME_DELAY_2
TIME_DELAY_3
TIME_DELAY_4
TIME_DELAY_5
TIME_DELAY_6
TIME_DELAY_7
TIME_DELAY_8
TIME_DELAY_9
TIME_DELAY_10
TIME_DELAY_11
TIME_DELAY_12
TIME_DELAY_13
TIME_DELAY_14
TIME_DELAY_15
TIME_DELAY_16
TIME_DELAY_17
TIME_DELAY_18
TIME_DELAY_19
TIME_DELAY_20
TIME_DELAY_21
TIME_DELAY_22
TIME_DELAY_23
TIME_DELAY_24
TIME_DELAY_25
TIME_DELAY_26
TIME_DELAY_27
TIME_DELAY_28
TIME_DELAY_29
TIME_DELAY_30
TIME_DELAY_31
TIME_DELAY_LAST

}

Enumeration to represent possible time delays values for WDT interrupt time and system reset time with 32 KHZ clock freq.

Typedefs

typedef void(* watchdog_timer_callback_t)(void)

Typedef for the function pointer of the callback function.

typedef AON_CLK_T low_freq_fsm_clock_t

Renaming low frequency fsm-clock type enum.

typedef FSM_CLK_T high_freq_fsm_clock_t

Renaming high frequency fsm-clock type enum.

Functions

void sl_si91x_watchdog_init_timer(void)

Initialize the Watchdog timer (Power-up WDT and enables it to run during sleep mode).

sl_status_t sl_si91x_watchdog_configure_clock(watchdog_timer_clock_config_t *timer_clk_config_ptr)

Configure timer clock sources.

sl_status_t sl_si91x_watchdog_set_configuration(watchdog_timer_config_t *timer_config_ptr)

Configure clock sources & timer parameters like interrupt time(WDT restart time), system reset time & window time (another time stamp for WDT restart, if required).

sl_status_t	sl_si91x_watchdog_register_timeout_callback(watchdog_timer_callback_t on_timeout_callback) Register Watchdog timer timeout callback and enables NVIC.
sl_status_t	sl_si91x_watchdog_set_interrupt_time(time_delays_t interrupt_time) Set Watchdog timer interrupt time(upper time-stamp for WDT restart) duration.
uint8_t	sl_si91x_watchdog_get_interrupt_time(void) Read Watchdog timer interrupt time set value (in terms of power of two)
sl_status_t	sl_si91x_watchdog_set_system_reset_time(time_delays_t system_reset_time) Set Watchdog timer system-reset time duration.
uint8_t	sl_si91x_watchdog_get_system_reset_time(void) Read Watchdog timer system-reset time set value (in terms of power of two).
sl_status_t	sl_si91x_watchdog_set_window_time(time_delays_t window_time) Set Watchdog timer window time value (lower time-stamp for WDT restart).
uint8_t	sl_si91x_watchdog_get_window_time(void) Read Watchdog timer window time set value (in terms of power of two).
boolean_t	sl_si91x_watchdog_get_timer_system_reset_status(void) Read Watchdog timer system-reset status.
void	sl_si91x_watchdog_deinit_timer(void) De-initialize Watchdog timer, mask its interrupt, de-power and disable timer.
void	sl_si91x_watchdog_unregister_timeout_callback(void) Unregister Watchdog timer timeout callback.
sl_watchdog_timer_version_t	sl_si91x_watchdog_get_version() Get the release version of Watchdog timer.
__STATIC_INLINE void	sl_si91x_watchdog_start_timer(void) Start and enable the Watchdog timer.
__STATIC_INLINE void	sl_si91x_watchdog_stop_timer(void) Stop Watchdog timer by disabling it.
__STATIC_INLINE void	sl_si91x_watchdog_restart_timer(void) Restart (kick) an already running Watchdog timer.
__STATIC_INLINE void	sl_si91x_watchdog_enable_system_reset_on_processor_lockup(void) Enable the Watchdog timer to reset the system on processor lockup.
__STATIC_INLINE void	sl_si91x_watchdog_disable_system_reset_on_processor_lockup(void) Disable the Watchdog timer to reset system on processor lockup.

Enumeration Documentation

bg_pmu_clock_t

```
bg_pmu_clock_t
```

Enumeration to represent bg-pmu clock sources.

Enumerator	
RO_32KHZ_CLOCK	RO_32KHz_clock.
MCU_FSM__CLOCK	mcu_fsm__clock

Definition at line 66 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

time_delays_t

time_delays_t

Enumeration to represent possible time delays values for WDT interrupt time and system reset time with 32 KHZ clock freq.

Enumerator	
TIME_DELAY_0	for time delay of 0.03125 milliseconds
TIME_DELAY_1	for time delay of 0.0625 milliseconds
TIME_DELAY_2	for time delay of 0.125 milliseconds
TIME_DELAY_3	for time delay of 0.25 milliseconds
TIME_DELAY_4	for time delay of 0.5 milliseconds
TIME_DELAY_5	for time delay of 1 milliseconds
TIME_DELAY_6	for time delay of 2 milliseconds
TIME_DELAY_7	for time delay of 4 milliseconds
TIME_DELAY_8	for time delay of 8 milliseconds
TIME_DELAY_9	for time delay of 16 milliseconds
TIME_DELAY_10	for time delay of 32 milliseconds
TIME_DELAY_11	for time delay of 64 milliseconds
TIME_DELAY_12	for time delay of 128 milliseconds
TIME_DELAY_13	for time delay of 256 milliseconds
TIME_DELAY_14	for time delay of 512 milliseconds
TIME_DELAY_15	for time delay of 1.024 seconds
TIME_DELAY_16	for time delay of 2.048 seconds
TIME_DELAY_17	for time delay of 4.096 seconds
TIME_DELAY_18	for time delay of 8.192 seconds
TIME_DELAY_19	for time delay of 16.384 seconds
TIME_DELAY_20	for time delay of 32.768 seconds
TIME_DELAY_21	for time delay of 65.536 seconds
TIME_DELAY_22	for time delay of 131.072 seconds
TIME_DELAY_23	for time delay of 262.144 seconds
TIME_DELAY_24	for time delay of 524.288 seconds
TIME_DELAY_25	for time delay of 1048.576 seconds
TIME_DELAY_26	for time delay of 2097.152 seconds
TIME_DELAY_27	for time delay of 4194.304 seconds
TIME_DELAY_28	for time delay of 8388.60 seconds
TIME_DELAY_29	for time delay of 16777.216 seconds
TIME_DELAY_30	for time delay of 33554.432 seconds
TIME_DELAY_31	for time delay of 67108.864 seconds
TIME_DELAY_LAST	for time delay value validation

Definition at line `72` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h`

Typedef Documentation

watchdog_timer_callback_t

```
typedef void(* watchdog_timer_callback_t) (void) )(void)
```

Typedef for the function pointer of the callback function.

Parameters

N/A	data	(void *)extra parameter for user application

Definition at line `60` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h`

low_freq_fsm_clock_t

```
typedef AON_CLK_T low_freq_fsm_clock_t
```

Renaming low frequency fsm-clock type enum.

Definition at line `62` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h`

high_freq_fsm_clock_t

```
typedef FSM_CLK_T high_freq_fsm_clock_t
```

Renaming high frequency fsm-clock type enum.

Definition at line `63` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h`

Function Documentation

sl_si91x_watchdog_init_timer

```
void sl_si91x_watchdog_init_timer (void)
```

Initialize the Watchdog timer (Power-up WDT and enables it to run during sleep mode).

Parameters

[in]		

Also un-masks its interrupt & sets RTC clock time-period.

Returns

- none

Definition at line `142` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h`

sl_si91x_watchdog_configure_clock

```
sl_status_t sl_si91x_watchdog_configure_clock (watchdog_timer_clock_config_t *timer_clk_config_ptr)
```

Configure timer clock sources.

Parameters

[in]	timer_clk_config_ptr	Pointer to timer clock configuration structure

- Pre-conditions:
  - sl_si91x_watchdog_init_timer

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Timer configuration structure members have invalid values, for members watchdog_timer_clock_config_t
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success, timer clock-source parameters configured properly

Definition at line 157 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_set_configuration

> sl_status_t sl_si91x_watchdog_set_configuration (watchdog_timer_config_t *timer_config_ptr)

Configure clock sources & timer parameters like interrupt time(WDT restart time), system reset time & window time (another time stamp for WDT restart, if required).

Parameters

[in]	timer_config_ptr	Pointer to timer clock configuration structure

Also un-masks its interrupt & sets RTC clock time-period.

- Pre-conditions:
  - sl_si91x_watchdog_init_timer
  - sl_si91x_watchdog_configure_clock

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - Timer configuration structure members have invalid values, for members watchdog_timer_config_t
  - SL_STATUS_INVALID_CONFIGURATION (0x0023) - Timer configuration structure member 'system_reset_time' is less than or equal to 'interrupt_time'. It should be greater than interrupt time of timer.
  - SL_STATUS_NULL_POINTER (0x0022) - The parameter is a null pointer
  - SL_STATUS_OK (0x0000) - Success, timer parameters configured properly

Definition at line 177 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_register_timeout_callback

> sl_status_t sl_si91x_watchdog_register_timeout_callback (watchdog_timer_callback_t on_timeout_callback)

Register Watchdog timer timeout callback and enables NVIC.

Parameters

[in]	on_timeout_callback	(function pointer) Callback function pointer to be called when timer timeout interrupt occurred.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_NULL_POINTER (0x0022) - Callback function pointer parameter is a null pointer.

SL_STATUS_BUSY (0x0004) - The callback is already registered, the user must unregister the previous callback before registering again
- SL_STATUS_OK (0x0000) - Successfully registered timer timer-out callback

Definition at line 190 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_set_interrupt_time

```
sl_status_t sl_si91x_watchdog_set_interrupt_time (time_delays_t interrupt_time)
```

Set Watchdog timer interrupt time(upper time-stamp for WDT restart) duration.

Parameters

[in]	interrupt_time	(time_delays_t), timer timeout interrupt duration, Number of clock pulses = 2^(interrupt_time), time_delays_t

- Pre-conditions:
  - sl_si91x_watchdog_init_timer
  - sl_si91x_watchdog_configure_clock
  - sl_si91x_watchdog_set_system_reset_time

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - 'interrupt_time' parameter has an invalid value.
  - SL_STATUS_INVALID_CONFIGURATION (0x0023) - 'interrupt_time' value is less than window time or greater than system reset time
  - SL_STATUS_OK (0x0000) - Successfully set watchdog timer timeout time value

Definition at line 207 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_get_interrupt_time

```
uint8_t sl_si91x_watchdog_get_interrupt_time (void)
```

Read Watchdog timer interrupt time set value (in terms of power of two)

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_set_interrupt_time

Returns

- returns interrupt time (uint8_t) value, time_delays_t

Definition at line 218 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_set_system_reset_time

```
sl_status_t sl_si91x_watchdog_set_system_reset_time (time_delays_t system_reset_time)
```

Set Watchdog timer system-reset time duration.

Parameters

[in]	system_reset_time	(time_delays_t) timer system-reset duration, Number of clock pulses = 2^(system_reset_time), time_delays_t

- Pre-conditions:
  - sl_si91x_watchdog_init_timer
  - sl_si91x_watchdog_configure_clock

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - 'system_reset_time' parameter has an invalid value.
  - SL_STATUS_INVALID_CONFIGURATION (0x0023) - 'system_reset_time' value is less than window time or interrupt time
  - SL_STATUS_OK (0x0000) - Successfully set watchdog timer system-reset time value

Definition at line 234 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_get_system_reset_time

```
uint8_t sl_si91x_watchdog_get_system_reset_time (void)
```

Read Watchdog timer system-reset time set value (in terms of power of two).

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_set_system_reset_time

Returns

- returns system-reset time (uint8_t) value, time_delays_t

Definition at line 245 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_set_window_time

```
sl_status_t sl_si91x_watchdog_set_window_time (time_delays_t window_time)
```

Set Watchdog timer window time value (lower time-stamp for WDT restart).

Parameters

[in]	window_time	(time_delays_t), timer window time, Number of clock pulses = 2^(window_time),time_delays_t

- Pre-conditions:
  - sl_si91x_watchdog_init_timer
  - sl_si91x_watchdog_configure_clock
  - sl_si91x_watchdog_set_system_reset_time
  - sl_si91x_watchdog_set_interrupt_time

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_INVALID_PARAMETER (0x0021) - 'window_time' parameter has an invalid value.
  - SL_STATUS_INVALID_CONFIGURATION (0x0023) - 'window_time' value is greater than interrupt time or system reset time
  - SL_STATUS_OK (0x0000) - Successfully set watchdog timer window time value

Definition at line 263 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_get_window_time

```
uint8_t sl_si91x_watchdog_get_window_time (void)
```

Read Watchdog timer window time set value (in terms of power of two).

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_set_window_time

Returns

- returns window time (uint8_t) value, time_delays_t

Definition at line 274 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_get_timer_system_reset_status

```
boolean_t sl_si91x_watchdog_get_timer_system_reset_status (void)
```

Read Watchdog timer system-reset status.

Parameters

[in]		

Returns

- returns true if watchdog timer resets system, else returns false

Definition at line 282 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_deinit_timer

```
void sl_si91x_watchdog_deinit_timer (void)
```

De-initialize Watchdog timer, mask its interrupt, de-power and disable timer.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_init_timer
  - sl_si91x_watchdog_unregister_timeout_callback

Returns

- none

Definition at line 294 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_unregister_timeout_callback

```
void sl_si91x_watchdog_unregister_timeout_callback (void)
```

Unregister Watchdog timer timeout callback.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_register_timeout_callback

Returns

- none

Definition at line 305 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_get_version

```
sl_watchdog_timer_version_t sl_si91x_watchdog_get_version ()
```

Get the release version of Watchdog timer.

Parameters

[in]	none	

Returns

- (sl_watchdog_version_t) type structure

Definition at line 314 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_start_timer

```
__STATIC_INLINE void sl_si91x_watchdog_start_timer (void)
```

Start and enable the Watchdog timer.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_init_timer
  - sl_si91x_watchdog_configure_clock
  - sl_si91x_watchdog_register_timeout_callback

Returns

- none

Definition at line 339 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_stop_timer

```
__STATIC_INLINE void sl_si91x_watchdog_stop_timer (void)
```

Stop Watchdog timer by disabling it.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_start_timer

Returns

- none

Definition at line 362 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_restart_timer

```
__STATIC_INLINE void sl_si91x_watchdog_restart_timer (void)
```

Restart (kick) an already running Watchdog timer.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_init_timer
  - sl_si91x_watchdog_configure_clock
  - sl_si91x_watchdog_start_timer

Returns

- none

Definition at line 389 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_enable_system_reset_on_processor_lockup

```
__STATIC_INLINE void sl_si91x_watchdog_enable_system_reset_on_processor_lockup (void)
```

Enable the Watchdog timer to reset the system on processor lockup.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_init_timer

Returns

- none

Definition at line 411 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_si91x_watchdog_disable_system_reset_on_processor_lockup

```
__STATIC_INLINE void sl_si91x_watchdog_disable_system_reset_on_processor_lockup (void)
```

Disable the Watchdog timer to reset system on processor lockup.

Parameters

[in]		

- Pre-conditions:
  - sl_si91x_watchdog_init_timer
  - sl_si91x_watchdog_enable_system_reset_on_processor_lockup

Returns

- none

Definition at line `434` of file `components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h`

watchdog_timer_clock_config_t

Structure to hold the parameters of watchdog timer clock-source configurations.

Public Attributes

uint8_t	low_freq_fsm_clock_src low frequency FSM clock source, low_freq_fsm_clock_t
uint8_t	high_freq_fsm_clock_src high frequency FSM clock source, high_freq_fsm_clock_t
uint8_t	bg_pmu_clock_source bg_pmu clock source, bg_pmu_clock_t

Public Attribute Documentation

low_freq_fsm_clock_src

```
uint8_t watchdog_timer_clock_config_t::low_freq_fsm_clock_src
```

low frequency FSM clock source, low_freq_fsm_clock_t

Definition at line 110 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

high_freq_fsm_clock_src

```
uint8_t watchdog_timer_clock_config_t::high_freq_fsm_clock_src
```

high frequency FSM clock source, high_freq_fsm_clock_t

Definition at line 111 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

bg_pmu_clock_source

```
uint8_t watchdog_timer_clock_config_t::bg_pmu_clock_source
```

bg_pmu clock source, bg_pmu_clock_t

Definition at line 112 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

watchdog_timer_config_t

Structure to hold the parameters of watchdog timer configurations.

Public Attributes

uint8_t	interrupt_time timer timeout interrupt duration, number of clock pulses = 2^(system_reset_time), time_delays_t
uint8_t	system_reset_time timer system-reset duration, number of clock pulses = 2^(system_reset_time), time_delays_t
uint8_t	window_time timer window duration, number of clock pulses = 2^(window_time), time_delays_t, but should be less than TIME_DELAY_16

Public Attribute Documentation

interrupt_time

```
uint8_t watchdog_timer_config_t::interrupt_time
```

timer timeout interrupt duration, number of clock pulses = 2^(system_reset_time), time_delays_t

Definition at line 118 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

system_reset_time

```
uint8_t watchdog_timer_config_t::system_reset_time
```

timer system-reset duration, number of clock pulses = 2^(system_reset_time), time_delays_t

Definition at line 120 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

window_time

```
uint8_t watchdog_timer_config_t::window_time
```

timer window duration, number of clock pulses = 2^(window_time), time_delays_t, but should be less than TIME_DELAY_16

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

sl_watchdog_timer_version_t

Structure to hold the versions of peripheral API.

Public Attributes

uint8_t	release Release version number.
uint8_t	major sqa version number
uint8_t	minor dev version number

Public Attribute Documentation

release

```
uint8_t sl_watchdog_timer_version_t::release
```

Release version number.

Definition at line 127 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

major

```
uint8_t sl_watchdog_timer_version_t::major
```

sqa version number

Definition at line 128 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

minor

```
uint8_t sl_watchdog_timer_version_t::minor
```

dev version number

Definition at line 129 of file components/device/silabs/si91x/mcu/drivers/unified_api/inc/sl_si91x_watchdog_timer.h

Overview

Overview

The hardware driver APIs enable using devices that are external to the SiWx917™ chipset, such as sensors, joystick and MEMLCD display.

While the drivers support external devices on the Silicon Labs kits, the drivers are board agnostic and can be used with other boards that include the same devices.

APIs

APIs

This section provides a reference to the API for drivers on the SiWx91x chipset including functions, data types, and constants.

- Button functions to control the buttons on the device.
- Joystick functions to control the joystick peripheral on the device.
- LED functions to control the LEDs on the device.
- Memory LCD functions to control the memory liquid crystal display (LCD) on the device.
- Si70XX Sensor functions to control the Si70XX relative humidity and temperature sensor on the device.

Modules

Button

Joystick

LED

Memory LCD

Si70XX Sensor

Button

Button

Sample API functions for using push-buttons.

See sl_si91x_button.c for source code.

Modules

sl_button_t

Button State Definitions

A set of numerical definitions for use with the button APIs indicating the state of a button.

#define HIGH_LEVEL_INTERRUPT 1U
Interrupt on low/pressed button state can be configured.

#define LOW_LEVEL_INTERRUPT 2U
Interrupt on high/released button state.

#define HIGH_LEVEL_AND_LOW_LEVEL_INTERRUPT 3U
Interrupt on low/pressed and high/released button state.

#define RISE_EDGE_INTERRUPT 4U
Interrupt on rising edge of the button press.

#define FALL_EDGE_INTERRUPT 8U
Interrupt on falling edge of the button press.

#define RISE_EDGE_AND_FALL_EDGE_INTERRUPT 12U
Interrupt on rising edge and falling edge of the button press.

#define BUTTON_PRESSED 1
Button state is pressed.

#define BUTTON_RELEASED 0
Button state is released.

#define BUTTON_STATE_INVALID -1
Button state is invalid.

Functions

void sl_si91x_button_init(const sl_button_t *handle)
Initializes the buttons.

int8_t sl_si91x_button_state_get(uint8_t pin)
Returns the current state (pressed or released) of a button.

int8_t sl_si91x_button_pin_state(uint8_t pin)
Returns the current state (pressed or released) of the pin associated with a button.

void sl_si91x_button_pin_isr(uint8_t pin, uint8_t state)
A callback called in interrupt context whenever a button changes its state.

void sl_si91x_button_state_toggle(uint8_t pin)
Toggles the state (pressed or released) of the pin associated with a button.

void sl_si91x_button_state_set(uint8_t pin, int8_t state)
Sets the state (pressed or released) of the pin associated with a button.

void sl_si91x_button_isr(uint8_t pin, int8_t state)
A callback that user can modify for the application.

Button State Definitions Documentation

HIGH_LEVEL_INTERRUPT

```
#define HIGH_LEVEL_INTERRUPT
```

Value:

```
1U
```

Interrupt on low/pressed button state can be configured.

Definition at line 46 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

LOW_LEVEL_INTERRUPT

```
#define LOW_LEVEL_INTERRUPT
```

Value:

```
2U
```

Interrupt on high/released button state.

Definition at line 50 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

HIGH_LEVEL_AND_LOW_LEVEL_INTERRUPT

```
#define HIGH_LEVEL_AND_LOW_LEVEL_INTERRUPT
```

Value:

```
3U
```

Interrupt on low/pressed and high/released button state.

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

RISE_EDGE_INTERRUPT

```
#define RISE_EDGE_INTERRUPT
```

Value:

```
4U
```

Interrupt on rising edge of the button press.

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

FALL_EDGE_INTERRUPT

```
#define FALL_EDGE_INTERRUPT
```

Value:

```
8U
```

Interrupt on falling edge of the button press.

Definition at line 62 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

RISE_EDGE_AND_FALL_EDGE_INTERRUPT

```
#define RISE_EDGE_AND_FALL_EDGE_INTERRUPT
```

Value:

```
12U
```

Interrupt on rising edge and falling edge of the button press.

Definition at line 66 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

BUTTON_PRESSED

```
#define BUTTON_PRESSED
```

Value:

```
1
```

Button state is pressed.

Definition at line 70 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

BUTTON_RELEASED

```
#define BUTTON_RELEASED
```

Value:

```
0
```

Button state is released.

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

BUTTON_STATE_INVALID

```
#define BUTTON_STATE_INVALID
```

Value:

```
-1
```

Button state is invalid.

Definition at line 78 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

Function Documentation

sl_si91x_button_init

```
void sl_si91x_button_init (const sl_button_t *handle)
```

Initializes the buttons.

Parameters

[in]	handle	The pointer to button structure that has the specific button information.

This function is automatically called by ::hallnit().

Definition at line 95 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

sl_si91x_button_state_get

```
int8_t sl_si91x_button_state_get (uint8_t pin)
```

Returns the current state (pressed or released) of a button.

Parameters

[in]	pin	The button pin being queried, either BUTTON0 pin or BUTTON1 pin as defined.

Note

- This function is correlated with sl_si91x_button_isr() and so returns the shadow state rather than reading the actual state of the pin.

Returns

- BUTTON_PRESSED if the button is pressed or BUTTON_RELEASED if the button is not pressed.

Definition at line 108 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

sl_si91x_button_pin_state

```
int8_t sl_si91x_button_pin_state (uint8_t pin)
```

Returns the current state (pressed or released) of the pin associated with a button.

Parameters

[in]	pin	The button pin being queried, either BUTTON0 pin or BUTTON1 pin as defined.

This reads the actual state of the pin and can be used on startup to determine the initial position of the buttons.

Returns

- BUTTON_PRESSED if the button is pressed or BUTTON_RELEASED if the button is not pressed.

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

sl_si91x_button_pin_isr

```
void sl_si91x_button_pin_isr (uint8_t pin, uint8_t state)
```

A callback called in interrupt context whenever a button changes its state.

Parameters

[in]	pin	The button pin which has changed state, either BUTTON0 pin or BUTTON1 pin as defined.
[out]	state	The new state of the button referenced by the button parameter, either BUTTON_PRESSED if the button has been pressed or BUTTON_RELEASED if the button has been released.

@appusage Must be implemented by the application. This function should contain the functionality to be executed in response to changes of state in each of the buttons, or callbacks to the appropriate functionality.

Definition at line 138 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

sl_si91x_button_state_toggle

```
void sl_si91x_button_state_toggle (uint8_t pin)
```

Toggles the state (pressed or released) of the pin associated with a button.

Parameters

[in]	pin	The button pin which has changed state, either BUTTON0 pin or BUTTON1 pin as defined.

@appusage Must be implemented by the application. This function should contain the functionality to be executed in response to changes of state in each of the buttons, or callbacks to the appropriate functionality.

Definition at line 150 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

sl_si91x_button_state_set

```
void sl_si91x_button_state_set (uint8_t pin, int8_t state)
```

Sets the state (pressed or released) of the pin associated with a button.

Parameters

[in]	pin	The button pin which has changed state, either BUTTON0 pin or BUTTON1 pin as defined.
[in]	state	The new state of the button referenced by the button parameter, either BUTTON_PRESSED if the button has been pressed or BUTTON_RELEASED if the button has been released.

@appusage Must be implemented by the application. This function should contain the functionality to be executed in response to changes of state in each of the buttons, or callbacks to the appropriate functionality.

Definition at line 166 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

sl_si91x_button_isr

```
void sl_si91x_button_isr (uint8_t pin, int8_t state)
```

A callback that user can modify for the application.

Parameters

[in]	pin	The button pin which has changed state, either BUTTON0 pin or BUTTON1 pin as defined.
[in]	state	The new state of the button referenced by the button parameter, either BUTTON_PRESSED if the button has been pressed or BUTTON_RELEASED if the button has been released.

@appusage Must be implemented by the application. This function should contain the functionality to be executed in response to changes of state in each of the buttons, or callbacks to the appropriate functionality.

Definition at line 181 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

sl_button_t

Public Attributes

uint8_t	pin
uint8_t	port
uint8_t	button_number
uint8_t	pad
uint8_t	interrupt_config

Public Attribute Documentation

pin

```
uint8_t sl_button_t::pin
```

Definition at line `83` of file `components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h`

port

```
uint8_t sl_button_t::port
```

Definition at line `84` of file `components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h`

button_number

```
uint8_t sl_button_t::button_number
```

Definition at line `85` of file `components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h`

pad

```
uint8_t sl_button_t::pad
```

Definition at line `86` of file `components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h`

interrupt_config

```
uint8_t sl_button_t::interrupt_config
```

Definition at line 87 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/button/inc/sl_si91x_button.h

Joystick

Enumerations

enum sl_joystick_position_t {

SL_JOYSTICK_NONE
SL_JOYSTICK_C
SL_JOYSTICK_N
SL_JOYSTICK_E
SL_JOYSTICK_S
SL_JOYSTICK_W

}

Enumeration for finding the position of Joystick.

enum sl_joystick_state_t {

SL_JOYSTICK_DISABLED
SL_JOYSTICK_ENABLED

}

Enumeration for Joystick state (ENABLE / DISABLE) enum.

Functions

sl_status_t sl_si91x_joystick_init(void)

sl_status_t sl_si91x_joystick_init(void) Initialize Joystick.

sl_status_t sl_si91x_joystick_get_position(sl_joystick_state_t state, sl_joystick_position_t *pos)

sl_status_t sl_si91x_joystick_get_position(sl_joystick_state_t state, sl_joystick_position_t *pos) Getting the direction of Joystick.

sl_status_t sl_si91x_joystick_start(sl_joystick_state_t state)

sl_status_t sl_si91x_joystick_start(sl_joystick_state_t state) Start/Enable the Joystick.

sl_status_t sl_si91x_joystick_stop(sl_joystick_state_t state)

sl_status_t sl_si91x_joystick_stop(sl_joystick_state_t state) Stop/Disable the Joystick.

Enumeration Documentation

sl_joystick_position_t

> sl_joystick_position_t

Enumeration for finding the position of Joystick.

Enumerator	
SL_JOYSTICK_NONE	Not pressed.
SL_JOYSTICK_C	Center.
SL_JOYSTICK_N	North.

SL_JOYSTICK_E	East.
SL_JOYSTICK_S	South.
SL_JOYSTICK_W	West.

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/sl_joystick/inc/sl_si91x_joystick.h

sl_joystick_state_t

```
sl_joystick_state_t
```

Enumeration for Joystick state (ENABLE / DISABLE) enum.

Enumerator	
SL_JOYSTICK_DISABLED	Joystick data acquisition is disabled.
SL_JOYSTICK_ENABLED	Joystick data acquisition is enabled.

Definition at line 68 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/sl_joystick/inc/sl_si91x_joystick.h

Function Documentation

sl_si91x_joystick_init

```
sl_status_t sl_si91x_joystick_init (void)
```

sl_status_t sl_si91x_joystick_init(void) Initialize Joystick.

Parameters

[in]		

Returns

- status 0 if successful, SL_STATUS_OK (0x0000) - Success

Definition at line 85 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/sl_joystick/inc/sl_si91x_joystick.h

sl_si91x_joystick_get_position

```
sl_status_t sl_si91x_joystick_get_position (sl_joystick_state_t state, sl_joystick_position_t *pos)
```

sl_status_t sl_si91x_joystick_get_position(sl_joystick_state_t state, sl_joystick_position_t *pos) Getting the direction of Joystick.

Parameters

[in]	state	: Joystick enable/disable
[in]	pos	: position of joystick.

- Pre-conditions:
  - sl_si91x_joystick_init
- sl_si91x_joystick_start

Returns

- status 0 if successful, else error code as follow:

SL_STATUS_OK (0x0000) - Success

- SL_STATUS_NOT_READY (0x0003) - Module is not ready for requested operation.

Definition at line 104 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/sl_joystick/inc/sl_si91x_joystick.h

sl_si91x_joystick_start

```
sl_status_t sl_si91x_joystick_start (sl_joystick_state_t state)
```

sl_status_t sl_si91x_joystick_start(sl_joystick_state_t state) Start/Enable the Joystick.

Parameters

[in]	state	: Joystick enable/disable.

- Pre-conditions:
  - sl_si91x_joystick_init

Returns

- status 0 if successful, else error code as follow:
  - SL_STATUS_OK (0x0000) - Success
- SL_STATUS_ABORT (0x0006) - Operation aborted.

Definition at line 121 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/sl_joystick/inc/sl_si91x_joystick.h

sl_si91x_joystick_stop

```
sl_status_t sl_si91x_joystick_stop (sl_joystick_state_t state)
```

sl_status_t sl_si91x_joystick_stop(sl_joystick_state_t state) Stop/Disable the Joystick.

Parameters

[in]	state	: Joystick enable/disable.

- Pre-conditions:
  - sl_si91x_joystick_init
- sl_si91x_joystick_start

Returns

- status 0 if successful, else error code SL_STATUS_OK (0x0000) - Success
  - SL_STATUS_BUSY (0x0004) - Module is busy.

Definition at line 140 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/sl_joystick/inc/sl_si91x_joystick.h

LED

LED

Sample API funtions for controlling LEDs.

See sl_si91x_led.c for source code.

Functions

void sl_si91x_led_init(const sl_led_t *handle)
Configures GPIOs pertaining to the control of LEDs.

void sl_si91x_led_toggle(uint8_t pin)
Atomically wraps an XOR or similar operation for a single GPIO pin attached to an LED.

void sl_si91x_led_set(uint8_t pin)
Turns on (sets) a GPIO pin connected to an LED so that the LED turns on.

void sl_si91x_led_clear(uint8_t pin)
Turns off (clears) a GPIO pin connected to an LED, which turns off the LED.

void sl_si91x_led_StackIndicateActivity(bool turnOn)
Called by the stack to indicate activity over the radio (for both transmission and reception).

Function Documentation

sl_si91x_led_init

```
void sl_si91x_led_init (const sl_led_t *handle)
```

Configures GPIOs pertaining to the control of LEDs.

Parameters

[in]	handle	The pointer to led structure that has the specific led information.

Definition at line 42 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/led/inc/sl_si91x_led.h

sl_si91x_led_toggle

```
void sl_si91x_led_toggle (uint8_t pin)
```

Atomically wraps an XOR or similar operation for a single GPIO pin attached to an LED.

Parameters

[in]	pin	LED pin for the LED to be toggled.

Definition at line 49 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/led/inc/sl_si91x_led.h

sl_si91x_led_set

> void sl_si91x_led_set (uint8_t pin)

Turns on (sets) a GPIO pin connected to an LED so that the LED turns on.

Parameters

[in]	pin	LED pin for the LED to turn on.

Definition at line `56` of file `components/device/silabs/si91x/mcu/drivers/hardware_drivers/led/inc/sl_si91x_led.h`

sl_si91x_led_clear

> void sl_si91x_led_clear (uint8_t pin)

Turns off (clears) a GPIO pin connected to an LED, which turns off the LED.

Parameters

[in]	pin	LED pin for the LED to turn off.

Definition at line `63` of file `components/device/silabs/si91x/mcu/drivers/hardware_drivers/led/inc/sl_si91x_led.h`

sl_si91x_led_StackIndicateActivity

> void sl_si91x_led_StackIndicateActivity (bool turnOn)

Called by the stack to indicate activity over the radio (for both transmission and reception).

Parameters

N/A	turnOn	See Usage.

It is called once with `turnOn` true and shortly thereafter with `turnOn` false.

Typically does something interesting, such as change the state of an LED.

Definition at line `74` of file `components/device/silabs/si91x/mcu/drivers/hardware_drivers/led/inc/sl_si91x_led.h`

Memory LCD

Memory LCD

The memory liquid crystal display (LCD) driver provides the ability to render characters or monochrome pictures onto the display of the LCD peripheral on the SiWx91x™ device.

- Refer to https://docs.silabs.com/gecko-platform/3.1/hardware-driver/api/group-memlcd for more information.

Si70XX Sensor

Enumerations

enum sl_si70xx_commands {

SL_HUMIDITY_HM = 0xE5
SL_HUMIDITY_NHM = 0xF5
SL_TEMPERATURE_HM = 0xE3
SL_TEMPERATURE_NHM = 0xF3
SL_TEMPERATURE_AH = 0xE0
SL_SI70XX_RESET = 0xFE
SL_W_RHT_U_REG = 0xE6
SL_R_RHT_U_REG = 0xE7
SL_W_HEATER_C_REG = 0×51
SL_R_HEATER_C_REG = 0×11
SL_EID_BYTEL1 = 0xFA
SL_EID_BYTEL2 = 0×0F
SL_EID_BYTE21 = 0xFC
SL_EID_BYTE22 = 0xC9
SL_FIRMWARE_REV1 = 0×84
SL_FIRMWARE_REV2 = 0xB8

}

Enum for Si70xx commands.

enum sl_si70xx_measurement_type {

SL_HUMIDITY
SL_TEMPERATURE
SL_LAST_MEASUREMENT

}

Enum for Temperature, Humidity measurement.

enum sl_si70xx_eid_type {

SL_EID_FIRST_BYTE
SL_EID_SECOND_BYTE
SL_LAST_EID

}

Enum for electronic ID.

enum sl_si70xx_registers {

SL_RH_T_USER_REG
SL_HEATER_CONTROL_REG
SL_LAST_CONTROL_REG

}

Enum for User, Heater control registers.

Typedefs

typedef enum sl_si70xx_commands sl_si70xx_commands_t

Enum for Si70xx commands.

typedef enum sl_si70xx_measurement_type	sl_si70xx_measurement_type_t Enum for Temperature, Humidity measurement.
typedef enum sl_si70xx_eid_type	sl_si70xx_eid_type_t Enum for electronic ID.
typedef enum sl_si70xx_registers	sl_si70xx_registers_t Enum for User, Heater control registers.

Functions

sl_status_t	sl_si91x_si70xx_init(sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_eid_type_t eid) Initialize the Si70xx sensor.
sl_status_t	sl_si91x_si70xx_is_present(sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_eid_type_t eid) Check whether an Si7006/13/20/21 is present on the I2C bus (or) not.
sl_status_t	sl_si91x_si70xx_measure_rh_and_temp(sl_i2c_instance_t i2c_instance, uint8_t addr, uint32_t *humid_data, int32_t *temp_data) Measure relative humidity and temperature from Si7006/13/20/21 sensor.
sl_status_t	sl_si91x_si70xx_get_firmware_revision(sl_i2c_instance_t i2c_instance, uint8_t addr, uint8_t *firmware_revision) Read Firmware Revision from Si7006/13/20/21 sensor.
sl_status_t	sl_si91x_si70xx_read_temp_from_rh(sl_i2c_instance_t i2c_instance, uint8_t addr, uint32_t *humid_data, int32_t *temp_data) Reads temperature value from previous relative humidity measurement from Si7006/13/20/21 sensor.
sl_status_t	sl_si91x_si70xx_start_no_hold_measure_rh_or_temp(sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_measurement_type_t type, uint32_t *data) Start a no hold measurement of relative humidity (or) temperature from Si7006/13/20/21 sensor.
sl_status_t	sl_si91x_si70xx_measure_humidity(sl_i2c_instance_t i2c_instance, uint8_t addr, uint32_t *humid_data) Measure relative humidity from Si7006/13/20/21 sensor.
sl_status_t	sl_si91x_si70xx_measure_temperature(sl_i2c_instance_t i2c_instance, uint8_t addr, int32_t *temp_data) Measure temperature from Si7006/13/20/21 sensor.
sl_status_t	sl_si91x_si70xx_reset(sl_i2c_instance_t i2c_instance, uint8_t addr) Initiates a si70xx software reset by the appropriate command.
sl_status_t	sl_si91x_si70xx_read_control_register(sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_registers_t reg, uint8_t *data) Reads the user register 1 and heater control register data.
sl_status_t	sl_si91x_si70xx_write_control_register(sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_registers_t reg, uint8_t value) Writes data to user register 1 and heater control register.

Macros

#define	SI7006_ADDR 0X40 I2C device address for Si7006.
#define	SI7013_ADDR 0×41 I2C device address for Si7013.
#define	SI7020_ADDR 0X40 I2C device address for Si7020.

#define SI7021_ADDR 0×40
I2C device address for Si7021.

#define I2C_BASE I2C2
I2C2 base.

#define RX_LEN 2
Read buffer length 2 bytes.

#define TX_LEN 2
Write buffer length 2 bytes.

#define RD_BUF 6
Read buffer length 6 bytes.

#define WR_BUF 1
Write buffer length 1 byte.

Enumeration Documentation

sl_si70xx_commands

> sl_si70xx_commands

Enum for Si70xx commands.

Enumerator	
SL_HUMIDITY_HM	Measure Relative Humidity, Hold Master Mode.
SL_HUMIDITY_NHM	Measure Relative Humidity, No Hold Master Mode.
SL_TEMPERATURE_HM	Measure Temperature, Hold Master Mode.
SL_TEMPERATURE_NHM	Measure Temperature, No Hold Master Mode.
SL_TEMPERATURE_AH	Read Temperature Value from Previous RH Measurement.
SL_SI70XX_RESET	Si70XX Reset.
SL_W_RHT_U_REG	Write RH/T User Register 1.
SL_R_RHT_U_REG	Read RH/T User Register 1.
SL_W_HEATER_C_REG	Write Heater Control Register.
SL_R_HEATER_C_REG	Read Heater Control Register.
SL_EID_BYTEL1	Read Electronic ID 1st Byte, first part.
SL_EID_BYTEL2	Read Electronic ID 1st Byte, second part.
SL_EID_BYTE21	Read Electronic ID 2nd Byte, first part.
SL_EID_BYTE22	Read Electronic ID 2nd Byte, second part.
SL_FIRMWARE_REV1	Read Firmware Revision, first part.
SL_FIRMWARE_REV2	Read Firmware Revision, second part.

Definition at line 64 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si70xx_measurement_type

> sl_si70xx_measurement_type

Enum for Temperature, Humidity measurement.

Enumerator	
SL_HUMIDITY	Enumerator for humidity selection.
SL_TEMPERATURE	Enumerator for temperature selection.
SL_LAST_MEASUREMENT	Last enum for validation.

Definition at line 84 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si70xx_eid_type

> sl_si70xx_eid_type

Enum for electronic ID.

Enumerator	
SL_EID_FIRST_BYTE	Enumerator for sending electronic ID first byte.
SL_EID_SECOND_BYTE	Enumerator for sending electronic ID second byte.
SL_LAST_EID	Last enum for validation.

Definition at line 91 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si70xx_registers

> sl_si70xx_registers

Enum for User, Heater control registers.

Enumerator	
SL_RH_T_USER_REG	Enumerator for write RH/T user register selection.
SL_HEATER_CONTROL_REG	Enumerator for heater control register selection.
SL_LAST_CONTROL_REG	Last enum for validation.

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

Typedef Documentation

sl_si70xx_commands_t

> typedef enum sl_si70xx_commands sl_si70xx_commands_t

Enum for Si70xx commands.

Definition at line 81 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si70xx_measurement_type_t

> typedef enum sl_si70xx_measurement_type sl_si70xx_measurement_type_t

Enum for Temperature, Humidity measurement.

Definition at line 88 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si70xx_eid_type_t

```
typedef enum sl_si70xx_eid_type sl_si70xx_eid_type_t
```

Enum for electronic ID.

Definition at line 95 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si70xx_registers_t

```
typedef enum sl_si70xx_registers sl_si70xx_registers_t
```

Enum for User, Heater control registers.

Definition at line 102 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

Function Documentation

sl_si91x_si70xx_init

```
sl_status_t sl_si91x_si70xx_init (sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_eid_type_t eid)
```

Initialize the Si70xx sensor.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.
[in]	eid	: electronic ID of type sl_si70xx_eid_type_t.

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INITIALIZATION (0x0010) - no Si70xx device present
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument

Definition at line 123 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_is_present

```
sl_status_t sl_si91x_si70xx_is_present (sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_eid_type_t eid)
```

Check whether an Si7006/13/20/21 is present on the I2C bus (or) not.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.

[in]	eid	: electronic ID of type sl_si70xx_eid_type_t. If SL_EID_FIRST_BYTE is selected, then EID 1st byte is considered. If SL_EID_SECOND_BYTE is selected, then EID 2nd byte is considered. For EID 1st byte, EID 2nd byte commands please look into datasheet. Write device ID from SNB_3 if device responds. Pass in NULL to discard. Should be 0x0D for Si7013, 0x14 for Si7020 or 0x15 for Si7021

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument

Definition at line 144 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_measure_rh_and_temp

> sl_status_t sl_si91x_si70xx_measure_rh_and_temp (sl_i2c_instance_t i2c_instance, uint8_t addr, uint32_t *humid_data, int32_t *temp_data)

Measure relative humidity and temperature from Si7006/13/20/21 sensor.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.
[out]	humid_data	: The relative humidity in percentage obtained after doing conversion as per formula mentioned in datasheet.
[out]	temp_data	: The temperature in milliCelsius.

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line 164 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_get_firmware_revision

> sl_status_t sl_si91x_si70xx_get_firmware_revision (sl_i2c_instance_t i2c_instance, uint8_t addr, uint8_t *firmware_revision)

Read Firmware Revision from Si7006/13/20/21 sensor.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.
[out]	firmware_revision	: Internal firmware revision.

- Pre-conditions:

sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line 185 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_read_temp_from_rh

> sl_status_t sl_si91x_si70xx_read_temp_from_rh (sl_i2c_instance_t i2c_instance, uint8_t addr, uint32_t *humid_data, int32_t *temp_data)

Reads temperature value from previous relative humidity measurement from Si7006/13/20/21 sensor.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.
[out]	humid_data	: The relative humidity in percent (multiplied by 1000).
[out]	temp_data	: The temperature in milliCelsius.

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line 206 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_start_no_hold_measure_rh_or_temp

> sl_status_t sl_si91x_si70xx_start_no_hold_measure_rh_or_temp (sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_measurement_type_t type, uint32_t *data)

Start a no hold measurement of relative humidity (or) temperature from Si7006/13/20/21 sensor.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.
[in]	type	: measurement value of type sl_si70xx_measurement_type_t.
[out]	data	: The data read from the sensor.

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

-

The following values are returned:
- SL_STATUS_OK on success

- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument

Definition at line 226 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_measure_humidity

> sl_status_t sl_si91x_si70xx_measure_humidity (sl_i2c_instance_t i2c_instance, uint8_t addr, uint32_t *humid_data)

Measure relative humidity from Si7006/13/20/21 sensor.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.
[out]	humid_data	: The relative humidity measurement.

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line 247 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_measure_temperature

> sl_status_t sl_si91x_si70xx_measure_temperature (sl_i2c_instance_t i2c_instance, uint8_t addr, int32_t *temp_data)

Measure temperature from Si7006/13/20/21 sensor.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.
[out]	temp_data	: The temperature measurement.

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line 265 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_reset

sl_status_t sl_si91x_si70xx_reset (sl_i2c_instance_t i2c_instance, uint8_t addr)

Initiates a si70xx software reset by the appropriate command.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument

Definition at line 277 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_read_control_register

sl_status_t sl_si91x_si70xx_read_control_register (sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_registers_t reg, uint8_t *data)

Reads the user register 1 and heater control register data.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.
[in]	reg	: Register of type sl_si70xx_registers_t.
[out]	data	: The data read from the sensor.

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument
- SL_STATUS_NULL_POINTER (0x0022) - The parameter is null pointer

Definition at line 296 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

sl_si91x_si70xx_write_control_register

sl_status_t sl_si91x_si70xx_write_control_register (sl_i2c_instance_t i2c_instance, uint8_t addr, sl_si70xx_registers_t reg, uint8_t value)

Writes data to user register 1 and heater control register.

Parameters

[in]	i2c_instance	: I2C peripheral to use.
[in]	addr	: I2C address to probe.

[in]	reg	: Register of type sl_si70xx_registers_t.
[out]	value	: The value written into the register.

- Pre-conditions:
  - sl_si91x_si70xx_reset()

Returns

- The following values are returned:
  - SL_STATUS_OK on success
- SL_STATUS_INVALID_PARAMETER (0x0021) - The parameter is invalid argument

Definition at line 315 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

Macro Definition Documentation

SI7006_ADDR

```
#define SI7006_ADDR
```

Value:

```
0X40
```

I2C device address for Si7006.

Definition at line 50 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

SI7013_ADDR

```
#define SI7013_ADDR
```

Value:

```
0x41
```

I2C device address for Si7013.

Definition at line 51 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

SI7020_ADDR

```
#define SI7020_ADDR
```

Value:

```
0X40
```

I2C device address for Si7020.

Definition at line 52 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

SI7021_ADDR

```
#define SI7021_ADDR
```

Value:

```
0x40
```

I2C device address for Si7021.

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

I2C_BASE

```
#define I2C_BASE
```

Value:

```
I2C2
```

I2C2 base.

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

RX_LEN

```
#define RX_LEN
```

Value:

```
2
```

Read buffer length 2 bytes.

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

TX_LEN

```
#define TX_LEN
```

Value:

```
2
```

Write buffer length 2 bytes.

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

RD_BUF

```
#define RD_BUF
```

Value:

```
6
```

Read buffer length 6 bytes.

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

WR_BUF

```
#define WR_BUF
```

Value:

```
1
```

Write buffer length 1 byte.

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/hardware_drivers/si70xx_sensor/inc/sl_si91x_si70xx.h

Overview

Overview

Si91x-Platform's service layer provides APIs for effective power management, input-output operations, memory management, timer etc.

The service layer APIs can be used to add the above features to any application at ease, without knowing much about underlying layers.

APIs

APIs

This section provides a reference to the APIs for services on the SiWx91x™ chipset including functions, data types, and constants.

- Power Manager functions to use power manager functionality in optimizing/handling power consumption.
- Sensor Hub functions to generate sensorhub functionality.
- Sleep Timer functions to access software timers, delays, timekeeping, and date features using a low-frequency real-time clock.
- Input/Output Stream functions to perform input/output by creating streams.
- Non-volatile Memory functions to access the non-volatile memory (NVM3) driver to maintain key-value pairs in flash memory.

Modules

Power Manager

Sensor Hub

Sleep Timer

Input/Output Stream

Non-volatile Memory

Power Manager

Modules

- sl_power_ram_retention_config_t
- sl_power_peripheral_t
- sl_power_manager_ps_transition_event_info_t
- sl_power_manager_ps_transition_event_handle_t
- sli_power_sleep_config_t

Enumerations

enum sl_power_state_t {

SL_SI91X_POWER_MANAGER_PS0 = 0
SL_SI91X_POWER_MANAGER_PS1
SL_SI91X_POWER_MANAGER_PS2
SL_SI91X_POWER_MANAGER_PS3
SL_SI91X_POWER_MANAGER_PS4
SL_SI91X_POWER_MANAGER_SLEEP
SL_SI91X_POWER_MANAGER_STANDBY
LAST_ENUM_POWER_STATE
SL_SH_PS4TOPS2
SL_SH_PS2TOPS4
SL_SH_SLEEP_WAKEUP
SL_SH_DUMMY

}

Enumeration for the power states.

enum sl_clock_scaling_t {

SL_SI91X_POWER_MANAGER_POWERSAVE
SL_SI91X_POWER_MANAGER_PERFORMANCE
LAST_ENUM_CLOCK_SCALING

}

Enumeration for clock scaling parameters.

enum sli_power_sleep_mode_t {

SLI_SI91X_POWER_MANAGER_WAKEUP_FROM_FLASH_MODE = 1
SLI_SI91X_POWER_MANAGER_WAKEUP_WITHOUT_RETENTION = 2
SLI_SI91X_POWER_MANAGER_WAKEUP_WITH_RETENTION
SLI_SI91X_POWER_MANAGER_WAKEUP_WITH_RETENTION_WITHOUT_ULPSS_RAM
SLI_SI91X_POWER_MANAGER_WAKEUP_WITHOUT_RETENTION_WITHOUT_ULPSS_RAM
LAST_ENUM_POWER_SLEEP_MODE

}

Enumeration for the sleep modes.

enum sli_power_low_freq_clock_t {

SLI_SI91X_POWER_MANAGER_DISABLE_LF_MODE = DISABLE_LF_MODE
SLI_SI91X_POWER_MANAGER_LF_32_KHZ_RC = (uint32_t)1 << (0)
SLI_SI91X_POWER_MANAGER_LF_32_KHZ_XTAL = (uint32_t)1 << (1)
SLI_SI91X_POWER_MANAGER_EXTERNAL_CAP_MODE = (uint32_t)1 << (2)
LAST_ENUM_POWER_LOW_FREQ_CLOCK

}

Enumeration for low frequency clocks.

Typedefs

typedef uint32_t sl_power_manager_ps_transition_event_t

Mask of all the event(s) to listen to.

typedef void(* sl_power_manager_ps_transition_on_event_t)(sl_power_state_t from, sl_power_state_t to)

Typedef for the user supplied callback function which is called when an power state transition occurs.

Variables

uint16_t m4ss_ram_size_kb

M4SS RAM size that needs to be restored.

uint16_t ulpss_ram_size_kb

ULPSS RAM size that needs to be restored.

boolean_t configure_ram_retention

Set or Clear RAM retention.

boolean_t configure_ram_banks

Enable will set the RAM banks using size, disable will set RAM banks using bank number.

uint32_t m4ss_ram_banks

M4SS RAM bank number that needs to be restored.

uint32_t ulpss_ram_banks

ULPSS RAM bank number that needs to be restored.

uint32_t ram_retention_mode

RAM Retention modes based on the flags sl_power_ram_retention_mode_t.

uint32_t m4ss_peripheral

Masked value of M4SS Peripherals sl_power_m4ss_peripheral_t.

uint32_t ulpss_peripheral

Masked value of ULPSS Peripherals sl_power_ulpss_peripheral_t.

uint32_t npss_peripheral

Masked value of NPSS Peripherals sl_power_npss_peripheral_t.

const sl_power_manager_ps_transition_event_t event_mask

Mask of the transitions on which the callback should be called.

const sl_power_manager_ps_transition_on_event_t on_event

Function that must be called when the event occurs.

sl_slist_node_t	node List node.
sl_power_manager_ps_transition_event_info_t *	info Handle event info.

Functions

sl_status_t	sl_si91x_power_manager_init(void) Initialize the power manager service.
sl_status_t	sl_si91x_power_manager_add_ps_requirement(sl_power_state_t state) Adds requirement on power states.
sl_status_t	sl_si91x_power_manager_remove_ps_requirement(sl_power_state_t state) Removes requirement on power states.
sl_status_t	sl_si91x_power_manager_set_clock_scaling(sl_clock_scaling_t mode) Configures the clock scaling.
sl_status_t	sl_si91x_power_manager_add_peripheral_requirement(sl_power_peripheral_t *peripheral) Adds the peripheral requirement.
sl_status_t	sl_si91x_power_manager_remove_peripheral_requirement(sl_power_peripheral_t *peripheral) Removes the peripheral requirement.
sl_status_t	sl_si91x_power_manager_subscribe_ps_transition_event(sl_power_manager_ps_transition_event_handle_t *event_handle, const sl_power_manager_ps_transition_event_info_t *event_info) Registers a callback to be called on given Power state transition(s).
sl_status_t	sl_si91x_power_manager_unsubscribe_ps_transition_event(sl_power_manager_ps_transition_event_handle_t *event_handle, const sl_power_manager_ps_transition_event_info_t *event_info) Unregisters an event callback handle on Power State transition.
sl_status_t	sl_si91x_power_manager_sleep(void) Transit to sleep mode and waits for the peripheral set as wakeup source to trigger and wakeup the m4 soc.
void	sl_si91x_power_manager_standby(void) Transit to standby state and waits for the interrupt.
sl_status_t	sl_si91x_power_manager_set_wakeup_sources(uint32_t source, boolean_t add) Configures the wakeup sources.
sl_status_t	sl_si91x_power_manager_configure_ram_retention(sl_power_ram_retention_config_t *config) Retains the RAM in low power state either by using size or RAM bank as input parameter.
sl_power_state_t	sl_si91x_power_manager_get_current_state(void) Returns the current power state.
sl_status_t	sli_si91x_power_manager_change_power_state(sl_power_state_t from, sl_power_state_t to) Updates the Power State as per the from and to parameters.
sl_status_t	sli_si91x_power_manager_set_sleep_configuration(sl_power_state_t state) Configures the parameters for sleep and transit to sleep mode.
sl_status_t	sli_power_manager_update_peripheral(sl_power_peripheral_t *peripheral, boolean_t add) Updates the peripheral power state, i.e., enables and disables the peripheral as per requirements.
boolean_t	sli_si91x_power_manager_is_valid_transition(sl_power_state_t from, sl_power_state_t to) Validates the power state transitions.

sl_status_t sli_si91x_power_configure_wakeup_resource(uint32_t source, boolean_t add)
Configures the wakeup sources.

sl_status_t sli_si91x_power_manager_set_ram_retention_configuration(sl_power_ram_retention_config_t *sram_bank)
Sets the RAM retention as well as configures the RAM banks which needs to be retained.

sl_status_t sli_si91x_power_manager_configure_clock(sl_power_state_t state, boolean_t mode)
Configures the clock as per the input, i.e.

void sli_si91x_power_manager_init_hardware(void)
Sets the initial hardware configuration.

Macros

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS4 (1 << 0)
Event transition for entering PS4 state.

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS4 (1 << 1)
Event transition for leaving PS4 state.

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS3 (1 << 2)
Event transition for entering PS3 state.

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS3 (1 << 3)
Event transition for leaving PS3 state.

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS2 (1 << 4)
Event transition for entering PS2 state.

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS2 (1 << 5)
Event transition for leaving PS2 state.

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS1 (1 << 6)
Event transition for leaving PS1 state.

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_SLEEP (1 << 7)
Event transition for leaving sleep state.

#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_STANDBY (1 << 8)
Event transition for leaving standby state.

#define SL_SI91X_POWER_MANAGER_DST_WAKEUP DST_BASED_WAKEUP
Deep Sleep Timer based wakeup source.

#define SL_SI91X_POWER_MANAGER_HOST_WAKEUP HOST_BASED_WAKEUP
Host based wakeup source.

#define SL_SI91X_POWER_MANAGER_WIRELESS_WAKEUP WIRELESS_BASED_WAKEUP
Wireless based wakeup source.

#define SL_SI91X_POWER_MANAGER_M4_PROCESSOR_WAKEUP M4_PROCS_BASED_WAKEUP
M4 Processor based wakeup source.

#define SL_SI91X_POWER_MANAGER_GPIO_WAKEUP GPIO_BASED_WAKEUP
GPIO based wakeup source.

#define SL_SI91X_POWER_MANAGER_COMPARATOR_WAKEUP COMPR_BASED_WAKEUP
Comparator based wakeup source.

#define SL_SI91X_POWER_MANAGER_SYSRTC_WAKEUP SYSRTC_BASED_WAKEUP
Sysrtc based wakeup source.

#define SL_SI91X_POWER_MANAGER_ULPSS_WAKEUP ULPSS_BASED_WAKEUP
ULP peripheral based wakeup source.

#define SL_SI91X_POWER_MANAGER_SDCSS_WAKEUP SDCSS_BASED_WAKEUP
SDC (Sensor data collector) based wakeup source.

#define SL_SI91X_POWER_MANAGER_ALARM_WAKEUP ALARM_BASED_WAKEUP
Alarm based wakeup source.

#define SL_SI91X_POWER_MANAGER_SEC_WAKEUP SEC_BASED_WAKEUP
Second based wakeup source.

#define SL_SI91X_POWER_MANAGER_MSEC_WAKEUP MSEC_BASED_WAKEUP
Milli second based wakeup source.

#define SL_SI91X_POWER_MANAGER_WDT_WAKEUP WDT_INTR_BASED_WAKEUP
Watchdog interrupt based wakeup source.

#define SL_SI91X_POWER_MANAGER_HPSRAM_RETENTION_ULP_ENABLE HPSRAM_RET_ULP_MODE_EN
High Power ULP SRAM retention enable.

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_RETENTION_ENABLE M4SS_RAM_RETENTION_MODE_EN
M4SS RAM retention enable.

#define SL_SI91X_POWER_MANAGER_M4ULP_RAM_RETENTION_ENABLE M4ULP_RAM_RETENTION_MODE_EN
M4 ULP RAM retention enable.

#define SL_SI91X_POWER_MANAGER_TA_RAM_RETENTION_ENABLE TA_RAM_RETENTION_MODE_EN
TA RAM retention enable.

#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_RETENTION_ENABLE ULPSS_RAM_RETENTION_MODE_EN
ULPSS RAM retention enable.

#define SL_SI91X_POWER_MANAGER_M4ULP_RAM16K_RETENTION_ENABLE
M4ULP_RAM16K_RETENTION_MODE_EN
M4 ULP 16K RAM retention enable.

#define SL_SI91X_POWER_MANAGER_M4SS_PG_EFUSE M4SS_PWRGATE_ULP_EFUSE_PERI
M4SS EFUSE Power Gate.

#define SL_SI91X_POWER_MANAGER_M4SS_PG_RPDMA M4SS_PWRGATE_ULP_RPDMA
M4SS RPDMA Power Gate.

#define SL_SI91X_POWER_MANAGER_M4SS_PG_SDIO_SPI M4SS_PWRGATE_ULP_SDIO_SPI
M4SS SDIO SPI Power Gate.

#define SL_SI91X_POWER_MANAGER_M4SS_PG_QSPI M4SS_PWRGATE_ULP_QSPI_ICACHE
M4SS QSPI and ICACHE Power Gate.

#define SL_SI91X_POWER_MANAGER_M4SS_PG_IID M4SS_PWRGATE_ULP_IID
M4SS IID Power Gate.

#define SL_SI91X_POWER_MANAGER_M4SS_PG_M4_DEBUG M4SS_PWRGATE_ULP_M4_DEBUG_FPU
M4SS M4 Debug Power Gate.

#define SL_SI91X_POWER_MANAGER_M4SS_PG_M4_CORE M4SS_PWRGATE_ULP_M4_CORE
M4SS M4 Core Power Gate.

#define SL_SI91X_POWER_MANAGER_M4SS_PG_EXTERNAL_ROM M4SS_PWRGATE_ULP_EXT_ROM
M4SS External ROM Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_MISC ULPSS_PWRGATE_ULP_MISC
ULP Miscellaneous Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_CAP ULPSS_PWRGATE_ULP_CAP
ULP Capacitive Touch Sensor Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_UART ULPSS_PWRGATE_ULP_UART
ULP UART Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_SSI ULPSS_PWRGATE_ULP_SSI
ULP SSI Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_I2S ULPSS_PWRGATE_ULP_I2S
ULP I2S Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_I2C ULPSS_PWRGATE_ULP_I2C
ULP I2C Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_AUX ULPSS_PWRGATE_ULP_AUX
ULP AUX Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_IR ULPSS_PWRGATE_ULP_IR
ULP IR Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_UDMA ULPSS_PWRGATE_ULP_UDMA
ULP UDMA Power Gate.

#define SL_SI91X_POWER_MANAGER_ULPSS_PG_FIM ULPSS_PWRGATE_ULP_FIM
ULP FIM Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUBFFS SLPSS_PWRGATE_ULP_MCUBFFS
NPSS MCU BFFS (Battery FF's) Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUFSM SLPSS_PWRGATE_ULP_MCUFSM
NPSS MCU FSM Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCURTC SLPSS_PWRGATE_ULP_MCURTC
NPSS MCU RTC (Real Time Clock) Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUWDT SLPSS_PWRGATE_ULP_MCUWDT
NPSS MCU WDT (Watchdog Timer) Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUPS SLPSS_PWRGATE_ULP_MCUPS
NPSS MCU Process Sensor Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUTS SLPSS_PWRGATE_ULP_MCUTS
NPSS MCU Temperature Sensor Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE1 SLPSS_PWRGATE_ULP_MCUSTORE1
NPSS MCU Storage 1 Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE2 SLPSS_PWRGATE_ULP_MCUSTORE2
NPSS MCU Storage 2 Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE3 SLPSS_PWRGATE_ULP_MCUSTORE3
NPSS MCU Storage 3 Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_TIMEPERIOD SLPSS_PWRGATE_ULP_TIMEPERIOD
NPSS Time Period Power Gate.

#define SL_SI91X_POWER_MANAGER_NPSS_PG_NWPAPB_MCU_CTRL SLPSS_PWRGATE_ULP_NWPAPB_MCU_CTRL
NPSS MCU APB Control Power Gate.

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_1 RAM_BANK_0
4 KB (Bank 1 of first 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_2 RAM_BANK_1
4 KB (Bank 2 of first 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_3 RAM_BANK_2
4 KB (Bank 3 of first 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_4 RAM_BANK_3
4 KB (Bank 4 of first 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_5 RAM_BANK_4
4 KB (Bank 5 of first 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_6 RAM_BANK_5
32 KB (Bank 6-7 of first 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_7 RAM_BANK_6
64 KB (Bank 9-11 of first 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_8 RAM_BANK_7
64 KB (Bank 12-15 of first 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_9 RAM_BANK_8
64 KB (Bank 1-4 of second 192k chunk)

#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_10 RAM_BANK_9
64 KB (Bank 1-4 of third 192k chunk)

#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_1 ULPSS_2K_BANK_0
2 KB

#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_2 ULPSS_2K_BANK_1
2 KB

#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_3 ULPSS_2K_BANK_2
2 KB

#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_4 ULPSS_2K_BANK_3
2 KB

Enumeration Documentation

sl_power_state_t

> sl_power_state_t

Enumeration for the power states.

Enumerator	
SL_SI91X_POWER_MANAGER_PS0	PS0 Power State.
SL_SI91X_POWER_MANAGER_PS1	PS1 Power State.
SL_SI91X_POWER_MANAGER_PS2	PS2 Power State.
SL_SI91X_POWER_MANAGER_PS3	PS3 Power State.
SL_SI91X_POWER_MANAGER_PS4	PS4 Power State.
SL_SI91X_POWER_MANAGER_SLEEP	Sleep.
SL_SI91X_POWER_MANAGER_STANDBY	Standby.
LAST_ENUM_POWER_STATE	Last enum for validation.
SL_SH_PS4TOPS2	

SL_SH_PS2TOPS4	
SL_SH_SLEEP_WAKEUP	
SL_SH_DUMMY	

Definition at line 166 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_clock_scaling_t

> sl_clock_scaling_t

Enumeration for clock scaling parameters.

Enumerator	
SL_SI91X_POWER_MANAGER_POWERSAVE	Minimum supported frequency in a power state.
SL_SI91X_POWER_MANAGER_PERFORMANCE	Maximum supported frequency in a power state.
LAST_ENUM_CLOCK_SCALING	Last enum for validation.

Definition at line 178 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sli_power_sleep_mode_t

> sli_power_sleep_mode_t

Enumeration for the sleep modes.

Enumerator	
SLI_SI91X_POWER_MANAGER_WAKEUP_FROM_FLASH_MODE	
SLI_SI91X_POWER_MANAGER_WAKEUP_WITHOUT_RETENTION	
SLI_SI91X_POWER_MANAGER_WAKEUP_WITH_RETENTION	
SLI_SI91X_POWER_MANAGER_WAKEUP_WITH_RETENTION_WITHOUT_ULPSS_RAM	
SLI_SI91X_POWER_MANAGER_WAKEUP_WITHOUT_RETENTION_WITHOUT_ULPSS_RAM	
LAST_ENUM_POWER_SLEEP_MODE	

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

sli_power_low_freq_clock_t

> sli_power_low_freq_clock_t

Enumeration for low frequency clocks.

Enumerator	
SLI_SI91X_POWER_MANAGER_DISABLE_LF_MODE	
SLI_SI91X_POWER_MANAGER_LF_32_KHZ_RC	
SLI_SI91X_POWER_MANAGER_LF_32_KHZ_XTAL	
SLI_SI91X_POWER_MANAGER_EXTERNAL_CAP_MODE	
LAST_ENUM_POWER_LOW_FREQ_CLOCK	

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

Typedef Documentation

sl_power_manager_ps_transition_event_t

```
typedef uint32_t sl_power_manager_ps_transition_event_t
```

Mask of all the event(s) to listen to.

Definition at line 185 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_power_manager_ps_transition_on_event_t

```
typedef void(* sl_power_manager_ps_transition_on_event_t) (sl_power_state_t from, sl_power_state_t to) )
(sl_power_state_t from, sl_power_state_t to)
```

Typedef for the user supplied callback function which is called when an power state transition occurs.

Parameters

N/A	from	Power state we are leaving.
N/A	to	Power state we are entering.

Definition at line 194 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

Variable Documentation

m4ss_ram_size_kb

```
uint16_t sl_power_ram_retention_config_t::m4ss_ram_size_kb
```

M4SS RAM size that needs to be restored.

Definition at line 148 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

ulpss_ram_size_kb

```
uint16_t sl_power_ram_retention_config_t::ulpss_ram_size_kb
```

ULPSS RAM size that needs to be restored.

Definition at line 149 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

configure_ram_retention

```
boolean_t sl_power_ram_retention_config_t::configure_ram_retention
```

Set or Clear RAM retention.

Definition at line 150 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

configure_ram_banks

boolean_t sl_power_ram_retention_config_t::configure_ram_banks

Enable will set the RAM banks using size, disable will set RAM banks using bank number.

Definition at line 152 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

m4ss_ram_banks

uint32_t sl_power_ram_retention_config_t::m4ss_ram_banks

M4SS RAM bank number that needs to be restored.

Definition at line 153 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

ulpss_ram_banks

uint32_t sl_power_ram_retention_config_t::ulpss_ram_banks

ULPSS RAM bank number that needs to be restored.

Definition at line 154 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

ram_retention_mode

uint32_t sl_power_ram_retention_config_t::ram_retention_mode

RAM Retention modes based on the flags sl_power_ram_retention_mode_t.

Definition at line 155 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

m4ss_peripheral

uint32_t sl_power_peripheral_t::m4ss_peripheral

Masked value of M4SS Peripherals sl_power_m4ss_peripheral_t.

Definition at line 160 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

ulpss_peripheral

uint32_t sl_power_peripheral_t::ulpss_peripheral

Masked value of ULPSS Peripherals sl_power_ulpss_peripheral_t.

Definition at line 161 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

npss_peripheral

uint32_t sl_power_peripheral_t::npss_peripheral

Masked value of NPSS Peripherals sl_power_npss_peripheral_t.

Definition at line 162 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

event_mask

const sl_power_manager_ps_transition_event_t sl_power_manager_ps_transition_event_info_t::event_mask

Mask of the transitions on which the callback should be called.

Definition at line 199 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

on_event

const sl_power_manager_ps_transition_on_event_t sl_power_manager_ps_transition_event_info_t::on_event

Function that must be called when the event occurs.

Definition at line 200 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

node

sl_slist_node_t sl_power_manager_ps_transition_event_handle_t::node

List node.

Definition at line 205 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

info

sl_power_manager_ps_transition_event_info_t* sl_power_manager_ps_transition_event_handle_t::info

Handle event info.

Definition at line 206 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

Function Documentation

sl_si91x_power_manager_init

sl_status_t sl_si91x_power_manager_init (void)

Initialize the power manager service.

Parameters

[in]		

Configures PS4 state with 100 MHz system clock.

- Pre-conditions:
  - none

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.
  - SL_STATUS_OK (0x0000) Success.
  - SL_STATUS_ALREADY_INITIALIZED (0x0012) Power Manager is already initialized.

Definition at line 226 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_add_ps_requirement

```
sl_status_t sl_si91x_power_manager_add_ps_requirement (sl_power_state_t state)
```

Adds requirement on power states.

Parameters

[in]	state	Power state to add requirement: sl_power_state_t • SL_POWER_MANAGER_PS4 • SL_POWER_MANAGER_PS3 • SL_POWER_MANAGER_PS2 • SL_POWER_MANAGER_PS1

Default state for power manager is PS4. If any requirements are added then power manager switches to the state if it is a valid transition. Before transition from one state to another, make sure to remove requirements of previous states if any added. If any invalid state requirement is added then it returns SL_STATUS_INVALID_PARAMETER. If power manager service is not initialized then it returns SL_STATUS_NOT_INITIALIZED, to initialize call sl_si91x_power_manager_init. To get the requirements on all power states, call sl_si91x_power_manager_get_requirement_table. To know the current power state, use sl_si91x_power_manager_get_current_state.

- Pre-conditions:
  - sl_si91x_power_manager_init

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.

    - SL_STATUS_OK (0x0000) Success.

    - SL_STATUS_NOT_INITIALIZED (0x0011) Power Manager is not initialized.

    - SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.

Definition at line 261 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_remove_ps_requirement

```
sl_status_t sl_si91x_power_manager_remove_ps_requirement (sl_power_state_t state)
```

Removes requirement on power states.

Parameters

[in]	state	Power state to remove requirement: sl_power_state_t • SL_POWER_MANAGER_PS4 • SL_POWER_MANAGER_PS3 • SL_POWER_MANAGER_PS2 • SL_POWER_MANAGER_PS1

Default state for power manager is PS4. Removing requirement will not impact on power state transitions. If the current state is PS4 and no other requirements are added, and PS4 requirement is removed then it returns SL_STATUS_INVALID_PARAMETER. If power manager service is not initialized then it returns SL_STATUS_NOT_INITIALIZED, to initialize call sl_si91x_power_manager_init. To get the requirements on all power states, call sl_si91x_power_manager_get_requirement_table. To know the current power state, use sl_si91x_power_manager_get_current_state.

- Pre-conditions:
  - sl_si91x_power_manager_init

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.
    - SL_STATUS_OK (0x0000) Success.
    - SL_STATUS_NOT_INITIALIZED (0x0011) Power Manager is not initialized.
    - SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.

Definition at line 294 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_set_clock_scaling

```
sl_status_t sl_si91x_power_manager_set_clock_scaling (sl_clock_scaling_t mode)
```

Configures the clock scaling.

Parameters

[in]	mode	(sl_clock_scaling_t type enum) the clock scaling mode sl_clock_scaling_t

PS4 and PS3 states are supported only. Possible values for clock scaling are:

- POWERSAVE and PERFORMANCE
- PS4 Performance: 180 MHz clock
- PS4 Power-save: 100 MHz clock
- PS3 Performance: 80 MHz clock
- PS3 Power-save: 32 MHz clock
- For PS2 state, 20 MHz clock is default.
- If power manager service is not initialized then it returns SL_STATUS_NOT_INITIALIZED, to initialize call sl_si91x_power_manager_init.

- Pre-conditions:
  - sl_si91x_power_manager_init

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.

    - SL_STATUS_OK (0x0000) Success.

    - SL_STATUS_NOT_INITIALIZED (0x0011) Power Manager is not initialized.

    - SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.

- SL_STATUS_INVALID_CONFIGURATION (0×0023) Invalid configuration of mode.

Definition at line 331 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_add_peripheral_requirement

```
sl_status_t sl_si91x_power_manager_add_peripheral_requirement (sl_power_peripheral_t *peripheral)
```

Adds the peripheral requirement.

Parameters

[in]	peripheral	structure for different peripherals sl_power_peripheral_t

Power on the peripherals the valid peripherals passed in the structure. Structure member possible values: sl_power_peripheral_t

- m4ss_peripheral -> Accepts masked value of m4ss peripherals.
- ulpss_peripheral -> Accepts masked value of ulpss peripherals.
- npss_peripheral -> Accepts masked value of npss peripherals. The values of enums can be combined by using 'OR' operator and then passed to the variable.

- Pre-conditions:
  - sl_si91x_power_manager_init

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.

    - SL_STATUS_OK (0×0000) Success.
    - SL_STATUS_NOT_INITIALIZED (0×0011) Power Manager is not initialized.
    - SL_STATUS_INVALID_STATE (0×0002) Not a valid transition.
    - SL_STATUS_INVALID_PARAMETER (0×0021) Invalid Parameter is passed.

Definition at line 359 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_remove_peripheral_requirement

```
sl_status_t sl_si91x_power_manager_remove_peripheral_requirement (sl_power_peripheral_t *peripheral)
```

Removes the peripheral requirement.

Parameters

[in]	peripheral	Structure for different peripherals sl_power_peripheral_t

Power off the peripherals the valid peripherals passed in the structure. Structure member possible values: sl_power_peripheral_t

- m4ss_peripheral -> Accepts masked value of m4ss peripherals.
- ulpss_peripheral -> Accepts masked value of ulpss peripherals.
- npss_peripheral -> Accepts masked value of npss peripherals. The values of enums can be combined by using 'OR' operator and then passed to the variable.

- Pre-conditions:
  - sl_si91x_power_manager_init

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.

> - SL_STATUS_OK (0×0000) Success.
>
> - SL_STATUS_NOT_INITIALIZED (0×0011) Power Manager is not initialized.
>
> - SL_STATUS_INVALID_PARAMETER (0×0021) Invalid Parameter is passed.

Definition at line 385 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_subscribe_ps_transition_event

> sl_status_t sl_si91x_power_manager_subscribe_ps_transition_event (sl_power_manager_ps_transition_event_handle_t *event_handle, const sl_power_manager_ps_transition_event_info_t *event_info)

Registers a callback to be called on given Power state transition(s).

Parameters

[in]	event_handle	Event handle (no initialization needed).
[in]	event_info	Event info structure that contains the event mask and the callback that must be called.

If power manager service is not initialized then it returns SL_STATUS_NOT_INITIALIZED, to initialize call sl_si91x_power_manager_init.

- Pre-conditions:
  - sl_si91x_power_manager_init
  -

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.
    - SL_STATUS_OK (0x0000) Success.
    - SL_STATUS_NOT_INITIALIZED (0x0011) Power Manager is not initialized.
    - SL_STATUS_NULL_POINTER (0x0022) Null Pointer is passed.

Note

- Adding and removing power state transition requirement(s) from a callback on a transition event is not supported.
- The parameters passed must be persistent, meaning that they need to survive until the callback fires.
- An ASSERT is thrown if the handle is not found.

Usage example:

```
#define PS_EVENT_MASK        ( SL_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS4 \
                             | SL_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS4  \
                             | SL_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS3 \
                             | SL_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS3  \
                             | SL_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS2 \
                             | SL_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS2  \
                             | SL_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS0 \
                             | SL_POWER_MANAGER_EVENT_TRANSITION_LEAVING_SLEEP)

sl_power_manager_ps_transition_event_handle_t handle;
sl_power_manager_ps_transition_event_info_t info = { .event_mask = PS_EVENT_MASK,
                                     .on_event   = transition_callback };

void transition_callback(sl_power_state_t from, sl_power_state_t to)
```

```
{[...]}

void main(void){
 sl_status_t status;

 status =sl_si91x_power_manager_init();// Validate the status

 status =sl_si91x_power_manager_subscribe_ps_transition_event(&handle,&info);// Validate the status}
```

Definition at line 446 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_unsubscribe_ps_transition_event

> sl_status_t sl_si91x_power_manager_unsubscribe_ps_transition_event (sl_power_manager_ps_transition_event_handle_t *event_handle, const sl_power_manager_ps_transition_event_info_t *event_info)

Unregisters an event callback handle on Power State transition.

Parameters

[in]	event_handle	Event handle which must be unregistered (must have been registered previously).
[in]	event_info	Event info structure that contains the event mask and the callback that must be called.

If power manager service is not initialized then it returns SL_STATUS_NOT_INITIALIZED, to initialize call sl_si91x_power_manager_init.

- Pre-conditions:
  - sl_si91x_power_manager_init
  - sl_si91x_power_manager_subscribe_ps_transition

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.
    - SL_STATUS_OK (0x0000) Success.
    - SL_STATUS_NOT_INITIALIZED (0x0011) Power Manager is not initialized.
    - SL_STATUS_NULL_POINTER (0x0022) Null Pointer is passed.

Note

- An ASSERT is thrown if the handle is not found.

Definition at line 476 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_sleep

> sl_status_t sl_si91x_power_manager_sleep (void)

Transit to sleep mode and waits for the peripheral set as wakeup source to trigger and wakeup the m4 soc.

Parameters

[in]		

It supports PS4, PS3 and PS2 only, it cannot enter sleep mode from any other active states. If any error is there, it returns the error code and does not transit to sleep mode.

- Pre-conditions:
  - sl_si91x_power_manager_init
  - sl_si91x_power_manager_subscribe_ps_transition
  -

sl_si91x_power_manager_configure_ram_retention
- sl_si91x_power_manager_set_wakeup_sources

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.
    - SL_STATUS_OK (0x0000) Success.
    - SL_STATUS_NOT_INITIALIZED (0x0011) Power Manager is not initialized.
    - SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.
    - SL_STATUS_INVALID_STATE (0x0002) Not a valid transition.

Definition at line 509 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_standby

> void sl_si91x_power_manager_standby (void)

Transit to standby state and waits for the interrupt.

Parameters

[in]		

Standby transition is possible from PS4, PS3, PS2 state only. Transition from sleep, PS1 or PS0 is not supported.

- Pre-conditions:
  - sl_si91x_power_manager_init
  - sl_si91x_power_manager_subscribe_ps_transition

Returns

- The following values are returned:
  - none

Definition at line 527 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_set_wakeup_sources

> sl_status_t sl_si91x_power_manager_set_wakeup_sources (uint32_t source, boolean_t add)

Configures the wakeup sources.

Parameters

[in]	source	(sl_power_wakeup_sources_t type enum) Wakeup sources sl_power_wakeup_sources_t
[in]	add	(boolean_t) true enables and false disables wakeup source

One or many wakeup sources can be configured by using 'OR' operation. The initialization of the peripheral configured as wakeup source needs to be performed by user. Power Manager only sets it as a wakeup source.

- Pre-conditions:
  - sl_si91x_power_manager_init
  - sl_si91x_power_manager_configure_ram_retention
- sl_si91x_power_manager_sleep OR
  - sl_si91x_power_manager_standby

Returns

- The following values are returned:

status SL_STATUS_OK on success, else error code.
- SL_STATUS_OK (0x0000) Success.
- SL_STATUS_NOT_INITIALIZED (0x0011) Power Manager is not initialized.
- SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.

Definition at line 557 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_configure_ram_retention

```
sl_status_t sl_si91x_power_manager_configure_ram_retention (sl_power_ram_retention_config_t *config)
```

Retains the RAM in low power state either by using size or RAM bank as input parameter.

Parameters

[in]	config	Structure for the parameters of RAM retention sl_power_ram_retention_config_t.

Structure member possible values: sl_power_ram_retention_config_t

- configure_ram_retention -> Boolean to enable RAM retention. (Enable/Disable).
- configure_ram_banks -> Boolean to switch between RAM Bank retentions. Either by size or by RAM bank number.
- (Enable -> Use RAM Bank Number).
- (Disable -> Use Size).
-
- m4ss_ram_size_kb -> Retains m4ss RAM banks according to the size.
- less than 320 KB (Enter 100 for 100 KB).
- ulpss_ram_size_kb -> Retains ulpss RAM banks according to the size.
- less than 8 KB (Enter 5 for 5 KB).
- ram_bank_number -> Retains the m4ss and ulpss RAM Bank using bank number
- ram_retention_mode -> Configures the RAM retention mode

- Pre-conditions:
  - sl_si91x_power_manager_init
  - sl_si91x_power_manager_subscribe_ps_transition
- sl_si91x_power_manager_sleep

Returns

- The following values are returned:
  - status SL_STATUS_OK on success, else error code.
    - SL_STATUS_OK (0x0000) Success.
    - SL_STATUS_NOT_INITIALIZED (0x0011) Power Manager is not initialized.
    - SL_STATUS_NULL_POINTER (0x0022) Null Pointer is passed.

Definition at line 597 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_si91x_power_manager_get_current_state

```
sl_power_state_t sl_si91x_power_manager_get_current_state (void)
```

Returns the current power state.

Parameters

[in]		

Possible return values:

- 2 : SL_POWER_MANAGER_PS2, ///< PS2 Power State

3 : SL_POWER_MANAGER_PS3, ///< PS3 Power State
- 4 : SL_POWER_MANAGER_PS4, ///< PS4 Power State

Returns

- The following values are returned:
  - sl_power_state_t enum value indicating current power state

Definition at line 614 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sli_si91x_power_manager_change_power_state

```
sl_status_t sli_si91x_power_manager_change_power_state (sl_power_state_t from, sl_power_state_t to)
```

Updates the Power State as per the from and to parameters.

Parameters

[in]	from	(sl_power_state_t type enum) Power State from which the transition takes place.
[in]	to	(sl_power_state_t type enum) Power State to which the transition takes place.

Note

- FOR INTERNAL USE ONLY.

Returns

- status SL_STATUS_OK on success, else error code as follow:
  - ▪ SL_STATUS_OK (0x0000) Success.
  - ▪ SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.

Definition at line 99 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

sli_si91x_power_manager_set_sleep_configuration

```
sl_status_t sli_si91x_power_manager_set_sleep_configuration (sl_power_state_t state)
```

Configures the parameters for sleep and transit to sleep mode.

Parameters

[in]	state	(sl_power_state_t type enum) Current Power state

Note

- FOR INTERNAL USE ONLY.

Returns

- status SL_STATUS_OK on success, else error code.
  - ▪ SL_STATUS_OK (0x0000) Success.
  - ▪ SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.

Definition at line 115 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

sli_power_manager_update_peripheral

```
sl_status_t sli_power_manager_update_peripheral (sl_power_peripheral_t *peripheral, boolean_t add)
```

Updates the peripheral power state, i.e., enables and disables the peripheral as per requirements.

Parameters

[in]	peripheral	(sl_power_peripheral_t type structure)
[in]	add	(boolean) true -> add peripheral, false -> remove peripheral.

If flag is set, the peripherals which are passed in the sl_power_peripheral_t structures is be powered on. If flag is cleared, the peripherals which are passed in the sl_power_peripheral_t structures is be powered off.

Note

- FOR INTERNAL USE ONLY.

Returns

- status SL_STATUS_OK on success, else error code.
  - SL_STATUS_OK (0x0000) Success.
  - SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.
  - SL_STATUS_NULL_POINTER (0x0022) Null Pointer is passed.

Definition at line 139 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

sli_si91x_power_manager_is_valid_transition

```
boolean_t sli_si91x_power_manager_is_valid_transition (sl_power_state_t from, sl_power_state_t to)
```

Validates the power state transitions.

Parameters

[in]	from	(sl_power_state_t type enum) Power State from which the transition takes place.
[in]	to	(sl_power_state_t type enum) Power State to which the transition takes place.

If invalid, returns false. **Note**

- FOR INTERNAL USE ONLY.

Returns

- boolean value true or false.

Definition at line 150 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

sli_si91x_power_configure_wakeup_resource

```
sl_status_t sli_si91x_power_configure_wakeup_resource (uint32_t source, boolean_t add)
```

Configures the wakeup sources.

Parameters

[in]	source	(uint32_t) Ored value of wakeup sources.
[in]	add	(boolean) true -> add peripheral, false -> remove peripheral.

If add is set, the wakeup source which are passed in the source variable is set. If add is clear, the wakeup source which are passed in the source variable is clear.

Note

FOR INTERNAL USE ONLY.

Returns

- status SL_STATUS_OK on success, else error code.
  - SL_STATUS_OK (0x0000) Success.
  - SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.

Definition at line 169 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

sli_si91x_power_manager_set_ram_retention_configuration

```
sl_status_t sli_si91x_power_manager_set_ram_retention_configuration (sl_power_ram_retention_config_t *sram_bank)
```

Sets the RAM retention as well as configures the RAM banks which needs to be retained.

Parameters

[in]	sram_bank	(sl_power_ram_retention_config_t type struct)

Note

- FOR INTERNAL USE ONLY.

Returns

- status SL_STATUS_OK on success, else error code.
  - SL_STATUS_OK (0x0000) Success.
  - SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.
  - SL_STATUS_NULL_POINTER (0x0022) Null Pointer is passed.

Definition at line 186 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

sli_si91x_power_manager_configure_clock

```
sl_status_t sli_si91x_power_manager_configure_clock (sl_power_state_t state, boolean_t mode)
```

Configures the clock as per the input, i.e.

Parameters

[in]	state	(sl_power_state_t type enum) current power state.
[in]	mode	(boolean) true -> performance mode, false -> powersave mode.

powersave or performance.

Note

- FOR INTERNAL USE ONLY.

Returns

- status SL_STATUS_OK on success, else error code.
  - SL_STATUS_OK (0x0000) Success.
  - SL_STATUS_INVALID_PARAMETER (0x0021) Invalid Parameter is passed.

Definition at line 203 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

sli_si91x_power_manager_init_hardware

```
void sli_si91x_power_manager_init_hardware (void)
```

Sets the initial hardware configuration.

Parameters

[in]		

Note

- FOR INTERNAL USE ONLY.

Returns

- none

Definition at line 214 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

Macro Definition Documentation

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS4

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS4
```

Value:

```
(1 << 0)
```

Event transition for entering PS4 state.

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS4

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS4
```

Value:

```
(1 << 1)
```

Event transition for leaving PS4 state.

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS3

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS3
```

Value:

```
(1 << 2)
```

Event transition for entering PS3 state.

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS3

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS3
```

Value:

```
(1 << 3)
```

Event transition for leaving PS3 state.

Definition at line 59 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS2

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_ENTERING_PS2
```

Value:

```
(1 << 4)
```

Event transition for entering PS2 state.

Definition at line 60 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS2

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS2
```

Value:

```
(1 << 5)
```

Event transition for leaving PS2 state.

Definition at line 61 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS1

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_PS1
```

Value:

```
(1 << 6)
```

Event transition for leaving PS1 state.

Definition at line 62 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_SLEEP

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_SLEEP
```

Value:

```
(1 << 7)
```

Event transition for leaving sleep state.

Definition at line 63 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_STANDBY

```
#define SL_SI91X_POWER_MANAGER_EVENT_TRANSITION_LEAVING_STANDBY
```

Value:

```
(1 << 8)
```

Event transition for leaving standby state.

Definition at line 64 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_DST_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_DST_WAKEUP
```

Value:

```
DST_BASED_WAKEUP
```

Deep Sleep Timer based wakeup source.

Definition at line 67 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_HOST_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_HOST_WAKEUP
```

Value:

```
HOST_BASED_WAKEUP
```

Host based wakeup source.

Definition at line 68 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_WIRELESS_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_WIRELESS_WAKEUP
```

Value:

```
WIRELESS_BASED_WAKEUP
```

Wireless based wakeup source.

Definition at line 69 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4_PROCESSOR_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_M4_PROCESSOR_WAKEUP
```

Value:

```
M4_PROCS_BASED_WAKEUP
```

M4 Processor based wakeup source.

Definition at line 70 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_GPIO_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_GPIO_WAKEUP
```

Value:

```
GPIO_BASED_WAKEUP
```

GPIO based wakeup source.

Definition at line 71 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_COMPARATOR_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_COMPARATOR_WAKEUP
```

Value:

```
COMPR_BASED_WAKEUP
```

Comparator based wakeup source.

Definition at line 72 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_SYSRTC_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_SYSRTC_WAKEUP
```

Value:

```
SYSRTC_BASED_WAKEUP
```

Sysrtc based wakeup source.

Definition at line 73 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_ULPSS_WAKEUP
```

Value:

```
ULPSS_BASED_WAKEUP
```

ULP peripheral based wakeup source.

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_SDCSS_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_SDCSS_WAKEUP
```

Value:

```
SDCSS_BASED_WAKEUP
```

SDC (Sensor data collector) based wakeup source.

Definition at line 75 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ALARM_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_ALARM_WAKEUP
```

Value:

```
ALARM_BASED_WAKEUP
```

Alarm based wakeup source.

Definition at line 76 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_SEC_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_SEC_WAKEUP
```

Value:

```
SEC_BASED_WAKEUP
```

Second based wakeup source.

Definition at line 77 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_MSEC_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_MSEC_WAKEUP
```

Value:

```
MSEC_BASED_WAKEUP
```

Milli second based wakeup source.

Definition at line 78 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_WDT_WAKEUP

```
#define SL_SI91X_POWER_MANAGER_WDT_WAKEUP
```

Value:

```
WDT_INTR_BASED_WAKEUP
```

Watchdog interrupt based wakeup source.

Definition at line 79 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_HPSRAM_RETENTION_ULP_ENABLE

```
#define SL_SI91X_POWER_MANAGER_HPSRAM_RETENTION_ULP_ENABLE
```

Value:

```
HPSRAM_RET_ULP_MODE_EN
```

High Power ULP SRAM retention enable.

Definition at line 82 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_RETENTION_ENABLE

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_RETENTION_ENABLE
```

Value:

```
M4SS_RAM_RETENTION_MODE_EN
```

M4SS RAM retention enable.

Definition at line 84 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4ULP_RAM_RETENTION_ENABLE

```
#define SL_SI91X_POWER_MANAGER_M4ULP_RAM_RETENTION_ENABLE
```

Value:

```
M4ULP_RAM_RETENTION_MODE_EN
```

M4 ULP RAM retention enable.

Definition at line 85 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_TA_RAM_RETENTION_ENABLE

```
#define SL_SI91X_POWER_MANAGER_TA_RAM_RETENTION_ENABLE
```

Value:

```
TA_RAM_RETENTION_MODE_EN
```

TA RAM retention enable.

Definition at line 86 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_RAM_RETENTION_ENABLE

```
#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_RETENTION_ENABLE
```

Value:

```
ULPSS_RAM_RETENTION_MODE_EN
```

ULPSS RAM retention enable.

Definition at line 87 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4ULP_RAM16K_RETENTION_ENABLE

```
#define SL_SI91X_POWER_MANAGER_M4ULP_RAM16K_RETENTION_ENABLE
```

Value:

```
M4ULP_RAM16K_RETENTION_MODE_EN
```

M4 ULP 16K RAM retention enable.

Definition at line 88 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_PG_EFUSE

```
#define SL_SI91X_POWER_MANAGER_M4SS_PG_EFUSE
```

Value:

```
M4SS_PWRGATE_ULP_EFUSE_PERI
```

M4SS EFUSE Power Gate.

Definition at line 92 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_PG_RPDMA

```
#define SL_SI91X_POWER_MANAGER_M4SS_PG_RPDMA
```

Value:

```
M4SS_PWRGATE_ULP_RPDMA
```

M4SS RPDMA Power Gate.

Definition at line 93 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_PG_SDIO_SPI

```
#define SL_SI91X_POWER_MANAGER_M4SS_PG_SDIO_SPI
```

Value:

```
M4SS_PWRGATE_ULP_SDIO_SPI
```

M4SS SDIO SPI Power Gate.

Definition at line 94 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_PG_QSPI

```
#define SL_SI91X_POWER_MANAGER_M4SS_PG_QSPI
```

Value:

```
M4SS_PWRGATE_ULP_QSPI_ICACHE
```

M4SS QSPI and ICACHE Power Gate.

Definition at line 95 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_PG_IID

```
#define SL_SI91X_POWER_MANAGER_M4SS_PG_IID
```

Value:

```
M4SS_PWRGATE_ULP_IID
```

M4SS IID Power Gate.

Definition at line 96 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_PG_M4_DEBUG

```
#define SL_SI91X_POWER_MANAGER_M4SS_PG_M4_DEBUG
```

Value:

```
M4SS_PWRGATE_ULP_M4_DEBUG_FPU
```

M4SS M4 Debug Power Gate.

Definition at line 97 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_PG_M4_CORE

```
#define SL_SI91X_POWER_MANAGER_M4SS_PG_M4_CORE
```

Value:

```
M4SS_PWRGATE_ULP_M4_CORE
```

M4SS M4 Core Power Gate.

Definition at line 98 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_PG_EXTERNAL_ROM

```
#define SL_SI91X_POWER_MANAGER_M4SS_PG_EXTERNAL_ROM
```

Value:

```
M4SS_PWRGATE_ULP_EXT_ROM
```

M4SS External ROM Power Gate.

Definition at line 99 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_MISC

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_MISC
```

Value:

```
ULPSS_PWRGATE_ULP_MISC
```

ULP Miscellaneous Power Gate.

Definition at line 102 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_CAP

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_CAP
```

Value:

```
ULPSS_PWRGATE_ULP_CAP
```

ULP Capacitive Touch Sensor Power Gate.

Definition at line 103 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_UART

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_UART
```

Value:

```
ULPSS_PWRGATE_ULP_UART
```

ULP UART Power Gate.

Definition at line 104 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_SSI

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_SSI
```

Value:

```
ULPSS_PWRGATE_ULP_SSI
```

ULP SSI Power Gate.

Definition at line 105 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_I2S

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_I2S
```

Value:

```
ULPSS_PWRGATE_ULP_I2S
```

ULP I2S Power Gate.

Definition at line 106 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_I2C

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_I2C
```

Value:

```
ULPSS_PWRGATE_ULP_I2C
```

ULP I2C Power Gate.

Definition at line 107 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_AUX

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_AUX
```

Value:

```
ULPSS_PWRGATE_ULP_AUX
```

ULP AUX Power Gate.

Definition at line 108 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_IR

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_IR
```

Value:

```
ULPSS_PWRGATE_ULP_IR
```

ULP IR Power Gate.

Definition at line 109 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_UDMA

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_UDMA
```

Value:

```
ULPSS_PWRGATE_ULP_UDMA
```

ULP UDMA Power Gate.

Definition at line 110 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_PG_FIM

```
#define SL_SI91X_POWER_MANAGER_ULPSS_PG_FIM
```

Value:

```
ULPSS_PWRGATE_ULP_FIM
```

ULP FIM Power Gate.

Definition at line 111 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCUBFFS

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUBFFS
```

Value:

```
SLPSS_PWRGATE_ULP_MCUBFFS
```

NPSS MCU BFFS (Battery FF's) Power Gate.

Definition at line 114 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCUFSM

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUFSM
```

Value:

```
SLPSS_PWRGATE_ULP_MCUFSM
```

NPSS MCU FSM Power Gate.

Definition at line 115 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCURTC

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCURTC
```

Value:

```
SLPSS_PWRGATE_ULP_MCURTC
```

NPSS MCU RTC (Real Time Clock) Power Gate.

Definition at line 116 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCUWDT

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUWDT
```

Value:

```
SLPSS_PWRGATE_ULP_MCUWDT
```

NPSS MCU WDT (Watchdog Timer) Power Gate.

Definition at line 117 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCUPS

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUPS
```

Value:

```
SLPSS_PWRGATE_ULP_MCUPS
```

NPSS MCU Process Sensor Power Gate.

Definition at line 118 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCUTS

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUTS
```

Value:

```
SLPSS_PWRGATE_ULP_MCUTS
```

NPSS MCU Temperature Sensor Power Gate.

Definition at line 119 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE1

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE1
```

Value:

```
SLPSS_PWRGATE_ULP_MCUSTORE1
```

NPSS MCU Storage 1 Power Gate.

Definition at line 120 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE2

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE2
```

Value:

```
SLPSS_PWRGATE_ULP_MCUSTORE2
```

NPSS MCU Storage 2 Power Gate.

Definition at line 121 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE3

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_MCUSTORE3
```

Value:

```
SLPSS_PWRGATE_ULP_MCUSTORE3
```

NPSS MCU Storage 3 Power Gate.

Definition at line 122 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_TIMEPERIOD

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_TIMEPERIOD
```

Value:

```
SLPSS_PWRGATE_ULP_TIMEPERIOD
```

NPSS Time Period Power Gate.

Definition at line 123 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_NPSS_PG_NWPAPB_MCU_CTRL

```
#define SL_SI91X_POWER_MANAGER_NPSS_PG_NWPAPB_MCU_CTRL
```

Value:

```
SLPSS_PWRGATE_ULP_NWPAPB_MCU_CTRL
```

NPSS MCU APB Control Power Gate.

Definition at line 124 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_1

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_1
```

Value:

```
RAM_BANK_0
```

4 KB (Bank 1 of first 192k chunk)

Definition at line 128 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_2

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_2
```

Value:

```
RAM_BANK_1
```

4 KB (Bank 2 of first 192k chunk)

Definition at line 129 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_3

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_3
```

Value:

```
RAM_BANK_2
```

4 KB (Bank 3 of first 192k chunk)

Definition at line 130 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_4

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_4
```

Value:

```
RAM_BANK_3
```

4 KB (Bank 4 of first 192k chunk)

Definition at line 131 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_5

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_5
```

Value:

```
RAM_BANK_4
```

4 KB (Bank 5 of first 192k chunk)

Definition at line 132 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_6

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_6
```

Value:

```
RAM_BANK_5
```

32 KB (Bank 6-7 of first 192k chunk)

Definition at line 133 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_7

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_7
```

Value:

```
RAM_BANK_6
```

64 KB (Bank 9-11 of first 192k chunk)

Definition at line 134 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_8

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_8
```

Value:

```
RAM_BANK_7
```

64 KB (Bank 12-15 of first 192k chunk)

Definition at line 135 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_9

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_9
```

Value:

```
RAM_BANK_8
```

64 KB (Bank 1-4 of second 192k chunk)

Definition at line 136 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_10

```
#define SL_SI91X_POWER_MANAGER_M4SS_RAM_BANK_10
```

Value:

```
RAM_BANK_9
```

64 KB (Bank 1-4 of third 192k chunk)

Definition at line 137 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_1

```
#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_1
```

Value:

```
ULPSS_2K_BANK_0
```

2 KB

Definition at line 138 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_2

```
#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_2
```

Value:

```
ULPSS_2K_BANK_1
```

2 KB

Definition at line 139 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_3

```
#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_3
```

Value:

```
ULPSS_2K_BANK_2
```

2 KB

Definition at line 140 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_4

```
#define SL_SI91X_POWER_MANAGER_ULPSS_RAM_BANK_4
```

Value:

```
ULPSS_2K_BANK_3
```

2 KB

Definition at line 141 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_power_ram_retention_config_t

Struture to store configuration parameters for RAM retention.

Public Attributes

uint16_t	m4ss_ram_size_kb M4SS RAM size that needs to be restored.
uint16_t	ulpss_ram_size_kb ULPSS RAM size that needs to be restored.
boolean_t	configure_ram_retention Set or Clear RAM retention.
boolean_t	configure_ram_banks Enable will set the RAM banks using size, disable will set RAM banks using bank number.
uint32_t	m4ss_ram_banks M4SS RAM bank number that needs to be restored.
uint32_t	ulpss_ram_banks ULPSS RAM bank number that needs to be restored.
uint32_t	ram_retention_mode RAM Retention modes based on the flags sl_power_ram_retention_mode_t.

Public Attribute Documentation

m4ss_ram_size_kb

> uint16_t sl_power_ram_retention_config_t::m4ss_ram_size_kb

M4SS RAM size that needs to be restored.

Definition at line 148 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

ulpss_ram_size_kb

> uint16_t sl_power_ram_retention_config_t::ulpss_ram_size_kb

ULPSS RAM size that needs to be restored.

Definition at line 149 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

configure_ram_retention

> boolean_t sl_power_ram_retention_config_t::configure_ram_retention

Set or Clear RAM retention.

Definition at line 150 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

configure_ram_banks

```
boolean_t sl_power_ram_retention_config_t::configure_ram_banks
```

Enable will set the RAM banks using size, disable will set RAM banks using bank number.

Definition at line 152 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

m4ss_ram_banks

```
uint32_t sl_power_ram_retention_config_t::m4ss_ram_banks
```

M4SS RAM bank number that needs to be restored.

Definition at line 153 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

ulpss_ram_banks

```
uint32_t sl_power_ram_retention_config_t::ulpss_ram_banks
```

ULPSS RAM bank number that needs to be restored.

Definition at line 154 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

ram_retention_mode

```
uint32_t sl_power_ram_retention_config_t::ram_retention_mode
```

RAM Retention modes based on the flags sl_power_ram_retention_mode_t.

Definition at line 155 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_power_peripheral_t

Structure to store masked values of peripherals to enable/disable it.

Public Attributes

uint32_t	m4ss_peripheral Masked value of M4SS Peripherals sl_power_m4ss_peripheral_t.
uint32_t	ulpss_peripheral Masked value of ULPSS Peripherals sl_power_ulpss_peripheral_t.
uint32_t	npss_peripheral Masked value of NPSS Peripherals sl_power_npss_peripheral_t.

Public Attribute Documentation

m4ss_peripheral

```
uint32_t sl_power_peripheral_t::m4ss_peripheral
```

Masked value of M4SS Peripherals sl_power_m4ss_peripheral_t.

Definition at line 160 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

ulpss_peripheral

```
uint32_t sl_power_peripheral_t::ulpss_peripheral
```

Masked value of ULPSS Peripherals sl_power_ulpss_peripheral_t.

Definition at line 161 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

npss_peripheral

```
uint32_t sl_power_peripheral_t::npss_peripheral
```

Masked value of NPSS Peripherals sl_power_npss_peripheral_t.

Definition at line 162 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_power_manager_ps_transition_event_info_t

Structure representing power state transition event information.

Public Attributes

const sl_power_manager_ps_transition_event_t	event_mask Mask of the transitions on which the callback should be called.
const sl_power_manager_ps_transition_on_event_t	on_event Function that must be called when the event occurs.

Public Attribute Documentation

event_mask

```
const sl_power_manager_ps_transition_event_t sl_power_manager_ps_transition_event_info_t::event_mask
```

Mask of the transitions on which the callback should be called.

Definition at line 199 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

on_event

```
const sl_power_manager_ps_transition_on_event_t sl_power_manager_ps_transition_event_info_t::on_event
```

Function that must be called when the event occurs.

Definition at line 200 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sl_power_manager_ps_transition_event_handle_t

Structure representing power state transition event handle.

Public Attributes

sl_slist_node_t	node List node.
sl_power_manager_ps_transition_event_info_t *	info Handle event info.

Public Attribute Documentation

node

```
sl_slist_node_t sl_power_manager_ps_transition_event_handle_t::node
```

List node.

Definition at line 205 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

info

```
sl_power_manager_ps_transition_event_info_t* sl_power_manager_ps_transition_event_handle_t::info
```

Handle event info.

Definition at line 206 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sl_si91x_power_manager.h

sli_power_sleep_config_t

Structure to store configuration parameters for sleep.

Public Attributes

uint32_t	stack_address
uint32_t	vector_offset
uint32_t	wakeup_callback_address
uint32_t	mode
uint8_t	low_freq_clock

Public Attribute Documentation

stack_address

```
uint32_t sli_power_sleep_config_t::stack_address
```

Definition at line 52 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

vector_offset

```
uint32_t sli_power_sleep_config_t::vector_offset
```

Definition at line 53 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

wakeup_callback_address

```
uint32_t sli_power_sleep_config_t::wakeup_callback_address
```

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

mode

```
uint32_t sli_power_sleep_config_t::mode
```

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

low_freq_clock

uint8_t sli_power_sleep_config_t::low_freq_clock

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/service/power_manager/inc/sli_si91x_power_manager.h

Sensor Hub

Sensor Hub

Modules

- sl_sensorhub_errors_t
- sl_data_deliver_type_t
- sl_sensor_info_t
- sl_sensor_handle_t
- sl_sensor_list_t
- sl_intr_list_t
- sl_intr_list_map_t
- sl_em_event_t
- sl_sensor_cb_info_t
- sl_i2c_config_t
- sl_spi_config_t
- sl_adc_config
- sl_gpio_config_t
- sl_bus_intf_config_t

Enumerations

enum sl_sensor_mode_t {

SL_SH_POLLING_MODE
SL_SH_INTERRUPT_MODE
SL_SH_NO_MODE

}

Enumeration for Sensor HUB data reading mode.

enum sl_sensorhub_event_t {

SL_SENSOR_CREATION_FAILED
SL_SENSOR_STARTED
SL_SENSOR_STOPPED
SL_SENSOR_DATA_READY
SL_SENSOR_CNFG_INVALID
SL_SENSOR_START_FAILED
SL_SENSOR_STOP_FAILED
SL_SENSOR_DELETED
SL_SENSOR_DELETE_FAILED

}

enum sl_gpio_intr_type_t {

SL_SH_RISE_EDGE

```
    SL_SH_FALL_EDGE
    SL_SH_LOW_LEVEL
    SL_SH_HIGH_LEVEL
}
```

```
enum    sl_data_deliver_mode_t {
          SL_SH_THRESHOLD
          SL_SH_TIMEOUT
          SL_SH_NUM_OF_SAMPLES
          SL_SH_NO_DATA_MODE
        }

enum    sl_sensor_status_t {
          SL_SENSOR_INVALID
          SL_SENSOR_VALID
          SL_SENSOR_START
          SL_SENSOR_STOP
        }

enum    sl_power_state_t {
          SL_SI91X_POWER_MANAGER_PS0 = 0
          SL_SI91X_POWER_MANAGER_PS1
          SL_SI91X_POWER_MANAGER_PS2
          SL_SI91X_POWER_MANAGER_PS3
          SL_SI91X_POWER_MANAGER_PS4
          SL_SI91X_POWER_MANAGER_SLEEP
          SL_SI91X_POWER_MANAGER_STANDBY
          LAST_ENUM_POWER_STATE
          SL_SH_PS4TOPS2
          SL_SH_PS2TOPS4
          SL_SH_SLEEP_WAKEUP
          SL_SH_DUMMY
        }
```

Typedefs

```
typedef void(*    sl_sensor_signalEvent_t)(uint8_t sensor_id, uint8_t event, void *data)

typedef struct    sl_adc_cfg_t
sl_adc_config
```

Functions

```
sl_status_t    sl_si91x_sensorhub_init()
```
Initialization Peripherals of Sensor HUB.

```
sl_status_t    sl_si91x_sensorhub_detect_sensors(sl_sensor_id_t sensor_id_info[], uint8_t num_of_sensors)
```
Detect the sensors.

```
sl_status_t    sl_si91x_sensorhub_delete_sensor(sl_sensor_id_t sensor_id)
```
Delete the specific sensor from the Sensor HUB.

```
sl_status_t    sl_si91x_sensorhub_create_sensor(sl_sensor_id_t sensor_id)
```
Create a sensor instance in the sensor hub.

```
sl_status_t    sl_si91x_sensorhub_start_sensor(sl_sensor_id_t sensor_id)
```
Start the sensor operation for the given sensor.

sl_status_t	sl_si91x_sensorhub_stop_sensor(sl_sensor_id_t sensor_id) Stop the sensor operation for the given sensor.
void	sl_si91x_em_post_event(sl_sensor_id_t sensor_id, sl_sensorhub_event_t event, void *dataPtr, TickType_t ticks_to_wait) Post the events to event manager(EM) to be notified to the application.
void	sl_si91x_sensor_task(void) Task to handle the sensor operations.
void	sl_si91x_power_state_task(void) Task to handle the system power operations.
void	sl_si91x_em_task(void) Task to handle the operations of the Event Manager(EM)
int32_t	sli_si91x_i2c_init(void) Initialize the I2C Interface based on configuration.
int32_t	sli_si91x_spi_init(void) Initialize SPI Interface based on configuration.
int32_t	sli_si91x_i2c_sensors_scan(uint16_t address) Scan the I2C sensors.
sl_sensor_impl_type_t *	sli_si91x_get_sensor_implementation(int32_t sensor_id) to get sensor implementation.
int32_t	sli_si91x_create_sensor_list_index() Create the sensor list.
uint32_t	sli_si91x_get_sensor_index(sl_sensor_id_t sensor_id) Get the sensor index for the sensor list.
uint32_t	sli_si91x_delete_sensor_list_index(sl_sensor_id_t sensor_id) Delete the sensor index for the sensor list.
sl_sensor_info_t *	sli_si91x_get_sensor_info(sl_sensor_id_t sensor_id) get the sensor info from the sensor configuration structure.
sl_status_t	sl_si91x_sensorhub_notify_cb_register(sl_sensor_signalEvent_t cb_event, sl_sensor_id_t *cb_ack) Call back function to the application.
void	sl_si91x_sensors_timer_cb(TimerHandle_t xTimer) Sensor OS timer callback function.
sl_status_t	sl_si91x_gpio_interrupt_config(uint16_t gpio_pin, sl_gpio_intr_type_t intr_type) Configuring the different types of NPSS GPIO interrupts in the Sensor HUB.
void	sl_si91x_gpio_interrupt_start(uint16_t gpio_pin) Enable and set the priority of NPSS GPIO interrupt.
void	sl_si91x_gpio_interrupt_stop(uint16_t gpio_pin) Mask and Disable the NPSS GPIO interrupt.
sl_status_t	sl_si91x_sensor_hub_start(void) Start the sensor hub Tasks.
void	sli_si91x_set_alarm_intr_time(uint16_t interval) set the alarm interrupt time.
void	sli_si91x_init_m4alarm_config(void) initialize the Alarm block.

void	sli_si91x_config_wakeup_source(uint16_t sleep_time) Configure the wake-up source for the system.
void	sli_si91x_sleep_wakeup(uint16_t sh_sleep_time) Configures sleep/wakeup sources for the system.
void	sli_si91x_sensorhub_ps4tops2_state(void) Change the system status from PS4 to PS2.
void	sli_si91x_sensorhub_ps2tops4_state(void) Change the system status from PS2 to PS4.
sl_status_t	sli_si91x_adc_init(void) Initialize ADC Interface based on configuration.
void	vPortSetupTimerInterrupt(void) Initialize and configure the systic timer for the RTOS.
void	ARM_I2C_SignalEvent(uint32_t event) I2C event handler.
sl_adc_cfg_t *	sl_si91x_fetch_adc_bus_intf_info(void) Fetch ADC bus interface information.
void	sl_si91x_adc_callback(uint8_t channel_no, uint8_t event) ADC callback from RSI_ADC_InterruptHandler.

Macros

#define	SL_SH_SENSOR_TASK_STACK_SIZE (512 * 2) Sensor task stack size.
#define	SL_SH_EM_TASK_STACK_SIZE (512 * 2) EM task stack size.
#define	SL_SH_POWER_SAVE_TASK_STACK_SIZE (512 * 2) Power task stack size.
#define	SL_EM_TASK_RUN_TICKS osWaitForever Event task Messageque maximum waiting time.
#define	MAP_TABLE_SIZE 10 Size of the interrupt MAP table.
#define	NPSS_GPIO_IRQHandler IRQ021_Handler NPSS gpio IRQ handler.
#define	NPSS_GPIO_NVIC NPSS_TO_MCU_GPIO_INTR_IRQn NPSS gpio IRQ Number.
#define	SL_ALARM_PERIODIC_TIME 10 Periodic alarm configuration in milliseconds.
#define	RC_TRIGGER_TIME 5 RC clock trigger time.
#define	RO_TRIGGER_TIME 0 RO clock trigger time.
#define	NO_OF_HOURS_IN_A_DAY 24 Number of hours.

#define NO_OF_MINUTES_IN_AN_HOUR 60
Number of minutes.

#define NO_OF_SECONDS_IN_A_MINUTE 60
Number of Seconds.

#define NO_OF_MILLISECONDS_IN_A_SECOND 1000
Number of milliseconds.

#define NO_OF_MONTHS_IN_A_YEAR 12
Number of months.

#define BASE_YEAR 2000
Start year for alarm configuration.

#define NO_OF_DAYS_IN_A_MONTH_1 28
Number of days in month.

#define NO_OF_DAYS_IN_A_MONTH_2 29
Number of days in month.

#define NO_OF_DAYS_IN_A_MONTH_3 30
Number of days in month.

#define NO_OF_DAYS_IN_A_MONTH_4 31
Number of days in month.

#define RTC_ALARM_IRQHandler IRQ028_Handler
Alarm IRQ handler.

#define NVIC_RTC_ALARM MCU_CAL_ALARM_IRQn
Alarm IRQ number.

Enumeration Documentation

sl_sensor_mode_t

> sl_sensor_mode_t

Enumeration for Sensor HUB data reading mode.

Enumerator	
SL_SH_POLLING_MODE	POLLING_MODE.
SL_SH_INTERRUPT_MODE	INTERRUPT_MODE.
SL_SH_NO_MODE	

Definition at line `99` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sl_sensorhub_event_t

> sl_sensorhub_event_t

Enumerator	
SL_SENSOR_CREATION_FAILED	
SL_SENSOR_STARTED	
SL_SENSOR_STOPPED	
SL_SENSOR_DATA_READY	

SL_SENSOR_CNFG_INVALID	
SL_SENSOR_START_FAILED	
SL_SENSOR_STOP_FAILED	
SL_SENSOR_DELETED	
SL_SENSOR_DELETE_FAILED	

Definition at line 108 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_gpio_intr_type_t

> sl_gpio_intr_type_t

Enumerator	
SL_SH_RISE_EDGE	Interrupt at GPIO rise edge.
SL_SH_FALL_EDGE	Interrupt at GPIO fall edge.
SL_SH_LOW_LEVEL	Interrupt at GPIO low level.
SL_SH_HIGH_LEVEL	Interrupt at GPIO high level.

Definition at line 123 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_data_deliver_mode_t

> sl_data_deliver_mode_t

Enumerator	
SL_SH_THRESHOLD	Threshold value for sensor data delivery.
SL_SH_TIMEOUT	Timeout value for sensor data delivery.
SL_SH_NUM_OF_SAMPLES	Number of samples for sensor data delivery.
SL_SH_NO_DATA_MODE	

Definition at line 133 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_sensor_status_t

> sl_sensor_status_t

Enumerator	
SL_SENSOR_INVALID	Sensor is Invalid.
SL_SENSOR_VALID	Sensor is Valid.
SL_SENSOR_START	Sensor is Started.
SL_SENSOR_STOP	Sensor is Stoped.

Definition at line 143 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_power_state_t

> sl_power_state_t

Enumerator	
SL_SI91X_POWER_MANAGER_PS0	PS0 Power State.
SL_SI91X_POWER_MANAGER_PS1	PS1 Power State.
SL_SI91X_POWER_MANAGER_PS2	PS2 Power State.
SL_SI91X_POWER_MANAGER_PS3	PS3 Power State.
SL_SI91X_POWER_MANAGER_PS4	PS4 Power State.
SL_SI91X_POWER_MANAGER_SLEEP	Sleep.
SL_SI91X_POWER_MANAGER_STANDBY	Standby.
LAST_ENUM_POWER_STATE	Last enum for validation.
SL_SH_PS4TOPS2	
SL_SH_PS2TOPS4	
SL_SH_SLEEP_WAKEUP	
SL_SH_DUMMY	

Definition at line `153` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

Typedef Documentation

sl_sensor_signalEvent_t

```
typedef void(* sl_sensor_signalEvent_t) (uint8_t sensor_id, uint8_t event, void *data) )(uint8_t sensor_id, uint8_t event, void *data)
```

Definition at line `92` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sl_adc_cfg_t

```
typedef struct sl_adc_config sl_adc_cfg_t
```

Definition at line `292` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

Function Documentation

sl_si91x_sensorhub_init

```
sl_status_t sl_si91x_sensorhub_init ()
```

Initialization Peripherals of Sensor HUB.

Parameters

[in]	-	NULL
[out]	-	Return the Sensor Hub bus Initialization status

This function will initialize the Peripherals, based on the user configuration. like - I2C/SPI/ADC

Returns

- status 0 if successful, else error code as follow:
  - SL_STATUS_FAIL (0x0001)- Fail, Peripherals initialization failed
  - SL_STATUS_OK (0X000)- Success, Peripherals initialization done properly

Definition at line 332 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_sensorhub_detect_sensors

```
sl_status_t sl_si91x_sensorhub_detect_sensors (sl_sensor_id_t sensor_id_info[], uint8_t num_of_sensors)
```

Detect the sensors.

Parameters

[in]	sensor_id_info	- Sensor IDs
[in]	num_of_sensors	- Number of sensors given by user
[out]	-	Return the status of the Detect sensor

This function will detect the sensor, based on the user configuration. and it will store the scanned sensor ID in the structure.

Returns

- if successful, Return number of sensors is scanned else error code SL_STATUS_FAIL (0x0001)- Fail,

Definition at line 350 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_sensorhub_delete_sensor

```
sl_status_t sl_si91x_sensorhub_delete_sensor (sl_sensor_id_t sensor_id)
```

Delete the specific sensor from the Sensor HUB.

Parameters

[in]	sensor_id	- Sensor ID
[out]	-	Return the delete sensors status

This function will delete the specific sensor from the sensor list, modify the sensor status to invalid, and publish the events to the event task.

Returns

- status 0 if successful, else error code as follow:
  - SL_STATUS_FAIL (0x0001) - Fail, Delete sensor failed
  - SL_STATUS_OK (0X000) - Success, Delete sensor Success

Definition at line 368 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_sensorhub_create_sensor

```
sl_status_t sl_si91x_sensorhub_create_sensor (sl_sensor_id_t sensor_id)
```

Create a sensor instance in the sensor hub.

Parameters

[in]	sensor_id	- Id of sensor
[out]	-	Return the create sensors status

This function will create the sensor instances in the sensor HUB. Checking the sensor state and creating the sensor list index. Allocate max sample memory for the specific sensor. Creating the RTOS timer instance for the sensor sampling

interval.

Returns

- status 0 if successful, else error code as follow:
  - SL_STATUS_FAIL (0x0001)- Fail, Create sensor instance failed
  - SL_STATUS _OK (0X000) - Create sensor instanceSuccess ,

Definition at line 388 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_sensorhub_start_sensor

```
sl_status_t sl_si91x_sensorhub_start_sensor (sl_sensor_id_t sensor_id)
```

Start the sensor operation for the given sensor.

Parameters

[in]	sensor_id	- Id of sensor
[out]	-	Return the Start sensors status

This function will start the sensor operation for the given sensor. Based on the sensor mode it will start the Polling/interrupt mode operations. It will Start the RTOS timer in Polling mode for the sensor. It will enable IRQ handles for the interrupt mode operations. it will post the status events to the em task. It will send the Control commands to the sensor to perform the operations.

Returns

- status 0 if successful, else error code as follow:
  - SL_STATUS_FAIL (0x0001)- Fail, Sensor start failed
  - SL_STATUS _OK (0X000) - Success ,Sensor start done properly

Definition at line 410 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_sensorhub_stop_sensor

```
sl_status_t sl_si91x_sensorhub_stop_sensor (sl_sensor_id_t sensor_id)
```

Stop the sensor operation for the given sensor.

Parameters

[in]	sensor_id	- Id of sensor
[out]	-	Return the Stop sensor status

This function will stop the sensor operation for the given sensor. Based on the sensor mode it will stop the Polling/interrupt mode operations and updates the sensor statutes. It will Stop the RTOS timer in Polling mode for the sensor. It will disable IRQ handles for the interrupt mode operations. it will post the status events to the em task. It will send the Control commands to the sensor to perform the operations.

Returns

- status 0 if successful, else error code as follows:
  - SL_STATUS_FAIL (0x0001)- Fail, Sensor stop failed
- SL_STATUS _OK (0X000) - Success ,Sensor stop done properly

Definition at line 433 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_em_post_event

```
void sl_si91x_em_post_event (sl_sensor_id_t sensor_id, sl_sensorhub_event_t event, void *dataPtr, TickType_t ticks_to_wait)
```

Post the events to event manager(EM) to be notified to the application.

Parameters

[in]	sensor_id	- id of the sensor
[in]	event	- sensor hub events
[in]	dataPtr	- sensor data pointer
[in]	ticks_to_wait	- max time to wait for the post
[out]	-	None

This function will update the event queue based on the sensor event. It will acquire the events task mutex and update the events in the messageque and release the mutex.

Returns

- - NULL

Definition at line `452` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sl_si91x_sensor_task

```
void sl_si91x_sensor_task (void)
```

Task to handle the sensor operations.

Parameters

[in]		None
[out]	-	None

This function will handle the sensor operations. It will create the OS event and mutex to perform the sensor operations. It will sample the Sensor data based on the event flags. It will check the events and post the sensor data to the em task based on the sensor data delivery mode.

Returns

- - NULL

Definition at line `473` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sl_si91x_power_state_task

```
void sl_si91x_power_state_task (void)
```

Task to handle the system power operations.

Parameters

[in]		None
[out]	-	None

This function will handle the system power operations. It will change the system from one power save mode to another power save made like(PS4 to PS2),(PS2toPS4),(SLeep_mode). It will use the Binary semaphore for changing the power states.

Returns

- - NULL

Definition at line `490` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sl_si91x_em_task

```
void sl_si91x_em_task (void)
```

Task to handle the operations of the Event Manager(EM)

Parameters

[in]		None
[out]	-	None

This function will handle the Event Manager(EM) operations. It will create the osMessageQueue and mutex to perform the event operations. It will check the events in the osMessageQueue and send the sensor data to the application. It will check the acknowledgment from the application. **Returns**

- - NULL

Definition at line `507` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_i2c_init

```
int32_t sli_si91x_i2c_init (void)
```

Initialize the I2C Interface based on configuration.

Parameters

[in]		None
[out]	-	Return the I2C init status

This function will configure/Initialize I2C Interface based on configuration.

Returns

- Initialize status to the application

Definition at line `521` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_spi_init

```
int32_t sli_si91x_spi_init (void)
```

Initialize SPI Interface based on configuration.

Parameters

[in]		None

[out]	-	Return the SPI init status

This function will configure/Initialize SPI Interface based on configuration.

Returns

- Initialize status to the application.

Definition at line `535` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_i2c_sensors_scan

```
int32_t sli_si91x_i2c_sensors_scan (uint16_t address)
```

Scan the I2C sensors.

Parameters

[in]	address	- Address of sensor
[out]	-	Return the I2C sensors scan status

This function will scan the I2C sensors based on the sensor address.

Returns

- status 0 if successful, else it will wait for the sensor response.

Definition at line `550` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_get_sensor_implementation

```
sl_sensor_impl_type_t * sli_si91x_get_sensor_implementation (int32_t sensor_id)
```

to get sensor implementation.

Parameters

[in]	sensor_id	- Id of sensor
[out]	-	Return the Sensor ID from the implementation structure

This function will used to perform the sensor operation by using the sensor implementation structure.

Returns

- if successful, returns the Sensor implementation structure else it will return an error code.

Definition at line `565` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_create_sensor_list_index

```
int32_t sli_si91x_create_sensor_list_index ()
```

Create the sensor list.

Parameters

[in]	-	None
[out]	-	Return the Sensor ID from the implementation struct

This function will Create the sensor list index based on the sensor status For the maximum sensors available in the sensor hub

Returns

- if successful, returns the sensor index else it will break the sensor list.

Definition at line `581` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_get_sensor_index

```
uint32_t sli_si91x_get_sensor_index (sl_sensor_id_t sensor_id)
```

Get the sensor index for the sensor list.

Parameters

[in]	sensor_id	- Id of sensor
[out]	-	Return the Sensor Index from the list

This function will retrieve the sensor index from the sensor list based on the sensor status and sensor ID for the maximum number of sensors available in the sensor hub.

Returns

- if successful, returns the sensor index else it returns the 0XFF as a sensor index fail.

Definition at line `598` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_delete_sensor_list_index

```
uint32_t sli_si91x_delete_sensor_list_index (sl_sensor_id_t sensor_id)
```

Delete the sensor index for the sensor list.

Parameters

[in]	sensor_id	- Id of sensor
[out]	-	Return the deleted Sensor Index from the list

This function will delete the sensor from the sensor list based on The sensor ID for the maximum number of sensors available in the sensor hub.

Returns

- if successful, returns the sensor index else it returns the 0XFF as a sensor index fail.

Definition at line `614` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_get_sensor_info

```
sl_sensor_info_t * sli_si91x_get_sensor_info (sl_sensor_id_t sensor_id)
```

get the sensor info from the sensor configuration structure.

Parameters

[in]	sensor_id	- Id of sensor
[out]	-	Return the Match Sensor-id Index from the list

This function will retrieve sensor data from the sensor configuration structure to update the sensor list configuration structure for the maximum number of sensors available in the sensor hub.

Returns

- if successful, returns the sensor information. else it returns the NULL as a sensor ID not found.

Definition at line `631` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sl_si91x_sensorhub_notify_cb_register

```
sl_status_t sl_si91x_sensorhub_notify_cb_register (sl_sensor_signalEvent_t cb_event, sl_sensor_id_t *cb_ack)
```

Call back function to the application.

Parameters

[in]	cb_event	- Pointer to the callback event
[in]	cb_ack	- Pointer to callback acknowledge to the application
[out]	-	None

This function will update the callback function event and acknowledgment.

Returns

- - If successful returns SL_STATUS_OK,If fail returns error code

Definition at line `646` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sl_si91x_sensors_timer_cb

```
void sl_si91x_sensors_timer_cb (TimerHandle_t xTimer)
```

Sensor OS timer callback function.

Parameters

[in]	xTimer	- Timer handle
[out]	-	None

This function will set the event flag bits based on the sensor sampling intervals. In the polling mode call the timer call-back function.

Returns

- - NULL

Definition at line `661` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sl_si91x_gpio_interrupt_config

```
sl_status_t sl_si91x_gpio_interrupt_config (uint16_t gpio_pin, sl_gpio_intr_type_t intr_type)
```

Configuring the different types of NPSS GPIO interrupts in the Sensor HUB.

Parameters

[in]	gpio_pin	- NPSS gpio pin number
[in]	intr_type	NPSS gpio interrupt type
[out]	-	None

This function configures the NPSS gpios and enables the interrupt mode for the gpios.

Returns

- - If successful returns SL_STATUS_OK,If fail returns error code

Definition at line 676 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_gpio_interrupt_start

```
void sl_si91x_gpio_interrupt_start (uint16_t gpio_pin)
```

Enable and set the priority of NPSS GPIO interrupt.

Parameters

[in]	gpio_pin	- NPSS gpio pin number
[out]	-	None

This function configures and sets the priority of the NPSS GPIO interrupt. GPIO interrupt priority is= (configMAX_SYSCALL_INTERRUPT_PRIORITY - 1)

Returns

- - NULL

Definition at line 691 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_gpio_interrupt_stop

```
void sl_si91x_gpio_interrupt_stop (uint16_t gpio_pin)
```

Mask and Disable the NPSS GPIO interrupt.

Parameters

[in]	gpio_pin	- NPSS gpio pin number
[out]	-	None

This function masks and disables the IRQ handler the NPSS GPIO interrupt.

Returns

- - NULL

Definition at line 705 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_sensor_hub_start

```
sl_status_t sl_si91x_sensor_hub_start (void)
```

Start the sensor hub Tasks.

Parameters

[in]		None
[out]	-	None

This Starts the sensor hub Tasks.

Returns

- - If succesfull returns SL_STATUS_OK,If fail returns error code

Definition at line `719` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_set_alarm_intr_time

> void sli_si91x_set_alarm_intr_time (uint16_t interval)

set the alarm interrupt time.

Parameters

[in]	interval	- interval time
[out]	-	None

This function will set the alarm interrupt based on the periodic time.

Returns

- - NULL

Definition at line `733` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_init_m4alarm_config

> void sli_si91x_init_m4alarm_config (void)

initialize the Alarm block.

Parameters

[in]		None
[out]	-	None

This function will initialize the Alarm block.

Returns

- - NULL

Definition at line `747` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_config_wakeup_source

> void sli_si91x_config_wakeup_source (uint16_t sleep_time)

Configure the wake-up source for the system.

Parameters

[in]	sleep_time	- Sleep time for the sensor hub
[out]	-	None

This function will configure the wake-up source to the system.

Returns

- - NULL

Definition at line `761` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_sleep_wakeup

```
void sli_si91x_sleep_wakeup (uint16_t sh_sleep_time)
```

Configures sleep/wakeup sources for the system.

Parameters

[in]	sh_sleep_time	- Sleep time for the sensor hub
[out]	-	None

This function will configure sleep/wakeup sources.

Returns

- - NULL

Definition at line `778` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_sensorhub_ps4tops2_state

```
void sli_si91x_sensorhub_ps4tops2_state (void)
```

Change the system status from PS4 to PS2.

Parameters

[in]		None
[out]	-	None

This function will change the system status from PS4 to PS2.

Returns

- - NULL

Definition at line `792` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_sensorhub_ps2tops4_state

```
void sli_si91x_sensorhub_ps2tops4_state (void)
```

Change the system status from PS2 to PS4.

Parameters

[in]		None
[out]	-	None

This function will change the system status from PS2 to PS4.

Returns

- - NULL

Definition at line `806` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

sli_si91x_adc_init

```
sl_status_t sli_si91x_adc_init (void)
```

Initialize ADC Interface based on configuration.

Parameters

[in]		None

This function will configure/Initialize ADC Interface based on configuration.

Returns

- error code SL_STATUS_FAIL (0x0001)- Fail , SL_STATUS_OK (0X000)- Success

Definition at line `821` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

vPortSetupTimerInterrupt

```
void vPortSetupTimerInterrupt (void)
```

Initialize and configure the systic timer for the RTOS.

Parameters

[in]		None
[out]	-	None

Set up the systic timer to generate the tick interrupts at the required frequency.

Returns

- Initialize status to the application.

Definition at line `835` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

ARM_I2C_SignalEvent

```
void ARM_I2C_SignalEvent (uint32_t event)
```

I2C event handler.

Parameters

[in]	event	- event I2C transmit and receive events
[out]	-	None

Returns

- None

Definition at line 846 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_fetch_adc_bus_intf_info

```
uint8_t sl_si91x_fetch_adc_bus_intf_info (void)
```

Fetch ADC bus interface information.

Parameters

N/A		

This can be used by lower level layers **Returns**

- Pointer to ADC configuration structure

Definition at line 856 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_si91x_adc_callback

```
void sl_si91x_adc_callback (uint8_t channel_no, uint8_t event)
```

ADC callback from RSI_ADC_InterruptHandler.

Parameters

[in]	channel_no	- respective channel number
[in]	event	- callback event (ADC_STATIC_MODE_CALLBACK, ADC_THRSHOLD_CALLBACK, INTERNAL_DMA, FIFO_MODE_EVENT)

Definition at line 868 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

Macro Definition Documentation

SL_SH_SENSOR_TASK_STACK_SIZE

```
#define SL_SH_SENSOR_TASK_STACK_SIZE
```

Value:

```
(512 * 2)
```

Sensor task stack size.

Definition at line 54 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

SL_SH_EM_TASK_STACK_SIZE

```
#define SL_SH_EM_TASK_STACK_SIZE
```

Value:

```
(512 * 2)
```

EM task stack size.

Definition at line 55 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

SL_SH_POWER_SAVE_TASK_STACK_SIZE

```
#define SL_SH_POWER_SAVE_TASK_STACK_SIZE
```

Value:

```
(512 * 2)
```

Power task stack size.

Definition at line 56 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

SL_EM_TASK_RUN_TICKS

```
#define SL_EM_TASK_RUN_TICKS
```

Value:

```
osWaitForever
```

Event task Messageque maximum waiting time.

Definition at line 57 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

MAP_TABLE_SIZE

```
#define MAP_TABLE_SIZE
```

Value:

```
10
```

Size of the interrupt MAP table.

Definition at line 58 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NPSS_GPIO_IRQHandler

```
#define NPSS_GPIO_IRQHandler
```

Value:

```
IRQ021_Handler
```

NPSS gpio IRQ handler.

Definition at line `63` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

NPSS_GPIO_NVIC

```
#define NPSS_GPIO_NVIC
```

Value:

```
NPSS_TO_MCU_GPIO_INTR_IRQn
```

NPSS gpio IRQ Number.

Definition at line `64` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

SL_ALARM_PERIODIC_TIME

```
#define SL_ALARM_PERIODIC_TIME
```

Value:

```
10
```

Periodic alarm configuration in milliseconds.

Definition at line `69` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

RC_TRIGGER_TIME

```
#define RC_TRIGGER_TIME
```

Value:

```
5
```

RC clock trigger time.

Definition at line `70` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

RO_TRIGGER_TIME

```
#define RO_TRIGGER_TIME
```

Value:

```
0
```

RO clock trigger time.

Definition at line 71 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_HOURS_IN_A_DAY

```
#define NO_OF_HOURS_IN_A_DAY
```

Value:

```
24
```

Number of hours.

Definition at line 72 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_MINUTES_IN_AN_HOUR

```
#define NO_OF_MINUTES_IN_AN_HOUR
```

Value:

```
60
```

Number of minutes.

Definition at line 73 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_SECONDS_IN_A_MINUTE

```
#define NO_OF_SECONDS_IN_A_MINUTE
```

Value:

```
60
```

Number of Seconds.

Definition at line 74 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_MILLISECONDS_IN_A_SECOND

```
#define NO_OF_MILLISECONDS_IN_A_SECOND
```

Value:

```
1000
```

Number of milliseconds.

Definition at line 75 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_MONTHS_IN_A_YEAR

```
#define NO_OF_MONTHS_IN_A_YEAR
```

Value:

```
12
```

Number of months.

Definition at line 76 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

BASE_YEAR

```
#define BASE_YEAR
```

Value:

```
2000
```

Start year for alarm configuration.

Definition at line 77 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_DAYS_IN_A_MONTH_1

```
#define NO_OF_DAYS_IN_A_MONTH_1
```

Value:

```
28
```

Number of days in month.

Definition at line 78 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_DAYS_IN_A_MONTH_2

```
#define NO_OF_DAYS_IN_A_MONTH_2
```

Value:

```
29
```

Number of days in month.

Definition at line 79 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_DAYS_IN_A_MONTH_3

```
#define NO_OF_DAYS_IN_A_MONTH_3
```

Value:

```
30
```

Number of days in month.

Definition at line 80 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NO_OF_DAYS_IN_A_MONTH_4

```
#define NO_OF_DAYS_IN_A_MONTH_4
```

Value:

```
31
```

Number of days in month.

Definition at line 81 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

RTC_ALARM_IRQHandler

```
#define RTC_ALARM_IRQHandler
```

Value:

```
IRQ028_Handler
```

Alarm IRQ handler.

Definition at line 86 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

NVIC_RTC_ALARM

```
#define NVIC_RTC_ALARM
```

Value:

```
MCU_CAL_ALARM_IRQn
```

Alarm IRQ number.

Definition at line 87 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_sensorhub_errors_t

Public Attributes

bool i2c
I2C bus error status.

bool spi
SPI bus error status.

bool adc
ADC bus error status.

bool sensor_global_status
All buses error status.

Public Attribute Documentation

i2c

```
bool sl_sensorhub_errors_t::i2c
```

I2C bus error status.

Definition at line 163 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi

```
bool sl_sensorhub_errors_t::spi
```

SPI bus error status.

Definition at line 164 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

adc

```
bool sl_sensorhub_errors_t::adc
```

ADC bus error status.

Definition at line 165 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_global_status

```
bool sl_sensorhub_errors_t::sensor_global_status
```

All buses error status.

Definition at line 166 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_data_deliver_type_t

Public Attributes

sl_data_deliver_mode_t	data_mode
uint32_t	threshold Threshold value for the sensor.
uint32_t	timeout Timeout values for the sensor.
uint32_t	numofsamples Number of samples for the sensor.
union sl_data_deliver_type_t::@0	@1

Public Attribute Documentation

data_mode

```
sl_data_deliver_mode_t sl_data_deliver_type_t::data_mode
```

Definition at line 173 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

threshold

```
uint32_t sl_data_deliver_type_t::threshold
```

Threshold value for the sensor.

Definition at line 175 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

timeout

```
uint32_t sl_data_deliver_type_t::timeout
```

Timeout values for the sensor.

Definition at line 176 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

numofsamples

uint32_t sl_data_deliver_type_t::numofsamples

Number of samples for the sensor.

Definition at line 177 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

@1

union sl_data_deliver_type_t::@0 sl_data_deliver_type_t::@1

Definition at line 178 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_sensor_info_t

Public Attributes

char *	sensor_name Name of the sensor.
int16_t	sensor_intr_type GPIO Interrupt Configurations.
uint16_t	sampling_intr_req_pin GPIO pin for sampling the sensor data.
uint32_t	sampling_interval Sensor data sampling interval.
uint8_t	address Address of sensor.
uint16_t	channel Channel for adc.
union sl_sensor_info_t::@2	@3
sl_sensor_id_t	sensor_id Sensor id.
sl_sensor_bus_t	sensor_bus Protocol for the sensor(spi/i2c)
sl_sensor_mode_t	sensor_mode Sensor Mode(Enumeration)
sl_sensor_range_t	sensor_range Range of sensor.
sl_data_deliver_type_t	data_deliver Data delivery mode for the sensor.
sl_sensor_data_group_t *	sensor_data_ptr Sensor data storage structure.

Public Attribute Documentation

sensor_name

```
char* sl_sensor_info_t::sensor_name
```

Name of the sensor.

Definition at line 185 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_intr_type

```
int16_t sl_sensor_info_t::sensor_intr_type
```

GPIO Interrupt Configurations.

Definition at line 186 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sampling_intr_req_pin

```
uint16_t sl_sensor_info_t::sampling_intr_req_pin
```

GPIO pin for sampling the sensor data.

Definition at line 187 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sampling_interval

```
uint32_t sl_sensor_info_t::sampling_interval
```

Sensor data sampling interval.

Definition at line 188 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

address

```
uint8_t sl_sensor_info_t::address
```

Address of sensor.

Definition at line 190 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

channel

```
uint16_t sl_sensor_info_t::channel
```

Channel for adc.

Definition at line 191 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

@3

```
union sl_sensor_info_t::@2 sl_sensor_info_t::@3
```

Definition at line 192 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_id

sl_sensor_id_t sl_sensor_info_t::sensor_id

Sensor id.

Definition at line 193 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_bus

sl_sensor_bus_t sl_sensor_info_t::sensor_bus

Protocol for the sensor(spi/i2c)

Definition at line 194 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_mode

sl_sensor_mode_t sl_sensor_info_t::sensor_mode

Sensor Mode(Enumeration)

Definition at line 195 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_range

sl_sensor_range_t sl_sensor_info_t::sensor_range

Range of sensor.

Definition at line 196 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

data_deliver

sl_data_deliver_type_t sl_sensor_info_t::data_deliver

Data delivery mode for the sensor.

Definition at line 197 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_data_ptr

sl_sensor_data_group_t* sl_sensor_info_t::sensor_data_ptr

Sensor data storage structure.

Definition at line 198 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_sensor_handle_t

Public Attributes

void *	sensor_handle Sensor handle.
void *	ctrl_handle Sensor control handle.
uint8_t	sensor_event_bit Sensor event bits.
uint8_t	event_ack Sensor event acknowledge.
uint16_t	max_samples Maximum samples for the sensors.
sl_sensor_info_t *	config_st Sensor configuration structure.
sl_sensor_impl_type_t *	sensor_impl Sensor implementation structure.
sl_sensor_status_t	sensor_status Sensor status.
TimerHandle_t	timer_handle RTOS timer handle.

Public Attribute Documentation

sensor_handle

```
void* sl_sensor_handle_t::sensor_handle
```

Sensor handle.

Definition at line 205 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

ctrl_handle

```
void* sl_sensor_handle_t::ctrl_handle
```

Sensor control handle.

Definition at line 206 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_event_bit

uint8_t sl_sensor_handle_t::sensor_event_bit

Sensor event bits.

Definition at line 207 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

event_ack

uint8_t sl_sensor_handle_t::event_ack

Sensor event acknowledge.

Definition at line 208 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

max_samples

uint16_t sl_sensor_handle_t::max_samples

Maximum samples for the sensors.

Definition at line 209 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

config_st

sl_sensor_info_t* sl_sensor_handle_t::config_st

Sensor configuration structure.

Definition at line 210 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_impl

sl_sensor_impl_type_t* sl_sensor_handle_t::sensor_impl

Sensor implementation structure.

Definition at line 211 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_status

sl_sensor_status_t sl_sensor_handle_t::sensor_status

Sensor status.

Definition at line 212 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

timer_handle

> TimerHandle_t sl_sensor_handle_t::timer_handle

RTOS timer handle.

Definition at line 213 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_sensor_list_t

Public Attributes

uint8_t	sensor_index Sensor index.
sl_sensor_handle_t	sl_sensors_st Sensor handle structure.

Public Attribute Documentation

sensor_index

```
uint8_t sl_sensor_list_t::sensor_index
```

Sensor index.

Definition at line 220 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_sensors_st

```
sl_sensor_handle_t sl_sensor_list_t::sl_sensors_st[SL_MAX_NUM_SENSORS]
```

Sensor handle structure.

Definition at line 221 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_intr_list_t

Public Attributes

uint8_t	sensor_list_index Interrupt mode sensor index.
uint16_t	intr Interrupt GPIO Pin.
uint16_t	adc_intr_channel Channel number for ADC interrupt.

Public Attribute Documentation

sensor_list_index

```
uint8_t sl_intr_list_t::sensor_list_index
```

Interrupt mode sensor index.

Definition at line 228 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

intr

```
uint16_t sl_intr_list_t::intr
```

Interrupt GPIO Pin.

Definition at line 229 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

adc_intr_channel

```
uint16_t sl_intr_list_t::adc_intr_channel
```

Channel number for ADC interrupt.

Definition at line 230 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_intr_list_map_t

Public Attributes

uint8_t	map_index Map Table Index.
sl_intr_list_t	map_table Sensor list MAP Table.

Public Attribute Documentation

map_index

```
uint8_t sl_intr_list_map_t::map_index
```

Map Table Index.

Definition at line 237 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

map_table

```
sl_intr_list_t sl_intr_list_map_t::map_table[MAP_TABLE_SIZE]
```

Sensor list MAP Table.

Definition at line 238 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_em_event_t

Public Attributes

void *	em_sensor_data String the sensor data address.
sl_sensor_id_t	sensor_id Sensor id information.
sl_sensorhub_event_t	event Sensors HUB Callback Events.

Public Attribute Documentation

em_sensor_data

```
void* sl_em_event_t::em_sensor_data
```

String the sensor data address.

Definition at line 245 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sensor_id

```
sl_sensor_id_t sl_em_event_t::sensor_id
```

Sensor id information.

Definition at line 246 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

event

```
sl_sensorhub_event_t sl_em_event_t::event
```

Sensors HUB Callback Events.

Definition at line 247 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_sensor_cb_info_t

Public Attributes

sl_sensor_signalEvent_t	cb_event Event callback.
sl_sensor_id_t *	cb_event_ack Event callback acknowledge.

Public Attribute Documentation

cb_event

```
sl_sensor_signalEvent_t sl_sensor_cb_info_t::cb_event
```

Event callback.

Definition at line 254 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

cb_event_ack

```
sl_sensor_id_t* sl_sensor_cb_info_t::cb_event_ack
```

Event callback acknowledge.

Definition at line 255 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_i2c_config_t

Public Attributes

uint8_t	i2c_id I2C instances(I2C0/1/2)
uint8_t	i2c_power_state I2C power state.
uint8_t	i2c_control_mode I2C bus control configuration.
uint16_t	i2c_bus_speed I2C bus speed.
uint32_t	i2c_slave_addr I2C slave address.

Public Attribute Documentation

i2c_id

```
uint8_t sl_i2c_config_t::i2c_id
```

I2C instances(I2C0/1/2)

Definition at line `262` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

i2c_power_state

```
uint8_t sl_i2c_config_t::i2c_power_state
```

I2C power state.

Definition at line `263` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

i2c_control_mode

```
uint8_t sl_i2c_config_t::i2c_control_mode
```

I2C bus control configuration.

Definition at line `264` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

i2c_bus_speed

uint16_t sl_i2c_config_t::i2c_bus_speed

I2C bus speed.

Definition at line 265 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

i2c_slave_addr

uint32_t sl_i2c_config_t::i2c_slave_addr

I2C slave address.

Definition at line 266 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_spi_config_t

Public Attributes

uint8_t	spi_bit_width SPI bus width.
uint8_t	spi_mode SPI Mode(Master/Slave)
uint8_t	spi_power_state SPI bus power mode.
uint8_t	spi_cs_number Chip select number.
uint8_t	spi_cs_misc_mode SPI miscellaneous for chip select.
uint8_t	spi_sec_sel_sig SPI slave select signal definitions.
uint32_t	spi_baud SPI bus data transmission speed(clock)
uint64_t	spi_control_mode SPI bus phase and polarity.
uint64_t	spi_cs_mode SPI control slave select mode.

Public Attribute Documentation

spi_bit_width

> uint8_t sl_spi_config_t::spi_bit_width

SPI bus width.

Definition at line 273 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi_mode

> uint8_t sl_spi_config_t::spi_mode

SPI Mode(Master/Slave)

Definition at line 274 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi_power_state

uint8_t sl_spi_config_t::spi_power_state

SPI bus power mode.

Definition at line 275 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi_cs_number

uint8_t sl_spi_config_t::spi_cs_number

Chip select number.

Definition at line 276 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi_cs_misc_mode

uint8_t sl_spi_config_t::spi_cs_misc_mode

SPI miscellaneous for chip select.

Definition at line 277 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi_sec_sel_sig

uint8_t sl_spi_config_t::spi_sec_sel_sig

SPI slave select signal definitions.

Definition at line 278 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi_baud

uint32_t sl_spi_config_t::spi_baud

SPI bus data transmission speed(clock)

Definition at line 279 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi_control_mode

uint64_t sl_spi_config_t::spi_control_mode

SPI bus phase and polarity.

Definition at line 280 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

spi_cs_mode

uint64_t sl_spi_config_t::spi_cs_mode

SPI control slave select mode.

Definition at line 281 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_adc_config

Public Attributes

uint8_t	adc_init flag to know if adc is initialized, this is necessary to deinit
uint16_t	adc_data_ready flag to indicate data availability for all 16 channels
sl_adc_config_t	adc_cfg adc configuration
sl_adc_channel_config_t	adc_ch_cfg adc channel configuration

Public Attribute Documentation

adc_init

```
uint8_t sl_adc_config::adc_init
```

flag to know if adc is initialized, this is necessary to deinit

Definition at line `288` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

adc_data_ready

```
uint16_t sl_adc_config::adc_data_ready
```

flag to indicate data availability for all 16 channels

Definition at line `289` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

adc_cfg

```
sl_adc_config_t sl_adc_config::adc_cfg
```

adc configuration

Definition at line `290` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

adc_ch_cfg

```
sl_adc_channel_config_t sl_adc_config::adc_ch_cfg
```

adc channel configuration

Definition at line 291 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_gpio_config_t

Public Attributes

uint8_t	c
uint32_t	d

Public Attribute Documentation

c

```
uint8_t sl_gpio_config_t::c
```

Definition at line 298 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

d

```
uint32_t sl_gpio_config_t::d
```

Definition at line 299 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

sl_bus_intf_config_t

Public Attributes

sl_i2c_config_t	i2c_config I2C configuration structure.
sl_spi_config_t	spi_config SPI configuration structure.
sl_adc_cfg_t	adc_config ADC configuration structure.
sl_gpio_config_t	gpio_config GPIO configuration structure.

Public Attribute Documentation

i2c_config

```
sl_i2c_config_t sl_bus_intf_config_t::i2c_config
```

I2C configuration structure.

Definition at line `306` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

spi_config

```
sl_spi_config_t sl_bus_intf_config_t::spi_config
```

SPI configuration structure.

Definition at line `307` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

adc_config

```
sl_adc_cfg_t sl_bus_intf_config_t::adc_config
```

ADC configuration structure.

Definition at line `308` of file `components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h`

gpio_config

```
sl_gpio_config_t sl_bus_intf_config_t::gpio_config
```

GPIO configuration structure.

Definition at line 309 of file components/device/silabs/si91x/mcu/drivers/service/sensorhub/inc/sensor_hub.h

Sleep Timer

Sleep Timer

The sleep timer driver provides software timers, delays, timekeeping, and date functionalities using a low-frequency real-time clock peripheral.

Refer to the Sleep Timer Documentation for more information.

Input/Output Stream

Input/Output Stream

The input/output (I/O) stream service provides data streams for performing read and write operations on a physical communication interface.

Refer to the I/O Stream Documentation for more Information.

Non-volatile Memory

Non-volatile Memory

The non-volatile memory (NVM3) driver provides a means to store key-value pairs in flash memory.

Refer to the NVM3 Documentation for more information.

Overview

Overview

Provides the list of Crypto APIs.

APIs

APIs

This section provides a reference to the Crypto API including functions, data types and constants.

Modules

AES

AES

This section provides a reference to the AES Crypto API including functions, data types and constants.

Modules

Functions

Functions

This section provides a reference to the AES Crypto API Functions.

Functions

sl_status_t	sl_si91x_aes(sl_si91x_aes_config_t *config, uint8_t *output) This API is used to encrypt/decrypt the message according to the given configuration.

Function Documentation

sl_si91x_aes

```
sl_status_t sl_si91x_aes (sl_si91x_aes_config_t *config, uint8_t *output)
```

This API is used to encrypt/decrypt the message according to the given configuration.

Parameters

[in]	config	Configuration object of type sl_si91x_aes_config_t
[out]	output	Buffer to store the output.

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `118` of file `components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h`

Types

Types

This section provides a reference to the AES Crypto API types.

Modules

sl_si91x_aes_key_config_a0_t

sl_si91x_aes_key_config_b0_t

sl_si91x_aes_key_config_t

sl_si91x_aes_config_t

sl_si91x_aes_key_config_a0_t

Public Attributes

uint8_t *	key Pointer to the key.
uint16_t	key_length Length of the key.

Public Attribute Documentation

key

```
uint8_t* sl_si91x_aes_key_config_a0_t::key
```

Pointer to the key.

Definition at line 73 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

key_length

```
uint16_t sl_si91x_aes_key_config_a0_t::key_length
```

Length of the key.

Definition at line 74 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

sl_si91x_aes_key_config_b0_t

Public Attributes

sl_si91x_crypto_key_type_t	key_type Key type.
sl_si91x_aes_key_size_t	key_size Key size.
sl_si91x_crypto_key_slot_t	key_slot Key slot.
sl_si91x_crypto_wrap_mode_t	wrap_iv_mode Wrap mode.
uint8_t	wrap_iv IV used in SL_SI91X_AES_CBC and SL_SI91X_AES_CTR modes.
uint8_t	key_buffer Buffer to store the key.
uint32_t	reserved Reserved for future use.

Public Attribute Documentation

key_type

```
sl_si91x_crypto_key_type_t sl_si91x_aes_key_config_b0_t::key_type
```

Key type.

Definition at line 78 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

key_size

```
sl_si91x_aes_key_size_t sl_si91x_aes_key_config_b0_t::key_size
```

Key size.

Definition at line 79 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

key_slot

```
sl_si91x_crypto_key_slot_t sl_si91x_aes_key_config_b0_t::key_slot
```

Key slot.

Definition at line 80 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

wrap_iv_mode

```
sl_si91x_crypto_wrap_mode_t sl_si91x_aes_key_config_b0_t::wrap_iv_mode
```

Wrap mode.

Definition at line 81 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

wrap_iv

```
uint8_t sl_si91x_aes_key_config_b0_t::wrap_iv[SL_SI91X_IV_SIZE]
```

IV used in SL_SI91X_AES_CBC and SL_SI91X_AES_CTR modes.

Definition at line 82 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

key_buffer

```
uint8_t sl_si91x_aes_key_config_b0_t::key_buffer[SL_SI91X_KEY_BUFFER_SIZE]
```

Buffer to store the key.

Definition at line 83 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

reserved

```
uint32_t sl_si91x_aes_key_config_b0_t::reserved
```

Reserved for future use.

Definition at line 84 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

sl_si91x_aes_key_config_t

Public Attributes

sl_si91x_aes_key_config_a0_t	a0 Key configuration for non-B0 chip versions.
sl_si91x_aes_key_config_b0_t	b0 Key configuration for B0 chip versions.

Public Attribute Documentation

a0

> sl_si91x_aes_key_config_a0_t sl_si91x_aes_key_config_t::a0

Key configuration for non-B0 chip versions.

Definition at line 88 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

b0

> sl_si91x_aes_key_config_b0_t sl_si91x_aes_key_config_t::b0

Key configuration for B0 chip versions.

Definition at line 89 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

sl_si91x_aes_config_t

Public Attributes

sl_si91x_aes_mode_t	aes_mode AES Mode.
sl_si91x_aes_type_t	encrypt_decrypt Encryption or decryption.
const uint8_t *	msg Pointer to the input message.
uint16_t	msg_length Length of the message.
const uint8_t *	iv Pointer to the Initialization vector.
sl_si91x_aes_key_config_t	key_config Key configuration.

Public Attribute Documentation

aes_mode

```
sl_si91x_aes_mode_t sl_si91x_aes_config_t::aes_mode
```

AES Mode.

Definition at line 93 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

encrypt_decrypt

```
sl_si91x_aes_type_t sl_si91x_aes_config_t::encrypt_decrypt
```

Encryption or decryption.

Definition at line 94 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

msg

```
const uint8_t* sl_si91x_aes_config_t::msg
```

Pointer to the input message.

Definition at line 95 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

msg_length

```
uint16_t sl_si91x_aes_config_t::msg_length
```

Length of the message.

Definition at line `96` of file `components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h`

iv

```
const uint8_t* sl_si91x_aes_config_t::iv
```

Pointer to the Initialization vector.

Definition at line `97` of file `components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h`

key_config

```
sl_si91x_aes_key_config_t sl_si91x_aes_config_t::key_config
```

Key configuration.

Definition at line `98` of file `components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h`

Constants

Constants

This section provides a reference to the AES Crypto API constants.

Enumerations

enum sl_si91x_aes_mode_t {

SL_SI91X_AES_CBC = 1
SL_SI91X_AES_ECB
SL_SI91X_AES_CTR

}

enum sl_si91x_aes_type_t {

SL_SI91X_AES_ENCRYPT = 1
SL_SI91X_AES_DECRYPT

}

enum sl_si91x_aes_key_size_t {

SL_SI91X_AES_KEY_SIZE_128 = 16
SL_SI91X_AES_KEY_SIZE_192 = 24
SL_SI91X_AES_KEY_SIZE_256 = 32

}

Macros

#define SL_SI91X_AES_BLOCK_SIZE 16

AES BLOCK SIZE of 16 bytes or 128 bits.

Enumeration Documentation

sl_si91x_aes_mode_t

```
sl_si91x_aes_mode_t
```

Enumerator	
SL_SI91X_AES_CBC	AES CBC mode.
SL_SI91X_AES_ECB	AES ECB mode.
SL_SI91X_AES_CTR	AES CTR mode.

Definition at line 45 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

sl_si91x_aes_type_t

```
sl_si91x_aes_type_t
```

Enumerator	
SL_SI91X_AES_ENCRYPT	AES Encryption.
SL_SI91X_AES_DECRYPT	AES Decryption.

Definition at line 51 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

sl_si91x_aes_key_size_t

```
sl_si91x_aes_key_size_t
```

Enumerator	
SL_SI91X_AES_KEY_SIZE_128	key size of 128 bits
SL_SI91X_AES_KEY_SIZE_192	key size of 192 bits
SL_SI91X_AES_KEY_SIZE_256	key size of 256 bits

Definition at line 56 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

Macro Definition Documentation

SL_SI91X_AES_BLOCK_SIZE

```
#define SL_SI91X_AES_BLOCK_SIZE
```

Value:

```
16
```

AES BLOCK SIZE of 16 bytes or 128 bits.

Definition at line 43 of file components/device/silabs/si91x/wireless/crypto/aes/inc/sl_si91x_aes.h

Attestation

Attestation

This section provides a reference to the Crypto API functions for device attestation.

Functions

sl_status_t sl_si91x_attestation_get_token(uint8_t *token, uint16_t length, uint32_t *nonce)
\Process the nonce for attestation token

Function Documentation

sl_si91x_attestation_get_token

> sl_status_t sl_si91x_attestation_get_token (uint8_t *token, uint16_t length, uint32_t *nonce)

\Process the nonce for attestation token

Parameters

[in]	token	
[in]	length	
[out]	nonce	

Returns

- return sl_status code

Definition at line 52 of file components/device/silabs/si91x/wireless/crypto/attestation/inc/sl_si91x_attestation.h

ECDH

ECDH

This section provides a reference to the ECDH Crypto API functions, data types and constants.

Modules

Functions

Functions

This section provides a reference to the ECDH Crypto API functions.

Functions

sl_status_t	sl_si91x_ecdh_point_addition(sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint8_t *tx, uint8_t *ty, uint8_t *tz, uint8_t *rx, uint8_t *ry, uint8_t *rz) Compute the ECDH point addition vector.
sl_status_t	sl_si91x_ecdh_point_subtraction(sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint8_t *tx, uint8_t *ty, uint8_t *tz, uint8_t *rx, uint8_t *ry, uint8_t *rz) Compute the ECDH point subtraction vector.
sl_status_t	sl_si91x_ecdh_point_multiplication(sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *d, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint32_t affinity, uint8_t *rx, uint8_t *ry, uint8_t *rz, uint8_t reverse) Compute the ECDH point multiplication vector.
sl_status_t	sl_si91x_ecdh_point_double(sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint8_t *rx, uint8_t *ry, uint8_t *rz) Compute the ECDH point double vector.
sl_status_t	sl_si91x_ecdh_point_affine(sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint8_t *rx, uint8_t *ry, uint8_t *rz) Compute the ECDH point affinity vector.

Function Documentation

sl_si91x_ecdh_point_addition

> sl_status_t sl_si91x_ecdh_point_addition (sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint8_t *tx, uint8_t *ty, uint8_t *tz, uint8_t *rx, uint8_t *ry, uint8_t *rz)

Compute the ECDH point addition vector.

Parameters

[in]	ecdh_mode	ECDH mode of type sl_si91x_ecdh_mode_t
[in]	sx	Pointer to x coordinate of the point1 that needs to be added
[in]	sy	Pointer to y coordinate of the point1 that needs to be added
[in]	sz	Pointer to z coordinate of the point1 that needs to be added
[in]	tx	Pointer to x coordinate of the point2 that needs to be added
[in]	ty	Pointer to y coordinate of the point2 that needs to be added
[in]	tz	Pointer to z coordinate of the point2 that needs to be added
[out]	rx	Pointer to x coordinate of the result point
[out]	ry	Pointer to y coordinate of the result point
[out]	rz	Pointer to z coordinate of the result point

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 102 of file components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h

sl_si91x_ecdh_point_subtraction

> sl_status_t sl_si91x_ecdh_point_subtraction (sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint8_t *tx, uint8_t *ty, uint8_t *tz, uint8_t *rx, uint8_t *ry, uint8_t *rz)

Compute the ECDH point subtraction vector.

Parameters

[in]	ecdh_mode	ECDH mode of type sl_si91x_ecdh_mode_t
[in]	sx	Pointer to x coordinate of the point1 that needs to be subtracted
[in]	sy	Pointer to y coordinate of the point1 that needs to be subtracted
[in]	sz	Pointer to z coordinate of the point1 that needs to be subtracted
[in]	tx	Pointer to x coordinate of the point2 that needs to be subtracted
[in]	ty	Pointer to y coordinate of the point2 that needs to be subtracted
[in]	tz	Pointer to z coordinate of the point2 that needs to be subtracted
[out]	rx	Pointer to x coordinate of the result point
[out]	ry	Pointer to y coordinate of the result point
[out]	rz	Pointer to z coordinate of the result point

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 138 of file components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h

sl_si91x_ecdh_point_multiplication

> sl_status_t sl_si91x_ecdh_point_multiplication (sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *d, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint32_t affinity, uint8_t *rx, uint8_t *ry, uint8_t *rz, uint8_t reverse)

Compute the ECDH point multiplication vector.

Parameters

[in]	ecdh_mode	ECDH mode of type sl_si91x_ecdh_mode_t
[in]	d	Pointer to scalar value that needs to be multiplied
[in]	sx	Pointer to x coordinate of the point to be multiplied with scalar ‘d’
[in]	sy	Pointer to y coordinate of the point to be multiplied with scalar ‘d’
[in]	sz	Pointer to z coordinate of the point to be multiplied with scalar ‘d’
[in]	affinity	• 0 : no affinity • 1 : affinity on input • 2 : affinity on output • 3 : affinity on both input and output
[out]	rx	Pointer to x coordinate of the result point
[out]	ry	Pointer to y coordinate of the result point

[out]	rz	Pointer to z coordinate of the result point
[in]	reverse	Setting this will perform reverse_8 on the inputs and outputs.

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 178 of file components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h

sl_si91x_ecdh_point_double

> sl_status_t sl_si91x_ecdh_point_double (sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint8_t *rx, uint8_t *ry, uint8_t *rz)

Compute the ECDH point double vector.

Parameters

[in]	ecdh_mode	ECDH mode of type sl_si91x_ecdh_mode_t
[in]	sx	Pointer to x coordinate of the point1 that needs to be doubled
[in]	sy	Pointer to y coordinate of the point1 that needs to be doubled
[in]	sz	Pointer to z coordinate of the point1 that needs to be doubled
[out]	rx	Pointer to x coordinate of the result point
[out]	ry	Pointer to y coordinate of the result point
[out]	rz	Pointer to z coordinate of the result point

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 209 of file components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h

sl_si91x_ecdh_point_affine

> sl_status_t sl_si91x_ecdh_point_affine (sl_si91x_ecdh_mode_t ecdh_mode, uint8_t *sx, uint8_t *sy, uint8_t *sz, uint8_t *rx, uint8_t *ry, uint8_t *rz)

Compute the ECDH point affinity vector.

Parameters

[in]	ecdh_mode	ECDH mode of type sl_si91x_ecdh_mode_t
[in]	sx	Pointer to x coordinate of the point1
[in]	sy	Pointer to y coordinate of the point1
[in]	sz	Pointer to z coordinate of the point1
[out]	rx	Pointer to x coordinate of the result point
[out]	ry	Pointer to y coordinate of the result point
[out]	rz	Pointer to z coordinate of the result point

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 236 of file components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h

Constants

@defgroup CRYPTO_ECDH_TYPES Types
@ingroup CRYPTO_ECDH

This section provides a reference to the ECDH Crypto API types.
‍/

/*!

This section provides a reference to the ECDH Crypto API constants.

Enumerations

enum sl_si91x_ecdh_mode_t {

SL_SI91X_ECDH_192 = 1
SL_SI91X_ECDH_224 = 2
SL_SI91X_ECDH_256 = 4

}

enum sl_si91x_ecdh_sub_mode_t {

SL_SI91X_ECDH_MUL = 1
SL_SI91X_ECDH_ADD = 2
SL_SI91X_ECDH_SUB = 3
SL_SI91X_ECDH_DOUBLE = 4
SL_SI91X_ECDH_AFFINITY = 5

}

enum sl_si91x_ecdh_vector_size_t {

SL_SI91X_ECDH_VECTOR_SIZE_192 = 24
SL_SI91X_ECDH_VECTOR_SIZE_224 = 28
SL_SI91X_ECDH_VECTOR_SIZE_256 = 32

}

enum sl_si91x_ecdh_curve_type_t {

SL_SI91X_ECDH_CURVE_P = 0
SL_SI91X_ECDH_CURVE_K = 1
SL_SI91X_ECDH_CURVE_B = 2

}

Macros

#define SL_SI91X_ECDH_MAX_VECTOR_SIZE 32
Maximum size of an ECDH vector.

Enumeration Documentation

sl_si91x_ecdh_mode_t

> sl_si91x_ecdh_mode_t

Enumerator	
SL_SI91X_ECDH_192	ECDH 192 mode.
SL_SI91X_ECDH_224	ECDH 224 mode.
SL_SI91X_ECDH_256	ECDH 256 mode.

Definition at line `44` of file `components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h`

sl_si91x_ecdh_sub_mode_t

> sl_si91x_ecdh_sub_mode_t

Enumerator	
SL_SI91X_ECDH_MUL	ECDH multiplication mode.
SL_SI91X_ECDH_ADD	ECDH addition mode.
SL_SI91X_ECDH_SUB	ECDH subtraction mode.
SL_SI91X_ECDH_DOUBLE	ECDH double mode.
SL_SI91X_ECDH_AFFINITY	ECDH affinity mode.

Definition at line `50` of file `components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h`

sl_si91x_ecdh_vector_size_t

> sl_si91x_ecdh_vector_size_t

Enumerator	
SL_SI91X_ECDH_VECTOR_SIZE_192	192 bits or 24 bytes vector size
SL_SI91X_ECDH_VECTOR_SIZE_224	224 bits or 28 bytes vector size
SL_SI91X_ECDH_VECTOR_SIZE_256	256 bits or 32 bytes vector size

Definition at line `58` of file `components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h`

sl_si91x_ecdh_curve_type_t

> sl_si91x_ecdh_curve_type_t

Enumerator	
SL_SI91X_ECDH_CURVE_P	Prime Field Curves.
SL_SI91X_ECDH_CURVE_K	Binary Field Curves.
SL_SI91X_ECDH_CURVE_B	Edwards Curves.

Definition at line `64` of file `components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h`

Macro Definition Documentation

SL_SI91X_ECDH_MAX_VECTOR_SIZE

```
#define SL_SI91X_ECDH_MAX_VECTOR_SIZE
```

Value:

```
32
```

Maximum size of an ECDH vector.

Definition at line 42 of file components/device/silabs/si91x/wireless/crypto/ecdh/inc/sl_si91x_ecdh.h

CCM

CCM

This section provides a reference to the CCM Crypto API including functions, data types and constants.

Modules

Functions

Types

Constants

Functions

Functions

This section provides a reference to the CCM Crypto API functions.

Functions

sl_status_t	sl_si91x_ccm(sl_si91x_ccm_config_t *config, uint8_t *output) This API is used to encrypt/decrypt the message according to the given configuration.

Function Documentation

sl_si91x_ccm

```
sl_status_t sl_si91x_ccm (sl_si91x_ccm_config_t *config, uint8_t *output)
```

This API is used to encrypt/decrypt the message according to the given configuration.

Parameters

[in]	config	Configuration object of type sl_si91x_ccm_config_t
[out]	output	Buffer to store the output.

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `114` of file `components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h`

Types

Types

This section provides a reference to the CCM Crypto API types.

Modules

sl_si91x_ccm_key_config_a0_t

sl_si91x_ccm_key_config_b0_t

sl_si91x_ccm_key_config_t

sl_si91x_ccm_config_t

sl_si91x_ccm_key_config_a0_t

Public Attributes

uint8_t *	key Pointer to the key.
uint16_t	key_length Length of the key.

Public Attribute Documentation

key

```
uint8_t* sl_si91x_ccm_key_config_a0_t::key
```

Pointer to the key.

Definition at line 65 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

key_length

```
uint16_t sl_si91x_ccm_key_config_a0_t::key_length
```

Length of the key.

Definition at line 66 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

sl_si91x_ccm_key_config_b0_t

Public Attributes

sl_si91x_crypto_key_type_t	key_type Key type.
sl_si91x_ccm_key_size_t	key_size Key size.
sl_si91x_crypto_key_slot_t	key_slot Key slot.
sl_si91x_crypto_wrap_mode_t	wrap_iv_mode Wrap mode.
uint8_t	wrap_iv IV used in SL_SI91X_AES_CBC and SL_SI91X_AES_CTR modes.
uint8_t	key_buffer Buffer to store the key.
uint32_t	reserved Reserved for future use.

Public Attribute Documentation

key_type

```
sl_si91x_crypto_key_type_t sl_si91x_ccm_key_config_b0_t::key_type
```

Key type.

Definition at line 70 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

key_size

```
sl_si91x_ccm_key_size_t sl_si91x_ccm_key_config_b0_t::key_size
```

Key size.

Definition at line 71 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

key_slot

```
sl_si91x_crypto_key_slot_t sl_si91x_ccm_key_config_b0_t::key_slot
```

Key slot.

Definition at line 72 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

wrap_iv_mode

```
sl_si91x_crypto_wrap_mode_t sl_si91x_ccm_key_config_b0_t::wrap_iv_mode
```

Wrap mode.

Definition at line 73 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

wrap_iv

```
uint8_t sl_si91x_ccm_key_config_b0_t::wrap_iv[SL_SI91X_IV_SIZE]
```

IV used in SL_SI91X_AES_CBC and SL_SI91X_AES_CTR modes.

Definition at line 74 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

key_buffer

```
uint8_t sl_si91x_ccm_key_config_b0_t::key_buffer[SL_SI91X_KEY_BUFFER_SIZE]
```

Buffer to store the key.

Definition at line 75 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

reserved

```
uint32_t sl_si91x_ccm_key_config_b0_t::reserved
```

Reserved for future use.

Definition at line 76 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

sl_si91x_ccm_key_config_t

Public Attributes

sl_si91x_ccm_key_config_a0_t	a0 Key configuration for non-B0 chip versions.
sl_si91x_ccm_key_config_b0_t	b0 Key configuration for B0 chip versions.

Public Attribute Documentation

a0

> sl_si91x_ccm_key_config_a0_t sl_si91x_ccm_key_config_t::a0

Key configuration for non-B0 chip versions.

Definition at line 80 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

b0

> sl_si91x_ccm_key_config_b0_t sl_si91x_ccm_key_config_t::b0

Key configuration for B0 chip versions.

Definition at line 81 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

sl_si91x_ccm_config_t

Public Attributes

sl_si91x_ccm_type_t	encrypt_decrypt Encryption or decryption.
const uint8_t *	msg Pointer to the input message.
uint16_t	msg_length Length of the message.
const uint8_t *	nonce Pointer to the Initialization vector.
const uint8_t *	tag Pointer to the tag.
const uint8_t *	ad Pointer to the additional data.
uint16_t	nonce_length Length of the Initialization vector.
uint16_t	tag_length Length of the tag.
uint16_t	ad_length Length of the additional data.
sl_si91x_ccm_key_config_t	key_config Key configuration.

Public Attribute Documentation

encrypt_decrypt

```
sl_si91x_ccm_type_t sl_si91x_ccm_config_t::encrypt_decrypt
```

Encryption or decryption.

Definition at line `85` of file `components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h`

msg

```
const uint8_t* sl_si91x_ccm_config_t::msg
```

Pointer to the input message.

Definition at line `86` of file `components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h`

msg_length

```
uint16_t sl_si91x_ccm_config_t::msg_length
```

Length of the message.

Definition at line 87 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

nonce

```
const uint8_t* sl_si91x_ccm_config_t::nonce
```

Pointer to the Initialization vector.

Definition at line 88 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

tag

```
const uint8_t* sl_si91x_ccm_config_t::tag
```

Pointer to the tag.

Definition at line 89 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

ad

```
const uint8_t* sl_si91x_ccm_config_t::ad
```

Pointer to the additional data.

Definition at line 90 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

nonce_length

```
uint16_t sl_si91x_ccm_config_t::nonce_length
```

Length of the Initialization vector.

Definition at line 91 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

tag_length

```
uint16_t sl_si91x_ccm_config_t::tag_length
```

Length of the tag.

Definition at line 92 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

ad_length

> uint16_t sl_si91x_ccm_config_t::ad_length

Length of the additional data.

Definition at line 93 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

key_config

> sl_si91x_ccm_key_config_t sl_si91x_ccm_config_t::key_config

Key configuration.

Definition at line 94 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

Constants

Constants

This section provides a reference to the CCM Crypto API constants.

Enumerations

enum sl_si91x_ccm_type_t {
SL_SI91X_CCM_ENCRYPT = 0
SL_SI91X_CCM_DECRYPT
}

enum sl_si91x_ccm_key_size_t {
SL_SI91X_CCM_KEY_SIZE_128 = 16
SL_SI91X_CCM_KEY_SIZE_192 = 24
SL_SI91X_CCM_KEY_SIZE_256 = 32
}

Enumeration Documentation

sl_si91x_ccm_type_t

> sl_si91x_ccm_type_t

Enumerator	
SL_SI91X_CCM_ENCRYPT	CCM Encryption.
SL_SI91X_CCM_DECRYPT	CCM Decryption.

Definition at line 43 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

sl_si91x_ccm_key_size_t

> sl_si91x_ccm_key_size_t

Enumerator	
SL_SI91X_CCM_KEY_SIZE_128	key size of 128 bits
SL_SI91X_CCM_KEY_SIZE_192	key size of 192 bits
SL_SI91X_CCM_KEY_SIZE_256	key size of 256 bits

Definition at line 48 of file components/device/silabs/si91x/wireless/crypto/ccm/inc/sl_si91x_ccm.h

ChaChaPoly

ChaChaPoly

This section provides a reference to the CHACHAPOLY Crypto API including functions, data types and constants.

Modules

Functions

Functions

This section provides a reference to the CHACHAPOLY Crypto API functions.

Functions

sl_status_t sl_si91x_chachapoly(sl_si91x_chachapoly_config_t *config, uint8_t *output)
This API is used to encrypt/decrypt the message according to the given configuration.

Function Documentation

sl_si91x_chachapoly

```
sl_status_t sl_si91x_chachapoly (sl_si91x_chachapoly_config_t *config, uint8_t *output)
```

This API is used to encrypt/decrypt the message according to the given configuration.

Parameters

[in]	config	Configuration object of type sl_si91x_chachapoly_config_t
[out]	output	Buffer to store the output.

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `124` of file `components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h`

Types

Types

This section provides a reference to the CHACHAPOLY Crypto API types.

Modules

sl_si91x_chachapoly_key_config_a0_t

sl_si91x_chachapoly_key_config_b0_t

sl_si91x_chachapoly_key_config_t

sl_si91x_chachapoly_config_t

sl_si91x_chachapoly_key_config_a0_t

Public Attributes

uint8_t	key_chacha
uint8_t	keyr_in
uint8_t	keys_in

Public Attribute Documentation

key_chacha

```
uint8_t sl_si91x_chachapoly_key_config_a0_t::key_chacha[SL_SI91X_KEY_BUFFER_SIZE]
```

Definition at line `75` of file `components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h`

keyr_in

```
uint8_t sl_si91x_chachapoly_key_config_a0_t::keyr_in[16]
```

Definition at line `76` of file `components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h`

keys_in

```
uint8_t sl_si91x_chachapoly_key_config_a0_t::keys_in[16]
```

Definition at line `77` of file `components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h`

sl_si91x_chachapoly_key_config_b0_t

Public Attributes

sl_si91x_crypto_key_type_t	key_type Key type.
sl_si91x_chachapoly_key_size_t	key_size Key size.
sl_si91x_crypto_key_slot_t	key_slot Key slot.
sl_si91x_crypto_wrap_mode_t	wrap_iv_mode Wrap mode.
uint8_t	wrap_iv IV used in SL_SI91X_AES_CBC and SL_SI91X_AES_CTR modes.
uint8_t	key_buffer Buffer to store the key.
uint32_t	reserved Reserved for future use.

Public Attribute Documentation

key_type

```
sl_si91x_crypto_key_type_t sl_si91x_chachapoly_key_config_b0_t::key_type
```

Key type.

Definition at line 81 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

key_size

```
sl_si91x_chachapoly_key_size_t sl_si91x_chachapoly_key_config_b0_t::key_size
```

Key size.

Definition at line 82 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

key_slot

```
sl_si91x_crypto_key_slot_t sl_si91x_chachapoly_key_config_b0_t::key_slot
```

Key slot.

Definition at line 83 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

wrap_iv_mode

> sl_si91x_crypto_wrap_mode_t sl_si91x_chachapoly_key_config_b0_t::wrap_iv_mode

Wrap mode.

Definition at line 84 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

wrap_iv

> uint8_t sl_si91x_chachapoly_key_config_b0_t::wrap_iv[SL_SI91X_IV_SIZE]

IV used in SL_SI91X_AES_CBC and SL_SI91X_AES_CTR modes.

Definition at line 85 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

key_buffer

> uint8_t sl_si91x_chachapoly_key_config_b0_t::key_buffer[SL_SI91X_KEY_BUFFER_SIZE]

Buffer to store the key.

Definition at line 86 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

reserved

> uint32_t sl_si91x_chachapoly_key_config_b0_t::reserved

Reserved for future use.

Definition at line 87 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

sl_si91x_chachapoly_key_config_t

Public Attributes

sl_si91x_chachapoly_key_config_a0_t	a0 Key configuration for non-B0 chip versions.
sl_si91x_chachapoly_key_config_b0_t	b0 Key configuration for B0 chip versions.

Public Attribute Documentation

a0

> sl_si91x_chachapoly_key_config_a0_t sl_si91x_chachapoly_key_config_t::a0

Key configuration for non-B0 chip versions.

Definition at line `91` of file `components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h`

b0

> sl_si91x_chachapoly_key_config_b0_t sl_si91x_chachapoly_key_config_t::b0

Key configuration for B0 chip versions.

Definition at line `92` of file `components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h`

sl_si91x_chachapoly_config_t

Public Attributes

sl_si91x_chachapoly_type_t	encrypt_decrypt Encryption or decryption.
sl_si91x_chachapoly_mode_t	chachapoly_mode
sl_si91x_chachapoly_dma_use_t	dma_use DMA Disable or Enable.
const uint8_t *	msg Pointer to the input message.
uint16_t	msg_length Length of the message.
const uint8_t *	nonce Pointer to the Initialization vector.
const uint8_t *	ad Pointer to the additional data.
uint16_t	ad_length Length of the additional data.
sl_si91x_chachapoly_key_config_t	key_config Key configuration.

Public Attribute Documentation

encrypt_decrypt

> sl_si91x_chachapoly_type_t sl_si91x_chachapoly_config_t::encrypt_decrypt

Encryption or decryption.

Definition at line 96 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

chachapoly_mode

> sl_si91x_chachapoly_mode_t sl_si91x_chachapoly_config_t::chachapoly_mode

Definition at line 97 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

dma_use

```
sl_si91x_chachapoly_dma_use_t sl_si91x_chachapoly_config_t::dma_use
```

DMA Disable or Enable.

Definition at line 98 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

msg

```
const uint8_t* sl_si91x_chachapoly_config_t::msg
```

Pointer to the input message.

Definition at line 99 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

msg_length

```
uint16_t sl_si91x_chachapoly_config_t::msg_length
```

Length of the message.

Definition at line 100 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

nonce

```
const uint8_t* sl_si91x_chachapoly_config_t::nonce
```

Pointer to the Initialization vector.

Definition at line 101 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

ad

```
const uint8_t* sl_si91x_chachapoly_config_t::ad
```

Pointer to the additional data.

Definition at line 102 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

ad_length

```
uint16_t sl_si91x_chachapoly_config_t::ad_length
```

Length of the additional data.

Definition at line 103 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

key_config

sl_si91x_chachapoly_key_config_t sl_si91x_chachapoly_config_t::key_config

Key configuration.

Definition at line 104 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

Constants

Constants

This section provides a reference to the CHACHAPOLY Crypto API constants.

Enumerations

enum sl_si91x_chachapoly_type_t {
SL_SI91X_CHACHAPOLY_ENCRYPT = 0
SL_SI91X_CHACHAPOLY_DECRYPT
}

enum sl_si91x_chachapoly_mode_t {
SL_SI91X_CHACHA20POLY1305_MODE = 0
SL_SI91X_CHACHA20_MODE
SL_SI91X_CHACHAPOLY_POLY1305_KEYR_KEYS_MODE
SL_SI91X_POLY1305_MODE
}

enum sl_si91x_chachapoly_dma_use_t {
SL_SI91X_DMA_DISABLE = 0
SL_SI91X_DMA_ENABLE
}

enum sl_si91x_chachapoly_key_size_t {
SL_SI91X_CHACHAPOLY_KEY_SIZE_256 = 32
}

Enumeration Documentation

sl_si91x_chachapoly_type_t

> sl_si91x_chachapoly_type_t

Enumerator	
SL_SI91X_CHACHAPOLY_ENCRYPT	CHACHAPOLY Encryption.
SL_SI91X_CHACHAPOLY_DECRYPT	CHACHAPOLY Decryption.

Definition at line 43 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

sl_si91x_chachapoly_mode_t

> sl_si91x_chachapoly_mode_t

Enumerator	
SL_SI91X_CHACHA20POLY1305_MODE	CHACHA20POLY1305 mode.

SL_SI91X_CHACHA20_MODE	CHACHA20 Mode.
SL_SI91X_CHACHAPOLY_POLY1305_KEYR_KEYS_MODE	
SL_SI91X_POLY1305_MODE	

Definition at line 48 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

sl_si91x_chachapoly_dma_use_t

> sl_si91x_chachapoly_dma_use_t

Enumerator

SL_SI91X_DMA_DISABLE	Disable DMA.
SL_SI91X_DMA_ENABLE	Enable DMA.

Definition at line 55 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

sl_si91x_chachapoly_key_size_t

> sl_si91x_chachapoly_key_size_t

Enumerator

SL_SI91X_CHACHAPOLY_KEY_SIZE_256	key size of 256 bits

Definition at line 60 of file components/device/silabs/si91x/wireless/crypto/chachapoly/inc/sl_si91x_chachapoly.h

GCM

GCM

This section provides a reference to the GCM Crypto API including functions, data types and constants.

Modules

Functions

Functions

This section provides a reference to the GCM Crypto API functions.

Functions

sl_status_t	sl_si91x_gcm(sl_si91x_gcm_config_t *config, uint8_t *output) This API is used to encrypt/decrypt the message according to the given configuration.

Function Documentation

sl_si91x_gcm

```
sl_status_t sl_si91x_gcm (sl_si91x_gcm_config_t *config, uint8_t *output)
```

This API is used to encrypt/decrypt the message according to the given configuration.

Parameters

[in]	config	Configuration object of type sl_si91x_gcm_config_t
[out]	output	Buffer to store the output.

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `126` of file `components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h`

Types

Types

This section provides a reference to the GCM Crypto API types.

Modules

sl_si91x_gcm_key_config_a0_t

sl_si91x_gcm_key_config_b0_t

sl_si91x_gcm_key_config_t

sl_si91x_gcm_config_t

sl_si91x_gcm_key_config_a0_t

Public Attributes

uint8_t *	key Pointer to the key.
uint16_t	key_length Length of the key.

Public Attribute Documentation

key

```
uint8_t* sl_si91x_gcm_key_config_a0_t::key
```

Pointer to the key.

Definition at line 75 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

key_length

```
uint16_t sl_si91x_gcm_key_config_a0_t::key_length
```

Length of the key.

Definition at line 76 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

sl_si91x_gcm_key_config_b0_t

Public Attributes

sl_si91x_crypto_key_type_t	key_type Key type.
sl_si91x_gcm_key_size_t	key_size Key size.
sl_si91x_crypto_key_slot_t	key_slot Key slot.
sl_si91x_crypto_wrap_mode_t	wrap_iv_mode Wrap mode.
uint8_t	wrap_iv IV used in SL_SI91X_AES_CBC and SL_SI91X_AES_CTR modes.
uint8_t	key_buffer Buffer to store the key.
uint32_t	reserved Reserved for future use.

Public Attribute Documentation

key_type

```
sl_si91x_crypto_key_type_t sl_si91x_gcm_key_config_b0_t::key_type
```

Key type.

Definition at line 80 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

key_size

```
sl_si91x_gcm_key_size_t sl_si91x_gcm_key_config_b0_t::key_size
```

Key size.

Definition at line 81 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

key_slot

```
sl_si91x_crypto_key_slot_t sl_si91x_gcm_key_config_b0_t::key_slot
```

Key slot.

Definition at line 82 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

wrap_iv_mode

```
sl_si91x_crypto_wrap_mode_t sl_si91x_gcm_key_config_b0_t::wrap_iv_mode
```

Wrap mode.

Definition at line 83 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

wrap_iv

```
uint8_t sl_si91x_gcm_key_config_b0_t::wrap_iv[SL_SI91X_IV_SIZE]
```

IV used in SL_SI91X_AES_CBC and SL_SI91X_AES_CTR modes.

Definition at line 84 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

key_buffer

```
uint8_t sl_si91x_gcm_key_config_b0_t::key_buffer[SL_SI91X_KEY_BUFFER_SIZE]
```

Buffer to store the key.

Definition at line 85 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

reserved

```
uint32_t sl_si91x_gcm_key_config_b0_t::reserved
```

Reserved for future use.

Definition at line 86 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

sl_si91x_gcm_key_config_t

Public Attributes

sl_si91x_gcm_key_config_a0_t	a0 Key configuration for non-B0 chip versions.
sl_si91x_gcm_key_config_b0_t	b0 Key configuration for B0 chip versions.

Public Attribute Documentation

a0

```
sl_si91x_gcm_key_config_a0_t sl_si91x_gcm_key_config_t::a0
```

Key configuration for non-B0 chip versions.

Definition at line 90 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

b0

```
sl_si91x_gcm_key_config_b0_t sl_si91x_gcm_key_config_t::b0
```

Key configuration for B0 chip versions.

Definition at line 91 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

sl_si91x_gcm_config_t

Public Attributes

sl_si91x_gcm_type_t	encrypt_decrypt Encryption or decryption.
sl_si91x_gcm_dma_use_t	dma_use DMA Disable or Enable.
const uint8_t *	msg Pointer to the input message.
uint16_t	msg_length Length of the message.
const uint8_t *	nonce Pointer to the Initialization vector.
const uint8_t *	ad Pointer to the additional data.
uint16_t	nonce_length Length of the Initialization vector.
uint16_t	ad_length Length of the additional data.
sl_si91x_gcm_key_config_t	key_config Key configuration.

Public Attribute Documentation

encrypt_decrypt

```
sl_si91x_gcm_type_t sl_si91x_gcm_config_t::encrypt_decrypt
```

Encryption or decryption.

Definition at line `95` of file `components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h`

dma_use

```
sl_si91x_gcm_dma_use_t sl_si91x_gcm_config_t::dma_use
```

DMA Disable or Enable.

Definition at line `99` of file `components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h`

msg

```
const uint8_t* sl_si91x_gcm_config_t::msg
```

Pointer to the input message.

Definition at line 100 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

msg_length

```
uint16_t sl_si91x_gcm_config_t::msg_length
```

Length of the message.

Definition at line 101 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

nonce

```
const uint8_t* sl_si91x_gcm_config_t::nonce
```

Pointer to the Initialization vector.

Definition at line 102 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

ad

```
const uint8_t* sl_si91x_gcm_config_t::ad
```

Pointer to the additional data.

Definition at line 103 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

nonce_length

```
uint16_t sl_si91x_gcm_config_t::nonce_length
```

Length of the Initialization vector.

Definition at line 104 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

ad_length

```
uint16_t sl_si91x_gcm_config_t::ad_length
```

Length of the additional data.

Definition at line 105 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

key_config

sl_si91x_gcm_key_config_t sl_si91x_gcm_config_t::key_config

Key configuration.

Definition at line 106 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

Constants

Constants

This section provides a reference to the GCM Crypto API constants.

Enumerations

enum sl_si91x_gcm_type_t {
  SL_SI91X_GCM_ENCRYPT = 0
  SL_SI91X_GCM_DECRYPT
}

enum sl_si91x_gcm_mode_t {
  SL_SI91X_GCM_MODE = 0
  SL_SI91X_CMAC_MODE
}

enum sl_si91x_gcm_dma_use_t {
  SL_SI91X_DMA_DISABLE = 0
  SL_SI91X_DMA_ENABLE
}

enum sl_si91x_gcm_key_size_t {
  SL_SI91X_GCM_KEY_SIZE_128 = 16
  SL_SI91X_GCM_KEY_SIZE_192 = 24
  SL_SI91X_GCM_KEY_SIZE_256 = 32
}

Enumeration Documentation

sl_si91x_gcm_type_t

> sl_si91x_gcm_type_t

Enumerator	
SL_SI91X_GCM_ENCRYPT	GCM Encryption.
SL_SI91X_GCM_DECRYPT	GCM Decryption.

Definition at line 43 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

sl_si91x_gcm_mode_t

> sl_si91x_gcm_mode_t

Enumerator	
SL_SI91X_GCM_MODE	GCM mode.

SL_SI91X_CMAC_MODE	CMAC Mode.

Definition at line 48 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

sl_si91x_gcm_dma_use_t

> sl_si91x_gcm_dma_use_t

Enumerator

SL_SI91X_DMA_DISABLE	Disable DMA.
SL_SI91X_DMA_ENABLE	Enable DMA.

Definition at line 53 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

sl_si91x_gcm_key_size_t

> sl_si91x_gcm_key_size_t

Enumerator

SL_SI91X_GCM_KEY_SIZE_128	key size of 128 bits
SL_SI91X_GCM_KEY_SIZE_192	key size of 192 bits
SL_SI91X_GCM_KEY_SIZE_256	key size of 256 bits

Definition at line 58 of file components/device/silabs/si91x/wireless/crypto/gcm/inc/sl_si91x_gcm.h

HMAC

HMAC

This section provides a reference to the HMAC Crypto API including functions, data types and constants.

Modules

Functions

Types

Constants

Functions

Functions

This section provides a reference to the HMAC Crypto API functions.

Functions

sl_status_t	sl_si91x_hmac(sl_si91x_hmac_config_t *config, uint8_t *output) This API will provide the HMAC output for the given configuration.

Function Documentation

sl_si91x_hmac

```
sl_status_t sl_si91x_hmac (sl_si91x_hmac_config_t *config, uint8_t *output)
```

This API will provide the HMAC output for the given configuration.

Parameters

[in]	config	Configuration object of type sl_si91x_hmac_config_t
[out]	output	Buffer to store the output.

This is a blocking API. **Returns**

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `110` of file `components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h`

Types

Types

This section provides a reference to the HMAC Crypto API types.

Modules

sl_si91x_hmac_key_config_A0_t

sl_si91x_hmac_key_config_B0_t

sl_si91x_hmac_key_config_t

sl_si91x_hmac_config_t

sl_si91x_hmac_key_config_A0_t

Public Attributes

uint8_t *	key Pointer to the key.
uint32_t	key_length Length of the key.

Public Attribute Documentation

key

```
uint8_t* sl_si91x_hmac_key_config_A0_t::key
```

Pointer to the key.

Definition at line 68 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

key_length

```
uint32_t sl_si91x_hmac_key_config_A0_t::key_length
```

Length of the key.

Definition at line 69 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

sl_si91x_hmac_key_config_B0_t

Public Attributes

sl_si91x_crypto_key_type_t	key_type Key type.
uint32_t	key_size Key size.
sl_si91x_crypto_key_slot_t	key_slot Key slot.
sl_si91x_crypto_wrap_mode_t	wrap_iv_mode Wrap mode.
uint8_t	wrap_iv Wrap IV.
uint8_t *	key Pointer to the key.
uint32_t	reserved Reserved for future use.

Public Attribute Documentation

key_type

```
sl_si91x_crypto_key_type_t sl_si91x_hmac_key_config_B0_t::key_type
```

Key type.

Definition at line 73 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

key_size

```
uint32_t sl_si91x_hmac_key_config_B0_t::key_size
```

Key size.

Definition at line 74 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

key_slot

```
sl_si91x_crypto_key_slot_t sl_si91x_hmac_key_config_B0_t::key_slot
```

Key slot.

Definition at line 75 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

wrap_iv_mode

```
sl_si91x_crypto_wrap_mode_t sl_si91x_hmac_key_config_B0_t::wrap_iv_mode
```

Wrap mode.

Definition at line 76 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

wrap_iv

```
uint8_t sl_si91x_hmac_key_config_B0_t::wrap_iv[SL_SI91X_IV_SIZE]
```

Wrap IV.

Definition at line 77 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

key

```
uint8_t* sl_si91x_hmac_key_config_B0_t::key
```

Pointer to the key.

Definition at line 78 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

reserved

```
uint32_t sl_si91x_hmac_key_config_B0_t::reserved
```

Reserved for future use.

Definition at line 79 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

sl_si91x_hmac_key_config_t

Public Attributes

sl_si91x_hmac_key_config_A0_t	A0 Key configuration for non-B0 chip versions.
sl_si91x_hmac_key_config_B0_t	B0 Key configuration for B0 chip versions.

Public Attribute Documentation

A0

```
sl_si91x_hmac_key_config_A0_t sl_si91x_hmac_key_config_t::A0
```

Key configuration for non-B0 chip versions.

Definition at line 83 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

B0

```
sl_si91x_hmac_key_config_B0_t sl_si91x_hmac_key_config_t::B0
```

Key configuration for B0 chip versions.

Definition at line 84 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

sl_si91x_hmac_config_t

Public Attributes

sl_si91x_hmac_mode_t	hmac_mode HMAC Mode.
const uint8_t *	msg Pointer to the input message.
uint32_t	msg_length Length of the message.
sl_si91x_hmac_key_config_t	key_config Key configuration.

Public Attribute Documentation

hmac_mode

```
sl_si91x_hmac_mode_t sl_si91x_hmac_config_t::hmac_mode
```

HMAC Mode.

Definition at line `88` of file `components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h`

msg

```
const uint8_t* sl_si91x_hmac_config_t::msg
```

Pointer to the input message.

Definition at line `89` of file `components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h`

msg_length

```
uint32_t sl_si91x_hmac_config_t::msg_length
```

Length of the message.

Definition at line `90` of file `components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h`

key_config

```
sl_si91x_hmac_key_config_t sl_si91x_hmac_config_t::key_config
```

Key configuration.

Definition at line 91 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

Constants

Constants

This section provides a reference to the HMAC Crypto API constants.

Enumerations

enum sl_si91x_hmac_mode_t {

SL_SI91X_HMAC_SHA_1 = 1
SL_SI91X_HMAC_SHA_256
SL_SI91X_HMAC_SHA_384
SL_SI91X_HMAC_SHA_512

}

enum sl_si91x_hmac_digest_len_t {

SL_SI91X_HMAC_SHA_1_DIGEST_LEN = 20
SL_SI91X_HMAC_SHA_256_DIGEST_LEN = 32
SL_SI91X_HMAC_SHA_384_DIGEST_LEN = 48
SL_SI91X_HMAC_SHA_512_DIGEST_LEN = 64

}

Enumeration Documentation

sl_si91x_hmac_mode_t

> sl_si91x_hmac_mode_t

Enumerator	
SL_SI91X_HMAC_SHA_1	HMAC SHA 1 mode.
SL_SI91X_HMAC_SHA_256	HMAC SHA 256 mode.
SL_SI91X_HMAC_SHA_384	HMAC SHA 384 mode.
SL_SI91X_HMAC_SHA_512	HMAC SHA 512 mode.

Definition at line 43 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

sl_si91x_hmac_digest_len_t

> sl_si91x_hmac_digest_len_t

Enumerator	
SL_SI91X_HMAC_SHA_1_DIGEST_LEN	
SL_SI91X_HMAC_SHA_256_DIGEST_LEN	
SL_SI91X_HMAC_SHA_384_DIGEST_LEN	
SL_SI91X_HMAC_SHA_512_DIGEST_LEN	

Definition at line 50 of file components/device/silabs/si91x/wireless/crypto/hmac/inc/sl_si91x_hmac.h

SHA

SHA

This section provides a reference to the SHA Crypto API including functions, data types and constants.

Modules

Functions

Functions

This section provides a reference to the SHA Crypto API functions.

Functions

sl_status_t sl_si91x_sha(uint8_t sha_mode, uint8_t *msg, uint16_t msg_length, uint8_t *digest)
Decide whether the SHA message can be sent once or requires multiple calls to send.

Function Documentation

sl_si91x_sha

```
sl_status_t sl_si91x_sha (uint8_t sha_mode, uint8_t *msg, uint16_t msg_length, uint8_t *digest)
```

Decide whether the SHA message can be sent once or requires multiple calls to send.

Parameters

[in]	sha_mode	• 1 – For SHA1 • 2 – For SHA256 • 3 – For SHA384 • 4 – For SHA512 • 5– For SHA224
[in]	msg	- Pointer to message
[in]	msg_length	- Total message length
[out]	digest	- Output parameter to hold computed digest from SHA

This is a blocking API. **Returns**

- The following values are returned:
  - 0 - Success
  - Non-Zero Value - Failure

Note

- Refer Error Codes section for above error codes error-codes.

Definition at line `88` of file `components/device/silabs/si91x/wireless/crypto/sha/inc/sl_si91x_sha.h`

Constants

Constants

This section provides a reference to the SHA Crypto API constants.

Enumerations

enum sl_si91x_crypto_sha_mode_t {

SL_SI91x_SHA_1 = 1
SL_SI91x_SHA_256
SL_SI91x_SHA_384
SL_SI91x_SHA_512
SL_SI91x_SHA_224

}

SHA modes.

enum sl_si91x_sha_length_t {

SL_SI91x_SHA_1_DIGEST_LEN = 20
SL_SI91x_SHA_256_DIGEST_LEN = 32
SL_SI91x_SHA_384_DIGEST_LEN = 48
SL_SI91x_SHA_512_DIGEST_LEN = 64
SL_SI91x_SHA_224_DIGEST_LEN = 28

}

SHA mode lenghts.

Enumeration Documentation

sl_si91x_crypto_sha_mode_t

> sl_si91x_crypto_sha_mode_t

SHA modes.

Enumerator	
SL_SI91x_SHA_1	
SL_SI91x_SHA_256	
SL_SI91x_SHA_384	
SL_SI91x_SHA_512	
SL_SI91x_SHA_224	

Definition at line 40 of file components/device/silabs/si91x/wireless/crypto/sha/inc/sl_si91x_sha.h

sl_si91x_sha_length_t

> sl_si91x_sha_length_t

SHA mode lenghts.

Enumerator	
SL_SI91x_SHA_1_DIGEST_LEN	
SL_SI91x_SHA_256_DIGEST_LEN	
SL_SI91x_SHA_384_DIGEST_LEN	
SL_SI91x_SHA_512_DIGEST_LEN	
SL_SI91x_SHA_224_DIGEST_LEN	

Definition at line 49 of file components/device/silabs/si91x/wireless/crypto/sha/inc/sl_si91x_sha.h

TRNG

TRNG

This section provides a reference to the TRNG Crypto API including functions, data types and constants.

Modules

Functions

Functions

This section provides a reference to the TRNG Crypto API functions.

Functions

sl_status_t	sl_si91x_trng_init(sl_si91x_trng_config_t *config, uint32_t *output) This API Initializes the TRNG hardware engine.
sl_status_t	sl_si91x_trng_entropy(void) This API checks the Entropy of TRNG and verifies TRNG functioning.
sl_status_t	sl_si91x_trng_program_key(uint32_t *trng_key, uint16_t key_length) This API initializes and programs the key required for TRNG hardware engine.
sl_status_t	sl_si91x_trng_get_random_num(uint32_t *random_number, uint16_t length) This API generates random number of desired length.
sl_status_t	sl_si91x_duplicate_element(uint32_t *a, uint32_t length) This API checks if there are any repeating elements in the Array.

Function Documentation

sl_si91x_trng_init

```
sl_status_t sl_si91x_trng_init (sl_si91x_trng_config_t *config, uint32_t *output)
```

This API Initializes the TRNG hardware engine.

Parameters

[in]	config	Configuration object of type sl_si91x_trng_config_t
[out]	output	Buffer to store the output.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 62 of file components/device/silabs/si91x/wireless/crypto/trng/inc/sl_si91x_trng.h

sl_si91x_trng_entropy

```
sl_status_t sl_si91x_trng_entropy (void)
```

This API checks the Entropy of TRNG and verifies TRNG functioning.

Parameters

N/A		

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 70 of file components/device/silabs/si91x/wireless/crypto/trng/inc/sl_si91x_trng.h

sl_si91x_trng_program_key

> sl_status_t sl_si91x_trng_program_key (uint32_t *trng_key, uint16_t key_length)

This API initializes and programs the key required for TRNG hardware engine.

Parameters

[in]	trng_key	- Pointer to trng_key
[in]	key_length	- key_length - key length in Dwords (uint32_t)

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 80 of file components/device/silabs/si91x/wireless/crypto/trng/inc/sl_si91x_trng.h

sl_si91x_trng_get_random_num

> sl_status_t sl_si91x_trng_get_random_num (uint32_t *random_number, uint16_t length)

This API generates random number of desired length.

Parameters

[in]	random_number	- Address for Random number
[in]	length	- length of random number in bytes

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 90 of file components/device/silabs/si91x/wireless/crypto/trng/inc/sl_si91x_trng.h

sl_si91x_duplicate_element

> sl_status_t sl_si91x_duplicate_element (uint32_t *a, uint32_t length)

This API checks if there are any repeating elements in the Array.

Parameters

[in]	a	
[in]	length	of the Array

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 100 of file components/device/silabs/si91x/wireless/crypto/trng/inc/sl_si91x_trng.h

Types

Types

This section provides a reference to the TRNG Crypto API types.

Modules

sl_si91x_trng_config_t

sl_si91x_trng_config_t

Public Attributes

uint32_t *	trng_test_data
uint16_t	input_length
uint32_t *	trng_key

Public Attribute Documentation

trng_test_data

```
uint32_t* sl_si91x_trng_config_t::trng_test_data
```

Definition at line 36 of file components/device/silabs/si91x/wireless/crypto/trng/inc/sl_si91x_trng.h

input_length

```
uint16_t sl_si91x_trng_config_t::input_length
```

Definition at line 37 of file components/device/silabs/si91x/wireless/crypto/trng/inc/sl_si91x_trng.h

trng_key

```
uint32_t* sl_si91x_trng_config_t::trng_key
```

Definition at line 38 of file components/device/silabs/si91x/wireless/crypto/trng/inc/sl_si91x_trng.h

Key Wrap

Key Wrap

This section provides a reference to the WRAP Crypto API including functions, data types and constants.

Modules

Functions

Functions

This section provides a reference to the WRAP Crypto API functions.

Functions

sl_status_t sl_si91x_wrap(sl_si91x_wrap_config_t *config, uint8_t *output)
This API is used to get wrap version of the plain key.

Function Documentation

sl_si91x_wrap

```
sl_status_t sl_si91x_wrap (sl_si91x_wrap_config_t *config, uint8_t *output)
```

This API is used to get wrap version of the plain key.

Parameters

N/A	config	Configuration object of type sl_si91x_wrap_config_t
N/A	output	Buffer to store the output.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 70 of file components/device/silabs/si91x/wireless/crypto/wrap/inc/sl_si91x_wrap.h

Types

Types

This section provides a reference to the WRAP Crypto API types.

Modules

sl_si91x_wrap_config_t

sl_si91x_wrap_config_t

Public Attributes

uint32_t	key_type Key type.
uint32_t	reserved Reserved for future use.
uint32_t	key_size Key size.
sl_si91x_crypto_wrap_mode_t	wrap_iv_mode Wrap IV mode.
uint8_t	wrap_iv Buffer to store IV.
uint8_t	key_buffer Key data wrapped/ Plain text.

Public Attribute Documentation

key_type

```
uint32_t sl_si91x_wrap_config_t::key_type
```

Key type.

Definition at line 45 of file components/device/silabs/si91x/wireless/crypto/wrap/inc/sl_si91x_wrap.h

reserved

```
uint32_t sl_si91x_wrap_config_t::reserved
```

Reserved for future use.

Definition at line 46 of file components/device/silabs/si91x/wireless/crypto/wrap/inc/sl_si91x_wrap.h

key_size

```
uint32_t sl_si91x_wrap_config_t::key_size
```

Key size.

Definition at line 47 of file components/device/silabs/si91x/wireless/crypto/wrap/inc/sl_si91x_wrap.h

wrap_iv_mode

```
sl_si91x_crypto_wrap_mode_t sl_si91x_wrap_config_t::wrap_iv_mode
```

Wrap IV mode.

Definition at line 48 of file components/device/silabs/si91x/wireless/crypto/wrap/inc/sl_si91x_wrap.h

wrap_iv

```
uint8_t sl_si91x_wrap_config_t::wrap_iv[SL_SI91X_IV_SIZE]
```

Buffer to store IV.

Definition at line 49 of file components/device/silabs/si91x/wireless/crypto/wrap/inc/sl_si91x_wrap.h

key_buffer

```
uint8_t sl_si91x_wrap_config_t::key_buffer[SL_SI91X_KEY_BUFFER_SIZE]
```

Key data wrapped/ Plain text.

Definition at line 50 of file components/device/silabs/si91x/wireless/crypto/wrap/inc/sl_si91x_wrap.h

Overview

Overview

The Ping application programming interface (API) provides the ability to use the Internet Control Message Protocol (ICMP) to check for the existence of other devices in the local area network (LAN).

APIs

APIs

This section provides a reference to the Internet Control Message Protocol (ICMP) or Ping protocol API.

Modules

Functions

Functions

Functions

This section provides a reference to the Ping API functions.

Functions

sl_status_t	sl_si91x_send_ping(sl_ip_address_t ip_address, uint16_t ping_packet_size) Send an ICMP ping request.

Function Documentation

sl_si91x_send_ping

```
sl_status_t sl_si91x_send_ping (sl_ip_address_t ip_address, uint16_t ping_packet_size)
```

Send an ICMP ping request.

Parameters

[in]	ip_address	Destination IP address (IPv4 or IPv6) and SL IP type identified by sl_ip_address_t
[in]	ping_packet_size	Size of a ping packet

- Pre-conditions:
  - sl_net_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- ping_packet_size valid range: [0, 300]
- This is an asynchronous API. The response is recieved via sl_net_event_handler_t with SL_NET_PING_RESPONSE_EVENT as event

Definition at line 56 of file components/device/silabs/si91x/wireless/icmp/sl_net_ping.h

Overview

Overview

The SiWx91x™ chipset family runs the Simple Network Time Protocol(SNTP) on its internal network processor and provides the ability for the application processor to exchange data with SNTP servers.

The WiSeConnect™ SDK v3.x provides an application programming interface (API) to access the SNTP features of the device.

APIs

APIs

This section provides a reference to the SNTP API including functions, data types, and constants.

Modules

Functions

Functions

This section provides a reference to the SNTP API functions.

Functions

sl_status_t	sl_sntp_client_start(sl_sntp_client_config_t *config, uint32_t timeout) Start SNTP client.
sl_status_t	sl_sntp_client_get_time_date(uint8_t *data, uint16_t data_length, uint32_t timeout) Get time and date information from NTP.
sl_status_t	sl_sntp_client_get_server_info(sl_sntp_server_info_t *data, uint32_t timeout) Get NTP server information.
sl_status_t	sl_sntp_client_stop(uint32_t timeout) Stop SNTP client.

Function Documentation

sl_sntp_client_start

```
sl_status_t sl_sntp_client_start (sl_sntp_client_config_t *config, uint32_t timeout)
```

Start SNTP client.

Parameters

[in]	config	Valid pointer to client configuration structure of type sl_sntp_client_config_t . This shall not be null.
[in]	timeout	Timeout for starting SNTP client. This is blocking API for timeout > 0, else results are returned in sl_sntp_client_event_handler_t callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- This API needs to be called before calling any other SNTP API

Definition at line `153` of file `components/service/sntp/inc/sl_sntp.h`

sl_sntp_client_get_time_date

```
sl_status_t sl_sntp_client_get_time_date (uint8_t *data, uint16_t data_length, uint32_t timeout)
```

Get time and date information from NTP.

Parameters

[in]	data	Valid pointer to data buffer which should be greater than or equal to 50 bytes. In synchronous case, the date and time is returned in this buffer in string format. In async call, this parameter is given to user in user_data parameter of event handler.
[in]	data_length	Data buffer length. In async call, this parameter is given to user in user_data_length parameter of event handler.
[in]	timeout	Timeout for getting time and date. This is blocking API for timeout > 0, else results are returned in sl_sntp_client_event_handler_t callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 164 of file components/service/sntp/inc/sl_sntp.h

sl_sntp_client_get_server_info

```
sl_status_t sl_sntp_client_get_server_info (sl_sntp_server_info_t *data, uint32_t timeout)
```

Get NTP server information.

Parameters

[in]	data	Valid pointer to data buffer of type __attribute__. In async call, this parameter is given to user in user_data with its the structure size (i.e sizeof(sl_sntp_client_event_handler_t)) in user_data_length parameter of event handler.
[in]	timeout	Timeout for getting server information. This is blocking API for timeout > 0, else results are returned in sl_sntp_client_event_handler_t callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 174 of file components/service/sntp/inc/sl_sntp.h

sl_sntp_client_stop

```
sl_status_t sl_sntp_client_stop (uint32_t timeout)
```

Stop SNTP client.

Parameters

[in]	timeout	Timeout for stop SNTP client. This is blocking API for timeout > 0, else results are returned in sl_sntp_client_event_handler_t callback

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 183 of file components/service/sntp/inc/sl_sntp.h

Types

Types

This section provides a reference to the SNTP API data types.

Modules

__attribute__

sl_sntp_client_response_t

sl_sntp_client_config_t

Typedefs

typedef void(* sl_sntp_client_event_handler_t)(sl_sntp_client_response_t *response, uint8_t *user_data, uint16_t user_data_length)
SNTP response handler.

typedef void(* sl_sntp_set_time_sync_notification_handler_t)(const uint8_t cmd_type, sl_status_t status, const uint8_t *buffer)
SNTP time sync notification handler.

Typedef Documentation

sl_sntp_client_event_handler_t

```
sl_sntp_client_event_handler_t )(sl_sntp_client_response_t *response, uint8_t *user_data, uint16_t user_data_length)
```

SNTP response handler.

Parameters

[in]	response	SNTP response of type sl_sntp_client_response_t
[in]	user_data	User data passed in sl_sntp_client_get_time_date(), sl_sntp_client_get_server_info() APIs.
[in]	user_data_length	User data length in sl_sntp_client_get_time_date(), sl_sntp_client_get_server_info() APIs

Definition at line 102 of file components/service/sntp/inc/sl_sntp.h

sl_sntp_set_time_sync_notification_handler_t

```
sl_sntp_set_time_sync_notification_handler_t )(const uint8_t cmd_type, sl_status_t status, const uint8_t *buffer)
```

SNTP time sync notification handler.

Parameters

[in]	cmd_type	command type
[in]	status	sl_status_t.See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.
[in]	buffer	Data buffer

Note

- This features is not supported currently in Si917 chipsets

Definition at line 114 of file components/service/sntp/inc/sl_sntp.h

__attribute__

SNTP Server Info response.

Public Attributes

uint8_t	command_type command type
uint8_t	ip_version Ip version of the SNTP server. 4 for IPV4 and 6 for IPV6.
uint8_t	ipv4_address Ipv4 address of the SNTP server.
unsigned long	ipv6_address Ipv6 address of the SNTP server.
uint8_t	sntp_method SNTP server method.

Public Functions

union __attribute__::@0	__attribute__((__packed__)) server_ip_address

Public Attribute Documentation

command_type

```
uint8_t __attribute__::command_type
```

command type

Definition at line 76 of file components/service/sntp/inc/sl_sntp.h

ip_version

```
uint8_t __attribute__::ip_version
```

Ip version of the SNTP server. 4 for IPV4 and 6 for IPV6.

Definition at line 77 of file components/service/sntp/inc/sl_sntp.h

ipv4_address

```
uint8_t __attribute__::ipv4_address[4]
```

Ipv4 address of the SNTP server.

Definition at line 79 of file components/service/sntp/inc/sl_sntp.h

ipv6_address

```
unsigned long __attribute__::ipv6_address[4]
```

Ipv6 address of the SNTP server.

Definition at line 80 of file components/service/sntp/inc/sl_sntp.h

sntp_method

```
uint8_t __attribute__::sntp_method
```

SNTP server method.

Definition at line 82 of file components/service/sntp/inc/sl_sntp.h

Public Function Documentation

__attribute__

```
union __attribute__::@0 __attribute__::__attribute__ ((__packed__)) server_ip_address
```

Parameters

N/A		

Definition at line 77 of file components/service/sntp/inc/sl_sntp.h

sl_sntp_client_response_t

SNTP Response.

Public Attributes

uint8_t	event_type Type of event being received indicated by sl_sntp_client_events_t.
sl_status_t	status Status of the call which triggered this response. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.
uint8_t *	data Pointer to response data. Incase of SL_SNTP_CLIENT_GET_TIME_DATE it is a (uint8_t *) buffer and incase of SL_SNTP_CLIENT_GET_SERVER_INFO it is a pointer to __attribute__ structure.
uint16_t	data_length Length of response data.

Public Attribute Documentation

event_type

```
uint8_t sl_sntp_client_response_t::event_type
```

Type of event being received indicated by sl_sntp_client_events_t.

Definition at line 87 of file components/service/sntp/inc/sl_sntp.h

status

```
sl_status_t sl_sntp_client_response_t::status
```

Status of the call which triggered this response. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 89 of file components/service/sntp/inc/sl_sntp.h

data

```
uint8_t* sl_sntp_client_response_t::data
```

Pointer to response data. Incase of SL_SNTP_CLIENT_GET_TIME_DATE it is a (uint8_t *) buffer and incase of SL_SNTP_CLIENT_GET_SERVER_INFO it is a pointer to __attribute__ structure.

Definition at line 91 of file components/service/sntp/inc/sl_sntp.h

data_length

```
uint16_t sl_sntp_client_response_t::data_length
```

Length of response data.

Definition at line `92` of file `components/service/sntp/inc/sl_sntp.h`

sl_sntp_client_config_t

SNTP configuration.

Public Attributes

uint8_t *	server_host_name SNTP host name or address.
uint8_t	sntp_method SNTP method. One of the defines from SNTP Methods.
uint16_t	sntp_timeout SNTP timeout.
uint8_t	flags SNTP flags.
sl_sntp_client_event_handler_t	event_handler SNTP response handler of type sl_sntp_client_event_handler_t.
sl_sntp_set_time_sync_notification_handler_t	time_sync_notifiication_handler Set to true if smooth sync required of type sl_sntp_set_time_sync_notification_handler_t.
bool	smooth_sync Set to true if smooth sync required.
bool	server_from_dhcp Set to true to request NTP server config from DHCP.
bool	renew_servers_after_new_ip This is used to refresh server list if NTP provided by DHCP.

Public Attribute Documentation

server_host_name

```
uint8_t* sl_sntp_client_config_t::server_host_name
```

SNTP host name or address.

Definition at line 120 of file components/service/sntp/inc/sl_sntp.h

sntp_method

```
uint8_t sl_sntp_client_config_t::sntp_method
```

SNTP method. One of the defines from SNTP Methods.

Definition at line 121 of file components/service/sntp/inc/sl_sntp.h

sntp_timeout

uint16_t sl_sntp_client_config_t::sntp_timeout

SNTP timeout.

Definition at line 122 of file components/service/sntp/inc/sl_sntp.h

flags

uint8_t sl_sntp_client_config_t::flags

SNTP flags.

Combination of values from SNTP Flags. **Note**

- Bits 1-7 are reserved. User must set them to NULL.

Definition at line 123 of file components/service/sntp/inc/sl_sntp.h

event_handler

sl_sntp_client_event_handler_t sl_sntp_client_config_t::event_handler

SNTP response handler of type sl_sntp_client_event_handler_t.

Definition at line 125 of file components/service/sntp/inc/sl_sntp.h

time_sync_notifiication_handler

sl_sntp_set_time_sync_notification_handler_t sl_sntp_client_config_t::time_sync_notifiication_handler

Set to true if smooth sync required of type sl_sntp_set_time_sync_notification_handler_t.

Note

- This feature is currently not supported in Si917 chipsets

Definition at line 127 of file components/service/sntp/inc/sl_sntp.h

smooth_sync

bool sl_sntp_client_config_t::smooth_sync

Set to true if smooth sync required.

Note

- This feature is currently not supported in Si917 chipsets

Definition at line 129 of file components/service/sntp/inc/sl_sntp.h

server_from_dhcp

```
bool sl_sntp_client_config_t::server_from_dhcp
```

Set to true to request NTP server config from DHCP.

Note

- This feature is currently not supported in Si917 chipsets

Definition at line 131 of file components/service/sntp/inc/sl_sntp.h

renew_servers_after_new_ip

```
bool sl_sntp_client_config_t::renew_servers_after_new_ip
```

This is used to refresh server list if NTP provided by DHCP.

Note

- This feature is currently not supported in 917 chipsets

Definition at line 133 of file components/service/sntp/inc/sl_sntp.h

Constants

Constants

This section provides a reference to the SNTP API constants.

Modules

SNTP Methods

SNTP Flags

Enumerations

enum sl_sntp_client_events_t {
  SL_SNTP_CLIENT_START = 1
  SL_SNTP_CLIENT_GET_TIME_DATE
  SL_SNTP_CLIENT_GET_SERVER_INFO
  SL_SNTP_CLIENT_STOP
}

SNTP Client Events.

Enumeration Documentation

sl_sntp_client_events_t

> sl_sntp_client_events_t

SNTP Client Events.

Enumerator	
SL_SNTP_CLIENT_START	Event for SNTP client started.
SL_SNTP_CLIENT_GET_TIME_DATE	Event for SNTP get time date.
SL_SNTP_CLIENT_GET_SERVER_INFO	Event for SNTP client get server info.
SL_SNTP_CLIENT_STOP	Event for SNTP client stopped.

Definition at line 57 of file components/service/sntp/inc/sl_sntp.h

SNTP Methods

SNTP Methods

Macros

#define SL_SNTP_BROADCAST_MODE 1
SNTP broadcast mode.

#define SL_SNTP_UNICAST_MODE 2
SNTP unicast mode.

Macro Definition Documentation

SL_SNTP_BROADCAST_MODE

```
#define SL_SNTP_BROADCAST_MODE
```

Value:

```
1
```

SNTP broadcast mode.

Definition at line 37 of file components/service/sntp/inc/sl_sntp.h

SL_SNTP_UNICAST_MODE

```
#define SL_SNTP_UNICAST_MODE
```

Value:

```
2
```

SNTP unicast mode.

Definition at line 40 of file components/service/sntp/inc/sl_sntp.h

SNTP Flags

SNTP Flags

Macros

#define SL_SNTP_ENABLE_IPV6 1
SNTP enable IPv6.

Macro Definition Documentation

SL_SNTP_ENABLE_IPV6

```
#define SL_SNTP_ENABLE_IPV6
```

Value:

```
1
```

SNTP enable IPv6.

Note

- By default IPv4 is enabled

Definition at line 49 of file components/service/sntp/inc/sl_sntp.h

Overview

Overview

The SiWx91x™ chipset family runs the HyperText Transfer Protocol (HTTP) on its internal network processor and provides the ability for the application processor to exchange data with HTTP servers.

The WiSeConnect™ SDK v3.x provides an application programming interface (API) to access the HTTP features of the device.

APIs

APIs

This section provides a reference to the HyperText Transfer Protocol (HTTP) API including functions, data types, and constants.

Modules

Functions

Functions

This section provides a reference to the HTTP API functions.

Functions

sl_status_t	sl_http_client_init(const sl_http_client_configuration_t *configuration, sl_http_client_t *client) Initialize an HTTP client.
sl_status_t	sl_http_client_deinit(sl_http_client_t *client) Deinitialize an HTTP client and release resources used by HTTP client.
sl_status_t	sl_http_client_request_init(sl_http_client_request_t *request, sl_http_client_event_handler_t event_handler, void *request_context) Initialize a callback function for the requested HTTP method.
sl_status_t	sl_http_client_add_header(sl_http_client_request_t *request, const char *key, const char *value) Add extended header with key and value in client request.
sl_status_t	sl_http_client_delete_header(sl_http_client_request_t *request, const char *key) Delete a specified header field from the extended header based on key of an HTTP client request.
sl_status_t	sl_http_client_delete_all_headers(sl_http_client_request_t *request) Delete all the headers from the extended header of an HTTP client request.
sl_status_t	sl_http_client_send_request(const sl_http_client_t *client, const sl_http_client_request_t *request) Send HTTP request.
sl_status_t	sl_http_client_write_chunked_data(const sl_http_client_t *client, uint8_t *data, uint32_t data_length, bool flush_now) Send an HTTP POST and PUT chunked data.

Function Documentation

sl_http_client_init

> sl_status_t sl_http_client_init (const sl_http_client_configuration_t *configuration, sl_http_client_t *client)

Initialize an HTTP client.

Parameters

[in]	configuration	HTTP client configuration of type sl_http_client_configuration_t
[out]	client	sl_http_client_t object which will be initialized with client handle.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line `210` of file `components/service/http_client/inc/sl_http_client.h`

sl_http_client_deinit

sl_status_t sl_http_client_deinit (sl_http_client_t *client)

Deinitialize an HTTP client and release resources used by HTTP client.

Parameters

[in]	client	HTTP client's handle of type sl_http_client_t

- Pre-conditions:
  - sl_http_client_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- User must call this API to release the resources once the HTTP client is no longer needed. This function deletes extended headers if any exists during deinitialization.

Definition at line 226 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_request_init

sl_status_t sl_http_client_request_init (sl_http_client_request_t *request, sl_http_client_event_handler_t event_handler, void *request_context)

Initialize a callback function for the requested HTTP method.

Parameters

[in]	request	HTTP client request configuration of type sl_http_client_request_t
[in]	event_handler	HTTP client callback event handler of type sl_http_client_event_handler_t
[in]	request_context	User defined context pointer. Memory space for the user defined context must be valid till the response is received.

- Pre-conditions:
  - sl_http_client_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 244 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_add_header

sl_status_t sl_http_client_add_header (sl_http_client_request_t *request, const char *key, const char *value)

Add extended header with key and value in client request.

Parameters

[in]	request	HTTP client request configuration of type sl_http_client_request_t
[in]	key	Header key.
[in]	value	Header value.

Pre-conditions:
- sl_http_client_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- If a request does not contain any extended header, this API allocates memory for it.

Definition at line 265 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_delete_header

```
sl_status_t sl_http_client_delete_header (sl_http_client_request_t *request, const char *key)
```

Delete a specified header field from the extended header based on key of an HTTP client request.

Parameters

[in]	request	HTTP client request configuration of type sl_http_client_request_t
[in]	key	Header key.

- Pre-conditions:
  - sl_http_client_add_header should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 280 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_delete_all_headers

```
sl_status_t sl_http_client_delete_all_headers (sl_http_client_request_t *request)
```

Delete all the headers from the extended header of an HTTP client request.

Parameters

[in]	request	HTTP client request configuration of type sl_http_client_request_t

- Pre-conditions:
  - sl_http_client_add_header should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- User must call this API to free the memory allocated to headers.

Definition at line 295 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_send_request

```
sl_status_t sl_http_client_send_request (const sl_http_client_t *client, const sl_http_client_request_t *request)
```

Send HTTP request.

Parameters

[in]	client	HTTP client handle of type sl_http_client_t
[in]	request	HTTP client request configuration of type sl_http_client_request_t

- Pre-conditions:
  - sl_http_client_request_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- 1. HTTP HEAD and DELETE methods are not supported in Si91x specific chipsets.
  2. body_length header in request by default internally in Si91x specific chipsets.
  3. HTTP PUT does not support sending body through this API, it is mandatory to call sl_http_client_write_chunked_data() in Si91x specific chipsets.

Definition at line `314` of file `components/service/http_client/inc/sl_http_client.h`

sl_http_client_write_chunked_data

> sl_status_t sl_http_client_write_chunked_data (const sl_http_client_t *client, uint8_t *data, uint32_t data_length, bool flush_now)

Send an HTTP POST and PUT chunked data.

Parameters

[in]	client	HTTP client handle of type sl_http_client_t
[in]	data	Buffer pointer of data to be written.
[in]	data_length	Length of data/chunk to be sent.
[in]	flush_now	

- Pre-conditions:
  - sl_http_client_send_request should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Note

- : flush_now feature is not supported in Si91x specific chipsets.

Definition at line `334` of file `components/service/http_client/inc/sl_http_client.h`

Types

Types

This section provides a reference to the HTTP API data types.

Modules

sl_http_client_credentials_t

sl_http_client_configuration_t

sl_http_client_header_t

sl_http_client_request_t

sl_http_client_response_t

Typedefs

typedef uint32_t **sl_http_client_t**
HTTP Client handle.

typedef sl_status_t(* **sl_http_client_event_handler_t**)(const sl_http_client_t *client, sl_http_client_event_t event, void *data, void *request_context)
Callback invoked when an HTTP response is received.

Typedef Documentation

sl_http_client_t

```
typedef uint32_t sl_http_client_t
```

HTTP Client handle.

Definition at line 101 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_event_handler_t

```
sl_http_client_event_handler_t )(const sl_http_client_t *client, sl_http_client_event_t event, void *data, void *request_context)
```

Callback invoked when an HTTP response is received.

Parameters

[out]	client	HTTP client handle of type sl_http_client_t
[out]	event	HTTP event which has occurred of type sl_http_client_event_t
[out]	data	HTTP response data.
[out]	request_context	User defined context pointer.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 118 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_credentials_t

HTTP client credentials.

Public Attributes

uint16_t	username_length Length of username.
uint16_t	password_length Length of password.
uint8_t	data A flexible array to store both username and password.

Public Attribute Documentation

username_length

```
uint16_t sl_http_client_credentials_t::username_length
```

Length of username.

Definition at line 128 of file components/service/http_client/inc/sl_http_client.h

password_length

```
uint16_t sl_http_client_credentials_t::password_length
```

Length of password.

Definition at line 129 of file components/service/http_client/inc/sl_http_client.h

data

```
uint8_t sl_http_client_credentials_t::data[]
```

A flexible array to store both username and password.

Definition at line 130 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_configuration_t

HTTP client configurations.

Public Attributes

uint8_t	certificate_index HTTPS client Certificate index.
sl_http_client_tls_version_t	tls_version HTTP client TLS version. sl_http_client_tls_version_t.
sl_http_client_version_t	http_version HTTP version. sl_http_client_version_t.
bool	https_enable HTTP security option.
bool	https_use_sni HTTPs SNI extension.
sl_ip_address_type_t	ip_version HTTP client IP version. sl_ip_address_type_t.
sl_net_interface_t	network_interface HTTP client network interface.sl_net_interface_t

Public Attribute Documentation

certificate_index

> uint8_t sl_http_client_configuration_t::certificate_index

HTTPS client Certificate index.

Definition at line 135 of file components/service/http_client/inc/sl_http_client.h

tls_version

> sl_http_client_tls_version_t sl_http_client_configuration_t::tls_version

HTTP client TLS version. sl_http_client_tls_version_t.

Definition at line 136 of file components/service/http_client/inc/sl_http_client.h

http_version

> sl_http_client_version_t sl_http_client_configuration_t::http_version

HTTP version. sl_http_client_version_t.

Definition at line 137 of file components/service/http_client/inc/sl_http_client.h

https_enable

```
bool sl_http_client_configuration_t::https_enable
```

HTTP security option.

Definition at line 138 of file components/service/http_client/inc/sl_http_client.h

https_use_sni

```
bool sl_http_client_configuration_t::https_use_sni
```

HTTPs SNI extension.

Definition at line 139 of file components/service/http_client/inc/sl_http_client.h

ip_version

```
sl_ip_address_type_t sl_http_client_configuration_t::ip_version
```

HTTP client IP version. sl_ip_address_type_t.

Definition at line 140 of file components/service/http_client/inc/sl_http_client.h

network_interface

```
sl_net_interface_t sl_http_client_configuration_t::network_interface
```

HTTP client network interface.sl_net_interface_t

Definition at line 142 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_header_t

Structure of HTTP client extended header node.

Public Attributes

struct sl_http_client_header_t *	next Link to the next client header.
char *	key Header key name.
char *	value Header value.

Public Attribute Documentation

next

```
struct sl_http_client_header_t* sl_http_client_header_t::next
```

Link to the next client header.

Definition at line 147 of file components/service/http_client/inc/sl_http_client.h

key

```
char* sl_http_client_header_t::key
```

Header key name.

Definition at line 148 of file components/service/http_client/inc/sl_http_client.h

value

```
char* sl_http_client_header_t::value
```

Header value.

Definition at line 149 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_request_t

HTTP client request configurations.

Public Attributes

sl_http_client_method_type_t	**http_method_type** HTTP request method. sl_http_client_method_type_t.
uint8_t *	**ip_address** HTTP server IP address.
uint8_t *	**resource** URL string for requested resource.
uint16_t	**port** HTTP server port number.
si91x_socket_type_length_value_t *	**sni_extension** SNI extension address. si91x_socket_type_length_value_t.
uint8_t *	**body** HTTP body to be sent to the server. Setting this to null will process the request in chunked encoding.
uint32_t	**body_length** Body length of data to be posted. In case of chunked request, body length should be equal to total content length.
sl_http_client_header_t *	**extended_header** User-defined extended header. If null, default extended header will be added internally. sl_http_client_header_t.
uint16_t	**timeout_ms** HTTP request timeout period. (Si91x chipsets does not support this feature).
uint16_t	**retry_count** HTTP request maximum retry count after timeout. (Si91x chipsets does not support this feature).
uint16_t	**retry_period_ms** Retry period after max retry count reach. (Si91x chipsets does not support this feature).
bool	**tcp_connection_reuse** Flag to use same TCP socket for connection. (Si91x chipsets does not support this feature).
void *	**context** User defined context.
sl_http_client_event_handler_t	**event_handler** Callback method. sl_http_client_event_handler_t.

Public Attribute Documentation

http_method_type

```
sl_http_client_method_type_t sl_http_client_request_t::http_method_type
```

HTTP request method. sl_http_client_method_type_t.

Definition at line 154 of file components/service/http_client/inc/sl_http_client.h

ip_address

```
uint8_t* sl_http_client_request_t::ip_address
```

HTTP server IP address.

Definition at line 155 of file components/service/http_client/inc/sl_http_client.h

resource

```
uint8_t* sl_http_client_request_t::resource
```

URL string for requested resource.

Definition at line 156 of file components/service/http_client/inc/sl_http_client.h

port

```
uint16_t sl_http_client_request_t::port
```

HTTP server port number.

Definition at line 157 of file components/service/http_client/inc/sl_http_client.h

sni_extension

```
si91x_socket_type_length_value_t* sl_http_client_request_t::sni_extension
```

SNI extension address. si91x_socket_type_length_value_t.

Definition at line 158 of file components/service/http_client/inc/sl_http_client.h

body

```
uint8_t* sl_http_client_request_t::body
```

HTTP body to be sent to the server. Setting this to null will process the request in chunked encoding.

Definition at line 160 of file components/service/http_client/inc/sl_http_client.h

body_length

```
uint32_t sl_http_client_request_t::body_length
```

Body length of data to be posted. In case of chunked request, body length should be equal to total content length.

Definition at line 162 of file components/service/http_client/inc/sl_http_client.h

extended_header

```
sl_http_client_header_t* sl_http_client_request_t::extended_header
```

User-defined extended header. If null, default extended header will be added internally. sl_http_client_header_t.

Definition at line 164 of file components/service/http_client/inc/sl_http_client.h

timeout_ms

```
uint16_t sl_http_client_request_t::timeout_ms
```

HTTP request timeout period. (Si91x chipsets does not support this feature).

Definition at line 165 of file components/service/http_client/inc/sl_http_client.h

retry_count

```
uint16_t sl_http_client_request_t::retry_count
```

HTTP request maximum retry count after timeout. (Si91x chipsets does not support this feature).

Definition at line 167 of file components/service/http_client/inc/sl_http_client.h

retry_period_ms

```
uint16_t sl_http_client_request_t::retry_period_ms
```

Retry period after max retry count reach. (Si91x chipsets does not support this feature).

Definition at line 169 of file components/service/http_client/inc/sl_http_client.h

tcp_connection_reuse

```
bool sl_http_client_request_t::tcp_connection_reuse
```

Flag to use same TCP socket for connection. (Si91x chipsets does not support this feature).

Definition at line 171 of file components/service/http_client/inc/sl_http_client.h

context

```
void* sl_http_client_request_t::context
```

User defined context.

Definition at line 172 of file components/service/http_client/inc/sl_http_client.h

event_handler

```
sl_http_client_event_handler_t sl_http_client_request_t::event_handler
```

Callback method. sl_http_client_event_handler_t.

Definition at line 173 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_response_t

HTTP client response structure.

Public Attributes

uint32_t	status Request status.
uint8_t *	data_buffer Response data.
uint16_t	data_length Length of received data.
uint32_t	end_of_data End of data indication.
uint16_t	http_response_code HTTP server response. (Si91x chipsets does not support this feature for SL_HTTP_PUT).
uint8_t	version HTTP version. (Si91x chipsets does not support this feature)
sl_http_client_header_t *	response_headers HTTP response headers. sl_http_client_header_t (Si91x chipsets does not support this feature).

Public Attribute Documentation

status

```
uint32_t sl_http_client_response_t::status
```

Request status.

Definition at line 178 of file components/service/http_client/inc/sl_http_client.h

data_buffer

```
uint8_t* sl_http_client_response_t::data_buffer
```

Response data.

Definition at line 179 of file components/service/http_client/inc/sl_http_client.h

data_length

```
uint16_t sl_http_client_response_t::data_length
```

Length of received data.

Definition at line 180 of file components/service/http_client/inc/sl_http_client.h

end_of_data

```
uint32_t sl_http_client_response_t::end_of_data
```

End of data indication.

Definition at line 181 of file components/service/http_client/inc/sl_http_client.h

http_response_code

```
uint16_t sl_http_client_response_t::http_response_code
```

HTTP server response. (Si91x chipsets does not support this feature for SL_HTTP_PUT).

Definition at line 183 of file components/service/http_client/inc/sl_http_client.h

version

```
uint8_t sl_http_client_response_t::version
```

HTTP version. (Si91x chipsets does not support this feature)

Definition at line 184 of file components/service/http_client/inc/sl_http_client.h

response_headers

```
sl_http_client_header_t* sl_http_client_response_t::response_headers
```

HTTP response headers. sl_http_client_header_t (Si91x chipsets does not support this feature).

Definition at line 186 of file components/service/http_client/inc/sl_http_client.h

Constants

Constants

This section provides a reference to the HTTP API constants.

Enumerations

enum sl_http_client_method_type_t {

SL_HTTP_GET = 0
SL_HTTP_POST = 1
SL_HTTP_HEAD = 2
SL_HTTP_PUT = 3
SL_HTTP_DELETE = 4

}

HTTP client methods.

enum sl_http_client_tls_version_t {

SL_TLS_V_1_0 = 0
SL_TLS_V_1_1 = 1
SL_TLS_V_1_2 = 2
SL_TLS_V_1_3 = 3

}

HTTPS client TLS versions.

enum sl_http_client_version_t {

SL_HTTP_V_1_0 = 0
SL_HTTP_V_1_1 = 1

}

HTTP versions.

enum sl_http_client_event_t {

SL_HTTP_CLIENT_GET_RESPONSE_EVENT = 0
SL_HTTP_CLIENT_POST_RESPONSE_EVENT
SL_HTTP_CLIENT_PUT_RESPONSE_EVENT
SL_HTTP_CLIENT_MAX_EVENT

}

HTTP client events.

Macros

#define SL_HTTP_CLIENT_MAX_WRITE_BUFFER_LENGTH 900

MAX buffer length for write data.

#define SL_HTTPS_CLIENT_DEFAULT_CERTIFICATE_INDEX 0

HTTPS client Certificate Index constants.

#define SL_HTTPS_CLIENT_CERTIFICATE_INDEX_1 1

HTTPS CLIENT 1st Certificate Index.

#define SL_HTTPS_CLIENT_CERTIFICATE_INDEX_2 2
HTTPS CLIENT 2nd Certificate Index.

Enumeration Documentation

sl_http_client_method_type_t

> sl_http_client_method_type_t

HTTP client methods.

Note

- 917 does not support SL_HTTP_HEAD and SL_HTTP_DELETE in current release.

Enumerator	
SL_HTTP_GET	HTTP Request Method type GET.
SL_HTTP_POST	HTTP Request Method type POST.
SL_HTTP_HEAD	HTTP Request Method type HEAD.
SL_HTTP_PUT	HTTP Request Method type PUT.
SL_HTTP_DELETE	HTTP Request Method type DELETE.

Definition at line 57 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_tls_version_t

> sl_http_client_tls_version_t

HTTPS client TLS versions.

Enumerator	
SL_TLS_V_1_0	Use TLS Version 1.0 for HTTP Client.
SL_TLS_V_1_1	Use TLS Version 1.1 for HTTP Client.
SL_TLS_V_1_2	Use TLS Version 1.2 for HTTP Client.
SL_TLS_V_1_3	Use TLS Version 1.3 for HTTP Client.

Definition at line 68 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_version_t

> sl_http_client_version_t

HTTP versions.

Enumerator	
SL_HTTP_V_1_0	Use HTTP Protocol version 1.0.
SL_HTTP_V_1_1	Use HTTP Protocol version 1.1.

Definition at line 76 of file components/service/http_client/inc/sl_http_client.h

sl_http_client_event_t

sl_http_client_event_t

HTTP client events.

Enumerator	
SL_HTTP_CLIENT_GET_RESPONSE_EVENT	HTTP Client GET response Event.
SL_HTTP_CLIENT_POST_RESPONSE_EVENT	HTTP Client POST response Event.
SL_HTTP_CLIENT_PUT_RESPONSE_EVENT	HTTP Client PUT response Event.
SL_HTTP_CLIENT_MAX_EVENT	HTTP Client Maximum number of events.

Definition at line 82 of file components/service/http_client/inc/sl_http_client.h

Macro Definition Documentation

SL_HTTP_CLIENT_MAX_WRITE_BUFFER_LENGTH

#define SL_HTTP_CLIENT_MAX_WRITE_BUFFER_LENGTH

Value:

```
900
```

MAX buffer length for write data.

Definition at line 38 of file components/service/http_client/inc/sl_http_client.h

SL_HTTPS_CLIENT_DEFAULT_CERTIFICATE_INDEX

#define SL_HTTPS_CLIENT_DEFAULT_CERTIFICATE_INDEX

Value:

```
0
```

HTTPS client Certificate Index constants.

Definition at line 41 of file components/service/http_client/inc/sl_http_client.h

SL_HTTPS_CLIENT_CERTIFICATE_INDEX_1

#define SL_HTTPS_CLIENT_CERTIFICATE_INDEX_1

Value:

```
1
```

HTTPS CLIENT 1st Certificate Index.

Definition at line 44 of file components/service/http_client/inc/sl_http_client.h

SL_HTTPS_CLIENT_CERTIFICATE_INDEX_2

```
#define SL_HTTPS_CLIENT_CERTIFICATE_INDEX_2
```

Value:

```
2
```

HTTPS CLIENT 2nd Certificate Index.

Definition at line 47 of file components/service/http_client/inc/sl_http_client.h

Overview

Overview

The SiWx91x™ chipset family runs the Message Queue Telemetry Transport (MQTT) on its internal network processor and provides the ability for the application processor to connect to an MQTT broker and access MQTT protocol features such as subscribing or publishing to MQTT topics.

The WiSeConnect™ SDK v3.x provides an application programming interface (API) to access the MQTT features of the device.

APIs

APIs

This section provides a reference to the Message Queue Telemetry Transport (MQTT) API including functions, data types, and constants.

Modules

Functions

Types

Constants

Functions

Functions

This section provides a reference to the MQTT API functions.

Functions

sl_status_t	sl_mqtt_client_init(sl_mqtt_client_t *client, sl_mqtt_client_event_handler_t event_handler) Initialize the mqtt_client and registers the event_handler.
sl_status_t	sl_mqtt_client_deinit(sl_mqtt_client_t *client) Deinitialize the MQTT client.
sl_status_t	sl_mqtt_client_connect(sl_mqtt_client_t *client, const sl_mqtt_broker_t *broker, const sl_mqtt_client_last_will_message_t *last_will_message, const sl_mqtt_client_configuration_t *configuration, uint32_t timeout) Connect the client to MQTT broker.
sl_status_t	sl_mqtt_client_disconnect(sl_mqtt_client_t *client, uint32_t timeout) Disconnect the client from MQTT broker.
sl_status_t	sl_mqtt_client_publish(sl_mqtt_client_t *client, const sl_mqtt_client_message_t *message, uint32_t timeout, void *context) Publish a message.
sl_status_t	sl_mqtt_client_subscribe(sl_mqtt_client_t *client, const uint8_t *topic, uint16_t topic_length, sl_mqtt_qos_t qos_level, uint32_t timeout, sl_mqtt_client_message_received_t message_handler, void *context) Indicate which client can subscribe to a topic.
sl_status_t	sl_mqtt_client_unsubscribe(sl_mqtt_client_t *client, const uint8_t *topic, uint16_t length, uint32_t timeout, void *context) Indicate which client can unsubscribe a topic.

Function Documentation

sl_mqtt_client_init

```
sl_status_t sl_mqtt_client_init (sl_mqtt_client_t *client, sl_mqtt_client_event_handler_t event_handler)
```

Initialize the mqtt_client and registers the event_handler.

Parameters

[in]	client	Valid pointer to the client structure of type sl_mqtt_client_t. This shall not be null.
[in]	event_handler	Event handler of type sl_mqtt_client_event_handler_t which will be called for various events on client.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 38 of file components/service/mqtt/inc/sl_mqtt_client.h

sl_mqtt_client_deinit

```
sl_status_t sl_mqtt_client_deinit (sl_mqtt_client_t *client)
```

Deinitialize the MQTT client.

Parameters

[in]	client	Valid pointer to the client structure of type sl_mqtt_client_t

- Pre-conditions:
  - sl_mqtt_client_init should be called before this API.

Returns

- sl_status_t. See https://docs.silabs.com/gecko-platform/4.1/common/api/group-status for details.

Definition at line 51 of file components/service/mqtt/inc/sl_mqtt_client.h

sl_mqtt_client_connect

```
sl_status_t sl_mqtt_client_connect (sl_mqtt_client_t *client, const sl_mqtt_broker_t *broker, const sl_mqtt_client_last_will_message_t *last_will_message, const sl_mqtt_client_configuration_t *configuration, uint32_t timeout)
```

Connect the client to MQTT broker.

Parameters

[in]	client	sl_mqtt_client_t client which would be connected to the broker.
[in]	broker	Broker configuration of type sl_mqtt_broker_t
[in]	last_will_message	Last will message of the client of type sl_mqtt_client_last_will_message_t
[in]	configuration	Client configuration of type sl_mqtt_client_configuration_t
[in]	timeout	Timeout for the API. If the value is zero, the API will be asynchronous.

- Pre-conditions:
  - sl_mqtt_client_init should be called before this API.

Returns

- sl_status_t. If called asynchronously, SL_STATUS_IN_PROGRESS will be returned as status.

Note

- In subsequent calls to connect, it is optional to provide broker, last_will_message, and configuration parameters.
- If broker and configuration parameters are given null, then value given in the first connect() call would be retained.
- In case of last_will_message parameter, values given in each connect() call would be considered. If value of last_will_message parameter is given as null, then no will message would be sent to the broker parameter.
- Only is_clean_session, credential_id, client_id, client_id_length of sl_mqtt_client_configuration_t are considered.

Definition at line 80 of file components/service/mqtt/inc/sl_mqtt_client.h

sl_mqtt_client_disconnect

```
sl_status_t sl_mqtt_client_disconnect (sl_mqtt_client_t *client, uint32_t timeout)
```

Disconnect the client from MQTT broker.

Parameters

[in]	client	sl_mqtt_client_t client which needs to be disconnected from the broker.
[in]	timeout	Timeout for the API. If the value is zero, the API will be asynchronous.

- Pre-conditions:
  - sl_mqtt_client_connect should be called before this API.

Returns

- sl_status_t. If called asynchronously, SL_STATUS_IN_PROGRESS will be returned as status.

Definition at line 99 of file components/service/mqtt/inc/sl_mqtt_client.h

sl_mqtt_client_publish

> sl_status_t sl_mqtt_client_publish (sl_mqtt_client_t *client, const sl_mqtt_client_message_t *message, uint32_t timeout, void *context)

Publish a message.

Parameters

[in]	client	sl_mqtt_client_t client which requires to publish the message.
[in]	message	sl_mqtt_client_message_t Message which needs to be published.
[in]	timeout	Timeout for the API. If the value is zero, the API will be asynchronous.
[in]	context	Context which would be returned in event handler if the API is called asynchronously. The caller must ensure that the lifecycle of context is retained until the callback is invoked. The deallocation of context is also the responsibility of the caller.

- Pre-conditions:
  - sl_mqtt_client_connect should be called before this API.

Returns

- sl_status_t. If called asynchronously, SL_STATUS_IN_PROGRESS will be returned as status.

Definition at line 118 of file components/service/mqtt/inc/sl_mqtt_client.h

sl_mqtt_client_subscribe

> sl_status_t sl_mqtt_client_subscribe (sl_mqtt_client_t *client, const uint8_t *topic, uint16_t topic_length, sl_mqtt_qos_t qos_level, uint32_t timeout, sl_mqtt_client_message_received_t message_handler, void *context)

Indicate which client can subscribe to a topic.

Parameters

[in]	client	sl_mqtt_client_t client which needs to subscribe to the topic.
[in]	topic	Topic of interest.
[in]	topic_length	Length of the topic.
[in]	qos_level	Quality of service of type sl_mqtt_qos_t using which client subscribes.
[in]	timeout	Timeout for API. If the value is zero, the API will be asynchronous.
[in]	message_handler	sl_mqtt_client_message_received_t message handler which invokes for messages published on the topic.

[in]	context	Context that is sent back to the message handler. The caller must ensure that the lifecycle of the context is retained until the callback is invoked. The deallocation of context is also the responsibility of the caller.

- Pre-conditions:
  - sl_mqtt_client_connect should be called before this API.

Returns

- sl_status_t. If called asynchronously, SL_STATUS_IN_PROGRESS will be returned as status.

Note

- The maximum length of the topic cannot be greater than SI91X_MQTT_CLIENT_TOPIC_MAXIMUM_LENGTH.

Definition at line 148 of file components/service/mqtt/inc/sl_mqtt_client.h

sl_mqtt_client_unsubscribe

> sl_status_t sl_mqtt_client_unsubscribe (sl_mqtt_client_t *client, const uint8_t *topic, uint16_t length, uint32_t timeout, void *context)

Indicate which client can unsubscribe a topic.

Parameters

[in]	client	sl_mqtt_client_t client that unsubscribes the topic.
[in]	topic	Topic from which client needs to unsubscribe.
[in]	length	Length of the topic.
[in]	timeout	Timeout for the API. If the value is zero, the API will be asynchronous.
[in]	context	Context that returns in event handler if the API is called asynchronously. The caller must ensure that the lifecycle of the context is retained until the callback is invoked. The deallocation of context is also the responsibility of the caller.

- Pre-conditions:
  - sl_mqtt_client_subscribe should be called before this API.

Returns

- sl_status_t. If called asynchronously, SL_STATUS_IN_PROGRESS will be returned as status.

Definition at line 175 of file components/service/mqtt/inc/sl_mqtt_client.h

Types

Types

This section provides a reference to the MQTT API data types.

Modules

sl_mqtt_client_last_will_message_t

sl_mqtt_client_message_t

sl_mqtt_broker_t

sl_mqtt_client_credentials_t

sl_mqtt_client_configuration_t

sl_mqtt_client_topic_subscription_info_t

sl_mqtt_client_t

Typedefs

typedef void(* sl_mqtt_client_event_handler_t)(void *client, sl_mqtt_client_event_t event, void *event_data, void *context)
Handler for MQTT client Events.

typedef void(* sl_mqtt_client_message_received_t)(void *client, sl_mqtt_client_message_t *message_to_be_published, void *context)
Handler for MQTT client RX message.

Typedef Documentation

sl_mqtt_client_event_handler_t

```
sl_mqtt_client_event_handler_t )(void *client, sl_mqtt_client_event_t event, void *event_data, void *context)
```

Handler for MQTT client Events.

Parameters

N/A	client	Client on which the event occurred.
N/A	event	Event occurred of type sl_mqtt_client_event_t
N/A	event_data	Event data for the data. This parameter would be non-null only for MQTT_CLIENT_MESSAGED_RECEIVED and MQTT_CLIENT_ERROR.
N/A	context	Context provided by user at the time of API call. The caller must ensure that the lifecycle of the context is retained until the callback is invoked. The deallocation of context is also the responsibility of the caller.

Definition at line 151 of file components/service/mqtt/inc/sl_mqtt_client_types.h

sl_mqtt_client_message_received_t

```
sl_mqtt_client_message_received_t )(void *client, sl_mqtt_client_message_t *message_to_be_published, void *context)
```

Handler for MQTT client RX message.

Parameters

N/A	client	Client which has received message.
N/A	message_to_be_published	Message received of type sl_mqtt_client_message_t
N/A	context	Context provided by user at the time of Subscribe API call. The caller must ensure that the lifecycle of the context is retained until the callback is invoked. The deallocation of context is also the responsibility of the caller.

Note

- Due to the constraints from the firmware, is_retained, qos_level, packet_identifer, duplicate_message of sl_mqtt_client_message_t are not populated and shall be ignored while processing the message.

Definition at line `169` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

sl_mqtt_client_last_will_message_t

MQTT Client last will message.

Public Attributes

bool	is_retained Flag whether to retain the will message or not.
sl_mqtt_qos_t	will_qos_level Quality of subscription.
uint8_t *	will_topic Name of will topic.
uint16_t	will_topic_length Length of topic.
uint8_t *	will_message Pointer to will message.
uint32_t	will_message_length Length of the will message.

Public Attribute Documentation

is_retained

```
bool sl_mqtt_client_last_will_message_t::is_retained
```

Flag whether to retain the will message or not.

Definition at line `84` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

will_qos_level

```
sl_mqtt_qos_t sl_mqtt_client_last_will_message_t::will_qos_level
```

Quality of subscription.

Definition at line `85` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

will_topic

```
uint8_t* sl_mqtt_client_last_will_message_t::will_topic
```

Name of will topic.

Definition at line `86` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

will_topic_length

```
uint16_t sl_mqtt_client_last_will_message_t::will_topic_length
```

Length of topic.

Definition at line `87` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

will_message

```
uint8_t* sl_mqtt_client_last_will_message_t::will_message
```

Pointer to will message.

Definition at line `88` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

will_message_length

```
uint32_t sl_mqtt_client_last_will_message_t::will_message_length
```

Length of the will message.

Definition at line `89` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

sl_mqtt_client_message_t

MQTT Client Message structure.

Public Attributes

sl_mqtt_qos_t	qos_level Quality of subscription.
uint16_t	pakcet_identifer Packed id of the received message.
bool	is_retained Retained flag of message as sent by the broker.
bool	is_duplicate_message Whether this is a duplicate message.
uint8_t *	topic Pointer to topic name.
uint16_t	topic_length Length of the topic.
uint8_t *	content Pointer to content.
uint32_t	content_length Length of the content.

Public Attribute Documentation

qos_level

```
sl_mqtt_qos_t sl_mqtt_client_message_t::qos_level
```

Quality of subscription.

Definition at line `94` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

pakcet_identifer

```
uint16_t sl_mqtt_client_message_t::pakcet_identifer
```

Packed id of the received message.

Definition at line `95` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

is_retained

bool sl_mqtt_client_message_t::is_retained

Retained flag of message as sent by the broker.

Definition at line 96 of file components/service/mqtt/inc/sl_mqtt_client_types.h

is_duplicate_message

bool sl_mqtt_client_message_t::is_duplicate_message

Whether this is a duplicate message.

Definition at line 97 of file components/service/mqtt/inc/sl_mqtt_client_types.h

topic

uint8_t* sl_mqtt_client_message_t::topic

Pointer to topic name.

Definition at line 98 of file components/service/mqtt/inc/sl_mqtt_client_types.h

topic_length

uint16_t sl_mqtt_client_message_t::topic_length

Length of the topic.

Definition at line 99 of file components/service/mqtt/inc/sl_mqtt_client_types.h

content

uint8_t* sl_mqtt_client_message_t::content

Pointer to content.

Definition at line 100 of file components/service/mqtt/inc/sl_mqtt_client_types.h

content_length

uint32_t sl_mqtt_client_message_t::content_length

Length of the content.

Definition at line 101 of file components/service/mqtt/inc/sl_mqtt_client_types.h

sl_mqtt_broker_t

MQTT Client broker information structure.

Public Attributes

sl_ip_address_t	ip IP address of broker.
uint16_t	port Port number of broker.
bool	is_connection_encrypted Whether to use MQTT or MQTTS.
uint16_t	connect_timeout MQTT Connection timeout.
uint16_t	keep_alive_interval Keep alive timeout of MQTT connection.
uint16_t	keep_alive_retries MQTT ping retries.

Public Attribute Documentation

ip

```
sl_ip_address_t sl_mqtt_broker_t::ip
```

IP address of broker.

Definition at line `106` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

port

```
uint16_t sl_mqtt_broker_t::port
```

Port number of broker.

Definition at line `107` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

is_connection_encrypted

```
bool sl_mqtt_broker_t::is_connection_encrypted
```

Whether to use MQTT or MQTTS.

Definition at line `108` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

connect_timeout

> uint16_t sl_mqtt_broker_t::connect_timeout

MQTT Connection timeout.

Definition at line 109 of file components/service/mqtt/inc/sl_mqtt_client_types.h

keep_alive_interval

> uint16_t sl_mqtt_broker_t::keep_alive_interval

Keep alive timeout of MQTT connection.

Definition at line 110 of file components/service/mqtt/inc/sl_mqtt_client_types.h

keep_alive_retries

> uint16_t sl_mqtt_broker_t::keep_alive_retries

MQTT ping retries.

Definition at line 111 of file components/service/mqtt/inc/sl_mqtt_client_types.h

sl_mqtt_client_credentials_t

MQTT Client credentials structure.

Public Attributes

uint16_t	username_length Length of the username. Length should not exceed 62 bytes including NULL termination character.
uint16_t	password_length Length of the password. Length should not exceed 62 bytes including NULL termination character.
uint8_t	data A flexible array to store both username and password.

Public Attribute Documentation

username_length

```
uint16_t sl_mqtt_client_credentials_t::username_length
```

Length of the username. Length should not exceed 62 bytes including NULL termination character.

Definition at line 117 of file components/service/mqtt/inc/sl_mqtt_client_types.h

password_length

```
uint16_t sl_mqtt_client_credentials_t::password_length
```

Length of the password. Length should not exceed 62 bytes including NULL termination character.

Definition at line 119 of file components/service/mqtt/inc/sl_mqtt_client_types.h

data

```
uint8_t sl_mqtt_client_credentials_t::data[]
```

A flexible array to store both username and password.

Definition at line 120 of file components/service/mqtt/inc/sl_mqtt_client_types.h

sl_mqtt_client_configuration_t

MQTT Client Configuration structure.

Public Attributes

bool	auto_reconnect Whether to automatically connect back to broker in case of disconnection.
uint8_t	retry_count Maximum retry count of auto reconnect.
uint16_t	minimum_back_off_time Minimum back off time between two successive reconnect attempts.
uint16_t	maximun_back_off_time Maximum back off time between two successive reconnect attempts.
bool	is_clean_session Clean session flag to send to broker in connect request.
sl_mqtt_version_t	mqt_version Version of client MQTT.
uint16_t	client_port Port number of client.
sl_net_credential_id_t	credential_id Credential id for username and password of MQTT connect request.
uint8_t *	client_id Pointer to MQTT client id.
uint8_t	client_id_length Length of client id. Length should not exceed 62 bytes including NULL termination character.

Public Attribute Documentation

auto_reconnect

```
bool sl_mqtt_client_configuration_t::auto_reconnect
```

Whether to automatically connect back to broker in case of disconnection.

Definition at line 125 of file components/service/mqtt/inc/sl_mqtt_client_types.h

retry_count

```
uint8_t sl_mqtt_client_configuration_t::retry_count
```

Maximum retry count of auto reconnect.

Definition at line 126 of file components/service/mqtt/inc/sl_mqtt_client_types.h

minimum_back_off_time

> uint16_t sl_mqtt_client_configuration_t::minimum_back_off_time

Minimum back off time between two successive reconnect attempts.

Definition at line 127 of file components/service/mqtt/inc/sl_mqtt_client_types.h

maximun_back_off_time

> uint16_t sl_mqtt_client_configuration_t::maximun_back_off_time

Maximum back off time between two successive reconnect attempts.

Definition at line 128 of file components/service/mqtt/inc/sl_mqtt_client_types.h

is_clean_session

> bool sl_mqtt_client_configuration_t::is_clean_session

Clean session flag to send to broker in connect request.

Definition at line 129 of file components/service/mqtt/inc/sl_mqtt_client_types.h

mqt_version

> sl_mqtt_version_t sl_mqtt_client_configuration_t::mqt_version

Version of client MQTT.

Definition at line 130 of file components/service/mqtt/inc/sl_mqtt_client_types.h

client_port

> uint16_t sl_mqtt_client_configuration_t::client_port

Port number of client.

Definition at line 131 of file components/service/mqtt/inc/sl_mqtt_client_types.h

credential_id

> sl_net_credential_id_t sl_mqtt_client_configuration_t::credential_id

Credential id for username and password of MQTT connect request.

Definition at line 132 of file components/service/mqtt/inc/sl_mqtt_client_types.h

client_id

```
uint8_t* sl_mqtt_client_configuration_t::client_id
```

Pointer to MQTT client id.

Definition at line 133 of file components/service/mqtt/inc/sl_mqtt_client_types.h

client_id_length

```
uint8_t sl_mqtt_client_configuration_t::client_id_length
```

Length of client id. Length should not exceed 62 bytes including NULL termination character.

Definition at line 135 of file components/service/mqtt/inc/sl_mqtt_client_types.h

sl_mqtt_client_topic_subscription_info_t

MQTT Client Topic Subscription Info structure.

Public Attributes

sl_slist_node_t	next_subscription Next node in the linked list.
sl_mqtt_client_message_received_t	topic_message_handler A function pointer to message handler.
sl_mqtt_qos_t	qos_of_subscription Quality of subscription.
uint16_t	topic_length Length of the subscribed topic.
uint8_t	topic A flexible array to store the topic.

Public Attribute Documentation

next_subscription

```
sl_slist_node_t sl_mqtt_client_topic_subscription_info_t::next_subscription
```

Next node in the linked list.

Definition at line 175 of file components/service/mqtt/inc/sl_mqtt_client_types.h

topic_message_handler

```
sl_mqtt_client_message_received_t sl_mqtt_client_topic_subscription_info_t::topic_message_handler
```

A function pointer to message handler.

Definition at line 176 of file components/service/mqtt/inc/sl_mqtt_client_types.h

qos_of_subscription

```
sl_mqtt_qos_t sl_mqtt_client_topic_subscription_info_t::qos_of_subscription
```

Quality of subscription.

Definition at line 177 of file components/service/mqtt/inc/sl_mqtt_client_types.h

topic_length

uint16_t sl_mqtt_client_topic_subscription_info_t::topic_length

Length of the subscribed topic.

Definition at line 178 of file components/service/mqtt/inc/sl_mqtt_client_types.h

topic

uint8_t sl_mqtt_client_topic_subscription_info_t::topic[]

A flexible array to store the topic.

Definition at line 179 of file components/service/mqtt/inc/sl_mqtt_client_types.h

sl_mqtt_client_t

MQTT Client Handle structure.

Public Attributes

sl_mqtt_client_connection_state_t	state Current state of MQTT client.
const sl_mqtt_broker_t *	broker Pointer to broker configuration, given at the time of connect() API.
const sl_mqtt_client_last_will_message_t *	last_will_message Pointer to last will message configuration, given at the time of connect() API.
const sl_mqtt_client_configuration_t *	client_configuration Pointer to client configuration, given at the time of connect() API.
sl_mqtt_client_topic_subscription_info_t *	subscription_list_head Pointer to the head of the subscription linked list.
sl_mqtt_client_event_handler_t	client_event_handler Function pointer to event handler given at the time of sl_mqtt_client_init.

Public Attribute Documentation

state

```
sl_mqtt_client_connection_state_t sl_mqtt_client_t::state
```

Current state of MQTT client.

Definition at line 184 of file components/service/mqtt/inc/sl_mqtt_client_types.h

broker

```
const sl_mqtt_broker_t* sl_mqtt_client_t::broker
```

Pointer to broker configuration, given at the time of connect() API.

Definition at line 185 of file components/service/mqtt/inc/sl_mqtt_client_types.h

last_will_message

```
const sl_mqtt_client_last_will_message_t* sl_mqtt_client_t::last_will_message
```

Pointer to last will message configuration, given at the time of connect() API.

Definition at line 187 of file components/service/mqtt/inc/sl_mqtt_client_types.h

client_configuration

```
const sl_mqtt_client_configuration_t* sl_mqtt_client_t::client_configuration
```

Pointer to client configuration, given at the time of connect() API.

Definition at line 189 of file components/service/mqtt/inc/sl_mqtt_client_types.h

subscription_list_head

```
sl_mqtt_client_topic_subscription_info_t* sl_mqtt_client_t::subscription_list_head
```

Pointer to the head of the subscription linked list.

Definition at line 191 of file components/service/mqtt/inc/sl_mqtt_client_types.h

client_event_handler

```
sl_mqtt_client_event_handler_t sl_mqtt_client_t::client_event_handler
```

Function pointer to event handler given at the time of sl_mqtt_client_init.

Definition at line 193 of file components/service/mqtt/inc/sl_mqtt_client_types.h

Constants

Constants

This section provides a reference to the MQTT API constants.

Enumerations

enum sl_mqtt_qos_t {

SL_MQTT_QOS_LEVEL_0
SL_MQTT_QOS_LEVEL_1
SL_MQTT_QOS_LEVEL_2

}

MQTT quality of service (QoS) levels.

enum sl_mqtt_client_connection_state_t {

SL_MQTT_CLIENT_DISCONNECTED
SL_MQTT_CLIENT_CONNECTING
SL_MQTT_CLIENT_CONNECTED
SL_MQTT_CLIENT_DISCONNECTING

}

MQTT Client connection states.

enum sl_mqtt_version_t {

SL_MQTT_VERSION_3
SL_MQTT_VERSION_3_1

}

MQTT Protocol version.

enum sl_mqtt_client_event_t {

SL_MQTT_CLIENT_CONNECTED_EVENT
SL_MQTT_CLIENT_DISCONNECTED_EVENT
SL_MQTT_CLIENT_MESSAGE_PUBLISHED_EVENT
SL_MQTT_CLIENT_MESSAGED_RECEIVED_EVENT
SL_MQTT_CLIENT_SUBSCRIBED_EVENT
SL_MQTT_CLIENT_UNSUBSCRIBED_EVENT
SL_MQTT_CLIENT_ERROR_EVENT

}

MQTT Client Events.

enum sl_mqtt_client_error_status_t {

SL_MQTT_CLIENT_CONNECT_FAILED
SL_MQTT_CLIENT_PUBLISH_FAILED
SL_MQTT_CLIENT_SUBSCRIBE_FAILED
SL_MQTT_CLIENT_UNSUBSCRIBED_FAILED
SL_MQTT_CLIENT_DISCONNECT_FAILED
SL_MQTT_CLIENT_UNKNKOWN_ERROR

}

MQTT Client error status.

Enumeration Documentation

sl_mqtt_qos_t

> sl_mqtt_qos_t

MQTT quality of service (QoS) levels.

Note

- Quality of service (QoS) level 2 not currently supported.

Enumerator	
SL_MQTT_QOS_LEVEL_0	MQTT QoS level 0.
SL_MQTT_QOS_LEVEL_1	MQTT QoS level 1.
SL_MQTT_QOS_LEVEL_2	MQTT QoS level 2 (not currently supported)

Definition at line `34` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

sl_mqtt_client_connection_state_t

> sl_mqtt_client_connection_state_t

MQTT Client connection states.

Enumerator	
SL_MQTT_CLIENT_DISCONNECTED	Initial state.
SL_MQTT_CLIENT_CONNECTING	Attains this state when the connect is called and await for results.
SL_MQTT_CLIENT_CONNECTED	Connection established with MQTT broker.
SL_MQTT_CLIENT_DISCONNECTING	Moves to this state when the disconnect is called and await for results.

Definition at line `41` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

sl_mqtt_version_t

> sl_mqtt_version_t

MQTT Protocol version.

Enumerator	
SL_MQTT_VERSION_3	MQTT Version 3.0.
SL_MQTT_VERSION_3_1	MQTT Version 3.1.

Definition at line `49` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

sl_mqtt_client_event_t

> sl_mqtt_client_event_t

MQTT Client Events.

Enumerator	
SL_MQTT_CLIENT_CONNECTED_EVENT	MQTT client connected event.
SL_MQTT_CLIENT_DISCONNECTED_EVENT	MQTT client disconnected event.
SL_MQTT_CLIENT_MESSAGE_PUBLISHED_EVENT	MQTT client message published event.
SL_MQTT_CLIENT_MESSAGED_RECEIVED_EVENT	MQTT client message received event.
SL_MQTT_CLIENT_SUBSCRIBED_EVENT	MQTT client subscribed event.
SL_MQTT_CLIENT_UNSUBSCRIBED_EVENT	MQTT client unsubscribed event.
SL_MQTT_CLIENT_ERROR_EVENT	MQTT client error event.

Definition at line `55` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

sl_mqtt_client_error_status_t

> sl_mqtt_client_error_status_t

MQTT Client error status.

Enumerator	
SL_MQTT_CLIENT_CONNECT_FAILED	MQTT client connect failed status.
SL_MQTT_CLIENT_PUBLISH_FAILED	MQTT client publish failed status.
SL_MQTT_CLIENT_SUBSCRIBE_FAILED	MQTT client subscribe failed status.
SL_MQTT_CLIENT_UNSUBSCRIBED_FAILED	MQTT client unsubscribe failed status.
SL_MQTT_CLIENT_DISCONNECT_FAILED	MQTT client disconnect failed status.
SL_MQTT_CLIENT_UNKNKOWN_ERROR	MQTT client unknown error status.

Definition at line `66` of file `components/service/mqtt/inc/sl_mqtt_client_types.h`

Overview

Overview

The WiSeConnect™ SDK v3.x uses Silicon Labs standard status codes. See the Gecko SDK Status Codes section for more details.

The following section provides a reference to additional status codes returned by the application programming interface (API) that are not included in the list of Silicon Labs standard status codes.

Additional Status Codes

Additional Status Codes

This section provides a reference to the additional status codes returned by the WiSeConnect™ SDK v3.x which are not included in the list of Silicon Labs standard status codes.

Macros

#define SL_STATUS_OS_OPERATION_FAILURE ((sl_status_t)0×0051)
OS operation failed.

#define SL_STATUS_BOOTUP_OPTIONS_NOT_SAVED ((sl_status_t)0×0052)
Bootup options not saved.

#define SL_STATUS_BOOTUP_OPTIONS_CHECKSUM_FAILURE ((sl_status_t)0×0053)
Bootup options checksum failed.

#define SL_STATUS_BOOTLOADER_VERSION_MISMATCH ((sl_status_t)0×0054)
Bootloader version mismatch.

#define SL_STATUS_WAITING_FOR_BOARD_READY ((sl_status_t)0×0055)
Waiting for board ready.

#define SL_STATUS_VALID_FIRMWARE_NOT_PRESENT ((sl_status_t)0×0056)
Invalid firmware.

#define SL_STATUS_INVALID_OPTION ((sl_status_t)0×0057)
Invalid option.

#define SL_STATUS_SPI_BUSY ((sl_status_t)0×0058)
SPI busy.

#define SL_STATUS_CARD_READY_TIMEOUT ((sl_status_t)0×0059)
Card ready not received.

#define SL_STATUS_FW_LOAD_OR_UPGRADE_TIMEOUT ((sl_status_t)0×005A)
Firmware upgrade timed out.

#define SL_STATUS_WIFI_DOES_NOT_EXIST ((sl_status_t)0×0B21)
Does not exist.

#define SL_STATUS_WIFI_NOT_AUTHENTICATED ((sl_status_t)0×0B22)
Not authenticated.

#define SL_STATUS_WIFI_NOT_KEYED ((sl_status_t)0×0B23)
Not keyed.

#define SL_STATUS_WIFI_IOCTL_FAIL ((sl_status_t)0×0B24)
IOCTL fail.

#define SL_STATUS_WIFI_BUFFER_UNAVAILABLE_TEMPORARY ((sl_status_t)0×0B25)
Buffer unavailable temporarily.

#define SL_STATUS_WIFI_BUFFER_UNAVAILABLE_PERMANENT ((sl_status_t)0×0B26)
Buffer unavailable permanently.

#define SL_STATUS_WIFI_WPS_PBC_OVERLAP ((sl_status_t)0×0B27)
WPS PBC overlap.

#define SL_STATUS_WIFI_CONNECTION_LOST ((sl_status_t)0×0B28)
Connection lost.

#define SL_STATUS_WIFI_OUT_OF_EVENT_HANDLER_SPACE ((sl_status_t)0×0B29)
Cannot add extra event handler.

#define SL_STATUS_WIFI_SEMAPHORE_ERROR ((sl_status_t)0×0B2A)
Error manipulating a semaphore.

#define SL_STATUS_WIFI_FLOW_CONTROLLED ((sl_status_t)0×0B2B)
Packet retrieval cancelled due to flow control.

#define SL_STATUS_WIFI_NO_CREDITS ((sl_status_t)0×0B2C)
Packet retrieval cancelled due to lack of bus credits.

#define SL_STATUS_WIFI_NO_PACKET_TO_SEND ((sl_status_t)0×0B2D)
Packet retrieval cancelled due to no pending packets.

#define SL_STATUS_WIFI_CORE_CLOCK_NOT_ENABLED ((sl_status_t)0×0B2E)
Core disabled due to no clock.

#define SL_STATUS_WIFI_CORE_IN_RESET ((sl_status_t)0×0B2F)
Core disabled - in reset.

#define SL_STATUS_WIFI_UNSUPPORTED ((sl_status_t)0×0B30)
Unsupported function.

#define SL_STATUS_WIFI_BUS_WRITE_REGISTER_ERROR ((sl_status_t)0×0B31)
Error writing to WLAN register.

#define SL_STATUS_WIFI_SDIO_BUS_UP_FAIL ((sl_status_t)0×0B32)
SDIO bus failed to come up.

#define SL_STATUS_WIFI_JOIN_IN_PROGRESS ((sl_status_t)0×0B33)
Join not finished yet.

#define SL_STATUS_WIFI_NETWORK_NOT_FOUND ((sl_status_t)0×0B34)
Specified network was not found.

#define SL_STATUS_WIFI_INVALID_JOIN_STATUS ((sl_status_t)0×0B35)
Join status error.

#define SL_STATUS_WIFI_UNKNOWN_INTERFACE ((sl_status_t)0×0B36)
Unknown interface specified.

#define SL_STATUS_WIFI_SDIO_RX_FAIL ((sl_status_t)0×0B37)
Error during SDIO receive.

#define SL_STATUS_WIFI_HWTAG_MISMATCH ((sl_status_t)0×0B38)
Hardware tag header corrupt.

#define SL_STATUS_WIFI_RX_BUFFER_ALLOC_FAIL ((sl_status_t)0×0B39)
Failed to allocate a buffer to receive into.

#define SL_STATUS_WIFI_BUS_READ_REGISTER_ERROR ((sl_status_t)0×0B3A)
Error reading a bus hardware register.

#define SL_STATUS_WIFI_THREAD_CREATE_FAILED ((sl_status_t)0×0B3B)
Failed to create a new thread.

#define SL_STATUS_WIFI_QUEUE_ERROR ((sl_status_t)0×0B3C)
Error manipulating a queue.

#define SL_STATUS_WIFI_BUFFER_POINTER_MOVE_ERROR ((sl_status_t)0×0B3D)
Error moving the current pointer of a buffer.

#define SL_STATUS_WIFI_BUFFER_SIZE_SET_ERROR ((sl_status_t)0×0B3E)
Error setting size of packet buffer.

#define SL_STATUS_WIFI_THREAD_STACK_NULL ((sl_status_t)0×0B3F)
Null stack pointer passed when non null was required.

#define SL_STATUS_WIFI_THREAD_DELETE_FAIL ((sl_status_t)0×0B40)
Error deleting a thread.

#define SL_STATUS_WIFI_SLEEP_ERROR ((sl_status_t)0×0B41)
Failed to put a thread to sleep.

#define SL_STATUS_WIFI_BUFFER_ALLOC_FAIL ((sl_status_t)0×0B42)
Failed to allocate a packet buffer.

#define SL_STATUS_WIFI_INTERFACE_NOT_UP ((sl_status_t)0×0B44)
Requested interface is not active.

#define SL_STATUS_WIFI_DELAY_TOO_LONG ((sl_status_t)0×0B45)
Requested delay is too long.

#define SL_STATUS_WIFI_INVALID_DUTY_CYCLE ((sl_status_t)0×0B46)
Duty cycle is outside limit 0 to 0.

#define SL_STATUS_WIFI_PMK_WRONG_LENGTH ((sl_status_t)0×0B47)
Returned pmk was the wrong length.

#define SL_STATUS_WIFI_UNKNOWN_SECURITY_TYPE ((sl_status_t)0×0B48)
AP security type was unknown.

#define SL_STATUS_WIFI_WEP_NOT_ALLOWED ((sl_status_t)0×0B49)
AP not allowed to use WEP - use Open instead.

#define SL_STATUS_WIFI_WPA_KEYLEN_BAD ((sl_status_t)0×0B4A)
WPA / WPA2 key length must be between 8 & 64 bytes.

#define SL_STATUS_WIFI_FILTER_NOT_FOUND ((sl_status_t)0×0B4B)
Specified filter id not found.

#define SL_STATUS_WIFI_SPI_ID_READ_FAIL ((sl_status_t)0×0B4C)
Failed to read 0xfeedbead SPI id from chip.

#define SL_STATUS_WIFI_SPI_SIZE_MISMATCH ((sl_status_t)0×0B4D)
Mismatch in sizes between SPI and SDPCM header.

#define SL_STATUS_WIFI_ADDRESS_ALREADY_REGISTERED ((sl_status_t)0×0B4E)
Attempt to register a multicast address twice.

#define SL_STATUS_WIFI_SDIO_RETRIES_EXCEEDED ((sl_status_t)0×0B4F)
SDIO transfer failed too many times.

#define SL_STATUS_WIFI_NULL_PTR_ARG ((sl_status_t)0×0B50)
Null Pointer argument passed to function.

#define SL_STATUS_WIFI_THREAD_FINISH_FAIL ((sl_status_t)0×0B51)
Error deleting a thread.

#define SL_STATUS_WIFI_WAIT_ABORTED ((sl_status_t)0×0B52)
Semaphore/Mutex wait has been aborted.

#define SL_STATUS_WIFI_QUEUE_MESSAGE_UNALIGNED ((sl_status_t)0×0B53)
Unaligned message in the queue.

#define SL_STATUS_WIFI_MUTEX_ERROR ((sl_status_t)0×0B54)
Error while Mutex operation.

#define SL_STATUS_WIFI_SECURE_LINK_DECRYPT_ERROR ((sl_status_t)0×0B57)
Error while decryption over secure link.

#define SL_STATUS_WIFI_SECURE_LINK_KEY_RENEGOTIATION_ERROR ((sl_status_t)0×0B59)
Error while renegotiation of key over secure link.

#define SL_STATUS_WIFI_INVALID_OPERMODE ((sl_status_t)0×0B60)
Invalid opermode provided.

#define SL_STATUS_WIFI_INVALID_ENCRYPTION_METHOD ((sl_status_t)0×0B61)
Invalid security encryption method provided.

#define SL_STATUS_TRNG_DUPLICATE_ENTROPY ((sl_status_t)0×0B62)
TRNG duplicate elements error.

#define SL_STATUS_CRYPTO_INVALID_PARAMETER ((sl_status_t)0×1CCFE)
Return when parameter passed to Crypto SAPI is invalid.

#define SL_STATUS_CRYPTO_INVALID_SIGNATURE ((sl_status_t)0×1CC9A)
Return in AEAD (CCM, GCM, Chachapoly) decryption function, when MAC generated during decryption does not match the MAC passed.

#define SL_STATUS_SI91X_SCAN_ISSUED_IN_ASSOCIATED_STATE ((sl_status_t)0×10002)
Scan command issued while device is already associated with an access point.

#define SL_STATUS_SI91X_NO_AP_FOUND ((sl_status_t)0×10003)
No AP found.

#define SL_STATUS_SI91X_INVALID_PSK_IN_WEP_SECURITY ((sl_status_t)0×10004)
Wrong PSK is issued while the device client tries to join an access point with WEP security enabled.

#define SL_STATUS_SI91X_INVALID_BAND ((sl_status_t)0×10005)
Invalid band.

#define SL_STATUS_SI91X_UNASSOCIATED ((sl_status_t)0×10006)
Association not done or in unassociated state.

#define SL_STATUS_SI91X_DEAUTHENTICATION_RECEIVED_FROM_AP ((sl_status_t)0×10008)
De-authentication received from AP.

#define SL_STATUS_SI91X_ASSOCIATION_FAILED ((sl_status_t)0×10009)
Failed to associate to access point during "Join".

#define SL_STATUS_SI91X_INVALID_CHANNEL ((sl_status_t)0×1000A)
Invalid channel.

#define SL_STATUS_SI91X_JOIN_AUTHENTICATION_FAILED ((sl_status_t)0×1000E)
Authentication failure during "Join". Unable to find AP during join which was found during scan.

#define SL_STATUS_SI91X_BEACON_MISSED_FROM_AP_DURING_JOIN ((sl_status_t)0×1000F)
Missed beacon from AP during join.

#define SL_STATUS_SI91X_INVALID_MAC_SUPPLIED ((sl_status_t)0×10013)
Non-existent MAC address supplied in "Disassociate" command.

#define SL_STATUS_SI91X_EAP_CONFIG_NOT_DONE ((sl_status_t)0×10014)
EAP configuration is not done.

#define SL_STATUS_SI91X_MEMORY_FAILED_FROM_MODULE ((sl_status_t)0×10015)
Memory allocation failed or Store configuration check sum failed.

#define SL_STATUS_SI91X_INSUFFICIENT_INFO ((sl_status_t)0×10016)
Information is wrong or insufficient in join command.

#define SL_STATUS_SI91X_NOT_AP_INTERFACE ((sl_status_t)0×10017)
Not an AP interface.

#define SL_STATUS_SI91X_INVALID_PUSH_BUTTON_SEQUENCE ((sl_status_t)0×10018)
Push button command given before the expiry of previous push button command.

#define SL_STATUS_SI91X_REJOIN_FAILURE ((sl_status_t)0×10019)
Access point not found. Rejoin failure.

#define SL_STATUS_SI91X_FREQUENCY_NOT_SUPPORTED ((sl_status_t)0×1001A)
Frequency not supported.

#define SL_STATUS_SI91X_INVALID_OPERMODE ((sl_status_t)0×1001B)
Invalid opermode.

#define SL_STATUS_SI91X_EAP_CONFIG_FAILED ((sl_status_t)0×1001C)
EAP configuration failed.

#define SL_STATUS_SI91X_P2P_CONFIG_FAILED ((sl_status_t)0×1001D)
P2P configuration failed.

#define SL_STATUS_SI91X_GROUP_OWNER_NEGOTIATION_FAILED ((sl_status_t)0×1001E)
Unable to start Group Owner negotiation.

#define SL_STATUS_SI91X_JOIN_TIMEOUT ((sl_status_t)0×10020)
Join timeout.

#define SL_STATUS_SI91X_COMMAND_GIVEN_IN_INVALID_STATE ((sl_status_t)0×10021)
Command given in incorrect state.

#define SL_STATUS_SI91X_INVALID_QUERY_GO_PARAMS ((sl_status_t)0×10022)
Query GO parameters issued in incorrect operating mode.

#define SL_STATUS_SI91X_ACCESS_POINT_FAILED ((sl_status_t)0×10023)
Unable to form access point.

#define SL_STATUS_SI91X_INVALID_SCAN_INFO ((sl_status_t)0×10024)
Wrong scan input parameters supplied to "Scan" command.

#define SL_STATUS_SI91X_COMMAND_ISSUED_IN_REJOIN_STATE ((sl_status_t)0×10025)
Command issued during re-join in progress.

#define SL_STATUS_SI91X_WRONG_PARAMETERS ((sl_status_t)0×10026)
Wrong parameters the command request.

#define SL_STATUS_SI91X_PROVISION_DISCOVERY_FAILED_IN_P2P ((sl_status_t)0×10027)
Provision discovery failed in P2P.

#define SL_STATUS_SI91X_INVALID_PSK_LENGTH ((sl_status_t)0×10028)
PSK length less than 8 bytes or more than 63 bytes.

#define SL_STATUS_SI91X_FAILED_TO_CLEAR_OR_SET_EAP_CERTIFICATE ((sl_status_t)0×10029)
Failed to clear or to set the Enterprise Certificate (Set Certificate).

#define SL_STATUS_SI91X_P2P_GO_NEGOTIATED_FAILED ((sl_status_t)0×1002A)
P2P Go negotiation failed.

#define SL_STATUS_SI91X_ASSOCIATION_TIMEOUT_IN_P2P_WPS_MODE ((sl_status_t)0×1002B)
Association between nodes failed in P2P WPS mode due to timeout.

#define SL_STATUS_SI91X_COMMAND_ISSUED_WHILE_INTERNAL_OPERATION ((sl_status_t)0×1002C)
If a command is issued by the Host when the device is internally executing auto-join or auto-create.

#define SL_STATUS_SI91X_INVALID_WEP_KEY_LEN ((sl_status_t)0×1002D)
WEP key is of wrong length.

#define SL_STATUS_SI91X_ICMP_REQUEST_TIMEOUT_ERROR ((sl_status_t)0×1002E)
ICMP request timeout error.

#define SL_STATUS_SI91X_ICMP_DATA_SIZE_EXCEED_MAX_LIMIT ((sl_status_t)0×1002F)
ICMP data size exceeds maximum limit.

#define SL_STATUS_SI91X_SEND_DATA_PACKET_EXCEED_LIMIT ((sl_status_t)0×10030)
Send data packet exceeded the limit or length that is mentioned (or) MQTT publish data and publish data length mismatched (or) MQTT Send data packet exceeded the limit.

#define SL_STATUS_SI91X_ARP_CACHE_NOT_FOUND ((sl_status_t)0×10031)
ARP cache entry not found.

#define SL_STATUS_SI91X_UART_COMMAND_TIMEOUT ((sl_status_t)0×10032)
UART command timeout happened.

#define SL_STATUS_SI91X_FIXED_DATA_RATE_NOT_SUPPORTED_BY_AP ((sl_status_t)0×10033)
Fixed data rate is not supported by connecting AP.

#define SL_STATUS_SI91X_USERNAME_PASSWORD_CLIENTID_TOPIC_MAX_LEN ((sl_status_t)0×10036)
Maximum length exceeded of username/password/client_id/topic in MQTT.

#define SL_STATUS_SI91X_INVALID_WPS_PIN ((sl_status_t)0×10037)
Wrong WPS PIN.

#define SL_STATUS_SI91X_INVALID_WPS_PIN_LEN ((sl_status_t)0×10038)
Wrong WPS PIN length.

#define SL_STATUS_SI91X_INVALID_PMK_LEN ((sl_status_t)0×10039)
Wrong PMK length.

#define SL_STATUS_SI91X_SSID_NOT_PRESENT_FOR_PMK_GENERATION ((sl_status_t)0×1003A)
SSID not present for PMK generation.

#define SL_STATUS_SI91X_SSID_INCORRECT_PMK_GENERATION ((sl_status_t)0×1003B)
SSID incorrect for PMK generation(more than 32 bytes).

#define SL_STATUS_SI91X_BAND_NOT_SUPPORTED ((sl_status_t)0×1003C)
Band not supported.

#define SL_STATUS_SI91X_INVALID_USR_STORE_CONFIGURATION_LEN ((sl_status_t)0×1003D)
User store configuration invalid length.

#define SL_STATUS_SI91X_INVALID_COMMAND_LEN ((sl_status_t)0×1003E)
Error in length of the command (Exceeds number of characters is mentioned in the PRM).

#define SL_STATUS_SI91X_DATA_PACKET_DROPPED ((sl_status_t)0×1003F)
Data packet dropped.

#define SL_STATUS_SI91X_WEP_KEY_NOT_GIVEN ((sl_status_t)0×10040)
WEP key not given.

#define SL_STATUS_SI91X_INVALID_STORE_CONFIG_PROFILE ((sl_status_t)0×10041)
Error in length of store config profile.

#define SL_STATUS_SI91X_MISSING_PSK_OR_PMK ((sl_status_t)0×10042)
PSK or PMK not given.

#define SL_STATUS_SI91X_INVALID_SECURITY_MODE_IN_JOIN_COMMAND ((sl_status_t)0×10043)
Security mode given in join command is invalid.

#define SL_STATUS_SI91X_MAX_BEACON_MISCOUNT ((sl_status_t)0×10044)
Beacon miscount reaches max beacon miscount (De-authentication due to beacon miss).

#define SL_STATUS_SI91X_DEAUTH_REQUEST_FROM_SUPPLICANT ((sl_status_t)0×10045)
De-authentication received from supplicant.

#define SL_STATUS_SI91X_DEAUTH_REQUEST_FROM_FROM_AP ((sl_status_t)0×10046)
De-authentication received from AP after channel switching.

#define SL_STATUS_SI91X_MISSED_SYNCHRONIZATION ((sl_status_t)0×10047)
Synchronization missed.

#define SL_STATUS_SI91X_AUTHENTICATION_TIMEOUT ((sl_status_t)0×10048)
Authentication timeout occurred.

#define SL_STATUS_SI91X_ASSOCIATION_TIMEOUT ((sl_status_t)0×10049)
Association timeout.

#define SL_STATUS_SI91X_BG_SCAN_NOT_ALLOWED ((sl_status_t)0×1004A)
BG scan in given channels is not allowed.

#define SL_STATUS_SI91X_SSID_MISMATCH ((sl_status_t)0×1004B)
Scanned SSID and SSID given in join are not matching.

#define SL_STATUS_SI91X_CLIENT_MAX_SUPPORTED_EXCEEDED ((sl_status_t)0×1004C)
Given number of clients exceeded max number of stations supported.

#define SL_STATUS_SI91X_HT_CAPABILITIES_NOT_SUPPORTED ((sl_status_t)0×1004D)
Given HT capabilities are not supported.

#define SL_STATUS_SI91X_UART_FLOW_NOT_SUPPORTED ((sl_status_t)0×1004E)
UART Flow control not supported.

#define SL_STATUS_SI91X_ZB_BT_BLE_PKT_RECEIVED ((sl_status_t)0×1004F)
ZB/BT/BLE packet received and protocol is not enabled.

#define SL_STATUS_SI91X_MGMT_PKT_DROPPED ((sl_status_t)0×10050)
MGMT pkt dropped.

#define SL_STATUS_SI91X_INVALID_RF_CURRENT_MODE ((sl_status_t)0×10051)
Invalid RF current mode.

#define SL_STATUS_SI91X_POWER_SAVE_NOT_SUPPORTED ((sl_status_t)0×10052)
Power save support is not present for a given interface.

#define SL_STATUS_SI91X_CONCURRENT_AP_IN_CONNECTED_STATE ((sl_status_t)0×10053)
Concurrent AP in connected state.

#define SL_STATUS_SI91X_CONNECTED_AP_OR_STATION_CHANNEL_MISMATCH ((sl_status_t)0×10054)
Connected AP or Station channel mismatch.

#define SL_STATUS_SI91X_IAP_COPROCESSOR_ERROR ((sl_status_t)0×10055)
IAP co processor error.

#define SL_STATUS_SI91X_WPS_NOT_SUPPORTED ((sl_status_t)0×10056)
WPS not supported in current operating mode.

#define SL_STATUS_SI91X_CONCURRENT_AP_CHANNEL_MISMATCH ((sl_status_t)0×10057)
Concurrent AP doesn't have same channel as connected station channel.

#define SL_STATUS_SI91X_PBC_SESSION_OVERLAP ((sl_status_t)0×10058)
PBC session overlap error.

#define SL_STATUS_SI91X_BT_FEATURE_BITMAP_INVALID ((sl_status_t)0×10059)
BT feature bit map invalid.

#define SL_STATUS_SI91X_FOUR_WAY_HANDSHAKE_FAILED ((sl_status_t)0×1005A)
4/4 confirmation of 4 way handshake failed.

#define SL_STATUS_SI91X_MAC_ADDRESS_NOT_PRESENT_IN_MAC_JOIN ((sl_status_t)0×1005B)
MAC address not present in MAC based join.

#define SL_STATUS_SI91X_CONCURRENT_MODE_DOWN ((sl_status_t)0×1005C)
Concurrent mode, both AP and client should be up, to enable configuration.

#define SL_STATUS_SI91X_CERTIFICATE_LOAD_NOT_ALLOWED_IN_FLASH ((sl_status_t)0×1005D)
Certificate load not allowed in flash.

#define SL_STATUS_SI91X_CERTIFICATE_LOAD_NOT_ALLOWED_IN_RAM ((sl_status_t)0×1005E)
Certificate load not allowed in RAM.

#define SL_STATUS_SI91X_WRONG_CERTIFICATE_LOAD_INDEX ((sl_status_t)0×1005F)
Certificate load failed due to wrong index.

#define SL_STATUS_SI91X_AP_HT_CAPS_NOT_ENABLED ((sl_status_t)0×10060)
AP HT caps not enabled.

#define SL_STATUS_SI91X_ADDRESS_FAMILY_NOT_SUPPORTED ((sl_status_t)0×10061)
Address family not supported by protocol.

#define SL_STATUS_SI91X_INVALID_BEACON_INTERVAL_OR_DTM_PERIOD ((sl_status_t)0×10062)
Invalid beacon interval or DTIM period provided.

#define SL_STATUS_SI91X_INVALID_CONFIG_RANGE_PROVIDED ((sl_status_t)0×10063)
Invalid range of the configuration provided.

#define SL_STATUS_SI91X_INVALID_CONFIG_TYPE ((sl_status_t)0×10064)
RTS THRESHOLD Config type is invalid.

#define SL_STATUS_SI91X_ERROR_WITH_MQTT_COMMAND ((sl_status_t)0×10065)
Error with MQTT command.

#define SL_STATUS_SI91X_HIGHER_LISTEN_INTERVAL ((sl_status_t)0×10066)
listen interval in power save is greater than that of join.

#define SL_STATUS_SI91X_WLAN_RADIO_DEREGISTERED ((sl_status_t)0×10067)
WLAN radio deregistered.

#define SL_STATUS_SI91X_SAE_FAILURE_DUE_TO_MULTIPLE_CONFIRM_FRAMES_FROM_AP ((sl_status_t)0×10069)
SAE failure due to multiple confirm frames from AP.

#define SL_STATUS_SI91X_EC_GROUP_STATION_UNSUPPORTED_BY_AP ((sl_status_t)0×1006A)
AP does not support the EC-group set by station.

#define SL_STATUS_TWT_SUPPORT_NOT_ENABLED_ERR ((sl_status_t)0×10070)
Occurs when HE_PARAMS_SUPPORT and SLI_SI91X_ENABLE_TWT_FEATURE macros are not enabled.

#define SL_STATUS_TWT_SETUP_ERR_SESSION_ACTIVE ((sl_status_t)0×10071)
Occurs when user tries to give TWT config command when there is an already active TWT session.

#define SL_STATUS_TWT_TEARDOWN_ERR_FLOWID_NOT_MATCHED ((sl_status_t)0×10072)
Occurs when TWT teardown command is given with a flow ID that does not match existing session flow ID.

#define SL_STATUS_TWT_TEARDOWN_ERR_NOACTIVE_SESS ((sl_status_t)0×10073)
Occurs when teardown command is given while there is no active session.

#define SL_STATUS_TWT_SESSION_NOT_FEASIBLE ((sl_status_t)0×10074)
This error code indicates that TWT session is not feasible. It is thrown only when TWT Auto Selection API is used.

#define SL_STATUS_SI91X_RESCHEDULE_TWT_NOT_SUPPORTED ((sl_status_t)0×10075)
AP does not support TWT information frame reception.

#define SL_STATUS_SI91X_RESCHEDULE_TWT_ERR_NOACTIVE_SESS ((sl_status_t)0×10076)
No active TWT agreement corresponding to given flow id.

#define SL_STATUS_SI91X_TWT_RESCHEDULING_IN_PROGRESS ((sl_status_t)0×10077)
Suspend or resume TWT action is in progress.

#define SL_STATUS_SI91X_RESCHEDULE_TWT_PACKET_CREATION_FAILED ((sl_status_t)0×10078)
TWT information frame packet creation failed in firmware.

#define SL_STATUS_SI91X_MQTT_ERROR_UNACCEPTABLE_PROTOCOL ((sl_status_t)0×10081)
The Server does not support the level of the MQTT protocol requested by the Client.

#define SL_STATUS_SI91X_MQTT_ERROR_IDENTIFIER_REJECTED ((sl_status_t)0×10082)
The Client identifier is correct UTF-8 but not allowed by the Server.

#define SL_STATUS_SI91X_MQTT_ERROR_SERVER_UNAVAILABLE ((sl_status_t)0×10083)
The Network Connection has been made but the MQTT service is unavailable.

#define SL_STATUS_SI91X_MQTT_ERROR_BAD_USERNAME_PASSWORD ((sl_status_t)0×10084)
The data in the user name or password is malformed.

#define SL_STATUS_SI91X_MQTT_ERROR_NOT_AUTHORIZED ((sl_status_t)0×10085)
The Client is not authorized to connect.

#define SL_STATUS_SI91X_DUPLICATE_ENTRY_EXISTS_IN_DNS_SERVER_TABLE ((sl_status_t)0×100AF)
Duplicate entry exists in DNS server table.

#define SL_STATUS_SI91X_NO_MEM_AVAILABLE ((sl_status_t)0×100B1)
Memory error: No memory available.

#define SL_STATUS_SI91X_INVALID_CHARACTERS_IN_JSON_OBJECT ((sl_status_t)0×100B2)
Invalid characters in JSON object.

#define SL_STATUS_SI91X_NO_KEY_FOUND ((sl_status_t)0×100B3)
Update commands: No such key found.

#define SL_STATUS_SI91X_NO_FILE_FOUND ((sl_status_t)0×100B4)
No such file found: Re-check filename.

#define SL_STATUS_SI91X_NO_WEB_PAGE_EXISTS_WITH_SAME_FILENAME ((sl_status_t)0×100B5)
No corresponding web page exists with same filename.

#define SL_STATUS_SI91X_SPACE_UNAVAILABLE_FOR_NEW_FILE ((sl_status_t)0×100B6)
Space unavailable for new file.

#define SL_STATUS_SI91X_INVALID_INPUT_DATA ((sl_status_t)0×100C1)
Invalid input data, Re-check filename, lengths, etc.

#define SL_STATUS_SI91X_NO_SPACE_AVAILABLE_FOR_NEW_FILE ((sl_status_t)0×100C2)
Space unavailable for new file.

#define SL_STATUS_SI91X_EXISTING_FILE_OVERWRITE ((sl_status_t)0×100C3)
Existing file overwrite: Exceeds size of previous file. Use erase and try again.

#define SL_STATUS_SI91X_NO_SUCH_FILE_FOUND ((sl_status_t)0×100C4)
No such file found. Re-check filename.

#define SL_STATUS_SI91X_MEMORY_ERROR ((sl_status_t)0×100C5)
Memory Error: No memory available.

#define SL_STATUS_SI91X_RECEIVED_MORE_WEB_PAGE_DATA ((sl_status_t)0×100C6)
Received more web page data than the total length initially specified.

#define SL_STATUS_SI91X_SET_REGION_ERROR ((sl_status_t)0×100C7)
Error in set region command.

#define SL_STATUS_SI91X_INVALID_WEBPAGE_CURRENT_CHUNK_LEN ((sl_status_t)0×100C8)
Web page current chunk length is incorrect.

#define SL_STATUS_SI91X_AP_SET_REGION_COMMAND_ERROR ((sl_status_t)0×100CA)
Error in AP set region command.

#define SL_STATUS_SI91X_AP_SET_REGION_COMMAND_PARAMETERS_ERROR ((sl_status_t)0×100CB)
Error in AP set region command parameters.

#define SL_STATUS_SI91X_REGION_CODE_NOT_SUPPORTED ((sl_status_t)0×100CC)
Region code not supported.

#define SL_STATUS_SI91X_EXTRACTING_COUNTRY_REGION_FROM_BEACON_FAILED ((sl_status_t)0×100CD)
Error in extracting country region from beacon.

#define SL_STATUS_SI91X_SELECTED_REGION_NOT_SUPPORTED ((sl_status_t)0×100CE)
Device does not have selected region support.

#define SL_STATUS_SI91X_SSL_TLS_CONTEXT_CREATION_FAILED ((sl_status_t)0×100D1)
SSL/TLS context create failed.

#define SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_FAIL ((sl_status_t)0×100D2)
SSL/TLS handshake failed. Socket will be closed.

#define SL_STATUS_SI91X_SSL_TLS_MAX_SOCKETS_REACHED ((sl_status_t)0×100D3)
SSL/TLS max sockets reached.

#define SL_STATUS_SI91X_FTP_CLIENT_NOT_CONNECTED ((sl_status_t)0×100D3)
FTP client is not connected.

#define SL_STATUS_SI91X_CIPHER_SET_FAILED ((sl_status_t)0×100D4)
Cipher set failure.

#define SL_STATUS_SI91X_HTTP_CREDENTIALS_MAX_LEN_EXCEEDED ((sl_status_t)0×100F1)
HTTP credentials maximum length exceeded.

#define SL_STATUS_SI91X_FEATURE_NOT_SUPPORTED ((sl_status_t)0×100F7)
Feature not supported.

#define SL_STATUS_SI91X_FLASH_WRITE_OR_FLASH_DATA_VERIFICATION_FAILED ((sl_status_t)0×100F8)
Unable to write to flash OR flash data verification failed.

#define SL_STATUS_SI91X_CALIBRATION_DATA_VERIFICATION_FAILED ((sl_status_t)0×100F9)
Calibration data verification failed.

#define SL_STATUS_SI91X_SNMP_INTERNAL_ERROR ((sl_status_t)0×10100)
SNMP internal error.

#define SL_STATUS_SI91X_SNMP_INVALID_IP_PROTOCOL ((sl_status_t)0×10104)
SNMP invalid IP protocol error.

#define SL_STATUS_SI91X_NO_DATA_RECEIVED_OR_RECEIVE_TIMEOUT ((sl_status_t)0×1BB01)
No data received or receive timeout.

#define SL_STATUS_SI91X_INSUFFICIENT_DATA_FOR_TIME_CONVERSION ((sl_status_t)0×1BB08)
Insufficient data for converting NTP time to mm-dd-yy time format.

#define SL_STATUS_SI91X_INVALID_SNTP_SERVER_ADDRESS ((sl_status_t)0×1BB0A)
Invalid SNTP server address.

#define SL_STATUS_SI91X_SNTP_CLIENT_NOT_STARTED ((sl_status_t)0×1BB0B)
SNTP client not started.

#define SL_STATUS_SI91X_SNTP_SERVER_UNAVAILABLE ((sl_status_t)0×1BB10)
SNTP server not available. Client will not get any time update service from current server.

#define SL_STATUS_SI91X_SNTP_SERVER_AUTHENTICATION_FAILED ((sl_status_t)0×1BB15)
SNTP server authentication failed.

#define SL_STATUS_SI91X_INTERNAL_ERROR ((sl_status_t)0×1BB0E)
Internal error.

#define SL_STATUS_SI91X_MULTICAST_IP_ADDRESS_ENTRY_NOT_FOUND ((sl_status_t)0×1BB16)
Entry not found for multicast IP address.

#define SL_STATUS_SI91X_MULTICAST_NO_ENTRIES_FOUND ((sl_status_t)0×1BB17)
No more entries found for multicast.

#define SL_STATUS_SI91X_IP_ADDRESS_ERROR ((sl_status_t)0×1BB21)
IP address error.

#define SL_STATUS_SI91X_SOCKET_ALREADY_BOUND ((sl_status_t)0×1BB22)
Socket already bound.

#define SL_STATUS_SI91X_PORT_UNAVAILABLE ((sl_status_t)0×1BB23)
Port not available.

#define SL_STATUS_SI91X_SOCKET_NOT_CREATED ((sl_status_t)0×1BB27)
Socket is not created.

#define SL_STATUS_SI91X_ICMP_REQUEST_FAILED ((sl_status_t)0×1BB29)
ICMP request failed.

#define SL_STATUS_SI91X_MAX_LISTEN_SOCKETS_REACHED ((sl_status_t)0×1BB33)
Maximum listen sockets reached.

#define SL_STATUS_SI91X_DHCP_DUPLICATE_LISTEN ((sl_status_t)0×1BB34)
DHCP duplicate listen.

#define SL_STATUS_SI91X_PORT_NOT_IN_CLOSE_STATE ((sl_status_t)0×1BB35)
Port not in closed state.

#define SL_STATUS_SI91X_SOCKET_CLOSED ((sl_status_t)0×1BB36)
Socket is closed or in process of closing.

#define SL_STATUS_SI91X_PROCESS_IN_PROGRESS ((sl_status_t)0×1BB37)
Process in progress.

#define SL_STATUS_SI91X_CONNECT_TO_NON_EXISTING_TCP_SERVER_SOCKET ((sl_status_t)0×1BB38)
Trying to connect non-existing TCP server socket.

#define SL_STATUS_SI91X_ERROR_IN_LEN_OF_THE_COMMAND ((sl_status_t)0×1BB3E)
Error in length of the command ('Exceeds number of characters' is mentioned in the PRM).

#define SL_STATUS_SI91X_WRONG_PACKET_INFO ((sl_status_t)0×1BB40)
Wrong packet info.

#define SL_STATUS_SI91X_SOCKET_STILL_BOUND ((sl_status_t)0×1BB42)
Socket is still bound.

#define SL_STATUS_SI91X_NO_FREE_PORT ((sl_status_t)0×1BB45)
No free port.

#define SL_STATUS_SI91X_INVALID_PORT ((sl_status_t)0×1BB46)
Invalid port.

#define SL_STATUS_SI91X_FEATURE_UNSUPPORTED ((sl_status_t)0×1BB4B)
Feature not supported.

#define SL_STATUS_SI91X_SOCKET_IN_UNCONNECTED_STATE ((sl_status_t)0×1BB50)
Socket is not in connected state. Disconnected from server. In case of FTP, user need to give destroy command after receiving this error.

#define SL_STATUS_SI91X_POP3_SESSION_CREATION_FAILED ((sl_status_t)0×1BB87)
POP3 session creation failed / POP3 session got terminated.

#define SL_STATUS_SI91X_DHCPV6_HANDSHAKE_FAIL ((sl_status_t)0×1BB9C)
DHCPv6 handshake failure.

#define SL_STATUS_SI91X_DHCP_INVALID_IP_RESPONSE ((sl_status_t)0×1BB9D)
DHCP invalid IP response.

#define SL_STATUS_SI91X_SMTP_AUTHENTICATION_ERROR ((sl_status_t)0×1BBA0)
SMTP authentication error.

#define SL_STATUS_SI91X_SMTP_OVER_SIZE_MAIL_DATA ((sl_status_t)0×1BBA1)
No DNS server was specified, SMTP over size mail data.

#define SL_STATUS_SI91X_SMTP_INVALID_SERVER_REPLY ((sl_status_t)0×1BBA2)
SMTP invalid server reply.

#define SL_STATUS_SI91X_SMTP_DNS_QUERY_FAILED ((sl_status_t)0×1BBA3)
DNS query failed, SMTP internal error.

#define SL_STATUS_SI91X_SMTP_BAD_DNS_ADDRESS ((sl_status_t)0×1BBA4)
Bad DNS address, SMTP server error code received.

#define SL_STATUS_SI91X_SMTP_INVALID_PARAMETERS ((sl_status_t)0×1BBA5)
SMTP invalid parameters.

#define SL_STATUS_SI91X_SMTP_PACKET_ALLOCATION_FAILED ((sl_status_t)0×1BBA6)
SMTP packet allocation failed.

#define SL_STATUS_SI91X_SMTP_GREET_REPLY_FAILED ((sl_status_t)0×1BBA7)
SMTP Greet reply failed.

#define SL_STATUS_SI91X_SMTP_PARAMETER_ERROR ((sl_status_t)0×1BBA8)
Parameter error, SMTP hello reply error.

#define SL_STATUS_SI91X_SMTP_MAIL_REPLY_ERROR ((sl_status_t)0×1BBA9)
SMTP mail reply error.

#define SL_STATUS_SI91X_SMTP_RCPT_REPLY_ERROR ((sl_status_t)0×1BBAA)
SMTP RCPT reply error.

#define SL_STATUS_SI91X_SMTP_MESSAGE_REPLY_ERROR ((sl_status_t)0×1BBAB)
SMTP message reply error.

#define SL_STATUS_SI91X_SMTP_DATA_REPLY_ERROR ((sl_status_t)0×1BBAC)
SMTP data reply error.

#define SL_STATUS_SI91X_SMTP_AUTH_REPLY_ERROR ((sl_status_t)0×1BBAD)
SMTP authentication reply error.

#define SL_STATUS_SI91X_SMTP_SERVER_REPLY_ERROR ((sl_status_t)0×1BBAE)
SMTP server error reply.

#define SL_STATUS_SI91X_DNS_DUPLICATE_ENTRY ((sl_status_t)0×1BBAF)
DNS duplicate entry.

#define SL_STATUS_SI91X_SMTP_OVERSIZE_SERVER_REPLY ((sl_status_t)0×1BBB1)
SMTP oversize server reply.

#define SL_STATUS_SI91X_SMTP_CLIENT_NOT_INITIALIZED ((sl_status_t)0×1BBB2)
SMTP client not initialized.

#define SL_STATUS_SI91X_DNS_IPV6_NOT_SUPPORTED ((sl_status_t)0×1BBB3)
DNS IPv6 not supported.

#define SL_STATUS_SI91X_INVALID_MAIL_INDEX_FOR_POP3_MAIL_RETRIEVE_COMMAND ((sl_status_t)0×1BBC5)
Invalid mail index for POP3 mail retrieve command.

#define SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_FAILED ((sl_status_t)0×1BBD2)
SSL/TLS handshake failed.

#define SL_STATUS_SI91X_FTP_CLIENT_DISCONNECTED ((sl_status_t)0×1BBD3)
FTP client is not connected or disconnected with the FTP server.

#define SL_STATUS_SI91X_FTP_CLIENT_NOT_DISCONNECTED ((sl_status_t)0×1BBD4)
FTP client is not disconnected.

#define SL_STATUS_SI91X_FTP_FILE_NOT_OPENED ((sl_status_t)0×1BBD5)
FTP file is not opened.

#define SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_TIMEOUT_OR_FTP_FILE_NOT_CLOSED ((sl_status_t)0×1BBD6)
SSL/TLS handshake timeout or FTP file is not closed.

#define SL_STATUS_SI91X_FTP_EXPECTED_1XX_RESPONSE_NOT_RECEIVED ((sl_status_t)0×1BBD9)
Expected [1XX response from FTP server but not received].

#define SL_STATUS_SI91X_FTP_EXPECTED_2XX_RESPONSE_NOT_RECEIVED ((sl_status_t)0×1BBDA)
Expected [2XX response from FTP server but not received].

#define SL_STATUS_SI91X_FTP_EXPECTED_22X_RESPONSE_NOT_RECEIVED ((sl_status_t)0×1BBDB)
Expected [22X response from FTP server but not received].

#define SL_STATUS_SI91X_FTP_EXPECTED_23X_RESPONSE_NOT_RECEIVED ((sl_status_t)0×1BBDC)
Expected [23X response from FTP server but not received].

#define SL_STATUS_SI91X_FTP_EXPECTED_3XX_RESPONSE_NOT_RECEIVED ((sl_status_t)0×1BBDD)
Expected [3XX response from FTP server but not received].

#define SL_STATUS_SI91X_FTP_EXPECTED_33X_RESPONSE_NOT_RECEIVED ((sl_status_t)0×1BBDE)
Expected [33X response from FTP server but not received].

#define SL_STATUS_SI91X_HTTP_TIMEOUT ((sl_status_t)0×1BBE1)
HTTP timeout.

#define SL_STATUS_SI91X_HTTP_FAILED ((sl_status_t)0×1BBE2)
HTTP failed.

#define SL_STATUS_SI91X_HTTP_PUT_CLIENT_TIMEOUT ((sl_status_t)0×1BBE7)
HTTP timeout for HTTP PUT client.

#define SL_STATUS_SI91X_AUTHENTICATION_ERROR ((sl_status_t)0×1BBEB)
Authentication error.

#define SL_STATUS_SI91X_INVALID_PACKET_LENGTH ((sl_status_t)0×1BBED)
Invalid packet length. Content length and received data length is mismatching.

#define SL_STATUS_SI91X_SERVER_RESPONDS_BEFORE_REQUEST_COMPLETE ((sl_status_t)0×1BBEF)
Server responds before HTTP client request is complete.

#define SL_STATUS_SI91X_HTTP_PASSWORD_TOO_LONG ((sl_status_t)0×1BBF0)
HTTP/HTTPS password is too long.

#define SL_STATUS_SI91X_MQTT_PING_TIMEOUT ((sl_status_t)0×1BBF1)
MQTT ping time out error.

#define SL_STATUS_SI91X_MQTT_COMMAND_SENT_IN_INCORRECT_STATE ((sl_status_t)0×1BBF2)
MQTT command sent in incorrect state.

#define SL_STATUS_SI91X_MQTT_ACK_TIMEOUT ((sl_status_t)0×1BBF3)
MQTT ACK time out error.

#define SL_STATUS_SI91X_POP3_INVALID_MAIL_INDEX ((sl_status_t)0×1BBFF)
POP3 error for invalid mail index.

#define SL_STATUS_SI91X_SOCKET_NOT_CONNECTED ((sl_status_t)0×1FFFF)
Listening TCP socket in device is not connected to the remote peer, or the LTCP socket is not yet opened in the device.

#define SL_STATUS_SI91X_SOCKET_LIMIT_EXCEEDED ((sl_status_t)0×1FFFE)
Sockets not available. The error comes if the host tries to open more than 10 sockets.

#define SL_STATUS_SI91X_HTTP_OTAF_INVALID_PACKET ((sl_status_t)0×1FFFD)
HTTP OTAF invalid packet.

#define SL_STATUS_SI91X_TCP_IP_INIT_FAILED ((sl_status_t)0×1FFFC)
TCP_IP initialization failed.

#define SL_STATUS_SI91X_CONCURRENT_IP_CREATION_ERROR ((sl_status_t)0×1FFFB)
Cannot create IP in same interface in concurrent mode.

#define SL_STATUS_SI91X_HTTP_OTAF_INCOMPLETE_PACKET ((sl_status_t)0×1FFF4)
HTTP OTAF incomplete packet.

#define SL_STATUS_SI91X_INVALID_STORE_CONFIGURATION_PROFILE ((sl_status_t)0×1FFF5)
Store configuration profile type mismatch or invalid profile type.

#define SL_STATUS_SI91X_MQTT_REMOTE_TERMINATE_ERROR ((sl_status_t)0×1FFF6)
MQTT remote terminate error.

#define SL_STATUS_SI91X_BYTE_STUFFING_ERROR_IN_AT_MODE ((sl_status_t)0×1FFF7)
Byte stuffing error in AT mode.

#define SL_STATUS_SI91X_INVALID_COMMAND_OR_OPERATION ((sl_status_t)0×1FFF8)
Invalid command (e.g. parameters insufficient or invalid in the command). Invalid operation (e.g. power save command with the same mode given twice, accessing wrong socket, creating more than allowed sockets).

#define SL_STATUS_SI91X_HTTP_OTAF_NO_PACKET ((sl_status_t)0×1FFF9)
HTTP OTAF no packet.

#define SL_STATUS_SI91X_TCP_SOCKET_NOT_CONNECTED ((sl_status_t)0×1FFFA)
TCP socket is not connected.

#define SL_STATUS_SI91X_MAX_STATION_COUNT_EXCEEDED ((sl_status_t)0×1FFC5)
Station count exceeded max station supported.

#define SL_STATUS_SI91X_UNABLE_TO_SEND_TCP_DATA ((sl_status_t)0×1FFC4)
Unable to send TCP data.

#define SL_STATUS_SI91X_SOCKET_BUFFER_TOO_SMALL ((sl_status_t)0×1FFBC)
Socket buffer too small.

#define SL_STATUS_SI91X_INVALID_CONTENT_IN_DNS_RESPONSE ((sl_status_t)0×1FFBB)
Invalid content in the DNS response to the DNS resolution query.

#define SL_STATUS_SI91X_DNS_CLASS_ERROR_IN_DNS_RESPONSE ((sl_status_t)0×1FFBA)
DNS class error in response to the DNS resolution query.

#define SL_STATUS_SI91X_DNS_COUNT_ERROR_IN_DNS_RESPONSE ((sl_status_t)0×1FFB8)
DNS count error in response to the DNS resolution query.

#define SL_STATUS_SI91X_DNS_RETURN_CODE_ERROR_IN_DNS_RESPONSE ((sl_status_t)0×1FFB7)
DNS return code error in the response to the DNS resolution query.

#define SL_STATUS_SI91X_DNS_OPCODE_ERROR_IN_DNS_RESPONSE ((sl_status_t)0×1FFB6)
DNS Opcode error in the response to the DNS resolution query.

#define SL_STATUS_SI91X_DNS_ID_MISMATCH ((sl_status_t)0×1FFB5)
DNS id mismatch between DNS resolution request and response.

#define SL_STATUS_SI91X_INVALID_INPUT_IN_DNS_QUERY ((sl_status_t)0×1FFAB)
An invalid input to the DNS resolution query.

#define SL_STATUS_SI91X_DNS_RESPONSE_TIMEOUT ((sl_status_t)0×1FF42)
DNS response was timed out.

#define SL_STATUS_SI91X_ARP_REQUEST_FAILURE ((sl_status_t)0×1FFA1)
ARP request failure.

#define SL_STATUS_SI91X_UNABLE_TO_UPDATE_TCP_WINDOW ((sl_status_t)0×1FF91)
Unable to update TCP window.

#define SL_STATUS_SI91X_DHCP_LEASE_EXPIRED ((sl_status_t)0×1FF9D)
DHCP lease time expired.

#define SL_STATUS_SI91X_DHCP_HANDSHAKE_FAILURE ((sl_status_t)0×1FF9C)
DHCP handshake failure.

#define SL_STATUS_SI91X_WEBSOCKET_CREATION_FAILED ((sl_status_t)0×1FF88)
This error is issued when WebSocket creation failed.

#define SL_STATUS_SI91X_TRYING_TO_CONNECT_NON_EXISTENT_TCP_SERVER_SOCKET ((sl_status_t)0×1FF87)
This error is issued when device tried to connect to a non-existent TCP server socket on the remote side.

#define SL_STATUS_SI91X_TRYING_TO_CLOSE_NON_EXISTENT_SOCKET ((sl_status_t)0×1FF86)
This error is issued when tried to close non-existent socket. or invalid socket descriptor.

#define SL_STATUS_SI91X_INVALID_SOCKET_PARAMETERS ((sl_status_t)0×1FF85)
Invalid socket parameters.

#define SL_STATUS_SI91X_FEATURE_NOT_AVAILABLE ((sl_status_t)0×1FF82)
Feature not supported.

#define SL_STATUS_SI91X_SOCKET_ALREADY_OPEN ((sl_status_t)0×1FF81)
Socket already open.

#define SL_STATUS_SI91X_MAX_SOCKETS_EXCEEDED ((sl_status_t)0×1FF80)
Attempt to open more than the maximum allowed number of sockets.

#define SL_STATUS_SI91X_DATA_LENGTH_EXCEEDS_MSS ((sl_status_t)0×1FF7E)
Data length exceeds mss.

#define SL_STATUS_SI91X_IP_CONFLICT_ERROR ((sl_status_t)0×1FF75)
DUT unable to configure IP address due to IP conflict.

#define SL_STATUS_SI91X_FEATURE_NOT_ENABLED ((sl_status_t)0×1FF74)
Feature not enabled.

#define SL_STATUS_SI91X_DHCP_SERVER_NOT_SET ((sl_status_t)0×1FF73)
DHCP server not set in AP mode.

#define SL_STATUS_SI91X_AP_SET_REGION_PARAM_ERROR ((sl_status_t)0×1FF71)
Error in AP set region command parameters.

#define SL_STATUS_SI91X_SSL_TLS_NOT_SUPPORTED ((sl_status_t)0×1FF70)
SSL/TLS not supported.

#define SL_STATUS_SI91X_JSON_NOT_SUPPORTED ((sl_status_t)0×1FF6F)
JSON not supported.

#define SL_STATUS_SI91X_INVALID_OPERATING_MODE ((sl_status_t)0×1FF6E)
Invalid operating mode.

#define SL_STATUS_SI91X_INVALID_SOCKET_CONFIG_PARAMS ((sl_status_t)0×1FF6D)
Invalid socket configuration parameters.

#define SL_STATUS_SI91X_WEBSOCKET_CREATION_TIMEOUT ((sl_status_t)0×1FF6C)
Web socket creation timeout.

#define SL_STATUS_SI91X_PARAM_MAX_VALUE_EXCEEDED ((sl_status_t)0×1FF6B)
Parameter maximum allowed value is exceeded.

#define SL_STATUS_SI91X_SOCKET_READ_TIMEOUT ((sl_status_t)0×1FF6A)
Socket read timeout.

#define SL_STATUS_SI91X_INVALID_COMMAND_SEQUENCE ((sl_status_t)0×1FF69)
Invalid command in sequence.

#define SL_STATUS_SI91X_DNS_RESPONSE_TIMEOUT_ERROR ((sl_status_t)0×1FF42)
DNS response timed out.

#define SL_STATUS_SI91X_HTTP_SOCKET_CREATION_FAILED ((sl_status_t)0×1FF41)
HTTP socket creation failed.

#define SL_STATUS_SI91X_TCP_CLOSE_BEFORE_RESPONSE_ERROR ((sl_status_t)0×1FF40)
TCP socket close command is issued before getting the response of the previous close command.

#define SL_STATUS_SI91X_WAIT_ON_HOST_FEATURE_NOT_ENABLED ((sl_status_t)0×1FF36)
'Wait On Host' feature not enabled.

#define SL_STATUS_SI91X_STORE_CONFIG_CHECKSUM_INVALID ((sl_status_t)0×1FF35)
Store configuration checksum validation failed.

#define SL_STATUS_SI91X_TCP_KEEP_ALIVE_TIMEOUT ((sl_status_t)0×1FF33)
TCP keep alive timed out.

#define SL_STATUS_SI91X_TCP_ACK_FAILED_FOR_SYN_ACK ((sl_status_t)0×1FF2D)
TCP ACK failed for TCP SYN-ACK.

#define SL_STATUS_SI91X_MEMORY_LIMIT_EXCEEDED ((sl_status_t)0×1FF2C)
Memory limit exceeded in a given operating mode.

#define SL_STATUS_SI91X_MEMORY_LIMIT_EXCEEDED_DURING_AUTO_JOIN ((sl_status_t)0×1FF2A)
Memory limit exceeded in an operating mode during auto join/create.

#define SL_STATUS_SI91X_PUF_OPERATION_BLOCKED ((sl_status_t)0×1CC2F)
PUF operation is blocked.

#define SL_STATUS_SI91X_PUF_ACTIVATION_CODE_INVALID ((sl_status_t)0×1CC31)
PUF activation code invalid.

#define SL_STATUS_SI91X_PUF_INPUT_PARAMETERS_INVALID ((sl_status_t)0×1CC32)
PUF input parameters invalid.

#define SL_STATUS_SI91X_PUF_IN_ERROR_STATE ((sl_status_t)0×1CC33)
PUF in error state.

#define SL_STATUS_SI91X_PUF_OPERATION_NOT_ALLOWED ((sl_status_t)0×1CC34)
PUF operation not allowed.

#define SL_STATUS_SI91X_PUF_OPERATION_FAILED ((sl_status_t)0×1CC35)
PUF operation failed.

#define SL_STATUS_SI91X_AUTO_JOIN_IN_PROGRESS ((sl_status_t)0×15A5A)
Auto join or user store configuration going on.

#define SL_STATUS_SI91X_RSNIE_FROM_AP_INVALID ((sl_status_t)0×1FFE1)
Improper RSNIE from AP to station.

#define SL_STATUS_SI91X_SNTP_MAX_ATTEMPTS_REACHED ((sl_status_t)0×1FF5F)
Reached maximum SNTP invalid attempts.

#define SL_STATUS_SI91X_FREQUENCY_OFFSET_ZER0 ((sl_status_t)0×100FC)
Frequency offset sent is zero.

#define SL_STATUS_SI91X_FREQUENCY_OFFSET_OUT_OF_LIMITS ((sl_status_t)0×100FB)
Frequency offset specified goes beyond upper or lower limits and indicates that frequency offset cannot be changed further.

Macro Definition Documentation

SL_STATUS_OS_OPERATION_FAILURE

```
#define SL_STATUS_OS_OPERATION_FAILURE
```

Value:

```
((sl_status_t)0×0051)
```

OS operation failed.

Definition at line 27 of file components/common/inc/sl_additional_status.h

SL_STATUS_BOOTUP_OPTIONS_NOT_SAVED

```
#define SL_STATUS_BOOTUP_OPTIONS_NOT_SAVED
```

Value:

```
((sl_status_t)0x0052)
```

Bootup options not saved.

Definition at line 28 of file components/common/inc/sl_additional_status.h

SL_STATUS_BOOTUP_OPTIONS_CHECKSUM_FAILURE

```
#define SL_STATUS_BOOTUP_OPTIONS_CHECKSUM_FAILURE
```

Value:

```
((sl_status_t)0x0053)
```

Bootup options checksum failed.

Definition at line 29 of file components/common/inc/sl_additional_status.h

SL_STATUS_BOOTLOADER_VERSION_MISMATCH

```
#define SL_STATUS_BOOTLOADER_VERSION_MISMATCH
```

Value:

```
((sl_status_t)0x0054)
```

Bootloader version mismatch.

Definition at line 30 of file components/common/inc/sl_additional_status.h

SL_STATUS_WAITING_FOR_BOARD_READY

```
#define SL_STATUS_WAITING_FOR_BOARD_READY
```

Value:

```
((sl_status_t)0x0055)
```

Waiting for board ready.

Definition at line 31 of file components/common/inc/sl_additional_status.h

SL_STATUS_VALID_FIRMWARE_NOT_PRESENT

```
#define SL_STATUS_VALID_FIRMWARE_NOT_PRESENT
```

Value:

```
((sl_status_t)0x0056)
```

Invalid firmware.

Definition at line 32 of file components/common/inc/sl_additional_status.h

SL_STATUS_INVALID_OPTION

```
#define SL_STATUS_INVALID_OPTION
```

Value:

```
((sl_status_t)0x0057)
```

Invalid option.

Definition at line 33 of file components/common/inc/sl_additional_status.h

SL_STATUS_SPI_BUSY

```
#define SL_STATUS_SPI_BUSY
```

Value:

```
((sl_status_t)0x0058)
```

SPI busy.

Definition at line 34 of file components/common/inc/sl_additional_status.h

SL_STATUS_CARD_READY_TIMEOUT

```
#define SL_STATUS_CARD_READY_TIMEOUT
```

Value:

```
((sl_status_t)0x0059)
```

Card ready not received.

Definition at line 35 of file components/common/inc/sl_additional_status.h

SL_STATUS_FW_LOAD_OR_UPGRADE_TIMEOUT

```
#define SL_STATUS_FW_LOAD_OR_UPGRADE_TIMEOUT
```

Value:

```
((sl_status_t)0x005A)
```

Firmware upgrade timed out.

Definition at line 36 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_DOES_NOT_EXIST

```
#define SL_STATUS_WIFI_DOES_NOT_EXIST
```

Value:

```
((sl_status_t)0x0B21)
```

Does not exist.

Definition at line 39 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_NOT_AUTHENTICATED

```
#define SL_STATUS_WIFI_NOT_AUTHENTICATED
```

Value:

```
((sl_status_t)0x0B22)
```

Not authenticated.

Definition at line 40 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_NOT_KEYED

```
#define SL_STATUS_WIFI_NOT_KEYED
```

Value:

```
((sl_status_t)0x0B23)
```

Not keyed.

Definition at line 41 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_IOCTL_FAIL

```
#define SL_STATUS_WIFI_IOCTL_FAIL
```

Value:

```
((sl_status_t)0x0B24)
```

IOCTL fail.

Definition at line 42 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_BUFFER_UNAVAILABLE_TEMPORARY

```
#define SL_STATUS_WIFI_BUFFER_UNAVAILABLE_TEMPORARY
```

Value:

```
((sl_status_t)0x0B25)
```

Buffer unavailable temporarily.

Definition at line 43 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_BUFFER_UNAVAILABLE_PERMANENT

```
#define SL_STATUS_WIFI_BUFFER_UNAVAILABLE_PERMANENT
```

Value:

```
((sl_status_t)0x0B26)
```

Buffer unavailable permanently.

Definition at line 44 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_WPS_PBC_OVERLAP

```
#define SL_STATUS_WIFI_WPS_PBC_OVERLAP
```

Value:

```
((sl_status_t)0x0B27)
```

WPS PBC overlap.

Definition at line 45 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_CONNECTION_LOST

```
#define SL_STATUS_WIFI_CONNECTION_LOST
```

Value:

```
((sl_status_t)0x0B28)
```

Connection lost.

Definition at line 46 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_OUT_OF_EVENT_HANDLER_SPACE

```
#define SL_STATUS_WIFI_OUT_OF_EVENT_HANDLER_SPACE
```

Value:

```
((sl_status_t)0x0B29)
```

Cannot add extra event handler.

Definition at line 47 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_SEMAPHORE_ERROR

```
#define SL_STATUS_WIFI_SEMAPHORE_ERROR
```

Value:

```
((sl_status_t)0x0B2A)
```

Error manipulating a semaphore.

Definition at line 48 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_FLOW_CONTROLLED

```
#define SL_STATUS_WIFI_FLOW_CONTROLLED
```

Value:

```
((sl_status_t)0x0B2B)
```

Packet retrieval cancelled due to flow control.

Definition at line 49 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_NO_CREDITS

```
#define SL_STATUS_WIFI_NO_CREDITS
```

Value:

```
((sl_status_t)0x0B2C)
```

Packet retrieval cancelled due to lack of bus credits.

Definition at line 50 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_NO_PACKET_TO_SEND

```
#define SL_STATUS_WIFI_NO_PACKET_TO_SEND
```

Value:

```
((sl_status_t)0×0B2D)
```

Packet retrieval cancelled due to no pending packets.

Definition at line 51 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_CORE_CLOCK_NOT_ENABLED

```
#define SL_STATUS_WIFI_CORE_CLOCK_NOT_ENABLED
```

Value:

```
((sl_status_t)0×0B2E)
```

Core disabled due to no clock.

Definition at line 53 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_CORE_IN_RESET

```
#define SL_STATUS_WIFI_CORE_IN_RESET
```

Value:

```
((sl_status_t)0×0B2F)
```

Core disabled - in reset.

Definition at line 54 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_UNSUPPORTED

```
#define SL_STATUS_WIFI_UNSUPPORTED
```

Value:

```
((sl_status_t)0×0B30)
```

Unsupported function.

Definition at line 55 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_BUS_WRITE_REGISTER_ERROR

```
#define SL_STATUS_WIFI_BUS_WRITE_REGISTER_ERROR
```

Value:

```
((sl_status_t)0×0B31)
```

Error writing to WLAN register.

Definition at line 56 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_SDIO_BUS_UP_FAIL

```
#define SL_STATUS_WIFI_SDIO_BUS_UP_FAIL
```

Value:

```
((sl_status_t)0x0B32)
```

SDIO bus failed to come up.

Definition at line 57 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_JOIN_IN_PROGRESS

```
#define SL_STATUS_WIFI_JOIN_IN_PROGRESS
```

Value:

```
((sl_status_t)0x0B33)
```

Join not finished yet.

Definition at line 58 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_NETWORK_NOT_FOUND

```
#define SL_STATUS_WIFI_NETWORK_NOT_FOUND
```

Value:

```
((sl_status_t)0x0B34)
```

Specified network was not found.

Definition at line 59 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_INVALID_JOIN_STATUS

```
#define SL_STATUS_WIFI_INVALID_JOIN_STATUS
```

Value:

```
((sl_status_t)0x0B35)
```

Join status error.

Definition at line 60 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_UNKNOWN_INTERFACE

```
#define SL_STATUS_WIFI_UNKNOWN_INTERFACE
```

Value:

```
((sl_status_t)0x0B36)
```

Unknown interface specified.

Definition at line 61 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_SDIO_RX_FAIL

```
#define SL_STATUS_WIFI_SDIO_RX_FAIL
```

Value:

```
((sl_status_t)0x0B37)
```

Error during SDIO receive.

Definition at line 62 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_HWTAG_MISMATCH

```
#define SL_STATUS_WIFI_HWTAG_MISMATCH
```

Value:

```
((sl_status_t)0x0B38)
```

Hardware tag header corrupt.

Definition at line 63 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_RX_BUFFER_ALLOC_FAIL

```
#define SL_STATUS_WIFI_RX_BUFFER_ALLOC_FAIL
```

Value:

```
((sl_status_t)0x0B39)
```

Failed to allocate a buffer to receive into.

Definition at line 64 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_BUS_READ_REGISTER_ERROR

```
#define SL_STATUS_WIFI_BUS_READ_REGISTER_ERROR
```

Value:

```
((sl_status_t)0x0B3A)
```

Error reading a bus hardware register.

Definition at line 65 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_THREAD_CREATE_FAILED

```
#define SL_STATUS_WIFI_THREAD_CREATE_FAILED
```

Value:

```
((sl_status_t)0x0B3B)
```

Failed to create a new thread.

Definition at line 66 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_QUEUE_ERROR

```
#define SL_STATUS_WIFI_QUEUE_ERROR
```

Value:

```
((sl_status_t)0x0B3C)
```

Error manipulating a queue.

Definition at line 67 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_BUFFER_POINTER_MOVE_ERROR

```
#define SL_STATUS_WIFI_BUFFER_POINTER_MOVE_ERROR
```

Value:

```
((sl_status_t)0x0B3D)
```

Error moving the current pointer of a buffer.

Definition at line 68 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_BUFFER_SIZE_SET_ERROR

```
#define SL_STATUS_WIFI_BUFFER_SIZE_SET_ERROR
```

Value:

```
((sl_status_t)0x0B3E)
```

Error setting size of packet buffer.

Definition at line 70 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_THREAD_STACK_NULL

```
#define SL_STATUS_WIFI_THREAD_STACK_NULL
```

Value:

```
((sl_status_t)0x0B3F)
```

Null stack pointer passed when non null was required.

Definition at line 71 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_THREAD_DELETE_FAIL

```
#define SL_STATUS_WIFI_THREAD_DELETE_FAIL
```

Value:

```
((sl_status_t)0x0B40)
```

Error deleting a thread.

Definition at line 73 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_SLEEP_ERROR

```
#define SL_STATUS_WIFI_SLEEP_ERROR
```

Value:

```
((sl_status_t)0x0B41)
```

Failed to put a thread to sleep.

Definition at line 74 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_BUFFER_ALLOC_FAIL

```
#define SL_STATUS_WIFI_BUFFER_ALLOC_FAIL
```

Value:

```
((sl_status_t)0x0B42)
```

Failed to allocate a packet buffer.

Definition at line 75 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_INTERFACE_NOT_UP

```
#define SL_STATUS_WIFI_INTERFACE_NOT_UP
```

Value:

```
((sl_status_t)0x0B44)
```

Requested interface is not active.

Definition at line 76 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_DELAY_TOO_LONG

```
#define SL_STATUS_WIFI_DELAY_TOO_LONG
```

Value:

```
((sl_status_t)0x0B45)
```

Requested delay is too long.

Definition at line 77 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_INVALID_DUTY_CYCLE

```
#define SL_STATUS_WIFI_INVALID_DUTY_CYCLE
```

Value:

```
((sl_status_t)0x0B46)
```

Duty cycle is outside limit 0 to 0.

Definition at line 78 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_PMK_WRONG_LENGTH

```
#define SL_STATUS_WIFI_PMK_WRONG_LENGTH
```

Value:

```
((sl_status_t)0x0B47)
```

Returned pmk was the wrong length.

Definition at line 79 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_UNKNOWN_SECURITY_TYPE

```
#define SL_STATUS_WIFI_UNKNOWN_SECURITY_TYPE
```

Value:

```
((sl_status_t)0x0B48)
```

AP security type was unknown.

Definition at line 80 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_WEP_NOT_ALLOWED

```
#define SL_STATUS_WIFI_WEP_NOT_ALLOWED
```

Value:

```
((sl_status_t)0x0B49)
```

AP not allowed to use WEP - use Open instead.

Definition at line 81 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_WPA_KEYLEN_BAD

```
#define SL_STATUS_WIFI_WPA_KEYLEN_BAD
```

Value:

```
((sl_status_t)0x0B4A)
```

WPA / WPA2 key length must be between 8 & 64 bytes.

Definition at line 82 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_FILTER_NOT_FOUND

```
#define SL_STATUS_WIFI_FILTER_NOT_FOUND
```

Value:

```
((sl_status_t)0x0B4B)
```

Specified filter id not found.

Definition at line 83 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_SPI_ID_READ_FAIL

```
#define SL_STATUS_WIFI_SPI_ID_READ_FAIL
```

Value:

```
((sl_status_t)0x0B4C)
```

Failed to read 0xfeedbead SPI id from chip.

Definition at line 84 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_SPI_SIZE_MISMATCH

```
#define SL_STATUS_WIFI_SPI_SIZE_MISMATCH
```

Value:

```
((sl_status_t)0x0B4D)
```

Mismatch in sizes between SPI and SDPCM header.

Definition at line 85 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_ADDRESS_ALREADY_REGISTERED

```
#define SL_STATUS_WIFI_ADDRESS_ALREADY_REGISTERED
```

Value:

```
((sl_status_t)0x0B4E)
```

Attempt to register a multicast address twice.

Definition at line 86 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_SDIO_RETRIES_EXCEEDED

```
#define SL_STATUS_WIFI_SDIO_RETRIES_EXCEEDED
```

Value:

```
((sl_status_t)0x0B4F)
```

SDIO transfer failed too many times.

Definition at line 88 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_NULL_PTR_ARG

```
#define SL_STATUS_WIFI_NULL_PTR_ARG
```

Value:

```
((sl_status_t)0x0B50)
```

Null Pointer argument passed to function.

Definition at line 89 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_THREAD_FINISH_FAIL

```
#define SL_STATUS_WIFI_THREAD_FINISH_FAIL
```

Value:

```
((sl_status_t)0x0B51)
```

Error deleting a thread.

Definition at line 90 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_WAIT_ABORTED

```
#define SL_STATUS_WIFI_WAIT_ABORTED
```

Value:

```
((sl_status_t)0x0B52)
```

Semaphore/Mutex wait has been aborted.

Definition at line 91 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_QUEUE_MESSAGE_UNALIGNED

```
#define SL_STATUS_WIFI_QUEUE_MESSAGE_UNALIGNED
```

Value:

```
((sl_status_t)0x0B53)
```

Unaligned message in the queue.

Definition at line 92 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_MUTEX_ERROR

```
#define SL_STATUS_WIFI_MUTEX_ERROR
```

Value:

```
((sl_status_t)0x0B54)
```

Error while Mutex operation.

Definition at line 93 of file components/common/inc/sl_additional_status.h

SL_STATUS_WIFI_SECURE_LINK_DECRYPT_ERROR

```
#define SL_STATUS_WIFI_SECURE_LINK_DECRYPT_ERROR
```

Value:

```
((sl_status_t)0x0B57)
```

Error while decryption over secure link.

Definition at line `94` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_WIFI_SECURE_LINK_KEY_RENEGOTIATION_ERROR

```
#define SL_STATUS_WIFI_SECURE_LINK_KEY_RENEGOTIATION_ERROR
```

Value:

```
((sl_status_t)0x0B59)
```

Error while renegotiation of key over secure link.

Definition at line `95` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_WIFI_INVALID_OPERMODE

```
#define SL_STATUS_WIFI_INVALID_OPERMODE
```

Value:

```
((sl_status_t)0x0B60)
```

Invalid opermode provided.

Definition at line `97` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_WIFI_INVALID_ENCRYPTION_METHOD

```
#define SL_STATUS_WIFI_INVALID_ENCRYPTION_METHOD
```

Value:

```
((sl_status_t)0x0B61)
```

Invalid security encryption method provided.

Definition at line `98` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_TRNG_DUPLICATE_ENTROPY

```
#define SL_STATUS_TRNG_DUPLICATE_ENTROPY
```

Value:

```
((sl_status_t)0x0B62)
```

TRNG duplicate elements error.

Definition at line 101 of file components/common/inc/sl_additional_status.h

SL_STATUS_CRYPTO_INVALID_PARAMETER

```
#define SL_STATUS_CRYPTO_INVALID_PARAMETER
```

Value:

```
((sl_status_t)0x1CCFE)
```

Return when parameter passed to Crypto SAPI is invalid.

Definition at line 104 of file components/common/inc/sl_additional_status.h

SL_STATUS_CRYPTO_INVALID_SIGNATURE

```
#define SL_STATUS_CRYPTO_INVALID_SIGNATURE
```

Value:

```
((sl_status_t)0x1CC9A)
```

Return in AEAD (CCM, GCM, Chachapoly) decryption function, when MAC generated during decryption does not match the MAC passed.

Definition at line 106 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SCAN_ISSUED_IN_ASSOCIATED_STATE

```
#define SL_STATUS_SI91X_SCAN_ISSUED_IN_ASSOCIATED_STATE
```

Value:

```
((sl_status_t)0x10002)
```

Scan command issued while device is already associated with an access point.

Definition at line 110 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_AP_FOUND

```
#define SL_STATUS_SI91X_NO_AP_FOUND
```

Value:

```
((sl_status_t)0x10003)
```

No AP found.

Definition at line 112 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_PSK_IN_WEP_SECURITY

```
#define SL_STATUS_SI91X_INVALID_PSK_IN_WEP_SECURITY
```

Value:

```
((sl_status_t)0x10004)
```

Wrong PSK is issued while the device client tries to join an access point with WEP security enabled.

Definition at line 113 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_BAND

```
#define SL_STATUS_SI91X_INVALID_BAND
```

Value:

```
((sl_status_t)0x10005)
```

Invalid band.

Definition at line 115 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_UNASSOCIATED

```
#define SL_STATUS_SI91X_UNASSOCIATED
```

Value:

```
((sl_status_t)0x10006)
```

Association not done or in unassociated state.

Definition at line 116 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DEAUTHENTICATION_RECEIVED_FROM_AP

```
#define SL_STATUS_SI91X_DEAUTHENTICATION_RECEIVED_FROM_AP
```

Value:

```
((sl_status_t)0x10008)
```

De-authentication received from AP.

Definition at line 117 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ASSOCIATION_FAILED

```
#define SL_STATUS_SI91X_ASSOCIATION_FAILED
```

Value:

```
((sl_status_t)0×10009)
```

Failed to associate to access point during "Join".

Definition at line 119 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_CHANNEL

```
#define SL_STATUS_SI91X_INVALID_CHANNEL
```

Value:

```
((sl_status_t)0×1000A)
```

Invalid channel.

Definition at line 121 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_JOIN_AUTHENTICATION_FAILED

```
#define SL_STATUS_SI91X_JOIN_AUTHENTICATION_FAILED
```

Value:

```
((sl_status_t)0×1000E)
```

Authentication failure during "Join". Unable to find AP during join which was found during scan.

Definition at line 122 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_BEACON_MISSED_FROM_AP_DURING_JOIN

```
#define SL_STATUS_SI91X_BEACON_MISSED_FROM_AP_DURING_JOIN
```

Value:

```
((sl_status_t)0×1000F)
```

Missed beacon from AP during join.

Definition at line 124 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_MAC_SUPPLIED

```
#define SL_STATUS_SI91X_INVALID_MAC_SUPPLIED
```

Value:

```
((sl_status_t)0×10013)
```

Non-existent MAC address supplied in "Disassociate" command.

Definition at line 125 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_EAP_CONFIG_NOT_DONE

```
#define SL_STATUS_SI91X_EAP_CONFIG_NOT_DONE
```

Value:

```
((sl_status_t)0x10014)
```

EAP configuration is not done.

Definition at line 127 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MEMORY_FAILED_FROM_MODULE

```
#define SL_STATUS_SI91X_MEMORY_FAILED_FROM_MODULE
```

Value:

```
((sl_status_t)0x10015)
```

Memory allocation failed or Store configuration check sum failed.

Definition at line 128 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INSUFFICIENT_INFO

```
#define SL_STATUS_SI91X_INSUFFICIENT_INFO
```

Value:

```
((sl_status_t)0x10016)
```

Information is wrong or insufficient in join command.

Definition at line 130 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NOT_AP_INTERFACE

```
#define SL_STATUS_SI91X_NOT_AP_INTERFACE
```

Value:

```
((sl_status_t)0x10017)
```

Not an AP interface.

Definition at line 132 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_PUSH_BUTTON_SEQUENCE

```
#define SL_STATUS_SI91X_INVALID_PUSH_BUTTON_SEQUENCE
```

Value:

```
((sl_status_t)0x10018)
```

Push button command given before the expiry of previous push button command.

Definition at line 133 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_REJOIN_FAILURE

```
#define SL_STATUS_SI91X_REJOIN_FAILURE
```

Value:

```
((sl_status_t)0x10019)
```

Access point not found. Rejoin failure.

Definition at line 135 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FREQUENCY_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_FREQUENCY_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0x1001A)
```

Frequency not supported.

Definition at line 136 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_OPERMODE

```
#define SL_STATUS_SI91X_INVALID_OPERMODE
```

Value:

```
((sl_status_t)0x1001B)
```

Invalid opermode.

Definition at line 137 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_EAP_CONFIG_FAILED

```
#define SL_STATUS_SI91X_EAP_CONFIG_FAILED
```

Value:

```
((sl_status_t)0x1001C)
```

EAP configuration failed.

Definition at line 138 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_P2P_CONFIG_FAILED

```
#define SL_STATUS_SI91X_P2P_CONFIG_FAILED
```

Value:

```
((sl_status_t)0x1001D)
```

P2P configuration failed.

Definition at line 139 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_GROUP_OWNER_NEGOTIATION_FAILED

```
#define SL_STATUS_SI91X_GROUP_OWNER_NEGOTIATION_FAILED
```

Value:

```
((sl_status_t)0x1001E)
```

Unable to start Group Owner negotiation.

Definition at line 140 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_JOIN_TIMEOUT

```
#define SL_STATUS_SI91X_JOIN_TIMEOUT
```

Value:

```
((sl_status_t)0x10020)
```

Join timeout.

Definition at line 142 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_COMMAND_GIVEN_IN_INVALID_STATE

```
#define SL_STATUS_SI91X_COMMAND_GIVEN_IN_INVALID_STATE
```

Value:

```
((sl_status_t)0x10021)
```

Command given in incorrect state.

Definition at line 143 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_QUERY_GO_PARAMS

```
#define SL_STATUS_SI91X_INVALID_QUERY_GO_PARAMS
```

Value:

```
((sl_status_t)0x10022)
```

Query GO parameters issued in incorrect operating mode.

Definition at line 144 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ACCESS_POINT_FAILED

```
#define SL_STATUS_SI91X_ACCESS_POINT_FAILED
```

Value:

```
((sl_status_t)0x10023)
```

Unable to form access point.

Definition at line 146 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_SCAN_INFO

```
#define SL_STATUS_SI91X_INVALID_SCAN_INFO
```

Value:

```
((sl_status_t)0x10024)
```

Wrong scan input parameters supplied to "Scan" command.

Definition at line 147 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_COMMAND_ISSUED_IN_REJOIN_STATE

```
#define SL_STATUS_SI91X_COMMAND_ISSUED_IN_REJOIN_STATE
```

Value:

```
((sl_status_t)0x10025)
```

Command issued during re-join in progress.

Definition at line 149 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WRONG_PARAMETERS

```
#define SL_STATUS_SI91X_WRONG_PARAMETERS
```

Value:

```
((sl_status_t)0x10026)
```

Wrong parameters the command request.

Definition at line 151 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_PROVISION_DISCOVERY_FAILED_IN_P2P

```
#define SL_STATUS_SI91X_PROVISION_DISCOVERY_FAILED_IN_P2P
```

Value:

```
((sl_status_t)0x10027)
```

Provision discovery failed in P2P.

Definition at line 152 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_PSK_LENGTH

```
#define SL_STATUS_SI91X_INVALID_PSK_LENGTH
```

Value:

```
((sl_status_t)0x10028)
```

PSK length less than 8 bytes or more than 63 bytes.

Definition at line 153 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FAILED_TO_CLEAR_OR_SET_EAP_CERTIFICATE

```
#define SL_STATUS_SI91X_FAILED_TO_CLEAR_OR_SET_EAP_CERTIFICATE
```

Value:

```
((sl_status_t)0x10029)
```

Failed to clear or to set the Enterprise Certificate (Set Certificate).

Definition at line 155 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_P2P_GO_NEGOTIATED_FAILED

```
#define SL_STATUS_SI91X_P2P_GO_NEGOTIATED_FAILED
```

Value:

```
((sl_status_t)0x1002A)
```

P2P Go negotiation failed.

Definition at line 157 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ASSOCIATION_TIMEOUT_IN_P2P_WPS_MODE

```
#define SL_STATUS_SI91X_ASSOCIATION_TIMEOUT_IN_P2P_WPS_MODE
```

Value:

```
((sl_status_t)0x1002B)
```

Association between nodes failed in P2P WPS mode due to timeout.

Definition at line 158 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_COMMAND_ISSUED_WHILE_INTERNAL_OPERATION

```
#define SL_STATUS_SI91X_COMMAND_ISSUED_WHILE_INTERNAL_OPERATION
```

Value:

```
((sl_status_t)0x1002C)
```

If a command is issued by the Host when the device is internally executing auto-join or auto-create.

Definition at line 160 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_WEP_KEY_LEN

```
#define SL_STATUS_SI91X_INVALID_WEP_KEY_LEN
```

Value:

```
((sl_status_t)0x1002D)
```

WEP key is of wrong length.

Definition at line 162 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ICMP_REQUEST_TIMEOUT_ERROR

```
#define SL_STATUS_SI91X_ICMP_REQUEST_TIMEOUT_ERROR
```

Value:

```
((sl_status_t)0x1002E)
```

ICMP request timeout error.

Definition at line 163 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ICMP_DATA_SIZE_EXCEED_MAX_LIMIT

```
#define SL_STATUS_SI91X_ICMP_DATA_SIZE_EXCEED_MAX_LIMIT
```

Value:

```
((sl_status_t)0x1002F)
```

ICMP data size exceeds maximum limit.

Definition at line 164 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SEND_DATA_PACKET_EXCEED_LIMIT

```
#define SL_STATUS_SI91X_SEND_DATA_PACKET_EXCEED_LIMIT
```

Value:

```
((sl_status_t)0x10030)
```

Send data packet exceeded the limit or length that is mentioned (or) MQTT publish data and publish data length mismatched (or) MQTT Send data packet exceeded the limit.

Definition at line 166 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ARP_CACHE_NOT_FOUND

```
#define SL_STATUS_SI91X_ARP_CACHE_NOT_FOUND
```

Value:

```
((sl_status_t)0x10031)
```

ARP cache entry not found.

Definition at line 168 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_UART_COMMAND_TIMEOUT

```
#define SL_STATUS_SI91X_UART_COMMAND_TIMEOUT
```

Value:

```
((sl_status_t)0x10032)
```

UART command timeout happened.

Definition at line 169 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FIXED_DATA_RATE_NOT_SUPPORTED_BY_AP

```
#define SL_STATUS_SI91X_FIXED_DATA_RATE_NOT_SUPPORTED_BY_AP
```

Value:

```
((sl_status_t)0x10033)
```

Fixed data rate is not supported by connecting AP.

Definition at line 170 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_USERNAME_PASSWORD_CLIENTID_TOPIC_MAX_LEN

```
#define SL_STATUS_SI91X_USERNAME_PASSWORD_CLIENTID_TOPIC_MAX_LEN
```

Value:

```
((sl_status_t)0x10036)
```

Maximum length exceeded of username/password/client_id/topic in MQTT.

Definition at line 172 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_WPS_PIN

```
#define SL_STATUS_SI91X_INVALID_WPS_PIN
```

Value:

```
((sl_status_t)0x10037)
```

Wrong WPS PIN.

Definition at line 174 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_WPS_PIN_LEN

```
#define SL_STATUS_SI91X_INVALID_WPS_PIN_LEN
```

Value:

```
((sl_status_t)0x10038)
```

Wrong WPS PIN length.

Definition at line 175 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_PMK_LEN

```
#define SL_STATUS_SI91X_INVALID_PMK_LEN
```

Value:

```
((sl_status_t)0×10039)
```

Wrong PMK length.

Definition at line 176 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SSID_NOT_PRESENT_FOR_PMK_GENERATION

```
#define SL_STATUS_SI91X_SSID_NOT_PRESENT_FOR_PMK_GENERATION
```

Value:

```
((sl_status_t)0×1003A)
```

SSID not present for PMK generation.

Definition at line 177 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SSID_INCORRECT_PMK_GENERATION

```
#define SL_STATUS_SI91X_SSID_INCORRECT_PMK_GENERATION
```

Value:

```
((sl_status_t)0×1003B)
```

SSID incorrect for PMK generation(more than 32 bytes).

Definition at line 179 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_BAND_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_BAND_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0×1003C)
```

Band not supported.

Definition at line 181 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_USR_STORE_CONFIGURATION_LEN

```
#define SL_STATUS_SI91X_INVALID_USR_STORE_CONFIGURATION_LEN
```

Value:

```
((sl_status_t)0×1003D)
```

User store configuration invalid length.

Definition at line 182 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_COMMAND_LEN

```
#define SL_STATUS_SI91X_INVALID_COMMAND_LEN
```

Value:

```
((sl_status_t)0x1003E)
```

Error in length of the command (Exceeds number of characters is mentioned in the PRM).

Definition at line 184 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DATA_PACKET_DROPPED

```
#define SL_STATUS_SI91X_DATA_PACKET_DROPPED
```

Value:

```
((sl_status_t)0x1003F)
```

Data packet dropped.

Definition at line 186 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WEP_KEY_NOT_GIVEN

```
#define SL_STATUS_SI91X_WEP_KEY_NOT_GIVEN
```

Value:

```
((sl_status_t)0x10040)
```

WEP key not given.

Definition at line 187 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_STORE_CONFIG_PROFILE

```
#define SL_STATUS_SI91X_INVALID_STORE_CONFIG_PROFILE
```

Value:

```
((sl_status_t)0x10041)
```

Error in length of store config profile.

Definition at line 188 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MISSING_PSK_OR_PMK

```
#define SL_STATUS_SI91X_MISSING_PSK_OR_PMK
```

Value:

```
((sl_status_t)0x10042)
```

PSK or PMK not given.

Definition at line 190 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_SECURITY_MODE_IN_JOIN_COMMAND

```
#define SL_STATUS_SI91X_INVALID_SECURITY_MODE_IN_JOIN_COMMAND
```

Value:

```
((sl_status_t)0x10043)
```

Security mode given in join command is invalid.

Definition at line 191 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MAX_BEACON_MISCOUNT

```
#define SL_STATUS_SI91X_MAX_BEACON_MISCOUNT
```

Value:

```
((sl_status_t)0x10044)
```

Beacon miscount reaches max beacon miscount (De-authentication due to beacon miss).

Definition at line 193 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DEAUTH_REQUEST_FROM_SUPPLICANT

```
#define SL_STATUS_SI91X_DEAUTH_REQUEST_FROM_SUPPLICANT
```

Value:

```
((sl_status_t)0x10045)
```

De-authentication received from supplicant.

Definition at line 195 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DEAUTH_REQUEST_FROM_FROM_AP

```
#define SL_STATUS_SI91X_DEAUTH_REQUEST_FROM_FROM_AP
```

Value:

```
((sl_status_t)0x10046)
```

De-authentication received from AP after channel switching.

Definition at line 197 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MISSED_SYNCHRONIZATION

```
#define SL_STATUS_SI91X_MISSED_SYNCHRONIZATION
```

Value:

```
((sl_status_t)0x10047)
```

Synchronization missed.

Definition at line 199 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_AUTHENTICATION_TIMEOUT

```
#define SL_STATUS_SI91X_AUTHENTICATION_TIMEOUT
```

Value:

```
((sl_status_t)0x10048)
```

Authentication timeout occurred.

Definition at line 200 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ASSOCIATION_TIMEOUT

```
#define SL_STATUS_SI91X_ASSOCIATION_TIMEOUT
```

Value:

```
((sl_status_t)0x10049)
```

Association timeout.

Definition at line 201 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_BG_SCAN_NOT_ALLOWED

```
#define SL_STATUS_SI91X_BG_SCAN_NOT_ALLOWED
```

Value:

```
((sl_status_t)0x1004A)
```

BG scan in given channels is not allowed.

Definition at line 202 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SSID_MISMATCH

```
#define SL_STATUS_SI91X_SSID_MISMATCH
```

Value:

```
((sl_status_t)0x1004B)
```

Scanned SSID and SSID given in join are not matching.

Definition at line 203 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CLIENT_MAX_SUPPORTED_EXCEEDED

```
#define SL_STATUS_SI91X_CLIENT_MAX_SUPPORTED_EXCEEDED
```

Value:

```
((sl_status_t)0x1004C)
```

Given number of clients exceeded max number of stations supported.

Definition at line 204 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HT_CAPABILITIES_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_HT_CAPABILITIES_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0x1004D)
```

Given HT capabilities are not supported.

Definition at line 206 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_UART_FLOW_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_UART_FLOW_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0x1004E)
```

UART Flow control not supported.

Definition at line 208 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ZB_BT_BLE_PKT_RECEIVED

```
#define SL_STATUS_SI91X_ZB_BT_BLE_PKT_RECEIVED
```

Value:

```
((sl_status_t)0x1004F)
```

ZB/BT/BLE packet received and protocol is not enabled.

Definition at line 209 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MGMT_PKT_DROPPED

```
#define SL_STATUS_SI91X_MGMT_PKT_DROPPED
```

Value:

```
((sl_status_t)0x10050)
```

MGMT pkt dropped.

Definition at line 211 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_RF_CURRENT_MODE

```
#define SL_STATUS_SI91X_INVALID_RF_CURRENT_MODE
```

Value:

```
((sl_status_t)0x10051)
```

Invalid RF current mode.

Definition at line 212 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_POWER_SAVE_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_POWER_SAVE_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0x10052)
```

Power save support is not present for a given interface.

Definition at line 213 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CONCURRENT_AP_IN_CONNECTED_STATE

```
#define SL_STATUS_SI91X_CONCURRENT_AP_IN_CONNECTED_STATE
```

Value:

```
((sl_status_t)0×10053)
```

Concurrent AP in connected state.

Definition at line 215 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CONNECTED_AP_OR_STATION_CHANNEL_MISMATCH

```
#define SL_STATUS_SI91X_CONNECTED_AP_OR_STATION_CHANNEL_MISMATCH
```

Value:

```
((sl_status_t)0×10054)
```

Connected AP or Station channel mismatch.

Definition at line 216 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_IAP_COPROCESSOR_ERROR

```
#define SL_STATUS_SI91X_IAP_COPROCESSOR_ERROR
```

Value:

```
((sl_status_t)0×10055)
```

IAP co processor error.

Definition at line 218 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WPS_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_WPS_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0×10056)
```

WPS not supported in current operating mode.

Definition at line 219 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CONCURRENT_AP_CHANNEL_MISMATCH

```
#define SL_STATUS_SI91X_CONCURRENT_AP_CHANNEL_MISMATCH
```

Value:

```
((sl_status_t)0×10057)
```

Concurrent AP doesn't have same channel as connected station channel.

Definition at line 220 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_PBC_SESSION_OVERLAP

```
#define SL_STATUS_SI91X_PBC_SESSION_OVERLAP
```

Value:

```
((sl_status_t)0x10058)
```

PBC session overlap error.

Definition at line 222 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_BT_FEATURE_BITMAP_INVALID

```
#define SL_STATUS_SI91X_BT_FEATURE_BITMAP_INVALID
```

Value:

```
((sl_status_t)0x10059)
```

BT feature bit map invalid.

Definition at line 223 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FOUR_WAY_HANDSHAKE_FAILED

```
#define SL_STATUS_SI91X_FOUR_WAY_HANDSHAKE_FAILED
```

Value:

```
((sl_status_t)0x1005A)
```

4/4 confirmation of 4 way handshake failed.

Definition at line 224 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MAC_ADDRESS_NOT_PRESENT_IN_MAC_JOIN

```
#define SL_STATUS_SI91X_MAC_ADDRESS_NOT_PRESENT_IN_MAC_JOIN
```

Value:

```
((sl_status_t)0x1005B)
```

MAC address not present in MAC based join.

Definition at line 226 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CONCURRENT_MODE_DOWN

```
#define SL_STATUS_SI91X_CONCURRENT_MODE_DOWN
```

Value:

```
((sl_status_t)0x1005C)
```

Concurrent mode, both AP and client should be up, to enable configuration.

Definition at line 228 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CERTIFICATE_LOAD_NOT_ALLOWED_IN_FLASH

```
#define SL_STATUS_SI91X_CERTIFICATE_LOAD_NOT_ALLOWED_IN_FLASH
```

Value:

```
((sl_status_t)0x1005D)
```

Certificate load not allowed in flash.

Definition at line 230 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CERTIFICATE_LOAD_NOT_ALLOWED_IN_RAM

```
#define SL_STATUS_SI91X_CERTIFICATE_LOAD_NOT_ALLOWED_IN_RAM
```

Value:

```
((sl_status_t)0x1005E)
```

Certificate load not allowed in RAM.

Definition at line 232 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WRONG_CERTIFICATE_LOAD_INDEX

```
#define SL_STATUS_SI91X_WRONG_CERTIFICATE_LOAD_INDEX
```

Value:

```
((sl_status_t)0x1005F)
```

Certificate load failed due to wrong index.

Definition at line 234 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_AP_HT_CAPS_NOT_ENABLED

```
#define SL_STATUS_SI91X_AP_HT_CAPS_NOT_ENABLED
```

Value:

```
((sl_status_t)0×10060)
```

AP HT caps not enabled.

Definition at line 236 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ADDRESS_FAMILY_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_ADDRESS_FAMILY_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0×10061)
```

Address family not supported by protocol.

Definition at line 237 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_BEACON_INTERVAL_OR_DTM_PERIOD

```
#define SL_STATUS_SI91X_INVALID_BEACON_INTERVAL_OR_DTM_PERIOD
```

Value:

```
((sl_status_t)0×10062)
```

Invalid beacon interval or DTIM period provided.

Definition at line 239 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_CONFIG_RANGE_PROVIDED

```
#define SL_STATUS_SI91X_INVALID_CONFIG_RANGE_PROVIDED
```

Value:

```
((sl_status_t)0×10063)
```

Invalid range of the configuration provided.

Definition at line 241 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_CONFIG_TYPE

```
#define SL_STATUS_SI91X_INVALID_CONFIG_TYPE
```

Value:

```
((sl_status_t)0×10064)
```

RTS THRESHOLD Config type is invalid.

Definition at line 243 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ERROR_WITH_MQTT_COMMAND

```
#define SL_STATUS_SI91X_ERROR_WITH_MQTT_COMMAND
```

Value:

```
((sl_status_t)0x10065)
```

Error with MQTT command.

Definition at line 244 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HIGHER_LISTEN_INTERVAL

```
#define SL_STATUS_SI91X_HIGHER_LISTEN_INTERVAL
```

Value:

```
((sl_status_t)0x10066)
```

listen interval in power save is greater than that of join.

Definition at line 245 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WLAN_RADIO_DEREGISTERED

```
#define SL_STATUS_SI91X_WLAN_RADIO_DEREGISTERED
```

Value:

```
((sl_status_t)0x10067)
```

WLAN radio deregistered.

Definition at line 247 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SAE_FAILURE_DUE_TO_MULTIPLE_CONFIRM_FRAMES_FROM_AP

```
#define SL_STATUS_SI91X_SAE_FAILURE_DUE_TO_MULTIPLE_CONFIRM_FRAMES_FROM_AP
```

Value:

```
((sl_status_t)0x10069)
```

SAE failure due to multiple confirm frames from AP.

Definition at line 248 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_EC_GROUP_STATION_UNSUPPORTED_BY_AP

```
#define SL_STATUS_SI91X_EC_GROUP_STATION_UNSUPPORTED_BY_AP
```

Value:

```
((sl_status_t)0x1006A)
```

AP does not support the EC-group set by station.

Definition at line 250 of file components/common/inc/sl_additional_status.h

SL_STATUS_TWT_SUPPORT_NOT_ENABLED_ERR

```
#define SL_STATUS_TWT_SUPPORT_NOT_ENABLED_ERR
```

Value:

```
((sl_status_t)0x10070)
```

Occurs when HE_PARAMS_SUPPORT and SLI_SI91X_ENABLE_TWT_FEATURE macros are not enabled.

Definition at line 252 of file components/common/inc/sl_additional_status.h

SL_STATUS_TWT_SETUP_ERR_SESSION_ACTIVE

```
#define SL_STATUS_TWT_SETUP_ERR_SESSION_ACTIVE
```

Value:

```
((sl_status_t)0x10071)
```

Occurs when user tries to give TWT config command when there is an already active TWT session.

Definition at line 254 of file components/common/inc/sl_additional_status.h

SL_STATUS_TWT_TEARDOWN_ERR_FLOWID_NOT_MATCHED

```
#define SL_STATUS_TWT_TEARDOWN_ERR_FLOWID_NOT_MATCHED
```

Value:

```
((sl_status_t)0x10072)
```

Occurs when TWT teardown command is given with a flow ID that does not match existing session flow ID.

Definition at line 256 of file components/common/inc/sl_additional_status.h

SL_STATUS_TWT_TEARDOWN_ERR_NOACTIVE_SESS

```
#define SL_STATUS_TWT_TEARDOWN_ERR_NOACTIVE_SESS
```

Value:

```
((sl_status_t)0×10073)
```

Occurs when teardown command is given while there is no active session.

Definition at line 258 of file components/common/inc/sl_additional_status.h

SL_STATUS_TWT_SESSION_NOT_FEASIBLE

```
#define SL_STATUS_TWT_SESSION_NOT_FEASIBLE
```

Value:

```
((sl_status_t)0×10074)
```

This error code indicates that TWT session is not feasible. It is thrown only when TWT Auto Selection API is used.

Definition at line 260 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_RESCHEDULE_TWT_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_RESCHEDULE_TWT_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0×10075)
```

AP does not support TWT information frame reception.

Definition at line 262 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_RESCHEDULE_TWT_ERR_NOACTIVE_SESS

```
#define SL_STATUS_SI91X_RESCHEDULE_TWT_ERR_NOACTIVE_SESS
```

Value:

```
((sl_status_t)0×10076)
```

No active TWT agreement corresponding to given flow id.

Definition at line 264 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_TWT_RESCHEDULING_IN_PROGRESS

```
#define SL_STATUS_SI91X_TWT_RESCHEDULING_IN_PROGRESS
```

Value:

```
((sl_status_t)0×10077)
```

Suspend or resume TWT action is in progress.

Definition at line 266 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_RESCHEDULE_TWT_PACKET_CREATION_FAILED

```
#define SL_STATUS_SI91X_RESCHEDULE_TWT_PACKET_CREATION_FAILED
```

Value:

```
((sl_status_t)0x10078)
```

TWT information frame packet creation failed in firmware.

Definition at line 268 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_ERROR_UNACCEPTABLE_PROTOCOL

```
#define SL_STATUS_SI91X_MQTT_ERROR_UNACCEPTABLE_PROTOCOL
```

Value:

```
((sl_status_t)0x10081)
```

The Server does not support the level of the MQTT protocol requested by the Client.

Definition at line 270 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_ERROR_IDENTIFIER_REJECTED

```
#define SL_STATUS_SI91X_MQTT_ERROR_IDENTIFIER_REJECTED
```

Value:

```
((sl_status_t)0x10082)
```

The Client identifier is correct UTF-8 but not allowed by the Server.

Definition at line 272 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_ERROR_SERVER_UNAVAILABLE

```
#define SL_STATUS_SI91X_MQTT_ERROR_SERVER_UNAVAILABLE
```

Value:

```
((sl_status_t)0x10083)
```

The Network Connection has been made but the MQTT service is unavailable.

Definition at line 274 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_ERROR_BAD_USERNAME_PASSWORD

```
#define SL_STATUS_SI91X_MQTT_ERROR_BAD_USERNAME_PASSWORD
```

Value:

```
((sl_status_t)0x10084)
```

The data in the user name or password is malformed.

Definition at line 276 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_ERROR_NOT_AUTHORIZED

```
#define SL_STATUS_SI91X_MQTT_ERROR_NOT_AUTHORIZED
```

Value:

```
((sl_status_t)0x10085)
```

The Client is not authorized to connect.

Definition at line 278 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DUPLICATE_ENTRY_EXISTS_IN_DNS_SERVER_TABLE

```
#define SL_STATUS_SI91X_DUPLICATE_ENTRY_EXISTS_IN_DNS_SERVER_TABLE
```

Value:

```
((sl_status_t)0x100AF)
```

Duplicate entry exists in DNS server table.

Definition at line 279 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_MEM_AVAILABLE

```
#define SL_STATUS_SI91X_NO_MEM_AVAILABLE
```

Value:

```
((sl_status_t)0x100B1)
```

Memory error: No memory available.

Definition at line 281 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_CHARACTERS_IN_JSON_OBJECT

```
#define SL_STATUS_SI91X_INVALID_CHARACTERS_IN_JSON_OBJECT
```

Value:

```
((sl_status_t)0x100B2)
```

Invalid characters in JSON object.

Definition at line 282 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_KEY_FOUND

```
#define SL_STATUS_SI91X_NO_KEY_FOUND
```

Value:

```
((sl_status_t)0x100B3)
```

Update commands: No such key found.

Definition at line 283 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_FILE_FOUND

```
#define SL_STATUS_SI91X_NO_FILE_FOUND
```

Value:

```
((sl_status_t)0x100B4)
```

No such file found: Re-check filename.

Definition at line 284 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_WEB_PAGE_EXISTS_WITH_SAME_FILENAME

```
#define SL_STATUS_SI91X_NO_WEB_PAGE_EXISTS_WITH_SAME_FILENAME
```

Value:

```
((sl_status_t)0x100B5)
```

No corresponding web page exists with same filename.

Definition at line 285 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SPACE_UNAVAILABLE_FOR_NEW_FILE

```
#define SL_STATUS_SI91X_SPACE_UNAVAILABLE_FOR_NEW_FILE
```

Value:

```
((sl_status_t)0x100B6)
```

Space unavailable for new file.

Definition at line 287 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_INPUT_DATA

```
#define SL_STATUS_SI91X_INVALID_INPUT_DATA
```

Value:

```
((sl_status_t)0x100C1)
```

Invalid input data, Re-check filename, lengths, etc.

Definition at line 288 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_SPACE_AVAILABLE_FOR_NEW_FILE

```
#define SL_STATUS_SI91X_NO_SPACE_AVAILABLE_FOR_NEW_FILE
```

Value:

```
((sl_status_t)0x100C2)
```

Space unavailable for new file.

Definition at line 290 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_EXISTING_FILE_OVERWRITE

```
#define SL_STATUS_SI91X_EXISTING_FILE_OVERWRITE
```

Value:

```
((sl_status_t)0x100C3)
```

Existing file overwrite: Exceeds size of previous file. Use erase and try again.

Definition at line 291 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_SUCH_FILE_FOUND

```
#define SL_STATUS_SI91X_NO_SUCH_FILE_FOUND
```

Value:

```
((sl_status_t)0x100C4)
```

No such file found. Re-check filename.

Definition at line 293 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MEMORY_ERROR

```
#define SL_STATUS_SI91X_MEMORY_ERROR
```

Value:

```
((sl_status_t)0x100C5)
```

Memory Error: No memory available.

Definition at line 294 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_RECEIVED_MORE_WEB_PAGE_DATA

```
#define SL_STATUS_SI91X_RECEIVED_MORE_WEB_PAGE_DATA
```

Value:

```
((sl_status_t)0x100C6)
```

Received more web page data than the total length initially specified.

Definition at line 295 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SET_REGION_ERROR

```
#define SL_STATUS_SI91X_SET_REGION_ERROR
```

Value:

```
((sl_status_t)0x100C7)
```

Error in set region command.

Definition at line 297 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_WEBPAGE_CURRENT_CHUNK_LEN

```
#define SL_STATUS_SI91X_INVALID_WEBPAGE_CURRENT_CHUNK_LEN
```

Value:

```
((sl_status_t)0x100C8)
```

Web page current chunk length is incorrect.

Definition at line 298 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_AP_SET_REGION_COMMAND_ERROR

```
#define SL_STATUS_SI91X_AP_SET_REGION_COMMAND_ERROR
```

Value:

```
((sl_status_t)0×100CA)
```

Error in AP set region command.

Definition at line 300 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_AP_SET_REGION_COMMAND_PARAMETERS_ERROR

```
#define SL_STATUS_SI91X_AP_SET_REGION_COMMAND_PARAMETERS_ERROR
```

Value:

```
((sl_status_t)0×100CB)
```

Error in AP set region command parameters.

Definition at line 301 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_REGION_CODE_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_REGION_CODE_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0×100CC)
```

Region code not supported.

Definition at line 303 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_EXTRACTING_COUNTRY_REGION_FROM_BEACON_FAILED

```
#define SL_STATUS_SI91X_EXTRACTING_COUNTRY_REGION_FROM_BEACON_FAILED
```

Value:

```
((sl_status_t)0×100CD)
```

Error in extracting country region from beacon.

Definition at line 304 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SELECTED_REGION_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_SELECTED_REGION_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0×100CE)
```

Device does not have selected region support.

Definition at line 306 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SSL_TLS_CONTEXT_CREATION_FAILED

```
#define SL_STATUS_SI91X_SSL_TLS_CONTEXT_CREATION_FAILED
```

Value:

```
((sl_status_t)0x100D1)
```

SSL/TLS context create failed.

Definition at line 308 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_FAIL

```
#define SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_FAIL
```

Value:

```
((sl_status_t)0x100D2)
```

SSL/TLS handshake failed. Socket will be closed.

Definition at line 309 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SSL_TLS_MAX_SOCKETS_REACHED

```
#define SL_STATUS_SI91X_SSL_TLS_MAX_SOCKETS_REACHED
```

Value:

```
((sl_status_t)0x100D3)
```

SSL/TLS max sockets reached.

Definition at line 311 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FTP_CLIENT_NOT_CONNECTED

```
#define SL_STATUS_SI91X_FTP_CLIENT_NOT_CONNECTED
```

Value:

```
((sl_status_t)0x100D3)
```

FTP client is not connected.

Definition at line 312 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CIPHER_SET_FAILED

```
#define SL_STATUS_SI91X_CIPHER_SET_FAILED
```

Value:

```
((sl_status_t)0x100D4)
```

Cipher set failure.

Definition at line 313 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_CREDENTIALS_MAX_LEN_EXCEEDED

```
#define SL_STATUS_SI91X_HTTP_CREDENTIALS_MAX_LEN_EXCEEDED
```

Value:

```
((sl_status_t)0x100F1)
```

HTTP credentials maximum length exceeded.

Definition at line 314 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FEATURE_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_FEATURE_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0x100F7)
```

Feature not supported.

Definition at line 316 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FLASH_WRITE_OR_FLASH_DATA_VERIFICATION_FAILED

```
#define SL_STATUS_SI91X_FLASH_WRITE_OR_FLASH_DATA_VERIFICATION_FAILED
```

Value:

```
((sl_status_t)0x100F8)
```

Unable to write to flash OR flash data verification failed.

Definition at line 317 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CALIBRATION_DATA_VERIFICATION_FAILED

```
#define SL_STATUS_SI91X_CALIBRATION_DATA_VERIFICATION_FAILED
```

Value:

```
((sl_status_t)0×100F9)
```

Calibration data verification failed.

Definition at line 319 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SNMP_INTERNAL_ERROR

```
#define SL_STATUS_SI91X_SNMP_INTERNAL_ERROR
```

Value:

```
((sl_status_t)0×10100)
```

SNMP internal error.

Definition at line 321 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SNMP_INVALID_IP_PROTOCOL

```
#define SL_STATUS_SI91X_SNMP_INVALID_IP_PROTOCOL
```

Value:

```
((sl_status_t)0×10104)
```

SNMP invalid IP protocol error.

Definition at line 322 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_DATA_RECEIVED_OR_RECEIVE_TIMEOUT

```
#define SL_STATUS_SI91X_NO_DATA_RECEIVED_OR_RECEIVE_TIMEOUT
```

Value:

```
((sl_status_t)0×1BB01)
```

No data received or receive timeout.

Definition at line 323 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INSUFFICIENT_DATA_FOR_TIME_CONVERSION

```
#define SL_STATUS_SI91X_INSUFFICIENT_DATA_FOR_TIME_CONVERSION
```

Value:

```
((sl_status_t)0×1BB08)
```

Insufficient data for converting NTP time to mm-dd-yy time format.

Definition at line 325 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_SNTP_SERVER_ADDRESS

```
#define SL_STATUS_SI91X_INVALID_SNTP_SERVER_ADDRESS
```

Value:

```
((sl_status_t)0x1BB0A)
```

Invalid SNTP server address.

Definition at line 327 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SNTP_CLIENT_NOT_STARTED

```
#define SL_STATUS_SI91X_SNTP_CLIENT_NOT_STARTED
```

Value:

```
((sl_status_t)0x1BB0B)
```

SNTP client not started.

Definition at line 328 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SNTP_SERVER_UNAVAILABLE

```
#define SL_STATUS_SI91X_SNTP_SERVER_UNAVAILABLE
```

Value:

```
((sl_status_t)0x1BB10)
```

SNTP server not available. Client will not get any time update service from current server.

Definition at line 329 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SNTP_SERVER_AUTHENTICATION_FAILED

```
#define SL_STATUS_SI91X_SNTP_SERVER_AUTHENTICATION_FAILED
```

Value:

```
((sl_status_t)0x1BB15)
```

SNTP server authentication failed.

Definition at line 331 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INTERNAL_ERROR

```
#define SL_STATUS_SI91X_INTERNAL_ERROR
```

Value:

```
((sl_status_t)0x1BB0E)
```

Internal error.

Definition at line 332 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MULTICAST_IP_ADDRESS_ENTRY_NOT_FOUND

```
#define SL_STATUS_SI91X_MULTICAST_IP_ADDRESS_ENTRY_NOT_FOUND
```

Value:

```
((sl_status_t)0x1BB16)
```

Entry not found for multicast IP address.

Definition at line 333 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MULTICAST_NO_ENTRIES_FOUND

```
#define SL_STATUS_SI91X_MULTICAST_NO_ENTRIES_FOUND
```

Value:

```
((sl_status_t)0x1BB17)
```

No more entries found for multicast.

Definition at line 335 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_IP_ADDRESS_ERROR

```
#define SL_STATUS_SI91X_IP_ADDRESS_ERROR
```

Value:

```
((sl_status_t)0x1BB21)
```

IP address error.

Definition at line 336 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_ALREADY_BOUND

```
#define SL_STATUS_SI91X_SOCKET_ALREADY_BOUND
```

Value:

```
((sl_status_t)0×1BB22)
```

Socket already bound.

Definition at line `337` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_PORT_UNAVAILABLE

```
#define SL_STATUS_SI91X_PORT_UNAVAILABLE
```

Value:

```
((sl_status_t)0×1BB23)
```

Port not available.

Definition at line `338` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_SOCKET_NOT_CREATED

```
#define SL_STATUS_SI91X_SOCKET_NOT_CREATED
```

Value:

```
((sl_status_t)0×1BB27)
```

Socket is not created.

Definition at line `339` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_ICMP_REQUEST_FAILED

```
#define SL_STATUS_SI91X_ICMP_REQUEST_FAILED
```

Value:

```
((sl_status_t)0×1BB29)
```

ICMP request failed.

Definition at line `340` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_MAX_LISTEN_SOCKETS_REACHED

```
#define SL_STATUS_SI91X_MAX_LISTEN_SOCKETS_REACHED
```

Value:

```
((sl_status_t)0×1BB33)
```

Maximum listen sockets reached.

Definition at line 341 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DHCP_DUPLICATE_LISTEN

```
#define SL_STATUS_SI91X_DHCP_DUPLICATE_LISTEN
```

Value:

```
((sl_status_t)0x1BB34)
```

DHCP duplicate listen.

Definition at line 342 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_PORT_NOT_IN_CLOSE_STATE

```
#define SL_STATUS_SI91X_PORT_NOT_IN_CLOSE_STATE
```

Value:

```
((sl_status_t)0x1BB35)
```

Port not in closed state.

Definition at line 343 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_CLOSED

```
#define SL_STATUS_SI91X_SOCKET_CLOSED
```

Value:

```
((sl_status_t)0x1BB36)
```

Socket is closed or in process of closing.

Definition at line 344 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_PROCESS_IN_PROGRESS

```
#define SL_STATUS_SI91X_PROCESS_IN_PROGRESS
```

Value:

```
((sl_status_t)0x1BB37)
```

Process in progress.

Definition at line 345 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CONNECT_TO_NON_EXISTING_TCP_SERVER_SOCKET

```
#define SL_STATUS_SI91X_CONNECT_TO_NON_EXISTING_TCP_SERVER_SOCKET
```

Value:

```
((sl_status_t)0x1BB38)
```

Trying to connect non-existing TCP server socket.

Definition at line 346 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ERROR_IN_LEN_OF_THE_COMMAND

```
#define SL_STATUS_SI91X_ERROR_IN_LEN_OF_THE_COMMAND
```

Value:

```
((sl_status_t)0x1BB3E)
```

Error in length of the command ('Exceeds number of characters' is mentioned in the PRM).

Definition at line 348 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WRONG_PACKET_INFO

```
#define SL_STATUS_SI91X_WRONG_PACKET_INFO
```

Value:

```
((sl_status_t)0x1BB40)
```

Wrong packet info.

Definition at line 350 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_STILL_BOUND

```
#define SL_STATUS_SI91X_SOCKET_STILL_BOUND
```

Value:

```
((sl_status_t)0x1BB42)
```

Socket is still bound.

Definition at line 351 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_NO_FREE_PORT

```
#define SL_STATUS_SI91X_NO_FREE_PORT
```

Value:

```
((sl_status_t)0×1BB45)
```

No free port.

Definition at line 352 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_PORT

```
#define SL_STATUS_SI91X_INVALID_PORT
```

Value:

```
((sl_status_t)0×1BB46)
```

Invalid port.

Definition at line 353 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FEATURE_UNSUPPORTED

```
#define SL_STATUS_SI91X_FEATURE_UNSUPPORTED
```

Value:

```
((sl_status_t)0×1BB4B)
```

Feature not supported.

Definition at line 354 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_IN_UNCONNECTED_STATE

```
#define SL_STATUS_SI91X_SOCKET_IN_UNCONNECTED_STATE
```

Value:

```
((sl_status_t)0×1BB50)
```

Socket is not in connected state. Disconnected from server. In case of FTP, user need to give destroy command after receiving this error.

Definition at line 355 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_POP3_SESSION_CREATION_FAILED

```
#define SL_STATUS_SI91X_POP3_SESSION_CREATION_FAILED
```

Value:

```
((sl_status_t)0×1BB87)
```

POP3 session creation failed / POP3 session got terminated.

Definition at line 357 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DHCPV6_HANDSHAKE_FAIL

```
#define SL_STATUS_SI91X_DHCPV6_HANDSHAKE_FAIL
```

Value:

```
((sl_status_t)0x1BB9C)
```

DHCPv6 handshake failure.

Definition at line 359 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DHCP_INVALID_IP_RESPONSE

```
#define SL_STATUS_SI91X_DHCP_INVALID_IP_RESPONSE
```

Value:

```
((sl_status_t)0x1BB9D)
```

DHCP invalid IP response.

Definition at line 360 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_AUTHENTICATION_ERROR

```
#define SL_STATUS_SI91X_SMTP_AUTHENTICATION_ERROR
```

Value:

```
((sl_status_t)0x1BBA0)
```

SMTP authentication error.

Definition at line 361 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_OVER_SIZE_MAIL_DATA

```
#define SL_STATUS_SI91X_SMTP_OVER_SIZE_MAIL_DATA
```

Value:

```
((sl_status_t)0x1BBA1)
```

No DNS server was specified, SMTP over size mail data.

Definition at line 362 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_INVALID_SERVER_REPLY

```
#define SL_STATUS_SI91X_SMTP_INVALID_SERVER_REPLY
```

Value:

```
((sl_status_t)0x1BBA2)
```

SMTP invalid server reply.

Definition at line `364` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_SMTP_DNS_QUERY_FAILED

```
#define SL_STATUS_SI91X_SMTP_DNS_QUERY_FAILED
```

Value:

```
((sl_status_t)0x1BBA3)
```

DNS query failed, SMTP internal error.

Definition at line `365` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_SMTP_BAD_DNS_ADDRESS

```
#define SL_STATUS_SI91X_SMTP_BAD_DNS_ADDRESS
```

Value:

```
((sl_status_t)0x1BBA4)
```

Bad DNS address, SMTP server error code received.

Definition at line `366` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_SMTP_INVALID_PARAMETERS

```
#define SL_STATUS_SI91X_SMTP_INVALID_PARAMETERS
```

Value:

```
((sl_status_t)0x1BBA5)
```

SMTP invalid parameters.

Definition at line `368` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_SMTP_PACKET_ALLOCATION_FAILED

```
#define SL_STATUS_SI91X_SMTP_PACKET_ALLOCATION_FAILED
```

Value:

```
((sl_status_t)0×1BBA6)
```

SMTP packet allocation failed.

Definition at line 369 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_GREET_REPLY_FAILED

```
#define SL_STATUS_SI91X_SMTP_GREET_REPLY_FAILED
```

Value:

```
((sl_status_t)0×1BBA7)
```

SMTP Greet reply failed.

Definition at line 370 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_PARAMETER_ERROR

```
#define SL_STATUS_SI91X_SMTP_PARAMETER_ERROR
```

Value:

```
((sl_status_t)0×1BBA8)
```

Parameter error, SMTP hello reply error.

Definition at line 371 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_MAIL_REPLY_ERROR

```
#define SL_STATUS_SI91X_SMTP_MAIL_REPLY_ERROR
```

Value:

```
((sl_status_t)0×1BBA9)
```

SMTP mail reply error.

Definition at line 372 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_RCPT_REPLY_ERROR

```
#define SL_STATUS_SI91X_SMTP_RCPT_REPLY_ERROR
```

Value:

```
((sl_status_t)0×1BBAA)
```

SMTP RCPT reply error.

Definition at line 373 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_MESSAGE_REPLY_ERROR

```
#define SL_STATUS_SI91X_SMTP_MESSAGE_REPLY_ERROR
```

Value:

```
((sl_status_t)0x1BBAB)
```

SMTP message reply error.

Definition at line 374 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_DATA_REPLY_ERROR

```
#define SL_STATUS_SI91X_SMTP_DATA_REPLY_ERROR
```

Value:

```
((sl_status_t)0x1BBAC)
```

SMTP data reply error.

Definition at line 375 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_AUTH_REPLY_ERROR

```
#define SL_STATUS_SI91X_SMTP_AUTH_REPLY_ERROR
```

Value:

```
((sl_status_t)0x1BBAD)
```

SMTP authentication reply error.

Definition at line 376 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_SERVER_REPLY_ERROR

```
#define SL_STATUS_SI91X_SMTP_SERVER_REPLY_ERROR
```

Value:

```
((sl_status_t)0x1BBAE)
```

SMTP server error reply.

Definition at line 377 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_DUPLICATE_ENTRY

```
#define SL_STATUS_SI91X_DNS_DUPLICATE_ENTRY
```

Value:

```
((sl_status_t)0×1BBAF)
```

DNS duplicate entry.

Definition at line 378 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_OVERSIZE_SERVER_REPLY

```
#define SL_STATUS_SI91X_SMTP_OVERSIZE_SERVER_REPLY
```

Value:

```
((sl_status_t)0×1BBB1)
```

SMTP oversize server reply.

Definition at line 379 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SMTP_CLIENT_NOT_INITIALIZED

```
#define SL_STATUS_SI91X_SMTP_CLIENT_NOT_INITIALIZED
```

Value:

```
((sl_status_t)0×1BBB2)
```

SMTP client not initialized.

Definition at line 380 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_IPV6_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_DNS_IPV6_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0×1BBB3)
```

DNS IPv6 not supported.

Definition at line 381 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_MAIL_INDEX_FOR_POP3_MAIL_RETRIEVE_COMMAND

```
#define SL_STATUS_SI91X_INVALID_MAIL_INDEX_FOR_POP3_MAIL_RETRIEVE_COMMAND
```

Value:

```
((sl_status_t)0×1BBC5)
```

Invalid mail index for POP3 mail retrieve command.

Definition at line `382` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_FAILED

```
#define SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_FAILED
```

Value:

```
((sl_status_t)0×1BBD2)
```

SSL/TLS handshake failed.

Definition at line `384` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_FTP_CLIENT_DISCONNECTED

```
#define SL_STATUS_SI91X_FTP_CLIENT_DISCONNECTED
```

Value:

```
((sl_status_t)0×1BBD3)
```

FTP client is not connected or disconnected with the FTP server.

Definition at line `385` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_FTP_CLIENT_NOT_DISCONNECTED

```
#define SL_STATUS_SI91X_FTP_CLIENT_NOT_DISCONNECTED
```

Value:

```
((sl_status_t)0×1BBD4)
```

FTP client is not disconnected.

Definition at line `387` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_FTP_FILE_NOT_OPENED

```
#define SL_STATUS_SI91X_FTP_FILE_NOT_OPENED
```

Value:

```
((sl_status_t)0×1BBD5)
```

FTP file is not opened.

Definition at line 388 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_TIMEOUT_OR_FTP_FILE_NOT_CLOSED

```
#define SL_STATUS_SI91X_SSL_TLS_HANDSHAKE_TIMEOUT_OR_FTP_FILE_NOT_CLOSED
```

Value:

```
((sl_status_t)0x1BBD6)
```

SSL/TLS handshake timeout or FTP file is not closed.

Definition at line 389 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FTP_EXPECTED_1XX_RESPONSE_NOT_RECEIVED

```
#define SL_STATUS_SI91X_FTP_EXPECTED_1XX_RESPONSE_NOT_RECEIVED
```

Value:

```
((sl_status_t)0x1BBD9)
```

Expected [1XX response from FTP server but not received].

Definition at line 391 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FTP_EXPECTED_2XX_RESPONSE_NOT_RECEIVED

```
#define SL_STATUS_SI91X_FTP_EXPECTED_2XX_RESPONSE_NOT_RECEIVED
```

Value:

```
((sl_status_t)0x1BBDA)
```

Expected [2XX response from FTP server but not received].

Definition at line 393 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FTP_EXPECTED_22X_RESPONSE_NOT_RECEIVED

```
#define SL_STATUS_SI91X_FTP_EXPECTED_22X_RESPONSE_NOT_RECEIVED
```

Value:

```
((sl_status_t)0x1BBDB)
```

Expected [22X response from FTP server but not received].

Definition at line 395 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FTP_EXPECTED_23X_RESPONSE_NOT_RECEIVED

```
#define SL_STATUS_SI91X_FTP_EXPECTED_23X_RESPONSE_NOT_RECEIVED
```

Value:

```
((sl_status_t)0x1BBDC)
```

Expected [23X response from FTP server but not received].

Definition at line 397 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FTP_EXPECTED_3XX_RESPONSE_NOT_RECEIVED

```
#define SL_STATUS_SI91X_FTP_EXPECTED_3XX_RESPONSE_NOT_RECEIVED
```

Value:

```
((sl_status_t)0x1BBDD)
```

Expected [3XX response from FTP server but not received].

Definition at line 399 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FTP_EXPECTED_33X_RESPONSE_NOT_RECEIVED

```
#define SL_STATUS_SI91X_FTP_EXPECTED_33X_RESPONSE_NOT_RECEIVED
```

Value:

```
((sl_status_t)0x1BBDE)
```

Expected [33X response from FTP server but not received].

Definition at line 401 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_TIMEOUT

```
#define SL_STATUS_SI91X_HTTP_TIMEOUT
```

Value:

```
((sl_status_t)0x1BBE1)
```

HTTP timeout.

Definition at line 403 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_FAILED

```
#define SL_STATUS_SI91X_HTTP_FAILED
```

Value:

```
((sl_status_t)0×1BBE2)
```

HTTP failed.

Definition at line 404 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_PUT_CLIENT_TIMEOUT

```
#define SL_STATUS_SI91X_HTTP_PUT_CLIENT_TIMEOUT
```

Value:

```
((sl_status_t)0×1BBE7)
```

HTTP timeout for HTTP PUT client.

Definition at line 405 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_AUTHENTICATION_ERROR

```
#define SL_STATUS_SI91X_AUTHENTICATION_ERROR
```

Value:

```
((sl_status_t)0×1BBEB)
```

Authentication error.

Definition at line 406 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_PACKET_LENGTH

```
#define SL_STATUS_SI91X_INVALID_PACKET_LENGTH
```

Value:

```
((sl_status_t)0×1BBED)
```

Invalid packet length. Content length and received data length is mismatching.

Definition at line 407 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SERVER_RESPONDS_BEFORE_REQUEST_COMPLETE

```
#define SL_STATUS_SI91X_SERVER_RESPONDS_BEFORE_REQUEST_COMPLETE
```

Value:

```
((sl_status_t)0×1BBEF)
```

Server responds before HTTP client request is complete.

Definition at line 409 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_PASSWORD_TOO_LONG

```
#define SL_STATUS_SI91X_HTTP_PASSWORD_TOO_LONG
```

Value:

```
((sl_status_t)0x1BBF0)
```

HTTP/HTTPS password is too long.

Definition at line 411 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_PING_TIMEOUT

```
#define SL_STATUS_SI91X_MQTT_PING_TIMEOUT
```

Value:

```
((sl_status_t)0x1BBF1)
```

MQTT ping time out error.

Definition at line 412 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_COMMAND_SENT_IN_INCORRECT_STATE

```
#define SL_STATUS_SI91X_MQTT_COMMAND_SENT_IN_INCORRECT_STATE
```

Value:

```
((sl_status_t)0x1BBF2)
```

MQTT command sent in incorrect state.

Definition at line 413 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_ACK_TIMEOUT

```
#define SL_STATUS_SI91X_MQTT_ACK_TIMEOUT
```

Value:

```
((sl_status_t)0x1BBF3)
```

MQTT ACK time out error.

Definition at line 415 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_POP3_INVALID_MAIL_INDEX

```
#define SL_STATUS_SI91X_POP3_INVALID_MAIL_INDEX
```

Value:

```
((sl_status_t)0×1BBFF)
```

POP3 error for invalid mail index.

Definition at line 416 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_NOT_CONNECTED

```
#define SL_STATUS_SI91X_SOCKET_NOT_CONNECTED
```

Value:

```
((sl_status_t)0×1FFFF)
```

Listening TCP socket in device is not connected to the remote peer, or the LTCP socket is not yet opened in the device.

Definition at line 417 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_LIMIT_EXCEEDED

```
#define SL_STATUS_SI91X_SOCKET_LIMIT_EXCEEDED
```

Value:

```
((sl_status_t)0×1FFFE)
```

Sockets not available. The error comes if the host tries to open more than 10 sockets.

Definition at line 419 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_OTAF_INVALID_PACKET

```
#define SL_STATUS_SI91X_HTTP_OTAF_INVALID_PACKET
```

Value:

```
((sl_status_t)0×1FFFD)
```

HTTP OTAF invalid packet.

Definition at line 421 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_TCP_IP_INIT_FAILED

```
#define SL_STATUS_SI91X_TCP_IP_INIT_FAILED
```

Value:

```
((sl_status_t)0×1FFFC)
```

TCP_IP initialization failed.

Definition at line 422 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_CONCURRENT_IP_CREATION_ERROR

```
#define SL_STATUS_SI91X_CONCURRENT_IP_CREATION_ERROR
```

Value:

```
((sl_status_t)0×1FFFB)
```

Cannot create IP in same interface in concurrent mode.

Definition at line 423 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_OTAF_INCOMPLETE_PACKET

```
#define SL_STATUS_SI91X_HTTP_OTAF_INCOMPLETE_PACKET
```

Value:

```
((sl_status_t)0×1FFF4)
```

HTTP OTAF incomplete packet.

Definition at line 425 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_STORE_CONFIGURATION_PROFILE

```
#define SL_STATUS_SI91X_INVALID_STORE_CONFIGURATION_PROFILE
```

Value:

```
((sl_status_t)0×1FFF5)
```

Store configuration profile type mismatch or invalid profile type.

Definition at line 426 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MQTT_REMOTE_TERMINATE_ERROR

```
#define SL_STATUS_SI91X_MQTT_REMOTE_TERMINATE_ERROR
```

Value:

```
((sl_status_t)0×1FFF6)
```

MQTT remote terminate error.

Definition at line 428 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_BYTE_STUFFING_ERROR_IN_AT_MODE

```
#define SL_STATUS_SI91X_BYTE_STUFFING_ERROR_IN_AT_MODE
```

Value:

```
((sl_status_t)0x1FFF7)
```

Byte stuffing error in AT mode.

Definition at line 429 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_COMMAND_OR_OPERATION

```
#define SL_STATUS_SI91X_INVALID_COMMAND_OR_OPERATION
```

Value:

```
((sl_status_t)0x1FFF8)
```

Invalid command (e.g. parameters insufficient or invalid in the command). Invalid operation (e.g. power save command with the same mode given twice, accessing wrong socket, creating more than allowed sockets).

Definition at line 430 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_OTAF_NO_PACKET

```
#define SL_STATUS_SI91X_HTTP_OTAF_NO_PACKET
```

Value:

```
((sl_status_t)0x1FFF9)
```

HTTP OTAF no packet.

Definition at line 432 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_TCP_SOCKET_NOT_CONNECTED

```
#define SL_STATUS_SI91X_TCP_SOCKET_NOT_CONNECTED
```

Value:

```
((sl_status_t)0x1FFFA)
```

TCP socket is not connected.

Definition at line 433 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MAX_STATION_COUNT_EXCEEDED

```
#define SL_STATUS_SI91X_MAX_STATION_COUNT_EXCEEDED
```

Value:

```
((sl_status_t)0x1FFC5)
```

Station count exceeded max station supported.

Definition at line 434 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_UNABLE_TO_SEND_TCP_DATA

```
#define SL_STATUS_SI91X_UNABLE_TO_SEND_TCP_DATA
```

Value:

```
((sl_status_t)0x1FFC4)
```

Unable to send TCP data.

Definition at line 436 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_BUFFER_TOO_SMALL

```
#define SL_STATUS_SI91X_SOCKET_BUFFER_TOO_SMALL
```

Value:

```
((sl_status_t)0x1FFBC)
```

Socket buffer too small.

Definition at line 437 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_CONTENT_IN_DNS_RESPONSE

```
#define SL_STATUS_SI91X_INVALID_CONTENT_IN_DNS_RESPONSE
```

Value:

```
((sl_status_t)0x1FFBB)
```

Invalid content in the DNS response to the DNS resolution query.

Definition at line 438 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_CLASS_ERROR_IN_DNS_RESPONSE

```
#define SL_STATUS_SI91X_DNS_CLASS_ERROR_IN_DNS_RESPONSE
```

Value:

```
((sl_status_t)0×1FFBA)
```

DNS class error in response to the DNS resolution query.

Definition at line 440 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_COUNT_ERROR_IN_DNS_RESPONSE

```
#define SL_STATUS_SI91X_DNS_COUNT_ERROR_IN_DNS_RESPONSE
```

Value:

```
((sl_status_t)0×1FFB8)
```

DNS count error in response to the DNS resolution query.

Definition at line 442 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_RETURN_CODE_ERROR_IN_DNS_RESPONSE

```
#define SL_STATUS_SI91X_DNS_RETURN_CODE_ERROR_IN_DNS_RESPONSE
```

Value:

```
((sl_status_t)0×1FFB7)
```

DNS return code error in the response to the DNS resolution query.

Definition at line 444 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_OPCODE_ERROR_IN_DNS_RESPONSE

```
#define SL_STATUS_SI91X_DNS_OPCODE_ERROR_IN_DNS_RESPONSE
```

Value:

```
((sl_status_t)0×1FFB6)
```

DNS Opcode error in the response to the DNS resolution query.

Definition at line 446 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_ID_MISMATCH

```
#define SL_STATUS_SI91X_DNS_ID_MISMATCH
```

Value:

```
((sl_status_t)0×1FFB5)
```

DNS id mismatch between DNS resolution request and response.

Definition at line 448 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_INPUT_IN_DNS_QUERY

```
#define SL_STATUS_SI91X_INVALID_INPUT_IN_DNS_QUERY
```

Value:

```
((sl_status_t)0x1FFAB)
```

An invalid input to the DNS resolution query.

Definition at line 450 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_RESPONSE_TIMEOUT

```
#define SL_STATUS_SI91X_DNS_RESPONSE_TIMEOUT
```

Value:

```
((sl_status_t)0x1FF42)
```

DNS response was timed out.

Definition at line 452 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_ARP_REQUEST_FAILURE

```
#define SL_STATUS_SI91X_ARP_REQUEST_FAILURE
```

Value:

```
((sl_status_t)0x1FFA1)
```

ARP request failure.

Definition at line 453 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_UNABLE_TO_UPDATE_TCP_WINDOW

```
#define SL_STATUS_SI91X_UNABLE_TO_UPDATE_TCP_WINDOW
```

Value:

```
((sl_status_t)0x1FF91)
```

Unable to update TCP window.

Definition at line 454 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DHCP_LEASE_EXPIRED

```
#define SL_STATUS_SI91X_DHCP_LEASE_EXPIRED
```

Value:

```
((sl_status_t)0×1FF9D)
```

DHCP lease time expired.

Definition at line 455 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DHCP_HANDSHAKE_FAILURE

```
#define SL_STATUS_SI91X_DHCP_HANDSHAKE_FAILURE
```

Value:

```
((sl_status_t)0×1FF9C)
```

DHCP handshake failure.

Definition at line 456 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WEBSOCKET_CREATION_FAILED

```
#define SL_STATUS_SI91X_WEBSOCKET_CREATION_FAILED
```

Value:

```
((sl_status_t)0×1FF88)
```

This error is issued when WebSocket creation failed.

Definition at line 457 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_TRYING_TO_CONNECT_NON_EXISTENT_TCP_SERVER_SOCKET

```
#define SL_STATUS_SI91X_TRYING_TO_CONNECT_NON_EXISTENT_TCP_SERVER_SOCKET
```

Value:

```
((sl_status_t)0×1FF87)
```

This error is issued when device tried to connect to a non-existent TCP server socket on the remote side.

Definition at line 459 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_TRYING_TO_CLOSE_NON_EXISTENT_SOCKET

```
#define SL_STATUS_SI91X_TRYING_TO_CLOSE_NON_EXISTENT_SOCKET
```

Value:

```
((sl_status_t)0×1FF86)
```

This error is issued when tried to close non-existent socket. or invalid socket descriptor.

Definition at line 461 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_SOCKET_PARAMETERS

```
#define SL_STATUS_SI91X_INVALID_SOCKET_PARAMETERS
```

Value:

```
((sl_status_t)0×1FF85)
```

Invalid socket parameters.

Definition at line 463 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FEATURE_NOT_AVAILABLE

```
#define SL_STATUS_SI91X_FEATURE_NOT_AVAILABLE
```

Value:

```
((sl_status_t)0×1FF82)
```

Feature not supported.

Definition at line 464 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_ALREADY_OPEN

```
#define SL_STATUS_SI91X_SOCKET_ALREADY_OPEN
```

Value:

```
((sl_status_t)0×1FF81)
```

Socket already open.

Definition at line 465 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MAX_SOCKETS_EXCEEDED

```
#define SL_STATUS_SI91X_MAX_SOCKETS_EXCEEDED
```

Value:

```
((sl_status_t)0×1FF80)
```

Attempt to open more than the maximum allowed number of sockets.

Definition at line 466 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DATA_LENGTH_EXCEEDS_MSS

```
#define SL_STATUS_SI91X_DATA_LENGTH_EXCEEDS_MSS
```

Value:

```
((sl_status_t)0x1FF7E)
```

Data length exceeds mss.

Definition at line 468 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_IP_CONFLICT_ERROR

```
#define SL_STATUS_SI91X_IP_CONFLICT_ERROR
```

Value:

```
((sl_status_t)0x1FF75)
```

DUT unable to configure IP address due to IP conflict.

Definition at line 469 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FEATURE_NOT_ENABLED

```
#define SL_STATUS_SI91X_FEATURE_NOT_ENABLED
```

Value:

```
((sl_status_t)0x1FF74)
```

Feature not enabled.

Definition at line 471 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DHCP_SERVER_NOT_SET

```
#define SL_STATUS_SI91X_DHCP_SERVER_NOT_SET
```

Value:

```
((sl_status_t)0x1FF73)
```

DHCP server not set in AP mode.

Definition at line 472 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_AP_SET_REGION_PARAM_ERROR

```
#define SL_STATUS_SI91X_AP_SET_REGION_PARAM_ERROR
```

Value:

```
((sl_status_t)0x1FF71)
```

Error in AP set region command parameters.

Definition at line 473 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SSL_TLS_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_SSL_TLS_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0x1FF70)
```

SSL/TLS not supported.

Definition at line 474 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_JSON_NOT_SUPPORTED

```
#define SL_STATUS_SI91X_JSON_NOT_SUPPORTED
```

Value:

```
((sl_status_t)0x1FF6F)
```

JSON not supported.

Definition at line 475 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_OPERATING_MODE

```
#define SL_STATUS_SI91X_INVALID_OPERATING_MODE
```

Value:

```
((sl_status_t)0x1FF6E)
```

Invalid operating mode.

Definition at line 476 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_SOCKET_CONFIG_PARAMS

```
#define SL_STATUS_SI91X_INVALID_SOCKET_CONFIG_PARAMS
```

Value:

```
((sl_status_t)0×1FF6D)
```

Invalid socket configuration parameters.

Definition at line 477 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WEBSOCKET_CREATION_TIMEOUT

```
#define SL_STATUS_SI91X_WEBSOCKET_CREATION_TIMEOUT
```

Value:

```
((sl_status_t)0×1FF6C)
```

Web socket creation timeout.

Definition at line 479 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_PARAM_MAX_VALUE_EXCEEDED

```
#define SL_STATUS_SI91X_PARAM_MAX_VALUE_EXCEEDED
```

Value:

```
((sl_status_t)0×1FF6B)
```

Parameter maximum allowed value is exceeded.

Definition at line 480 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SOCKET_READ_TIMEOUT

```
#define SL_STATUS_SI91X_SOCKET_READ_TIMEOUT
```

Value:

```
((sl_status_t)0×1FF6A)
```

Socket read timeout.

Definition at line 482 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_INVALID_COMMAND_SEQUENCE

```
#define SL_STATUS_SI91X_INVALID_COMMAND_SEQUENCE
```

Value:

```
((sl_status_t)0×1FF69)
```

Invalid command in sequence.

Definition at line 483 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_DNS_RESPONSE_TIMEOUT_ERROR

```
#define SL_STATUS_SI91X_DNS_RESPONSE_TIMEOUT_ERROR
```

Value:

```
((sl_status_t)0x1FF42)
```

DNS response timed out.

Definition at line 484 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_HTTP_SOCKET_CREATION_FAILED

```
#define SL_STATUS_SI91X_HTTP_SOCKET_CREATION_FAILED
```

Value:

```
((sl_status_t)0x1FF41)
```

HTTP socket creation failed.

Definition at line 485 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_TCP_CLOSE_BEFORE_RESPONSE_ERROR

```
#define SL_STATUS_SI91X_TCP_CLOSE_BEFORE_RESPONSE_ERROR
```

Value:

```
((sl_status_t)0x1FF40)
```

TCP socket close command is issued before getting the response of the previous close command.

Definition at line 486 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_WAIT_ON_HOST_FEATURE_NOT_ENABLED

```
#define SL_STATUS_SI91X_WAIT_ON_HOST_FEATURE_NOT_ENABLED
```

Value:

```
((sl_status_t)0x1FF36)
```

'Wait On Host' feature not enabled.

Definition at line 488 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_STORE_CONFIG_CHECKSUM_INVALID

#define SL_STATUS_SI91X_STORE_CONFIG_CHECKSUM_INVALID

Value:

((sl_status_t)0×1FF35)

Store configuration checksum validation failed.

Definition at line 489 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_TCP_KEEP_ALIVE_TIMEOUT

#define SL_STATUS_SI91X_TCP_KEEP_ALIVE_TIMEOUT

Value:

((sl_status_t)0×1FF33)

TCP keep alive timed out.

Definition at line 491 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_TCP_ACK_FAILED_FOR_SYN_ACK

#define SL_STATUS_SI91X_TCP_ACK_FAILED_FOR_SYN_ACK

Value:

((sl_status_t)0×1FF2D)

TCP ACK failed for TCP SYN-ACK.

Definition at line 492 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MEMORY_LIMIT_EXCEEDED

#define SL_STATUS_SI91X_MEMORY_LIMIT_EXCEEDED

Value:

((sl_status_t)0×1FF2C)

Memory limit exceeded in a given operating mode.

Definition at line 493 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_MEMORY_LIMIT_EXCEEDED_DURING_AUTO_JOIN

#define SL_STATUS_SI91X_MEMORY_LIMIT_EXCEEDED_DURING_AUTO_JOIN

Value:

```
((sl_status_t)0×1FF2A)
```

Memory limit exceeded in an operating mode during auto join/create.

Definition at line `495` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_PUF_OPERATION_BLOCKED

```
#define SL_STATUS_SI91X_PUF_OPERATION_BLOCKED
```

Value:

```
((sl_status_t)0×1CC2F)
```

PUF operation is blocked.

Definition at line `497` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_PUF_ACTIVATION_CODE_INVALID

```
#define SL_STATUS_SI91X_PUF_ACTIVATION_CODE_INVALID
```

Value:

```
((sl_status_t)0×1CC31)
```

PUF activation code invalid.

Definition at line `498` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_PUF_INPUT_PARAMETERS_INVALID

```
#define SL_STATUS_SI91X_PUF_INPUT_PARAMETERS_INVALID
```

Value:

```
((sl_status_t)0×1CC32)
```

PUF input parameters invalid.

Definition at line `499` of file `components/common/inc/sl_additional_status.h`

SL_STATUS_SI91X_PUF_IN_ERROR_STATE

```
#define SL_STATUS_SI91X_PUF_IN_ERROR_STATE
```

Value:

```
((sl_status_t)0×1CC33)
```

PUF in error state.

Definition at line 500 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_PUF_OPERATION_NOT_ALLOWED

```
#define SL_STATUS_SI91X_PUF_OPERATION_NOT_ALLOWED
```

Value:

```
((sl_status_t)0x1CC34)
```

PUF operation not allowed.

Definition at line 501 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_PUF_OPERATION_FAILED

```
#define SL_STATUS_SI91X_PUF_OPERATION_FAILED
```

Value:

```
((sl_status_t)0x1CC35)
```

PUF operation failed.

Definition at line 502 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_AUTO_JOIN_IN_PROGRESS

```
#define SL_STATUS_SI91X_AUTO_JOIN_IN_PROGRESS
```

Value:

```
((sl_status_t)0x15A5A)
```

Auto join or user store configuration going on.

Definition at line 503 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_RSNIE_FROM_AP_INVALID

```
#define SL_STATUS_SI91X_RSNIE_FROM_AP_INVALID
```

Value:

```
((sl_status_t)0x1FFE1)
```

Improper RSNIE from AP to station.

Definition at line 505 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_SNTP_MAX_ATTEMPTS_REACHED

```
#define SL_STATUS_SI91X_SNTP_MAX_ATTEMPTS_REACHED
```

Value:

```
((sl_status_t)0x1FF5F)
```

Reached maximum SNTP invalid attempts.

Definition at line 506 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FREQUENCY_OFFSET_ZER0

```
#define SL_STATUS_SI91X_FREQUENCY_OFFSET_ZER0
```

Value:

```
((sl_status_t)0x100FC)
```

Frequency offset sent is zero.

Definition at line 507 of file components/common/inc/sl_additional_status.h

SL_STATUS_SI91X_FREQUENCY_OFFSET_OUT_OF_LIMITS

```
#define SL_STATUS_SI91X_FREQUENCY_OFFSET_OUT_OF_LIMITS
```

Value:

```
((sl_status_t)0x100FB)
```

Frequency offset specified goes beyond upper or lower limits and indicates that frequency offset cannot be changed further.

Definition at line 508 of file components/common/inc/sl_additional_status.h

Examples

WiSeConnect™ SDK v3.x Examples

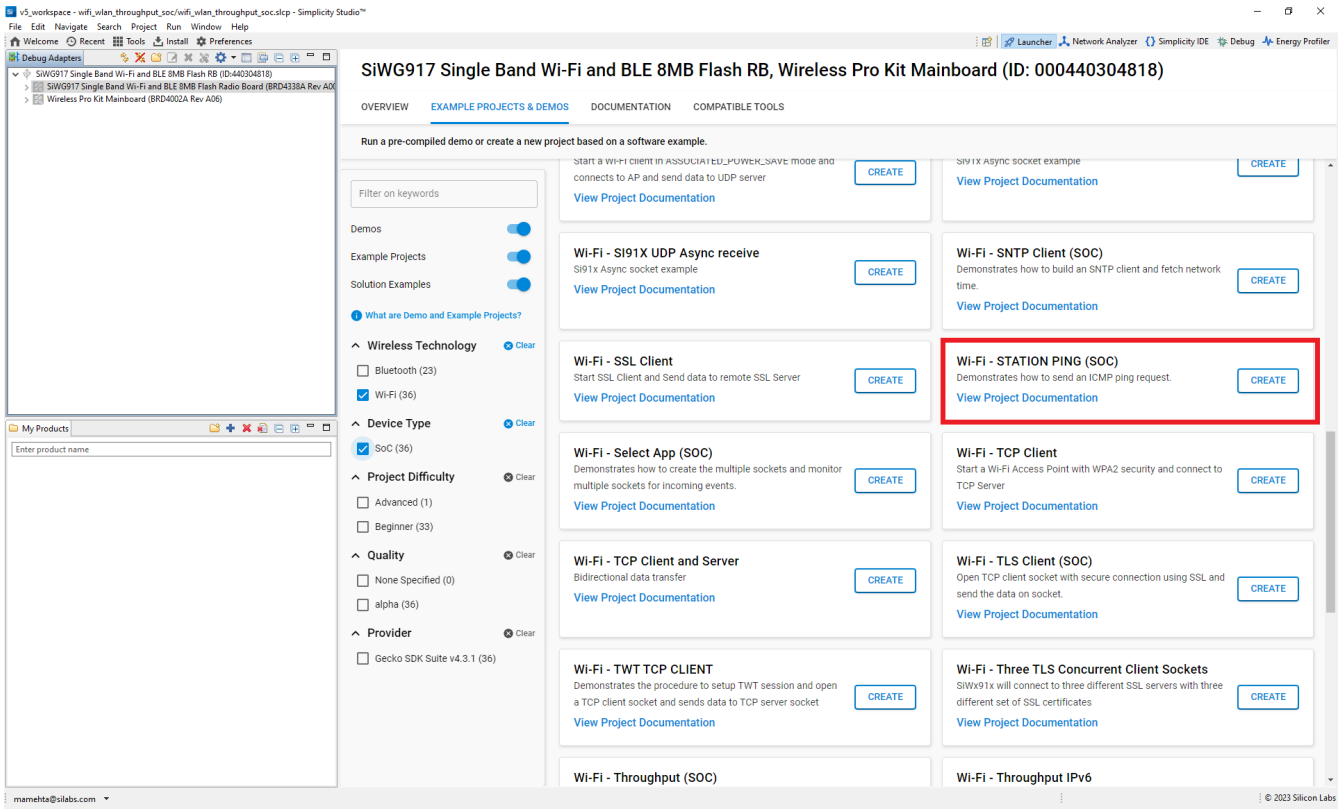
This section describes the example applications available with the WiSeConnect™ SDK v3.x.

The Example Folder Strucure section describes the folder structure of a Simplicity Studio example.

Subsequent sections provide information about each example available with WiSeConnect SDK v3.x.

In the examples tables below:

- The **Name** column indicates the name by which the example can be found in Simplicity Studio, not including suffixes such as "(SoC)".



> **Note**: The above image is for illustration only and example names may not match with the latest version of the **WiSeConnect 3** extension.

- The **Featured** column indicates whether the example is a fully-featured application.
- The **SoC** column indicates whether the example is available in System-on-chip (SoC) mode, where the application and connectivity stack run on the SiWx91x™ chipset. Simplicity Studio displays such examples in the **EXAMPLE PROJECTS & DEMOS** tab when the SiWx917 is connected to the computer running Studio.
- The **NCP** column indicates whether the example is available in Network Co-processor (NCP) mode with an external EFR32™ microcontroller unit (MCU) host, where the application runs on the EFR32 host, and connectivity stack runs on the SiWx91x

chipset. Simplicity Studio displays such examples in the **EXAMPLE PROJECTS & DEMOS** tab when an EFR32 host is connected to the computer running Studio.
- The **PSRAM** column indicates whether the example executes from pseudo-static random-access memory (PSRAM) in SoC mode on an SiWx917 chip variant that includes PSRAM. Simplicity Studio displays such examples in the **EXAMPLE PROJECTS & DEMOS** tab when the **BRD4342A** (SiWx917 with PSRAM) radio board is connected to the computer running Studio.

To find the example you need, browse the sections below. The same example may be mentioned in more than one section when it belongs to multiple categories.

The following examples demonstrate specific actions as described in the description column and are organized in alphabetical order by example name.

Example Folder Strucure

Folder / File	Description
`autogen`	Files auto-generated by Simplicity Studio such as configuration header files, linker files, and others.
`config`	Header files with platform-specific configuration values.
`gecko_sdk_4.3.2`	The sub-folders of this folder contain Gecko SDK platform-specific layer implementations, and third party libraries.
`|-- platform`	Platform layer implementations such as the hardware abstraction layer (HAL), Common Microcontroller Software Interface Standard (CMSIS) for real-time operating systems (RTOS), and common libraries.
`|-- util`	Third party libraries such as FreeRTOS, mbedTLS, and others.
`resources`	Resources used by the example such as images for the README file.
`wiseconnect3_sdk_3.1.1`	The sub-folders of this folder contain WiSeConnect SDK components and resources.
`|-- components`	WiSeConnect SDK components such as **Wi-Fi**, **BLE**, **Si91x MCU Subsystem**, and others.
`|-- resources`	WiSeConnect SDK resources such as certificates.
`app.c`	Source file with the code for the main application thread. Implement your application code here.
`app.h`	Header file for the main application thread.
`main.c`	Source file with the entry point function (`main()`) for application execution with start-up code such as wireless and platform initializations, and creation of the main application thread.
`readme.md`	README file for the example.

Wireless Examples

The following wireless examples are available:

- Wi-Fi Examples
- Wi-Fi + BLE Examples
- BLE Examples

Wi-Fi Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Wi-Fi - Access Point		X	X		Implementing a Wi-Fi Access Point (AP)	Go to README
Wi-Fi - Access Point V6		X			Implementing a Wi-Fi Access Point (AP) over Internet Protocol version 6 (IPv6)	Go to README
Wi-Fi - AWS Device Shadow	X	X	X		Establishing connection to the Amazon Web Services (AWS) Internet-of-Things (IoT) Core	Go to README
Wi-Fi - Calibration App		X	X		Calibrating carrier frequency offset and transmission (TX) gain offset and writing them to the eFuse	Go to README
Wi-Fi - Concurrent Mode		X	X		Implementing a Wi-Fi device both as an AP and station (STA)	Go to README
Wi-Fi - Enterprise Client Mode		X	X		Implementing a Wi-Fi enterprise client and connecting to an enterprise AP	Go to README
Wi-Fi - Firmware Update	X	X	X		Downloading and updating firmware as a TCP client	Go to README
Wi-Fi - HTTP OTAF Update		X	X		Downloading and updating firmware as an HTTP client	Go to README
Wi-Fi - LwIP TCP Client		X	X		Using the hosted mode of the WiSeConnect SDK v3.x, by utilizing the Lightweight Internet Protocol (LwIP) stack	Go to README
Wi-Fi - M4 Firmware Update		X			Downloading and updating the application processor's firmware	Go to README
Wi-Fi - Powersave Deep Sleep		X	X		Implementing the power-save deep sleep mode of the SiWx91x™ chipset	Go to README
Wi-Fi - Powersave Standby Associated	X	X	X	X	Running an application in `ASSOCIATED_POWER_SAVE` mode	Go to README
Wi-Fi - Select App		X	X		Opening a network socket and monitoring it for incoming data	Go to README
Wi-Fi - TCP Tx on Periodic Wakeup		X			Transmitting data from a sleepy device every time it wakes up	Go to README
Wi-Fi - Three TLS Concurrent Client Sockets		X	X		Maintaining three concurrent SSL connections using three sets of SSL certificates	Go to README
Wi-Fi - Throughput	X	X	X	X	Measuring the WLAN throughput	Go to README
Wi-Fi - Throughput (PSRAM)		X		X	Measuring WLAN throughput while executing from PSRAM	Go to README
Wi-Fi - Throughput IPv6		X	X		Measuring the WLAN throughout of an IPv6 client	Go to README
Wi-Fi - TLS Client		X	X		Implementing a Transport Layer Security (TLS) client and exchanging encrypted data with a server	Go to README
Wi-Fi - TWT TCP Client		X	X		Implementing a TCP client using the Wi-Fi Target Wake Time (TWT) power-save scheme	Go to README

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Wi-Fi - TWT Use case demo app		X	X		Implementing a Wi-Fi 6 client using TWT power-save	Go to README
Wi-Fi - TWT Use case remote app		X	X		TCP door lock or UDP camera communication between a client and server using TWT power-save	Go to README
Wi-Fi - User Gain Table		X	X		Transmitting data at varying data rates, power levels, and lengths according to a gain table	Go to README
Wi-Fi - Wlan RF Test		X	X		Transmitting Wi-Fi packets at different data rates and power levels for regulatory certification testing	Go to README

Wi-Fi + BLE Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Cli Demo		X	X		Using console commands to demonstrate various API functions in the SDK such as scanning for Access Points (APs), bringing up a network interface, and others	Go to README
Out of Box Demo		X			Commissioning a Wi-Fi device over BLE and exchanging data with it	Go to README
Wi-Fi Coex - Wi-Fi Client BLE Provisioning		X	X		Advertising a device over BLE, receiving Wi-Fi credentials and connecting to a Wi-Fi AP	Go to README
Wi-Fi Coex - Wi-Fi Client BLE Provisioning with AWS		X	X		Receiving Wi-Fi credentials over BLE, connecting to Wi-Fi and publishing data to an AWS MQTT broker	Go to README
Wi-Fi Coex - Wi-Fi Throughput BLE Dual Role		X			Measuring the WLAN and/or BLE throughput(s) while they are simultaneously connected	Go to README

BLE Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
BLE - Acceptlist		X	X		Maintaining an allow list of devices that are allowed to connect to a BLE device	Go to README
BLE - AE Central		X	X		Running a BLE central in Extended Advertising Central mode	Go to README
BLE - AE Peripheral		X	X		Running a BLE peripheral in Extended Advertising (AE) Peripheral mode	Go to README
BLE - Central		X	X		Running a BLE central and connecting to a BLE peripheral	Go to README
BLE - Datalength		X	X		Using BLE data length extension to increase packet length	Go to README
BLE - Gatt Long Read		X	X		BLE Generic Attribute Profile (GATT) client performing a long read from a BLE GATT server	Go to README

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
BLE - Heart Rate		X	X	X	Implementing the Heart Rate service both in the BLE central and peripheral roles	Go to README
BLE - HID On GATT		X	X		Implementing the Human Interface Device (HID) service both in the BLE central and peripheral roles	Go to README
BLE - iBeacon		X	X		Implementing an iBeacon-compatible BLE peripheral	Go to README
BLE - Long Range		X	X		Running a BLE central and updating physical layer (PHY) rates	Go to README
BLE - Multiconnection GATT Test		X			Multiple BLE connections both as a BLE central and peripheral	Go to README
BLE - PER	X	X	X	X	Getting BLE packet error rate (PER) statistics for transmitting and receiving packets	Go to README
BLE - Power Save		X	X	X	BLE power-save configuration both in Advertising and Connected modes	Go to README
BLE - Privacy		X	X		BLE privacy support implemented by changing the device address frequently	Go to README
BLE - Secure Connection		X	X		BLE peripheral implementing Security Manager Protocol (SMP) pairing and encryption	Go to README
BLE - Testmodes		X	X		Testing the BLE Generic Access Profile (GAP) peripheral role	Go to README
BLE - Throughput		X	X	X	Measuring the BLE throughput	Go to README
BLE - Unified AE Coex App		X			Implementing BLE AE mode with a central and peripheral	Go to README

SiWx91x Device Management Examples

Firmware Update Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Wi-Fi - Firmware Update	X	X	X		Downloading and updating firmware as a TCP client	Go to README
Wi-Fi - HTTP OTAF Update		X	X		Downloading and updating firmware as an HTTP client	Go to README
Wi-Fi - M4 Firmware Update		X			Downloading and updating the application processor's firmware	Go to README

SiWx91x MCU Examples

The following SiWx91x microcontroller unit host (MCU) examples are available:

- Peripheral Examples
- Driver Examples
- Service Examples

Peripheral Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Empty C Project		X			Creating an empty C project to which addtional SiWx91x components and functionality may be added	Go to README
Empty CPP Project		X			Creating an empty C++ project to which addtional SiWx91x components and functionality may be added	Go to README
SiW91x - Hello World		X			Creating a hello world application to which addtional SiWx91x components and functionality may be added	Go to README
Si91x - PSRAM Blink		X			Executing from pseudostatic random-access memory (PSRAM) and performing a light-emitting diode (LED) blink operation	Go to README
Si91x - PSRAM Driver example		X			Reading and writing from and to the PSRAM	Go to README
Si91x - SL_ADC		X			Converting analog signals using the analog-to-digital converter (ADC)	Go to README
Si91x - SL ADC MULTICHANNEL		X			Converting analog signals using the analog-to-digital converter (ADC)	Go to README
Si91x - SL_CALENDAR		X			Configuring the clock and implementing triggers, alarms, and time conversion	Go to README
Si91x - SL_Config_Timer		X			Configuring and using timers	Go to README
Si91x - SL_DMA		X			Data transfer using Direct Memory Access (DMA)	Go to README
Si91x - SL_DRIVER_GPIO		X			Using the general purpose input-outputs (GPIO's) in the HP, ULP and UULP domains	Go to README
Si91x - SL_EFUSE		X			Programming the eFuse and reading data from the eFuse	Go to README
Si91x - SL_GPIO		X			Using the GPIO ports of the SiWx91x chipset	Go to README
Si91x - SL_GSPI		X			Generic Serial Peripheral Interface (GSPI) communication.	Go to README
Si91x - SL_I2C_driver_Follower		X			Data transfer from an Inter-Integrated Circuit (I2C) leader to follower and subsequently follower to leader	Go to README
Si91x - SL_I2C_driver_Leader		X			Low level driver for an I2C leader transferring data to a follower	Go to README
Si91x - SL_I2C_Follower		X			Running an I2C follower and exchanging data with a leader	Go to README
Si91x - SL_I2C_Leader		X			Running an I2C leader and exchanging data with a follower	Go to README
Si91x - SL_I2S_LOOPBACK		X			Performing an Inter-Integrated Circuit Sound (I2S) transfer using the loopback mechanism	Go to README
Si91x - SL_I2S_LOWPOWER		X			Performing an I2S loopback transfer in power save mode	Go to README

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Si91x - SL_I2S_PRIMARY		X			Performing I2S primary device transfer	Go to README
Si91x - SL_I2S_SECONDARY		X			Performing I2S secondary device transfer	Go to README
Si91x - SL_PWM		X			Using pulse width modulation (PWM) to generate a periodic pulse waveform	Go to README
Si91x - SDIO Secondary		X			Implementing a Secure Digital Input Output (SDIO) secondary and exchanging data with an SDIO primary	Go to README
Si91x - SL_SIO		X			Serial Input-Output (SIO) over SPI and Universal Asynchronous Receiver-Transmitter (UART) interfaces	Go to README
Si91x - SL_SSI_Master		X			Running a Synchronous Serial Interface (SSI) main and exchanging data with a secondary	Go to README
Si91x - SL_SSI_Slave		X			Running an SSI secondary and exchanging data with a main	Go to README
Si91x - SL_SYSRTC		X			Using the system real-time clock (SYSRTC) on the Si91x device to toggle the LED at one-second intervals	Go to README
Si91x - SL_ULP ADC		X			Using the ADC converter in ultra low-power (ULP) mode	Go to README
Si91x - SL ULP ADC MULTICHANNEL		X			Using the ADC converter for multichannel in ultra low-power (ULP) mode	Go to README
Si91x - SL_ULP_CALENDAR		X			Using the clock in ULP mode	Go to README
Si91x - SL_ULP_DMA		X			Performing a DMA transfer in ULP mode	Go to README
Si91x - SL_ULP_GPIO		X			Using the Si91x GPIO ports in ULP mode	Go to README
Si91x - SL_ULP_I2C_Leader		X			Implementing an I2C leader in ULP mode	Go to README
Si91x - SL_ULP_SSI_MASTER		X			Implementing an SSI primary in ULP mode	Go to README
Si91x - SL_ULP_Timer		X			Using timers in ULP mode	Go to README
Si91x - SL_ULP_UART		X			Transferring data over the UART interface in ULP mode	Go to README
Si91x - SL_USART		X			Exchanging data over USART running on a real-time operating system (RTOS) host	Go to README
Si91x - SL_Watchdog_Timer		X			Implementing a watchdog timer which resets the system on an exception condition	Go to README

Driver Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Si91x - Blinky		X			Toggling the LED	Go to README
Si91x - Button Baremetal		X			Handling button presses by toggling LEDs	Go to README
Si91x - MEMLCD Baremetal		X			Displaying data on the Liquid Crystal Display (LCD) peripheral	Go to README
Si91x - SL_JOYSTICK		X			Using the joystick on the SiWx91x and reporting its position	Go to README
Si91x - SL_SI70xx		X			Measuring relative humidity and temperature with the Si70XX sensor	Go to README

Service Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Si91x - iostream usart baremetal		X			Exchanging data using Universal Synchronous/Asynchronous Receiver-Transmitter (USART) communication over the virtual COM (VCOM) port	Go to README
sl_si91x_nvm3_common_flash		X			Using the Non-Volatile Memory (NVM3) service of the SiWx91x in Common Flash configuration to maintain key-value pairs in the flash memory	Go to README
sl_si91x_nvm3_dual_flash		X			Using the NVM3 service of the SiWx91x Dual Flash configuration to maintain key-value pairs in the flash memory	Go to README
Si91x - SL_POWER_MANAGER_M4_WIRELESS		X			Using the power manager for the application processor	Go to README
sl_si91x_sensorhub - sensorhub example		X			Reading data from the sensors on the Si91x device	Go to README
Si91x - Sleep Timer		X			Implementing one-time and periodic sleep timers	Go to README

Cryptography Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Crypto - AES		X			Encrypting and decrypting data using the advanced encryption standard (AES) method	Go to README
Crypto - Attestation		X			Performing attestation on the SiWx91x using cryptographic methods	Go to README
Crypto - ECDH		X			Using the elliptic-curve Diffie-Hellman (ECDH) cryptographic APIs of the SiWx91x device	Go to README

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Crypto - HMAC		X			Generating a message digest using the HMAC cryptographic method	Go to README
Crypto - SHA		X			Generating a message digest using the SHA cryptographic method	Go to README
Si91x - SoC PSA AES Cipher		X			Implementing the Platform Security Architecture (PSA) Advanced Encryption Standard (AES)	Go to README
Si91x - SoC PSA asymmetric key storage		X			Generating and storing assymetric cryptographic keys	Go to README
Si91x - SoC PSA CCM		X			Implementing PSA with the counter with cipher block chaining message authentication code (CCM) encryption method	Go to README
Si91x - SoC PSA CHACHAPOLY		X			Implementing PSA with the ChaChaPoly encryption method	Go to README
Si91x - SoC PSA ECDH		X			Implementing PSA with the elliptic-curve Diffie-Hellman (ECDH) encryption method	Go to README
Si91x - SoC PSA GCM		X			Implementing PSA with the Galois/Counter Mode (GCM) of operation	Go to README
Si91x - SoC PSA HMAC		X			Implementing PSA with hash-based message authentication code (HMAC)	Go to README
Si91x - SoC PSA SHA		X			Implementing PSA with the secure hash algorithm (SHA) encryption method	Go to README
Si91x - SoC PSA symmetric key storage		X			Generating and storing assymetric cryptographic keys	Go to README

Network Protocol Examples

The following network protocol examples are available:

- Ping Examples
- SNTP Examples

Ping Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Wi-Fi - Station Ping		X	X		Implementing the Internet Control Message Protocol (ICMP), better known as the ping protocol	Go to README
Wi-Fi - Station Ping V6		X			Implementing the ping protocol (ICMP) over IPv6	Go to README

SNTP Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Wi-Fi - SNTP Client		X	X		Implementing the Simple Network Time Protocol (SNTP)	Go to README

Application Protocol Examples

The following application protocol examples are available:

- HTTP Examples
- MQTT Examples

HTTP Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Wi-Fi - HTTP Client		X	X		Implementing a Hyper-Text Transfer Protocol (HTTP) client to exchange data with an HTTP server	Go to README
Wi-Fi - HTTP OTAF Update		X	X		Downloading and updating firmware as an HTTP client	Go to README
Wi-Fi Coex - Wi-Fi Client HTTPS BLE Dual Role		X			Simultaneous connection over secure HTTP (HTTPS) and as a BLE central and peripheral	Go to README

MQTT Examples

Example	Featured	SoC	NCP	PSRAM	Description	Link to README page
Wi-Fi - AWS IoT MQTT Client		X	X		Connecting to an AWS MQTT broker and publishing data	Go to README
Wi-Fi - Embedded MQTT Client		X	X		Connecting to an MQTT broker, publishing, and subscribing to MQTT topics	Go to README
Wi-Fi - Embedded MQTT Client TWT		X	X		MQTT client using the Wi-Fi TWT power-save mode	Go to README