User Guide

N32WB03x API Function Guide

Introduction

The purpose of this document is to enable users to quickly become familiar with the usage of N32WB03x series Bluetooth SOC chip API functions, so as to reduce the preparation time for development and lower the difficulty of development.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Table of Contents

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

1 Bluetooth Application Module：ns_ble.h

API Directory：middlewares\Nationstech\ble_library\ns_library\ble

Source files：ns_ble.c, ns_ble.h, ns_ble_task.c, ns_ble_task.h

Introduction: Bluetooth application related APIs, communicating between user application code and Bluetooth protocol stack.

1.1 ns_ble_stack_init

Function:

Bluetooth protocol stack initialization, registering Bluetooth message callbacks and user defined message callbacks. In the struct ns_stack_cfg_t structure passed in as a parameter, ble_msg_handler is the Bluetooth message callback function that can handle messages declared in enum app_ble_msg. user_msg_handler is the user message callback function. Users can declare host messages, noting that the message number starts from APP_FREE_EVE_FOR_USER, specifically referring to enum user_msg_id. Users can immediately hang up a message through the function ke_msg_send_basic, or hang up a message at a timed interval through the function ke_timer_set. To have a recurring timed message, re-timed hangup of the message can be done again in message handling.

Syntax：

```
1.  void ns_ble_stack_init(struct ns_stack_cfg_t const* p_handler);
```

Parameters：[in] p_handler：Bluetooth application callback function configuration,
see struct ns_stack_cfg_t definition for details.

Return: None

```
1.  struct ns_stack_cfg_t app_handler;
2.  app_handler.ble_msg_handler  = app_ble_msg_handler;    //user ble msg handler
3.  app_handler.user_msg_handler = app_user_msg_handler;  //user custom msg handler
4.  ns_ble_stack_init(&app_handler);
```

User-defined message callback implementation:

```
5.  void app_user_msg_handler(ke_msg_id_t const msgid, void const *p_param)
6.  {
7.      switch (msgid)
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

```
8.      {
9.          case APP_CUSTS_TEST_EVT:
10.             app_usart_tx_process();
11.             break;
12.         default:
13.             break;
14.     }
15. }
```

Bluetooth message callback implementation:

```
16. void app_ble_msg_handler(struct ble_msg_t const *p_ble_msg)
17. {
18.     switch (p_ble_msg->msg_id)
19.     {
20.         case APP_BLE_OS_READY:
21.             NS_LOG_INFO("APP_BLE_OS_READY\r\n");
22.             break;
23.         case APP_BLE_GAP_CONNECTED:
24.             app_ble_connected();
25.             break;
26.         case APP_BLE_GAP_DISCONNECTED:
27.             app_ble_disconnected();
28.             break;
29.         default:
30.             break;
31.     }
32. }
```

1.2 ns_ble_gap_init

Function: Bluetooth common parameter configuration, such as Bluetooth MAC address, name, role, connection parameters, etc. See struct ns_gap_params_t definition for details.

Syntax：

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

```
1. void ns_ble_gap_init(struct ns_gap_params_t const* p_dev_info);
```

Paramters：[in] p_dev_info：Bluetooth common parameter structure pointer..

Return: None

Examples：

```
1. struct ns_gap_params_t dev_info = {0};
2. memcpy(dev_info.mac_addr.addr, "\x01\x02\x03\x04\x05\x06" , BD_ADDR_LEN);
3. dev_info.mac_addr_type = GAPM_STATIC_ADDR;
4. dev_info.appearance = 0;
5. dev_info.dev_role = GAP_ROLE_PERIPHERAL;
6. dev_info.dev_name_len = sizeof(CUSTOM_DEVICE_NAME)-1;
7. memcpy(dev_info.dev_name, CUSTOM_DEVICE_NAME, dev_info.dev_name_len);
8. dev_info.dev_conn_param.intv_min =
   MSECS_TO_UNIT(MIN_CONN_INTERVAL,MSECS_UNIT_1_25_MS);
9. dev_info.dev_conn_param.intv_max =
   MSECS_TO_UNIT(MAX_CONN_INTERVAL,MSECS_UNIT_1_25_MS);
10. dev_info.dev_conn_param.latency    = SLAVE_LATENCY;
11. dev_info.dev_conn_param.time_out = MSECS_TO_UNIT(CONN_SUP_TIMEOUT,MSECS_UNIT_10_MS);
12. dev_info.conn_param_update_delay = FIRST_CONN_PARAMS_UPDATE_DELAY;
13. ns_ble_gap_init(&dev_info);
```

1.3 ns_ble_add_prf_func_register

Function: Register service (profile) add function. The system will later call the registered service add function to add corresponding services.

Syntax：

```
1. bool ns_ble_add_prf_func_register(ns_ble_add_prf_func_t func);
```

Parameters: [in] func: Add service (profile) function, implement the function with reference to sample function app_dis_add_dis.

Return: true: Registration succeeded; false: Registration failed

Example:：

```
1. ns_ble_add_prf_func_register(app_dis_add_dis);
```

1.4 ns_ble_prf_task_register

Function: Register service (profile) subtask events to Bluetooth application event callback list.

Syntax：

```
1. bool ns_ble_prf_task_register(struct prf_task_t *prf);
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Parameters: [in] prf: Service (profile) subtask structure pointer, see struct prf_task_t definition for details.

Return: true: Registration succeeded; false: Registration failed

Example:

```
1. //register application subtask to app task
2. struct prf_task_t prf;
3. prf.prf_task_id = TASK_ID_DISS;
4. prf.prf_task_handler = &app_dis_handlers;
5. ns_ble_prf_task_register(&prf);
```

1.5 prf_get_itf_func_register

Function: Register service (profile) task interface function getting function. Note that this function is implemented and declared in prf.c file along with prf.h file.

Syntax:：

```
1. #include "prf.h"
2. bool prf_get_itf_func_register(struct prf_get_func_t *prf);
```

Parameters: [in] prf: Service task interface getting function pointer.

Return: true: Registration succeeded; false: Registration failed

Example:

```
1. //register get itf function to prf.c
2. struct prf_get_func_t get_func;
3. get_func.task_id = TASK_ID_DISS;
4. get_func.prf_itf_get_func = diss_prf_itf_get;
5. prf_get_itf_func_register(&get_func);
```

1.6 ns_ble_adv_init

Function: Initialize BLE Bluetooth advertising parameters and initialize.

Syntax:：

```
1. void ns_ble_adv_init(struct ns_adv_params_t const* p_adv_init);
```

Parameters: [in] p_adv_init: Advertising initialization parameter structure pointer.

Return: None

Example:

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

```
1.  struct ns_adv_params_t user_adv = {0};
2.  //init advertising data
3.  user_adv.adv_data_len = ADVERTISE_DATA_LEN;
4.  memcpy(user_adv.adv_data,ADVERTISE_DATA,ADVERTISE_DATA_LEN);
5.  user_adv.scan_rsp_data_len = ADV_SCNRSP_DATA_LEN;
6.  memcpy(user_adv.scan_rsp_data,ADV_SCNRSP_DATA,ADV_SCNRSP_DATA_LEN);
7.  user_adv.attach_appearance  = false;
8.  user_adv.attach_name        = true;
9.  user_adv.ex_adv_enable      = false;
10. user_adv.adv_phy            = PHY_1MBPS_VALUE;
11. user_adv.directed_adv.enable = false;
12. user_adv.fast_adv.enable    = true;
13. user_adv.fast_adv.duration  = CUSTOM_ADV_FAST_DURATION;
14. user_adv.fast_adv.adv_intv  = CUSTOM_ADV_FAST_INTERVAL;
15. ser_adv.slow_adv.enable     = true;
16. user_adv.slow_adv.duration  = CUSTOM_ADV_SLOW_DURATION;
17. user_adv.slow_adv.adv_intv  = CUSTOM_ADV_SLOW_INTERVAL;
18. user_adv.ble_adv_msg_handler = app_ble_adv_msg_handler;
19. ns_ble_adv_init(&user_adv);
```

1.7 ns_ble_adv_start

Function: Start (enable) Bluetooth advertising.

Syntax:：

```
1.  void ns_ble_adv_start(void);
```

Parameters: None

Return: None

Example:

```
1.  ns_ble_adv_start();
```

1.8 ns_ble_adv_stop

Function: Stop Bluetooth advertising.

Syntax:：

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

```
1. void ns_ble_adv_stop(void);
```

Parameters: None

Return: None

Example:：

```
1. ns_ble_adv_stop();
```

1.9 ns_ble_adv_data_set

Function: Set the content of the advertising data packet.

Syntax::：

```
1. void ns_ble_adv_data_set(uint8_t* p_dat, uint16_t len);
```

Parameters: [in] p_dat: Set advertising packet data. [in] len: Set length of advertising packet data.

Return: None

Example:：

```
1. ns_ble_adv_data_set(CUSTOM_USER_ADVERTISE_DATA,CUSTOM_USER_ADVERTISE_DATA_LEN);
```

1.10 ns_ble_scan_rsp_data_set

Function: Set the content of the advertising scan response data packet.

Syntax:：

```
1. void ns_ble_scan_rsp_data_set(uint8_t* p_dat, uint16_t len);
```

Parameters: [in] p_dat: Set advertising scan response packet data. [in] len: Set length of advertising scan response packet data.

Return: None

Example:：

```
1. ns_ble_scan_rsp_data_set(CUSTOM_USER_ADV_SCNRSP_DATA,CUSTOM_USER_ADV_SCNRSP_DATA_LEN);
```

1.11 ns_ble_ex_adv_data_set

Function: Set the content of the extended broadcast data packet.

Syntax:：

```
1. void ns_ble_ex_adv_data_set(uint8_t* p_dat, uint16_t len);
```

Parameters:

[in] p_dat: Set the extended broadcast packet data, note that the pointer to the local variable cannot be used here.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

[in] len: Set the length of the extended broadcast packet data.

Return: None

Example:

```
1. const static uint8_t ex_adv[] = {"\x29\xff""12345678901234567890123456789012345678901234567890"};
2. ns_ble_ex_adv_data_set((uint8_t*) ex_adv, sizeof(ex_adv)-1);
```

1.12 ns_ble_scan_init

Function: Initialize the parameters for the BLE scan function.

Syntax:：

```
1. void ns_ble_scan_init(struct ns_scan_params_t *p_init);
```

Parameters:

[in] p_init: Pointer to the scan function parameter structure, refer to the internal definition of the ns_scan_params_t structure for details.

Return: None

Example:

```
1. struct ns_scan_params_t init = {0};
2. static const uint8_t target_name[] = {"NS_RDTS_SERVER"};
3. init.type             = SCAN_PARAM_TYPE;
4. init.dup_filt_pol     = SCAN_PARAM_DUP_FILT_POL;
5. init.connect_enable   = SCAN_PARAM_CONNECT_EN;
6. init.prop_active_enable = SCAN_PARAM_PROP_ACTIVE;
7. init.scan_intv        = SCAN_PARAM_INTV;
8. init.scan_wd          = SCAN_PARAM_WD;
9. init.duration         = SCAN_PARAM_DURATION;
10. init.filter_type     = SCAN_FILTER_BY_NAME;
11. init.filter_data     = (uint8_t*)&target_name;
12. ns_ble_scan_init(&init);
```

1.13 ns_ble_start_scan

Function: Start the BLE scan function.

Syntax:

```
1. void ns_ble_start_scan(void);
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Parameters: None

Return: None

Example:

```
1.  ns_ble_start_scan();
```

1.14 ns_ble_stop_scan

Function: Stop the BLE scan function.

Syntax:

```
1.  void ns_ble_ stop_scan(void);
```

Parameters: None

Return: None

Example:

```
1.  ns_ble_ stop _scan();
```

1.15 ns_ble_start_init

Function: The host device actively initiates the BLE connection. Note that only the master device can call this function.

Syntax:：

```
1.  void ns_ble_start_init(uint8_t *addr, uint8_t addr_type);
```

Parameters: [in] addr: The address of the slave device to connect to. [in] addr_type: The address type of the device to connect to, the returned scan information contains the address type.

Return: None

Example:

```
1.  ns_ble_start_init("\x11\x11\x11\x11\x11\x11",GAPM_STATIC_ADDR);
```

1.16 ns_ble_update_param

Function: The master or slave device actively initiates the BLE connection parameter update request.

Syntax:

```
1.  void ns_ble_update_param(struct gapc_conn_param *conn_param);
```

Parameters: [in] conn_param: Pointer to the connection parameter structure.

Return: None

Example:

```
1.  struct gapc_conn_param conn_param;
2.  conn_param.intv_min = 12;  //15ms
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

```
3.  conn_param.intv_max = 12;  //15ms
4.  conn_param.latency = 5;
5.  conn_param.time_out = 500; //5000ms
6.  ns_ble_update_param(&conn_param);
```

1.17 ns_ble_mtu_set

Function: The master or slave device actively initiates the BLE mtu parameter update request. It should be noted that the valid data length of the user data packet is 3 bytes less than the MTU.

Syntax:：

```
1.  void ns_ble_mtu_set(uint16_t mtu);
```

Parameters: [in] mtu: The BLE mtu value, the maximum value is 517.

Return: None

Example:：

```
1.  ns_ble_mtu_set(247); //set mtu as 247
```

1.18 ns_ble_phy_set

Function: The master or slave device actively initiates the BLE phy parameter update request.

Syntax:

```
1.  void ns_ble_phy_set(enum gap_phy_val phy);
```

Parameters: [in] phy: The BLE phy parameter value, optional parameters refer to the declaration of enum gap_phy_val.

Return: None

Example:

```
1.  ns_ble_phy_set(GAP_PHY_125KBPS); //set phy as coded 125kbps
```

1.19 ns_ble_active_rssi

Function: The master or slave device actively initiates the BLE rssi read request. The read value is returned through the APP_BLE_GAP_RSSI_IND message in the Bluetooth event callback function.

Syntax:：

```
1.  void ns_ble_active_rssi(uint32_t interval);
```

Parameters: [in] interval: The BLE rssi read interval, the time unit is milliseconds, input 0 to read only once.

Return: None

Example:

```
1.  void app_ble_msg_handler(struct ble_msg_t const *p_ble_msg)
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

```
{
    switch (p_ble_msg->msg_id)
    {
        case APP_BLE_OS_READY:
            NS_LOG_INFO("APP_BLE_OS_READY\r\n");
            break;
        case APP_BLE_GAP_CONNECTED:
            app_ble_connected();
            ns_ble_active_rssi(40);  //enable rssi read every 1000ms
            break;
        case APP_BLE_GAP_DISCONNECTED:
            app_ble_disconnected();
            break;
        case APP_BLE_GAP_RSSI_IND:
            NS_LOG_INFO("rssi:%d\r\n",p_ble_msg->msg.p_gapc_rssi->rssi); //log rssi value
            break;
        default:
            break;
    }
}
```

1.20 ns_ble_disconnect

Function: The master or slave device actively initiates the BLE disconnection request. The successful disconnection will be returned through the APP_BLE_GAP_DISCONNECTED message in the Bluetooth event callback function.

Syntax:：

```
void ns_ble_disconnect(void);
```

Parameters: None

Return: None

Example:：

```
ns_ble_disconnect();
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

1.21 ns_ble_dle_set

Function: The master or slave device requests to set the DLE parameters. It is recommended to use the parameters in the example code if you need to change them.

Syntax:：

```
1.  void ns_ble_dle_set(uint16_t tx_octets, uint16_t tx_time);
```

Parameters: [in] tx_octets: The maximum data amount of a single link data channel PDU. [in] tx_time: The maximum number of microseconds to send a single link data channel PDU.

Return: None

Example:

```
1.  ns_ble_dle_set(251, 2120); //suggest parameter
```

1.22 rf_tx_power_set

Function: The master or slave device sets the transmission power of the radio signal.

Syntax:

```
1.  void rf_tx_power_set(rf_tx_power_t pwr);
```

Parameters: [in] pwr: The set radio signal transmit power, optional parameters are as follows.

```
1.  typedef enum
2.  {
3.      TX_POWER_0_DBM = 0, /*  0 dBm */
4.      TX_POWER_Neg2_DBM,  /* -2 dBm */
5.      TX_POWER_Neg4_DBM,  /* -4 dBm */
6.      TX_POWER_Neg8_DBM,  /* -8 dBm */
7.      TX_POWER_Neg15_DBM, /* -12 dBm */
8.      TX_POWER_Neg20_DBM, /* -20 dBm */
9.      TX_POWER_Pos2_DBM,  /* +2 dBm */
10.     TX_POWER_Pos3_DBM,  /* +3 dBm */
11.     TX_POWER_Pos4_DBM,  /* +4 dBm */
12.     TX_POWER_Pos6_DBM,  /* +6 dBm */
13. }rf_tx_power_t;
```

Return: None

Example

```
1.  rf_tx_power_set(TX_POWER_Pos4_DBM);
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

1.23 Process functions

void ns_ble_adv_fsm_next(void);

Function: Slave device broadcast state management function. User code does not need to call separately.

void ns_ble_create_scan(void);

Function: The host creates a scan activity. Calling the ns_ble_scan_init function correctly to initialize the scan module will create it automatically. User code does not need to call separately.

void ns_ble_delete_scan(void);

Function: The host deletes the scan activity. User code does not need to call separately.

void ns_ble_create_init(void);

Function: The host creates the master connection activity, which will be created automatically. User code does not need to call separately.

void ns_ble_delete_init(void);

Function: The host deletes the connection activity. User code does not need to call separately.

bool ns_ble_scan_data_find(uint8_t types, const uint8_t *p_filter_data, uint8_t *p_data, uint8_t len);

Function: The host finds the specified data in the returned broadcast data packet during scanning. Calling the ns_ble_scan_init function correctly to initialize the scan module will automatically call based on the filter configuration. User code does not need to call separately.

bool ns_ble_add_svc(void);

Function: Adding services (profiles) that have been initially registered during initialization. User code does not need to call separately.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

2 Bluetooth Security Encryption Module：ns_sec.h

API Directory: middlewares\Nationstech\ble_library\ns_library\sec

Source files: ns_sec.c, ns_sec.h

Introduction: Bluetooth security encryption related APIs, communicating between user application code and Bluetooth protocol stack.

2.1 ns_sec_init

Function: Initialize the Bluetooth security encryption module.

Syntax:：

```
1. void ns_sec_init(struct ns_sec_init_t const* init);
```

init: Bluetooth security module initialization parameter structure pointer. Refer to the internal definition of the structure ns_sec_init_t for details.

Return: None

Example:

```
1. struct ns_sec_init_t sec_init={0};
2. sec_init.rand_pin_enable = false;
3. sec_init.pin_code = 123456;
4. sec_init.pairing_feat.auth     = ( SEC_PARAM_BOND | (SEC_PARAM_MITM<<2) | (SEC_PARAM_LESC<
   <3) | (SEC_PARAM_KEYPRESS<<4) );
5. sec_init.pairing_feat.iocap    = SEC_PARAM_IO_CAPABILITIES;
6. sec_init.pairing_feat.key_size  = SEC_PARAM_KEY_SIZE;
7. sec_init.pairing_feat.oob      = SEC_PARAM_OOB;
8. sec_init.pairing_feat.ikey_dist = SEC_PARAM_IKEY;
9. sec_init.pairing_feat.rkey_dist = SEC_PARAM_RKEY;
10. sec_init.pairing_feat.sec_req   = SEC_PARAM_SEC_MODE_LEVEL;
11. sec_init.bond_enable          = BOND_STORE_ENABLE;
12. sec_init.bond_db_addr         = BOND_DATA_BASE_ADDR;
13. sec_init.bond_max_peer        = MAX_BOND_PEER;
14. sec_init.bond_sync_delay      = 5000;
15. sec_init.ns_sec_msg_handler    = NULL;
16. ns_sec_init(&sec_init);
```

2.2 ns_sec_get_bond_status

Function: Return the bonded status.

Syntax:：

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

```
1. bool ns_sec_get_bond_status(void);
```

Parameters: None.

Return: ture: Already bonded. false: Not bonded.

Example:

```
1. bool bond_status = ns_sec_get_bond_status();
```

2.3 ns_sec_get_iocap

Function: Return the device interface capability parameter for calling by related Bluetooth libraries, generally user code does not need to call.

Syntax:：

```
1. uint8_t ns_sec_get_iocap(void);
```

Parameters: None.

Return: The device interface capability parameter is the value set in by the ns_sec_init function initialization.

Example:

```
1. uint8_t io_cap = ns_sec_get_iocap();
```

2.4 ns_sec_bond_db_erase_all

Function: Erase all bonded devices. Because erasing flash will block running, it is recommended not to execute when already connected, which must affect the possibility of connection.

Syntax:

```
1. void ns_sec_bond_db_erase_all(void);
```

Parameters: None

Return: None

Example:

```
1. ns_sec_bond_db_erase_all();
```

2.5 ns_sec_send_security_req

Function: Send a security request to the master device. (Generally user code does not need to call.)

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

2.6 ns_sec_bond_store_evt_handler

Function: Callback function of the data storage task for the bonded device information, used internally.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

3 Software Timer：ns_timer.h

API Directory：middlewares\Nationstech\ble_library\ns_library\timer

Source files：ns_timer.c, ns_timer.h

Introduction: Software timer module, encapsulated by the ke_timer module in the Bluetooth protocol stack, which needs to be used after the protocol stack is initialized, that is, after the APP_BLE_OS_READY message is sent. Since it takes more than one low-speed clock tick (about 32us) to wake up from sleep before calling ke_timer_ser.

3.1 ns_timer_create

Function: Create a software timer that calls the fn function after a delay of delay milliseconds. For a cyclic timer, you can create another timer in the callback handler again. Note that it can only be used after the protocol stack is initialized, that is, after the APP_BLE_OS_READY message is sent out.

Syntax:

```
1. timer_hnd_t ns_timer_create(const uint32_t delay, timer_callback_t fn);
```

Parameters: [in] delay: The delay time in milliseconds. [in] fn: The callback function after a delay of delay milliseconds.

Return: The id of the timer

Example:

```
1. timer_hnd_t timer_id = ns_timer_create(1000, my_timer_callback);
```

3.2 ns_timer_modify

Function: Modify the delay time of a specified timer.

Syntax:：

```
1. timer_hnd_t ns_timer_modify(const timer_hnd_t timer_id, const uint32_t delay);
```

Parameters: [in] timer_id: The timer id to modify. [in] Delay: The modified delay time in milliseconds.

Return: The modified timer id.

Example:

```
1. timer_id = ns_timer_modify(timer_id, 2000);
```

3.3 ns_timer_cancel

Function: Cancel a specified timer.

Syntax:

```
1. void ns_timer_cancel(const timer_hnd_t timer_id);
```

Parameters: [in] timer_id: The timer id to cancel.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Return: None

Example:

```
1.  ns_timer_cancel(timer_id);
```

3.4 ns_timer_cancel_all

Function: Cancel all created timers

Syntax：

```
1.  void ns_timer_cancel_all(void);
```

Parameters：None

Return：None

Example：

```
1.  ns_timer_cancel_all();
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

4 Bluetooth Sleep Module：ns_sleep.h

API Directory：middlewares\Nationstech\ble_library\ns_library\sleep

Source files：ns_sleep.c, ns_sleep.h

Introduction: Bluetooth protocol stack sleep function module.

4.1 ns_sleep

Function: Bluetooth sleep function entry function. Based on the Bluetooth protocol stack working status, it will automatically enter sleep state if there is no task to execute, and resume system status after being woken up by timed task or hardware interrupt. It is usually called in the main function main loop, after the rwip_schedule dispatch function.

Syntax:

```
1.  void ns_sleep(void);
```

Parameters：None

Return：None

Example：

```
1.  ns_sleep();
```

4.2 ns_sleep_lock_acquire

Function: Bluetooth sleep mode lock request, i.e. apply for the system not to enter sleep mode. For example, to enable some high-speed peripherals (such as UART) that you do not want the system to turn them off when entering sleep, you can prohibit the system from entering sleep by requesting a sleep mode lock.

Syntax：

```
1.  uint8_t ns_sleep_lock_acquire(void);
```

Parameters: None

Return: true: Lock request succeeded. false: Lock request failed.

Example:：

```
1.  if(ns_sleep_lock_acquire())
2.  {
3.      //require sleep lock success
4.  }
```

4.3 ns_sleep_lock_release

Function: Release Bluetooth sleep mode lock. After all sleep mode locks are released, the system can enter sleep mode

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

when there is no pending task.

Syntax:

```
1.  uint8_t ns_sleep_lock_release(void);
```

Parameters: None

Return: true: Release lock successfully. false: Release lock failure, currently no lock to release.

Example:

```
1.  if(ns_sleep_lock_release())
2.  {
3.      //release sleep lock success
4.  }
```

4.4 Virtual functions for entering and exiting sleep mode

__weak void app_sleep_prepare_proc(void);

Function: Preset virtual function for users to reimplement tasks that need to be done before entering sleep.

__weak void app_sleep_resume_proc(void)

Function: Preset virtual function for users to reimplement tasks that need to be done after sleep wake up, for example, peripherals closed during sleep can be reinitialized.。

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

5 Debug Information Printing Module：log.h

API Directory：middlewares\Nationstech\ble_library\ns_library\log

Source files：ns_log_lpuart.c, ns_log_lpuart.h, ns_log_usart.c, ns_log_usart.h, ns_log.h

Introduction: Debug information output function module. Currently available output hardware is LPUART and USART. LPUART uses PB1 as its default TX pin, and USART uses PB6 of USART1 as its default TX pin.

5.1 NS_LOG_INIT

Function: Initialize the debug information printing module.

Syntax:

```
1. NS_LOG_INIT();
```

5.2 NS_LOG_DEINIT

Function: Cancel initialization of the debug information printing module.

Syntax:

```
1. NS_LOG_DEINIT ()
```

5.3 Debug information print output

Function: Call debug information print output functions of different levels. The syntax is the same as the print function. Output functions of different levels are controlled by different macros, NS_LOG_ERROR_ENABLE, NS_LOG_WARNING_ENABLE, NS_LOG_INFO_ENABLE and NS_LOG_DEBUG_ENABLE.

Syntax:

```
1. NS_LOG_DEBUG(...);
2. NS_LOG_INFO(...);
3. NS_LOG_WARNING(...);
4. NS_LOG_ERROR(...) ;
```

5.4 Related enabling macros

//Whether to enable LPUART (PB1) as the output hardware for the log module

#define NS_LOG_LPUART_ENABLE 0

//Whether to enable USART (PB6) as the output hardware for the log module

#define NS_LOG_USART_ENABLE 0

//Whether to enable ERROR level output functions

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

```
#define NS_LOG_ERROR_ENABLE 0

//Whether to enable WARNING level output functions

#define NS_LOG_WARNING_ENABLE 0

//Whether to enable INFO level output functions

#define NS_LOG_INFO_ENABLE 0

//Whether to enable DEBUG level output functions

#define NS_LOG_DEBUG_ENABLE 0

// Whether to enable output of debug information with color

#define PRINTF_COLOR_ENABLE 0
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

6 Hard Delay Module：ns_delay.h

API Directory: middlewares\Nationstech\ble_library\ns_library\delay

Source File: ns_delay.h

Introduction: Hard delay function module. The related functions declared in it are implemented in ROM code. Users can directly call them after adding the header file. It should be noted that the delay time is not accurate. For accurate delays, it is recommended to use hardware timer functions.

6.1 delay_cycles

Function: Delay and wait for the specified number of cycles to return.

Syntax：

```
1. void delay_cycles(uint32_t ui32Cycles);
```

Parameters: [in] ui32Cycles: The number of cycles to wait, one cycle takes about (10/110) microseconds.

Return: None

Example:

```
1. delay_cycles(1000);
```

6.2 delay_n_us

Function: Delay and wait for the specified number of microseconds.

Syntax:

```
1. void delay_n_us(uint32_t val);
```

Parameters: [in] val: Number of microseconds to wait.

Return: None

Example:

```
1. delay_n_us(1000);
```

6.3 delay_n_10us

Function: Delay and wait for the specified number of 10 microseconds.

Syntax:

```
1. void delay_n_10us(uint32_t val);
```

Parameters: [in] val: Number of 10 microseconds to wait.

Return: None

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Example:

```
1.  delay_n_10us(1000);
```

6.4 delay_n_100us

Function: Delay and wait for the specified number of 100 microseconds.

Syntax:

```
1.  void delay_n_100us(uint32_t val);
```

Parameters: [in] val: Number of 100 microseconds to wait.

Return: None

Example:

```
1.  delay_n_100us(1000);
```

6.5 delay_n_ms

Function: Delay and wait for the specified number of milliseconds.

Syntax:

```
1.  void delay_n_ms(uint32_t val);
```

Parameters: [in] val: Number of milliseconds to wait.

Return: None

Example:

```
2.  delay_n_ms(1000);
```


NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

7 Suggestions for Bluetooth Programming

- Do not continuously execute code that takes too long, which will hinder Bluetooth message handling. It is recommended to fragmentize user code as much as possible, with each task message or polling only taking up a small amount of time.
- Do not execute interrupt service functions for too long to avoid hindering Bluetooth message handling. It is recommended to implement logical code in tasks or polling by pending messages, timers or flag bits.
- Do not call hardware peripherals in interrupt service functions. If an interrupt event is pending after sleep wake-up, the interrupt service function will be called before app_sleep_resume_proc, and the peripheral has not been initialized after wake-up. Calling hardware peripherals in interrupt service functions will cause errors because the peripherals are not initialized.
- For projects containing Bluetooth, interrupt service functions need to be registered through ModuleIrqRegister, and the interrupt priority of user code can only use 2 and 3 (Bluetooth protocol stack uses 0 and 1).
- Note that high-speed peripherals (such as USART) will be turned off in low power sleep mode, and need to be initialized by the program after wake-up before they can be used.
- It is recommended that the user's logical code be implemented in task callback functions, which can be message event callbacks or timed task callbacks.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

8 Version History

Version	Date	Changes
V1.0	2021.12.27	Initial version

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

9 Disclaimer

This document is the exclusive property of Nations Technologies Inc. (Hereinafter referred to as NATIONS). This document, and the product of NATIONS described herein (Hereinafter referred to as the Product) are owned by NATIONS under the laws and treaties of the People's Republic of China and other applicable jurisdictions worldwide. NATIONS does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. Names and brands of third party may be mentioned or referred thereto (if any) for identification purposes only. NATIONS reserves the right to make changes, corrections, enhancements, modifications, and improvements to this document at any time without notice. Please contact NATIONS and obtain the latest version of this document before placing orders. Although NATIONS has attempted to provide accurate and reliable information, NATIONS assumes no responsibility for the accuracy and reliability of this document. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. In no event shall NATIONS be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this document or the Product. NATIONS Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage". Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, all types of safety devices, and other applications intended to support or sustain life. All Insecure Usage shall be made at user's risk. User shall indemnify NATIONS and hold NATIONS harmless from and against all claims, costs, damages, and other liabilities, arising from or related to any customer's Insecure Usage. Any express or implied warranty with regard to this document or the Product, including, but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement are disclaimed to the fullest extent permitted by law. Unless otherwise explicitly permitted by NATIONS, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg