UM1770
User manual

BlueNRG profile application interface

Introduction

This document describes the API interfaces and related events of the Bluetooth low energy (BLE) profiles.

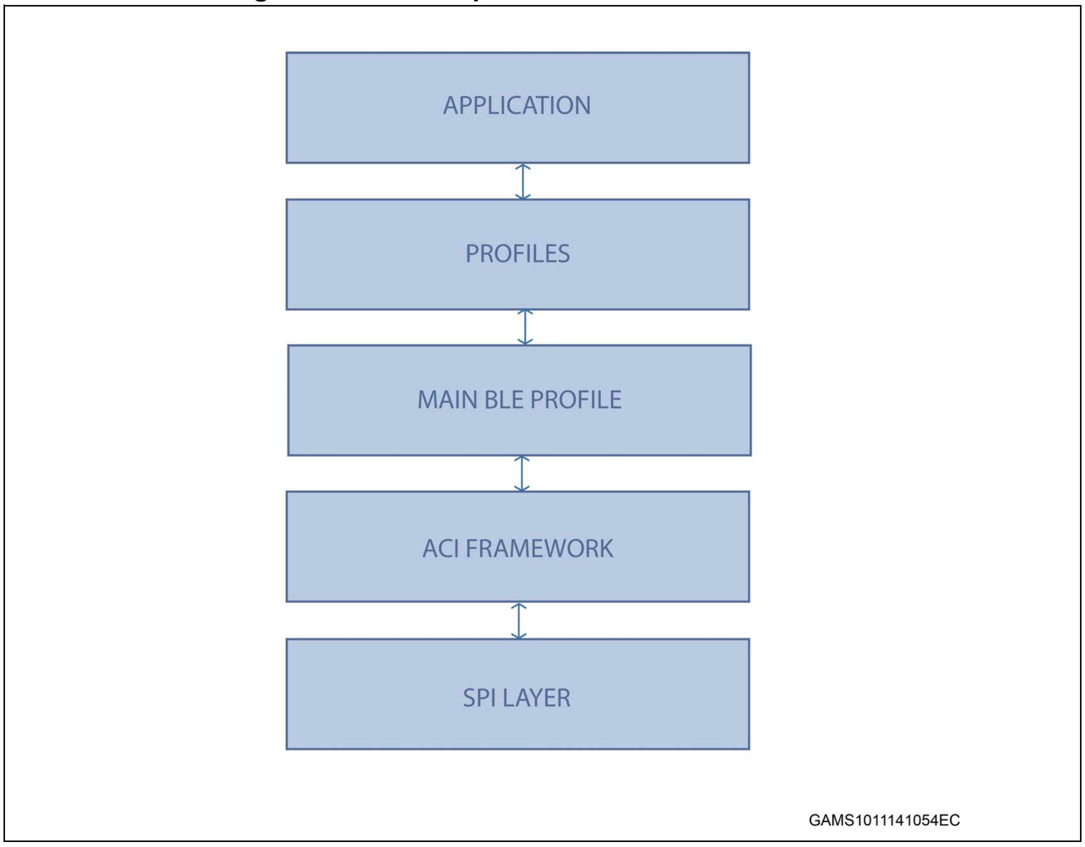
These APIs allow the management of communication between a user application and the available Bluetooth low energy profiles.

Contents

1 Architecture

Figure 1 describes the BlueNRG profile framework architecture:

Figure 1. BlueNRG profile framework architecture



The following is a description of each profile layer:

- Application:
  - user/test application using Bluetooth low energy (BLE) profiles framework
- Profiles:
  - specific profile implementation (alert notification, find me, etc.)
- Main BLE profile:
  - main/common BLE profile framework for all BLE profiles (file profile.c)
  - it provides functions for main profile initialization, profile registration, event handlers and notifies events to specific profiles.
- ACI framework:
  - it exposes functions to the upper layers to send the various commands supported by the BlueNRG (standard HCI and vendor specific ones)
  - all the commands are sent to the controller via the ACI framework (bluenrg_gap_aci.c, bluenrg_gatt_aci.c, bluenrg_l2cap.c, bluenrg_hal_aci.c).

Note: A specific profile may only require a subset of these commands (automatically stripped by the linker).

- SPI Layer
  - SPI layer APIs (read/write from/to BlueNRG SPI buffers)

Note: This profile framework only supports peripheral (slave) roles. No multiple profiles are supported at the same time.

2 ACI framework

All the BlueNRG commands are sent to the controller via the ACI framework. The ACI framework exposes functions to the upper layers to send the various commands supported by the BlueNRG and to get the events raised from the BlueNRG network coprocessor.

The ACI framework implements the ACI APIs according to the Bluetooth LE stack application command interface APIS defined on the UM1755 user manual on References Section.

Three types of events are handled within the profile framework:

1. BlueNRG events raised by the BlueNRG network coprocessor (refer to UM1755 user manual on *Section 7: List of references*);

2. General profile events which are used by all profiles and are not specific to any profile (refer to *Section 5.1: Generic events*);

3. Profile specific events defined by each profile.

3 Execution context

The BlueNRG host profiles framework implements a single task model for execution. The entire processing takes place in the ISR context and the main thread context. The execution context is a while(1) loop.

The following functions are processed in this loop:

1. HCI_Process(): it performs the processing of any pending events read. It is defined on file hci.c (ACI Framework);
2. Profile_Process(): it sends the commands during the initialization or pairing process and updates the main profile state machine.It is defined on file profile.c (main profile file);
3. Profile specific state machine: it is called for checking current main profile and profile state, substate and performing related actions and consequent status updates (functions *_StateMachine(void) on each specific profile);
4. Application state machine: it performs application-specific handling (sending data to profiles, enabling advertising, displaying to the user, etc.). An example of such function is the Host_Profile_Test_Application() defined within the profiles_test_application.c file (test application for profile PTS validation).

The PTS profile validation application (profiles_test_application.c) provides an example of such processing.

The BlueNRG events are notified to the main profile application (profile.c) through the HCI_Event_CB() callback which performs the required actions based on main profile state and substate. Further the HCI_Event_CB() function notifies the BlueNRG events to a specific profile by calling the profile callback function * _Event_Handler(). This function is called with the following instruction:

```
gMainProfileCtxt.bleProfileApplNotifyFunc(appNotifEvt,1,&appNotifStatus);
```

Each profile registers its BlueNRG event handler function (* _Event_Handler()) through the profile *_Init() function.

Following are more details about the main profile framework (profile.c) key functions:

- BLE_Profile_Init(): It initializes the main profile.
- BLE_Profiles_Evt_Notify_Cb(): it is the main profile callback function which is called by each profile to notify the profile specific events to the main profile application. Based on such events, main profile application can decide which actions to take in order to handle the specific profile events. This function is provided to each profile when the related *_Init() API is called.
- BLE_Profile_Register_Profile(): it allows registration of each profile callback; it is called within the profile initialization function named: *_Init() function.

Profile specific example: heart rate profile

- Heart rate profile header file: heart_rate.h file;
- Heart rate profile initialization function:HRProfile_Init(). This function performs the following operations:
- Set the main profile callback function for notifying profile specific events to the main profile application;
- Set the heart rate profile HRProfile_Rx_Event_Handler() callback function;
- Register the heart rate with BLE main Profile.

The HRProfile_Init() function is called on the profile main application as follow:

```
if (HRProfile_Init((uint8_t)0xFF,
                    BLE_Profiles_Evt_Notify_Cb,
                    0x04) == BLE_STATUS_SUCCESS)
{
   APPL_MESG_DBG(profiledbgfile,"Initialized Heart Rate Profile \n" );
}
```

- Heart rate profile callback function: HRProfile_Rx_Event_Handler(). This function allows Heart Rate profile to properly handling the BlueNRG events according to the profile state.

4 BlueNRG profile framework

The BlueNRG profile framework consists of the following main components:

1. Profile library
   – Profile_Library_Release.a file (release version)
   – Profile_Library_Debug.a file (debug version with debug messages)

This library provides supports for each of the following profiles (GAP peripheral role):

- Alert Notification Client
- Alert Notification Server
- Blood Pressure Sensor
- Find Me Locator
- Find Me Target
- Glucose Sensor
- Health Thermometer
- Hearth Rate
- Phone Alert
- Proximity Monitor
- Proximity Reporter
- Time Client
- Time Server

2. Main profile file
   – profile.c: it provides the common profile framework to all the profiles, and it implements the BlueNRG events callback HCI_Event_CB(void *pckt).
3. Generic profile interface header files:
   a) ble_events.h: it defines the generic and profiles specific events;
   b) ble_profile.h: main profile header file;
   c) ble_status.h: profile status and error codes;
   d) debug.h: function for specific profile debug messages
   e) host_config.h: define values for selecting each specific supported profile (through the BLE_CURRENT_PROFILE_ROLES definition)
   f) uuid.h: profile service & characteristics UUID as defined in the SIG specification (https://developer.bluetooth.org/gatt/profiles/Pages/ProfilesHome.aspx)

Profiles interfaces header files:

Table 1. Profiles interfaces header files

Profile	Header file
Alert Notification Client	alertNotification_Client.h
Alert Notification Server	alertNotification_Server.h
Blood Pressure Sensor	blood_pressure.h
Find Me Locator	findme_locator.h

Table 1. Profiles interfaces header files

Profile	Header file
Find Me Target	findme_target.h
Glucose Sensor	glucose_service.h glucose_sensor.h glucose_racp.h glucose_database.h
Health Thermometer	health_thermometer.h
Hearth Rate	heart_rate.h
Phone Alert	phone_alert_client.h
Proximity Monitor	proximity_monitor.h
Proximity Reporter	proximity_reporter.h
Time Client	time_client.h
Time Server	time_server.h time_profile_types.h

4. Profile test application for profile PTS validation: profiles_test_application.c. This file addresses the following features:
   - Set the profile security parameters and initialize the main profile by defining the main profile callback function:
   - Initialize the selected profile (through the BLE_CURRENT_PROFILE_ROLES definition) by calling the profile *_Init() function with the BLE_Profiles_Evt_Notify_Cb() as one of the initialization parameters;
   - It defines the while(1) loop where the HCI_process(), Profile_Process() and profile specific state machine (*_StateMachine()) functions are processed;
   - It allows to enter & process specific user commands (through serial I/O), in order to interact with each profile and performs specific actions (i.e. ask to profile to notify/indicate a characteristic). Such user input commands are used during profile PTS validation tests. The supported user input commands are defined within the profiles_test_application.c file.

Note: An EWARM project defining a workspace for each supported profile and including the profile library is available. By selecting the specific profile workspace, a profile test application supporting the selected profile is built. This application can be used for validating the profile using the PTS USB dongle and related Bluetooth PTS SW tool.

5 Application – profile interface

Function calls handle the communication between the application and profiles. Any application using the profiles should first initialize the base profile by calling the BLE_Profile_Init function, followed by a call to the profile specific initialization function. For example, HRProfile_Init() function is called for the heart rate profile. To enable execution, the application requires a loop that calls the HCI_Process() and Profile_Process() continuously.

The BLE_Profile_Init function takes two parameters:

- A pointer to the securityParameters; the security parameters should specify the io capabilities, mitm mode, bonding mode and encryption key size.
- A callback function; the callback registered should be of the form: typedef void (* BLE_CALLBACK_FUNCTION_TYPE)(tNotificationEvent event,uint8 evtLen,uint8* evtData).

This function is used by the profile to notify the application of events. When the application is notified of an event, it only reads the number of parameters specified in the evtLen parameter.

Below is the list of events sent by the various profiles to the application.

5.1 Generic events

The events in this category are not specific to any profile.

1. EVT_MP_BLUE_INITIALIZED: this event is sent to the application by the main profile when the controller has been initialized.
2. EVT_MP_CONNECTION_COMPLETE: this event is sent to the application by the main profile when a connection has been successfully established with the peer.
3. EVT_MP_PASSKEY_REQUEST: this event is sent to the application by the main profile when there is a request for passkey during the pairing process from the controller. This event has no parameters. The application must call the function BLE_Profile_Send_Pass_Key and send the passkey to the controller.
4. EVT_MP_PAIRING_COMPLETE: this event is sent to the application by the main profile when the device is successfully paired with the peer.
5. EVT_MP_DISCONNECTION_COMPLETE: this event is sent to the application by the main profile to notify the result of a disconnection procedure initiated by master/slave.
6. EVT_MP_ADVERTIZE_ERROR: this event is sent by any of the child profiles when enabling of advertising fails. It is the application's responsibility to restart advertising.
7. EVT_MP_ADVERTISING_TIMEOUT: this event is sent by the child profiles when the limited discoverable mode times out or the profile-specific advertising timeout occurs. It is the application's responsibility to restart advertising.

5.2 Heart rate profile events

The events under this category are those which are sent by the heart rate profile to the application.

1. EVT_HRS_INITIALIZED: this event is sent to the application when the heart rate profile has completed its initialization sequence and is ready to enable advertising or the initialization sequence failed. The evtData parameter contains the error code; 0X00 means the initialization was successful.
2. EVT_HRS_CHAR_UPDATE_CMPLT: this event is sent to the application whenever it has started a characteristic update procedure to update the heart rate measurement or body sensor location. The evtData contains the status, service handle, and characteristic handle. This has to be changed to give different events for each update since the application is not aware of the handles.
3. EVT_HRS_RESET_ENERGY_EXPENDED: this event is sent to the application when the peer writes a value of 0x01 to the control point characteristic. This event has no parameters. The application must restart accumulating the energy expended values from 0.

5.3 Proximity reporter events

The events under this category are those which are sent by the proximity profile in the reporter role to the application.

1. EVT_PR_INITIALIZED: this event is sent to the application when the proximity reporter has completed its initialization sequence and is ready to enable advertising, or the initialization sequence failed. The evtData parameter contains the error code; 0X00 means the initialization was successful.
2. EVT_PR_LINK_LOSS_ALERT: this event is sent to the application when a link loss is detected. The evtData contains the alert level. The application must start an alert for the level specified. The type of alert is decided by the application: the alert can continue for an application-specific duration or until another connection is established. The application must re-enable advertising to establish a new connection.
3. EVT_PR_PATH_LOSS_ALERT: this event is sent to the application by the proximity reporter when a path loss is detected. The evtData contains the alert level. When a path loss is detected, an alert of any type must be started – the desired user action would be to move the device closer to its connected peer. The alert should continue until another event with alert level 0 is issued.

5.4 Proximity monitor events

1. EVT_PM_INITIALIZED: this event is sent by the proximity monitor to the application when the initialization sequence is completed and the device is ready to start advertising.
2. EVT_PM_DISCOVERY_CMPLT: this event is sent by the proximity monitor after a connection is established. The evtData contains the error code.
   – 0x00: all the mandatory services, characteristics and descriptors as specified in the profile specification were discovered successfully.
   – 0x01: link loss service not found.
3. EVT_PM_LINK_LOSS_ALERT: this event is sent to the application when a link loss is detected. The evtData contains the alert level. The application must start an alert for the level specified. The type of alert is decided by the application: the alert can continue for an application-specific duration or until another connection is established. The application must re-enable advertising to establish a new connection.
4. EVT_PM_PATH_LOSS_ALERT: this event is sent to the application by the proximity monitor when a path loss is detected. The evtData contains the alert level. When a path loss is detected, the application can start an alert of any type for the alert level specified.

5.5 HID events

1. EVT_HID_INITIALIZED: this event is sent to the application when the HID has completed its initialization sequence and is ready to enable advertising, or the initialization sequence failed. The evtData parameter contains the error code; 0X00 means the initialization was successful.
2. EVT_HID_UPDATE_CMPLT: this event is sent to the application when an update previously started by the application completes. The status indicates whether the update was successful or it failed. The evtData also contains the service handle and the characteristic handle.
3. EVT_BATT_LEVEL_READ_REQ: this event is sent to the application when the client requests a battery level reading. On receiving this event, the application can update the battery level characteristic and then call the function Allow_BatteryLevel_Char_Read. If the process takes more than 30 minutes, the GATT channel is closed.

5.6 Find me target events

1. EVT_FMT_INITIALIZED: this event is sent to the application when the find me target has completed its initialization sequence and is ready to enable advertising, or the initialization sequence failed. The evtData parameter contains the error code; 0X00 means the initialization was successful.
2. EVT_FMT_ALERT: this event is sent to the application when the client writes to the alert level characteristic with a valid alert level. The application must start alerting if the alert level is 0x01 or 0x02, and stop when the alert level is 0x00.

5.7 Find me locator events

1. EVT_FML_INITIALIZED: this event is sent by the find me locator to the application when the initialization sequence is completed and the device is ready to start advertising.
2. EVT_FML_DISCOVERY_CMPLT: this event is sent by the find me locator after a connection is established. The evtData contains the error code:
   - 0x00: all the mandatory services, characteristics and descriptors as specified in the profile specification were discovered successfully.
   - 0x01: alert characteristic not found.
   - 0x02: immediate alert service not found.

5.8 Phone alert client events

1. EVT_PAC_INITIALIZED: this event is sent by the phone alert client to the application when the initialization sequence is completed and the device is ready to start advertising.
2. EVT_PAC_DISCOVERY_CMPLT: this event is sent by the phone alert client after a connection is established. The evtData contains the error code:

   0x00: all the mandatory services, characteristics and descriptors as specified in the profile specification were discovered successfully.

   0x01: PHONE_ALERT_SERVICE_NOT_FOUND.

   0x02: PHONE_ALERT_STATUS_CHARAC_NOT_FOUND.

   0x03: RINGER_CNTRL_POINT_CHARAC_NOT_FOUND.

   0x04: RINGER_SETTING_CHARAC_NOT_FOUND.

   0x05: PHONE_ALERT_STATUS_DESC_NOT_FOUND.

   0x06: RINGER_CNTRL_POINT_DESC_NOT_FOUND.

   0x07: RINGER_SETTING_DESC_NOT_FOUND.
3. EVT_PAC_ALERT_STATUS: the application can start a procedure to read the alert status characteristic on the peer server using the function PAC_Read_AlertStatus(). The response to this function call is returned in this event. The evtData contains the response received from the server.
4. EVT_PAC_RINGER_SETTING: the application can read the ringer setting on the server by using the function PAC_Read_RingerSetting(). The response to this function call is returned in this event. The evtData contains the response received from the server.

5.9 Time server events

1. EVT_TS_INITIALIZED: this event is sent by the time server to the application when the initialization sequence has completed and the device is ready to start advertising.
2. EVT_TS_CHAR_UPDATE_CMPLT: this event is sent to the application when an update previously started by the application completes. The status indicates whether the update was successful or it failed. The evtData also contains the service handle and the characteristic handle.
3. EVT_TS_START_REFTIME_UPDATE: this event is sent to the application when the GET_REFERENCE_UPDATE(0x01) command is written to the updateState characteristic by the time client.
4. EVT_TS_STOP_REFTIME_UPDATE: this event is sent to the application when the CANCEL_REFERENCE_UPDATE(0x02) command is written to the updateState characteristic by the time client.

5.10 Time client events

1. EVT_TC_INITIALIZED: this event is sent by the time client to the application when the initialization sequence is completed and the device is ready to start advertising.
2. EVT_TC_DISCOVERY_CMPLT: this event is sent by the time client after a connection is established and all the mandatory services, characteristics and descriptors as specified in the profile specification were discovered successfully.
3. EVT_TC_CUR_TIME_VAL_RECEIVED: this event is sent to the application when a notification for the current time characteristic is received by the time client. The event data contains all the fields of the current time characteristic.
4. EVT_TC_READ_CUR_TIME_CHAR: The application can read the current time characteristic on the server by using the function TimeClient_Get_Current_Time(). The response to this function call is returned in this event. The evtData contains the response received from the server.
5. EVT_TC_READ_REF_TIME_INFO_CHAR: The application can read the reference time characteristic on the server by using the function TimeClient_Get_Time_Accuracy_Info_Of_Server(). The response to this function call is returned in this event. The evtData contains the response received from the server.
6. EVT_TC_READ_LOCAL_TIME_INFO_CHAR: The application can read the local time information characteristic on the server by using the function TimeClient_Get_Local_Time_Information(). The response to this function call is returned in this event. The evtData contains the response received from the server.
7. EVT_TC_READ_TIME_WITH_DST_CHAR: The application can read the time with dst change characteristic on the server by using the function TimeClient_Get_Next_DST_Change_Time(). The response to this function call is returned in this event. The evtData contains the response received from the server.
8. EVT_TC_READ_TIME_UPDATE_STATE_CHAR: The application can read the time update state characteristic on the server by using the function TimeClient_Get_Server_Time_Update_State(). The response to this function call is returned in this event. The evtData contains the response received from the server.

5.11 Blood pressure sensor events

1. EVT_BPS_INITIALIZED: this event is sent by the blood pressure sensor to the application when the initialization sequence is completed and the device is ready to start advertising.
2. EVT_BPS_BPM_CHAR_UPDATE_CMPLT: this event is sent to the application when an update to the blood pressure measurement characteristic previously started by the application completes. The status indicates whether the update was successful or not.
3. EVT_BPS_ICP_CHAR_UPDATE_CMPLT: this event is sent to the application when an update to the intermediate cuff pressure characteristic previously started by the application completes. The status indicates whether the update was successful or not.
4. EVT_BPS_IDLE_CONNECTION_TIMEOUT: this event is sent to the application when there is no measurements to be sent to the collector for more than five seconds.

5.12 Health thermometer events

1. EVT_HT_INITIALIZED: this event is sent by the thermometer to the application when the initialization sequence is completed and the device is ready to start advertising.
2. EVT_HT_TEMPERATURE_CHAR_UPDATE_CMPLT: this event is sent to the application when an update to the temperature measurement characteristic previously started by the application completes. The status indicates whether the update was successful or it failed.
3. EVT_HT_INTERMEDIATE_TEMP_CHAR_UPDATE_CMPLT: this event is sent to the application when an update to the intermediate temperature measurement characteristic previously started by the application completes. The status indicates whether the update was successful or it failed.
4. EVT_HT_MEASUREMENT_INTERVAL_RECEIVED: this event is sent to the application when the collector writes to the measurement interval characteristic.
5. EVT_HT_MEASUREMENT_INTERVAL_UPDATE_CMPLT: this event is sent to the application when an update to the measurement interval characteristic previously started by the application completes. The status indicates whether the update was successful or it failed.
6. EVT_BPS_IDLE_CONNECTION_TIMEOUT: this event is sent to the application when there are no measurements to be sent to the collector for more than five seconds.

5.13 Alert notification server events

1. EVT_ANS_INITIALIZED: this event is sent by the alert notification server to the application when the initialization sequence has completed and the device is ready to start advertising.

5.14 Alert notification client events

1. EVT_ANC_INITIALIZED: this event is sent by the alert notification client to the application when the initialization sequence has completed and the device is ready to start advertising.
2. EVT_ANC_DISCOVERY_CMPLT: this event is sent by the alert notification client after a connection is established and all the mandatory services, characteristics and descriptors as specified in the profile specification were discovered successfully.
3. EVT_ANC_NEW_ALERT_RECEIVED: this event is sent to the application when a notification for the new alert is received by the alert notification client.
4. EVT_ANC_UNREAD_ALERT_STATUS_RECEIVED: this event is sent to the application when a notification for an unread alert is received by the alert notification client.

5.15 Glucose sensor events

1. EVT_GL_INITIALIZED: this event is sent by the glucose sensor to the application when the initialization sequence has completed and the device is ready to start advertising.
2. EVT_GL_IDLE_CONNECTION_TIMEOUT: this event is sent to the application when the connection is idle for more than five seconds.

6 Profile – application interface functions

6.1 Heart rate profile

6.1.1 HRProfile_Init()

Description

The application calls this function to initialize the heart rate profile. The initialization procedure returns BLE_STATUS_SUCCESS if started successfully or BLE_STATUS_FAILED if not. On successful initialization of the profile, the application is notified through the event EVT_HRS_INITIALIZED through the registered callback.

Parameters

- Feature support flag: the characteristic/feature mask supported by the heart rate profile during initialization. The various characteristics mask supported by the profile are:

1.	BODY_SENSOR_LOCATION_SUPPORT_MASK	(0x01)
2.	ENERGY_EXTENDED_INFO_SUPPORT_MASK	(0x02)

- BLE_CALLBACK_FUNCTION_TYPE: the callback function to be registered by the heart rate profile for notification/communication to the main BLE profile.
- Sensor location value: the Value for the body sensor location.

6.1.2 HR_Sensor_Make_Discoverable()

Description

This command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

- useBoundedDeviceList: set this flag to TRUE '1' if Profile needs to advertise to devices already bonded; otherwise, set it to FALSE '0'.

6.1.3 HRProfile_Send_HRM_Value()

Description

This function or command is called by the application to send the heart rate measurement value to the collector. The procedure to send a heart rate measurement value returns BLE_STATUS_SUCCESS if started successfully or BLE_STATUS_FAILED if not. When the measurement value is sent successfully, the application is notified through the event EVT_HRS_CHAR_UPDATE_CMPLT.

Parameters

- heartRateVal: The heart rate measurement structure with the following members:

1. valueformat – indicates the format of the heart measurement value.
   '0' if UINT8
   '1' if UINT16
2. sensorContact – this field indicates whether the sensor is in contact with the body
   '0' is no or poor contact.
   '1' contact o.k.
3. energyExpendedStatus – indicates whether the EE field is present in the current characteristic value.
   '0' not present.
   '1' present.
4. rrIntervalStatus – indicates whether RR interval values are present in the current characteristic value
   '0' not present.
   '1' present.
5. heartRateValue – the heart rate measurement value.
6. energyExpended – the energy expended value.
7. numOfRRIntervalvalues – the maximum length of RR interval values allowed is nine. If the maximum is exceeded, then only the last nine will be considered, assuming they correspond to the most recent collected data.
8. rrIntervalValues[9] – the buffer to hold the nine(9) most recent RR interval values provided by the application.

6.1.4 HRProfile_Set_Sensor_Contact_Support_Bit()

Description

The application should call this function before sending any data to the device in order to enable the sensor contact bit (BODY_SENSOR_LOCATION_SUPPORT_MASK) to include sensor contact information value in the heart rate measurement. It returns BLE_STATUS_SUCCESS when successfully set and BLE_STATUS_FAILED when not set.

6.1.5 HRProfile_Set_Body_Sensor_Location()

Description

Updates the body sensor location characteristic with the value provided. This should be called by the application when not in a connection; it returns BLE_STATUS_SUCCESS if successful.

Parameters

- bdsValue: The position of the body sensor location. The valid sensor location values are:

BODY_SENSOR_LOCATION_OTHER	(0x00)
BODY_SENSOR_LOCATION_CHEST	(0x01)
BODY_SENSOR_LOCATION_WRIST	(0x02)
BODY_SENSOR_LOCATION_FINGER	(0x03)
BODY_SENSOR_LOCATION_HAND	(0x04)
BODY_SENSOR_LOCATION_EAR_LOBE	(0x05)
BODY_SENSOR_LOCATION_FOOT	(0x06)

6.1.6 HRProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.2 Find me profile

Find me – Target

6.2.1 FindMeTarget_Init()

Description

The application calls this function to initialize the find me target profile. The initialization procedure returns BLE_STATUS_SUCCESS if started successfully or BLE_STATUS_FAILED if not. The application is notified of successful initialization of the profile by the event EVT_FMT_INITIALIZED through the registered callback.

Parameters

- BLE_CALLBACK_FUNCTION_TYPE: the callback through which the application is notified of events by the find me.

6.2.2 FMT_Advertize()

Description

The command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

6.2.3 FMT_Add_Device_To_WhiteList()

Description

This function is called by the application to add devices into the whitelist.

Parameters

- addrType: address type of the bdAddr to be added to the whitelist.
  - 0x00: PUBLIC ADDRESS
  - 0x01: RANDOM ADDRESS
- bdAddr: address of the peer device that must be added to the whitelist.

6.2.4 FMLProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

Find Me – Locator

6.2.5 FindMeLocator_Init()

Description

The application should call this function to initialize the Find Me Locator Profile. The initialization procedure returns BLE_STATUS_SUCCESS if started successfully, or BLE_STATUS_FAILED if not. The application is notified of successful initialization of the profile by the event EVT_FML_INITIALIZED through the registered callback.

Parameters

- bleSecReq: pointer to the structure denoting the security requirements of the profile.
- BLE_CALLBACK_FUNCTION_TYPE: the Callback function to be called to notify the application of the events.

6.2.6 FML_Advertize()

Description

The command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

6.2.7 FML_Add_Device_To_WhiteList()

Description

This function or command is called by the application to add devices to the whitelist.

Parameters

- addrType: address type of the bdAddr to be added to the whitelist.
  - 0x00: PUBLIC ADDRESS
  - 0x01: RANDOM ADDRESS
- bdAddr: address of the peer device that must be added to the whitelist.

6.2.8 FML_Alert_Target()

Description

The function is called by the application to start a write to the alert level on the find me target. It returns BLE_STATUS_SUCCESS if the procedure is started successfully, otherwise, it returns an error.

Parameters

- alertLevel: the alert level for the target must be configured according to the following alert levels:
  - NO_ALERT (0x00)
  - MILD_ALERT (0x01)
  - HIGH_ALERT (0x02)

6.2.9 FMTProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.3 Health thermometer profile

6.3.1 HT_Init()

Description

Initializes the health thermometer profile. Returns BLE_STATUS_SUCCESS if the procedure is started successfully, and then notifies the application of successful initialization of the profile through the event EVT_HT_INITIALIZED.

Parameters

- thermometerFeatures: bitmask for the characteristics to be added to the health thermometer service. The various bit masks for the characteristics are:
  - INTERMEDIATE_TEMPERATURE_CHAR (0x01)
  - MEASUREMENT_INTERVAL_CHAR (0x02)
  - TEMPERATURE_TYPE (0x04)
- minValidInterval: the minimum valid interval value for the measurement interval characteristic. This is only valid if the MEASUREMENT_INTERVAL_CHAR flag is set in the thermometer features.
- maxValidInterval: the maximum valid interval value for the measurement interval characteristic.
- BLE_CALLBACK_FUNCTION_TYPE: callback function to be called by the profile to notify the application of the events.

6.3.2 HT_Advertize()

Description

This command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

- useWhitelist: If the useWhiteList is set to TRUE, the device is configured to use the whitelist which is configured with bonded devices at the time of initialization; otherwise, the device enters limited discoverable mode to connect to any of the available devices.

6.3.3 HT_Send_Temperature_Measurement()

Description

The application calls this function to update the temperature measurement characteristic. The application is notified through the event EVT_HT_TEMPERATURE_CHAR_UPDATE_CMPLT when the update is complete.

Parameters

- tempMeasurementVal: The temperature measurement value structure contains the following members:

1. flags – bit mask of the fields supported in the characteristic.
   a) FLAG_TEMPERATURE_UNITS_FARENHEIT (0x01)
   b) FLAG_TIMESTAMP_PRESENT (0x02)
   c) FLAG_TEMPERATURE_TYPE (0x04)
2. Temperature – temperature measurement value(4 byte)
3. timeStamp – timestamp of the measurement
4. temperatureType – temperature type
   a) TEMP_MEASURED_AT_ARMPIT (0x01)
   b) TEMP_MEASURED_FOR_BODY (0x02)
   c) TEMP_MEASURED_AT_EAR (0x03)
   d) TEMP_MEASURED_AT_FINGER (0x04)
   e) TEMP_MEASURED_AT_GASTRO_INTESTINAL_TRACT (0x05)
   f) TEMP_MEASURED_AT_MOUTH (0x06)
   g) TEMP_MEASURED_AT_RECTUM (0x07)
   h) TEMP_MEASURED_AT_TOE (0x08)
   i) TEMP_MEASURED_AT_TYMPANUM (0x09)

6.3.4 HT_Send_Intermediate_Temperature()

Description

The application calls this function to update the intermediate temperature measurement characteristic. The application is notified through the event EVT_HT_INTERMEDIATE_TEMP_CHAR_UPDATE_CMPLT when the update completes.

Parameters

1. flags – bitmask of the fields supported in the characteristic. Refer to *Section 6.3.3: HT_Send_Temperature_Measurement()* for valid values.
2. Temperature – temperature measurement value.
3. timestamp – timestamp of the measurement.
4. temperatureType – temperature type. Refer to *Section 6.3.3: HT_Send_Temperature_Measurement()* for valid values.

6.3.5 HT_Update_Measurement_Interval()

Description

The application calls this to update the measurement interval value characteristic. This is the interval between the temperature updates sent to the collector. On completion of the update, the application is notified through the event EVT_HT_MEASUREMENT_INTERVAL_UPDATE_CMPLT.

Parameters

Interval – the gap interval after which the update of the measurement value is to be performed.

6.3.6 HT_Update_Temperature_Type()

Description

The application calls this to update the temperature type characteristic. The temperature type indicates the part of the body where the temperature is being measured. During an active connection, this setting must remain static and updates are not allowed. On successful update, the event EVT_HT_TEMP_TYPE_CHAR_UPDATE_CMPLT is sent to the application.

Parameters

1. Type – the type denotes the part of the body where the temperature is measured; below is the list of the type fields:
   - TEMP_MEASURED_AT_ARMPIT (0x01)
   - TEMP_MEASURED_FOR_BODY (0x02)
   - TEMP_MEASURED_AT_EAR (0x03)
   - TEMP_MEASURED_AT_FINGER (0x04)
   - TEMP_MEASURED_AT_GASTRO_INTESTINAL_TRACT (0x05)
   - TEMP_MEASURED_AT_MOUTH (0x06)
   - TEMP_MEASURED_AT_RECTUM (0x07)
   - TEMP_MEASURED_AT_TOE (0x08)
   - TEMP_MEASURED_AT_TYMPANUM (0x09)

6.3.7 HT_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.4 Alert Notification profile

Alert Notification – Client

6.4.1 ANC_Client_Init()

Description

Initializes the Alert Notification Profile. It returns BLE_STATUS_SUCCESS if the procedure is started successfully. Notification of successful initialization of the profile is sent to the application through the event EVT_ANC_INITIALIZED.

Parameters

- BLE_CALLBACK_FUNCTION_TYPE: callback function called by the profile to notify the application of the events.

6.4.2 ANC_Advertize()

Description

The function puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

- useWhitelist: if the useWhiteList is set to TRUE, the device is configured to use the whitelist which is configured with bonded devices at the time of initialization; otherwise, the device enters limited discoverable mode to connect to any of the available devices.

6.4.3 ANC_Write_Control_Point()

Description

The application calls this to write or update the control point characteristic. The application is notified through the event on successful update.

Parameters

1. Command – ID of the command to be sent. Below is the list of the different command IDs:
   – ENABLE_NEW_ALERT_NOTIFICATION (0x00)
   – ENABLE_UNREAD_ALERT_STATUS_NOTIFICATION (0x01)
   – DISABLE_NEW_ALERT_NOTIFICATION (0x02)
   – DISABLE_UNREAD_ALERT_STATUS_NOTIFICATION (0x03)
   – NOTIFY_NEW_ALERT_IMMEDIATELY (0x04)
   – NOTIFY_UNREAD_ALERT_STATUS_IMMEDIATELY (0x05)
2. Category ID – category which has to be affected by the command. Below is the list of the category IDs.
   – CATEGORY_ID_SIMPLE_ALERT (0x00)
   – CATEGORY_ID_EMAIL (0x01)
   – CATEGORY_ID_NEWS (0x02)
   – CATEGORY_ID_CALL (0x03)
   – CATEGORY_ID_MISSED_CALL (0x04)
   – CATEGORY_ID_SMS_MMS (0x05)
   – CATEGORY_ID_VOICE_MAIL (0x06)
   – CATEGORY_ID_SCHEDULE (0x07)
   – CATEGORY_ID_HIGH_PRIORITIZED_ALERT (0x08)
   – CATEGORY_ID_INSTANT_MESSAGE (0x09)

6.4.4 ANC_Enable_Disable_New_Alert_Notification()

Description

Enables the notifications for the new alert characteristic. After enabling this, the control point characteristic must also be written with the command and category to receive alerts from the peer.

Parameters

1. enable – if set to TRUE, it enables the notifications for the new alert characteristic.

6.4.5 ANC_Enable_Disable_Unread_Alert_Status_Notification()

Description

Enables the notifications for the unread alert status characteristic. After enabling this, the control point characteristic must also be written with the command and category to receive alerts from the peer.

Parameters

1. enable: if set to TRUE, it enables the notifications for the unread alert status characteristic.

6.4.6 ANCProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

Alert notification – Server

6.4.7 ANS_Init()

Description

Initializes the Alert Notification Profile in the server role. It returns BLE_STATUS_SUCCESS if the procedure is started successfully. Notification of successful initialization of the profile is sent to the application through the event EVT_ANS_INITIALIZED.

Parameters

- BLE_CALLBACK_FUNCTION_TYPE: callback function called by the profile to notify the application of the events.
- AlertCategory: bitmask of the categories supported for the new alert characteristic.
  - CATEGORY_ID_SIMPLE_ALERT (0x00)

  - CATEGORY_ID_EMAIL (0x01)
  - CATEGORY_ID_NEWS (0x02)
  - CATEGORY_ID_CALL (0x03)
  - CATEGORY_ID_MISSED_CALL (0x04)
  - CATEGORY_ID_SMS_MMS (0x05)
  - CATEGORY_ID_VOICE_MAIL (0x06)
  - CATEGORY_ID_SCHEDULE (0x07)
  - CATEGORY_ID_HIGH_PRIORITIZED_ALERT (0x08)
  - CATEGORY_ID_INSTANT_MESSAGE (0x09)
- unreadAlertCategory: bitmask of the categories supported for the unread alert status characteristic.
  - CATEGORY_ID_SIMPLE_ALERT (0x00)
  - CATEGORY_ID_EMAIL (0x01)
  - CATEGORY_ID_NEWS (0x02)
  - CATEGORY_ID_CALL (0x03)
  - CATEGORY_ID_MISSED_CALL (0x04)
  - CATEGORY_ID_SMS_MMS (0x05)
  - CATEGORY_ID_VOICE_MAIL (0x06)
  - CATEGORY_ID_SCHEDULE (0x07)
  - CATEGORY_ID_HIGH_PRIORITIZED_ALERT (0x08)
  - CATEGORY_ID_INSTANT_MESSAGE (0x09)

6.4.8 ANS_Advertize()

Description

The function puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

- useWhitelist: if the useWhiteList is set to TRUE, the device is configured to use the whitelist which is configured with bonded devices at the time of initialization; otherwise, the device enters limited discoverable mode to connect to any of the available devices.

6.4.9 ANS_Update_New_Alert_Category()

Description

The application calls this to update the alert category characteristic with the new bitmask. It returns BLE_STATUS_SUCCESS if the update is successfully started and BLE_STATUS_INVALID_PARAMS if a bitmask for an invalid category is requested.

Parameters

1. Length: length of the category field; it must be 0 or 1.
2. Category: bitmask of the categories supported. The bitmasks are split across two octets with the meanings described in the bluetooth assigned numbers documentation.

6.4.10 ANS_Update_Unread_Alert_Category()

Description

The application calls this to update the unread alert category with the new bitmask. It returns BLE_STATUS_SUCCESS if the update is successfully started and BLE_STATUS_INVALID_PARAMS if a bitmask for an invalid category is set.

Parameters

1. Length: length of the category field; it must be 0 or 1.
2. Category: bitmask of the categories supported. The bitmasks are split across two octets with the meanings described in the bluetooth assigned numbers documentation.

6.4.11 ANS_Update_New_Alert()

Description

The application calls this to update the number of new alerts for the category specified in the new alert characteristic. If the category ID specified is not valid or the text information is longer than 18 octets, BLE_STATUS_INVALID_PARAMS is returned. On successful write, BLE_STATUS_SUCCESS is returned.

Parameters

1. alertCount: alert count for the category specified. The application must maintain the count of new alerts.
2. categoryID: category which is affected by the command.
3. TextInfo: textual information corresponding to the alert.

6.4.12 ANS_Update_Unread_Alert_Status()

Description

The application calls this to update the number of unread alerts for the category specified in the new alert characteristic. If the category ID specified is not valid, BLE_STATUS_INVALID_PARAMS is returned. On successful write to the alert status, BLE_STATUS_SUCCESS is returned.

Parameters

1. categoryID: category which is affected by the command.
2. AlertCount: alert count for the category specified. The application must maintain the count of unread alerts.

6.4.13 ANSProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.5 Blood pressure profile

6.5.1 BPS_Init()

Description

Initializes and registers the blood pressure Sensor profile with the main Profile. It returns BLE_STATUS_SUCCESS if the procedure is started successfully, or BLE_STATUS_FAILED if not. The application is notified through the event EVT_BPS_INITIALIZED on completion of the initialization procedure.

Parameters

1. intermediateCuffPressureChar: indicates whether the blood pressure service should support the intermediate cuff pressure characteristic.
2. Feature: a bitmask representing the features supported by the device. Below is the list of the features supported by the device:
   - BODY_MOVEMENT_DETECTION_FLAG (0x01)
   - CUFF_FIT_DETECTION_FLAG (0x02)
   - IRREGULAR_PULSE_DETECTION_FLAG (0x04)
   - PULSE_RATE_RANGE_EXCEEDS_UPPER_LIMIT (0x08)
   - PULSE_RATE_RANGE_BELOW_LOWER_LIMIT (0x10)
   - MEASUREMENT_POSITION_DETECTION_FLAG (0x20)
3. BLE_CALLBACK_FUNCTION_TYPE: callback function type to be called by the profile to notify the application of the profile specific events.

6.5.2 BPS_Advertize()

Description

The function puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

useWhitelist: if useWhiteList is set to TRUE, the device is configured to use the whitelist which is configured with bonded devices at the time of initialization; otherwise, the device enters limited discoverable mode to connect to any of the available devices.

6.5.3 BPS_Send_Intermediate_Cuff_Pressure()

Description

This function is called to send the intermediate cuff pressure values during the measurement process until a stable value is obtained. The function can only be used only if the intermediate Cuff Pressure Char is set to ‘True’ during initialization. The application is notified of successful update through the event EVT_BPS_ICP_CHAR_UPDATE_CMPLT.

Parameters

- icpVal: the intermediate Cuff pressure value structure containing the following structure members:
  – Flags: the flags is a bitmask which tells the peer of the data to follow.

    Bit0 – a value of 1 indicates that the unit is kPa; a value of 0 indicates that the unit is mm Hg.

    Bit3 – a value of 1 implies there is a user ID field in the data

    Bit4 – a value of 1 implies there is a measurement status in the data.
  – Icp: intermediate cuff pressure value.
  – UserID: if Bit3 is set, then this field should contain the value of the USER ID.
  – MeasurementStatus: a structure containing the values of the various bit mask features supported by the device.

6.5.4 BPS_Send_Blood_Pressure_Measurement()

Description

This function is called to send the Blood Pressure measurement values. It updates the blood pressure measurement characteristic with the value if the device is connected. The application is notified through the event EVT_BPS_BPM_CHAR_UPDATE_CMPLT.

Parameters

- bpmval: the blood pressure value structure. The members are similar to those in icpValue.

6.5.5 BPS_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.6 Human interface device profile

6.6.1 HidDevice_Init()

Description

Initializes the Hid profile. It returns BLE_STATUS_SUCCESS if the procedure is started successfully, or BLE_STATUS_FAILED if not. The application is notified through the event EVT_HID_INITIALIZED on successful initialization.

Parameters

1. numOfHIDServices – number of HID services to be exposed in the profile.
2. HidServiceData – pointer to the structure tApplDataForHidServ containing the configuration parameters provided by the application at the time of Initialization.
3. NumOfBatteryServices – number of battery services to be exposed in the profile
4. scanParamServiceSupport – adds the scan parameters service during hid initialization if set to '1'.
5. scanRefreshCharSupport – adds the scan parameters refresh characteristics if set to '1'.
6. BLE_CALLBACK_FUNCTION_TYPE – callback function type to be called by the profile to notify the application of the events.

6.6.2 HidDevice_Make_Discoverable()

Description

The function puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

1. useBoundedDeviceList – if this is set to '1', advertising is performed on the devices already bonded using whitelist.

6.6.3 HidDevice_Update_Input_Report()

Description

This is called by the application to update the input report characteristic with the value specified in ipReportValue. It returns BLE_STATUS_SUCCESS if the update was successfully started, otherwise it returns error codes. On successful update of the characteristic, the application is notified through the event EVT_HID_UPDATE_CMPLT.

Parameters

1. hidServiceIndex – the index of the HID service whose report characteristic has to be updated.
2. IpReportIndex – the index of the input report to be updated
3. ipReportValLength – length of the input report
4. ipReportValue – value of the input report.

6.6.4 HidDevice_Update_Feature_Report()

Description

This is called by the application to update the feature report characteristic with the value specified in ftrReportValue. It returns BLE_STATUS_SUCCESS if the update was successfully started, otherwise it returns error codes. On successful update of the characteristic, the application is notified through the event EVT_HID_UPDATE_CMPLT.

Parameters

1. hidServiceIndex – the index of the HID service whose report characteristic has to be updated.
2. ftrReportIndex – the index of the input report to be updated
3. ftrReportValLength – length of the input report
4. ftrReportValue – value of the input report.

6.6.5 HidDevice_Update_Boot_Keyboard_Input_Report()

Description

This is called by the application to update the boot keyboard input report characteristic with the value specified in bootKbdIpReportValue. On successful update of the characteristic, the application is notified through the event EVT_HID_UPDATE_CMPLT.

Parameters

1. hidServiceIndex – the index of the HID service whose report characteristic must be updated.
2. BootKbdIpReportValLength – length of the boot keyboard input report
3. bootKbdIpReportValue – value of the boot keyboard input report

6.6.6 HidDevice_Update_Boot_Mouse_Input_Report()

Description

This is called by the application to update the boot mouse input report characteristic with the value specified in bootMouseIpReportValue. On successful update of the characteristic, the application is notified through the event EVT_HID_UPDATE_CMPLT.

Parameters

1. hidServiceIndex – the index of the HID service whose report characteristic must be updated.
2. BootMouseIpReportValLength – length of the boot mouse input report
3. bootMouseIpReportValue – value of the boot mouse input report

6.6.7 HidDevice_Update_Battery_Level()

Description

This is called by the application to start the update for the battery level characteristic. On successful update, the application is notified through the event EVT_BATTERY_LEVEL_UPDATE_CMPLT.

Parameters

1. batteryServiceIndex – the battery service whose characteristic must be updated.
2. BatteryLevel – value of the battery level characteristic.

6.6.8 HidDevice_Update_Scan_Refresh_Char()

Description

This is called by the application to start the update for the scan refresh characteristic. On successful update, the application is notified through the event EVT_SCAN_REFRESH_UPDATE_CMPLT.

Parameters

1. scanRefresh – the value of the scan refresh characteristic

6.6.9 Allow_BatteryLevel_Char_Read()

Description

This is called by the application when it receives an event EVT_BATT_LEVEL_READ_REQ from the profile. When EVT_BATT_LEVEL_READ_REQ is received, the application should first update the battery level characteristic if required and then call this function. The stack blocks the read response until this function is called by the application. For more details see the event description for EVT_BATT_LEVEL_READ_REQ.

Parameters

1. batteryServiceIndex – the battery service whose characteristic must be updated.

6.6.10 HIDProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.7 Phone alert profile

6.7.1 PAC_Init()

Description

Initializes the Phone Alert Status profile for client role and registers the Phone Alert Profile with the main profile. It returns BLE_STATUS_SUCCESS if the procedure is successfully started. The application is notified through the event EVT_PAC_INITIALIZED on successful initialization.

Parameters

1. BLE_CALLBACK_FUNCTION_TYPE – callback function to be called by the profile to notify the application of the events.

6.7.2 PAC_Add_Device_To_WhiteList()

Description

The application calls this function to add the devices to whitelist.

Parameters

1. bdAddr – the address of the peer device that must be added to the whitelist.

6.7.3 PAC_Advertize()

Description

The application calls this function to put the device into discoverable mode.

6.7.4 PAC_Configure_Ringer()

Description

The application calls this to write the ringer mode to the phone alert server. It returns BLE_STATUS_SUCCESS if the parameters are valid and the procedure has been started successfully, otherwise it returns error codes.

Parameters

1. ringerMode – the ringer mode to be set. The valid ringer modes are:
   - SILENT_MODE (0x01)
   - MUTE_ONCE (0x02)
   - CANCEL_SILENT_MODE (0x03)

6.7.5 PAC_Read_AlertStatus()

Description

When this function is called by the application, the profile starts a GATT procedure to read the characteristic value. The value read is returned via the event EVT_PAC_ALERT_STATUS to the application through the callback.

6.7.6 PAC_Read_RingerSetting()

Description

When this function is called by the application, the profile starts a GATT procedure to read the characteristic value. The value read is returned via the event EVT_PAC_RINGER_SETTING to the application.

6.7.7 PACProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.8 Time profile

Time profile – Client

6.8.1 TimeClient_Init()

Description

Initializes the time profile in client role. The application is notified through the event EVT_TC_INITIALIZED on successful initialization.

Parameters

1. BLE_CALLBACK_FUNCTION_TYPE – callback function to be called by the profile to notify the application of the events.

6.8.2 TimeClient_Make_Discoverable()

Description

The command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

1. useBoundedDeviceList – set this to '1' if advertising is to be performed on the devices already bonded.

6.8.3 TimeClient_Get_Current_Time()

Description

The application calls this function to read the current time characteristic. Once the read is complete, the read value is sent to the application through an event EVT_TC_READ_CUR_TIME_CHAR. The event data contains the following fields:

1. byte 0 and 1 – year
2. byte 2 – month
3. byte 3 – date
4. byte 4 – hours
5. byte 5 – minutes
6. byte 6 – seconds
7. byte 7 – day_of_week
8. byte 8 – fractions256
9. byte 9 – adjust_reason

6.8.4 TimeClient_Get_Local_Time_Information()

Description

The application calls this function to read the local time information characteristic. Once the read is complete, the read value is sent to the application through an event called EVT_TC_READ_LOCAL_TIME_INFO_CHAR.

6.8.5 TimeClient_Get_Time_Accuracy_Info_Of_Server()

Description

The application calls this function to read the reference time information characteristic. Once the read is complete, the read value is sent to the application through an event called EVT_TC_READ_REF_TIME_INFO_CHAR.

6.8.6 TimeClient_Get_Next_DST_Change_Time()

Description

The application calls this function to read the time with DST information characteristic on the server. Once the read is complete, the read value is sent to the application through an event called EVT_TC_READ_TIME_WITH_DST_CHAR.

6.8.7 TimeClient_Get_Server_Time_Update_State()

Description

The application should call this function to read the time update state characteristic on the server. Once the read is complete, the read value is sent to the application through an event called EVT_TC_READ_TIME_UPDATE_STATE_CHAR.

6.8.8 TimeClient_Update_Reference_Time_On_Server()

Description

The application calls this function to write the time update control point characteristic on the server.

Parameters

1. ctlValue: writing a value of 1 starts the update procedure; a value of 0 cancels the update procedure.

6.8.9 TimeClient_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

Time profile - Server

6.8.10 TimeServer_Init()

Description

Initializes the time profile in server role. On successful initialization, the application is notified through the event EVT_TS_INITIALIZED.

Parameters

1. BLE_CALLBACK_FUNCTION_TYPE – callback function to be called by the profile to notify the application of the events.
2. ServicesToBeSupported – the bit mask of the optional services to be supported. Below are the bitmask values:
   a) NEXT_DST_CHANGE_SERVICE_BITMASK (0x01)
   b) REFERENCE_TIME_UPDATE_SERVICE_BITMASK (0x02)

6.8.11 TimeServer_Make_Discoverable()

Description

The command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

1. useBoundedDeviceList- this is set to '1' if advertising is to be done to the devices already bonded.

6.8.12 TimeServer_Update_Current_Time_Value()

Description

This is called by the application to update the current time characteristic with the timeValue On completion of the update, the application is notified through the event EVT_TS_CHAR_UPDATE_CMPLT.

Parameters

- timeValue – current time structure containing the following structure members.

1. year
2. month
3. date
4. hours
5. minutes
6. seconds
7. day_of_week
8. fractions256
9. adjustReason: the adjust reason parameter can take any of the below values:
   a) ADJUST_REASON_NO_REASON (0x00)
   b) ADJUST_REASON_MANUAL_TIME_UPDATE (0x01)
   c) ADJUST_REASON_EXTERNAL_REFERENCE_TIME_UPDATE (0x02)
   d) ADJUST_REASON_CHANGE_OF_TIME_ZONE (0x04)
   e) ADJUST_REASON_CHANGE_OF_DST (0x08)

6.8.13 TimeServer_Update_Local_Time_Information()

Description

This is called by the application to update the localTimeInfo characteristic with the value specified and notifies the application with the EVT_TS_CHAR_UPDATE_CMPLT event on successful update.

Parameters

- localTimeInfo – local time structure containing the following structure members:
  – timeZone
  – dstOffset

6.8.14 TimeServer_Update_Reference_Time_Information()

Description

This is called by the application to update the reference time information characteristic with the value specified and the application is notified with the EVT_TS_START_REFTIME_UPDATE event on successful update.

Parameters

- refTimeInfo – the new reference time information. This structure containing the following members:
  - source
  - accuracy
  - daysSinceUpdate
  - hoursSinceUpdate

6.8.15 TimeServer_Update_Next_DST_Change()

Description

This is called by the application to update the next DST change characteristic with the value specified.

Parameters

- timeDST – the new DST information. This structure contains the following members:
  - year
  - month
  - date
  - hours
  - minutes
  - seconds
  - dstOffset

6.8.16 TimeServer_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.9 Device information service profile

Different profiles have different requirements of the characteristics for the device information service. The profiles specify the characteristics required during the initialization, but the update to these characteristics must be performed by the application.

6.9.1 BLE_Profile_Update_DIS_SystemID()

Description

The application calls this to update the System ID characteristic of the device information service.

Parameters

1. length – the Length of the characteristic to be updated
2. SystemID – the Characteristic value

6.9.2 BLE_Profile_Update_DIS_ModelNum()

Description

The application calls this to update the Model Number characteristic of device information service.

Parameters

1. length – The Length of the characteristic to be updated.
2. modelNum – The Characteristic value

6.9.3 BLE_Profile_Update_DIS_SerialNum()

Description

The application calls this to update the Serial Number characteristic of device information service.

Parameters

1. length – the Length of the characteristic to be updated.
2. serialNum – the Characteristic value

6.9.4 BLE_Profile_Update_DIS_FirmwareRev()

Description

The application calls this to update the Firmware Revision characteristic of device information service.

Parameters

1. length – the Length of the characteristic to be updated.
2. firmwareRev – the Characteristic value to be written or updated.

6.9.5 BLE_Profile_Update_DIS_HardwareRev()

Description

The application calls this to update the Hardware Revision characteristic of device information service.

Parameters

1. length – the Length of the characteristic to be updated.
2. hardwareRev – the Characteristic value.

6.9.6 BLE_Profile_Update_DIS_SoftwareRev()

Description

The application calls this to update the Software Revision characteristic of device information service.

Parameters

1. length – the Length of the characteristic to be updated.
2. softwareRev – the Characteristic value

6.9.7 BLE_Profile_Update_DIS_manufacturerName()

Description

The application calls this to update the Manufacture Name characteristic of device information service.

Parameters

1. length – the Length of the characteristic to be updated.
2. name – the Characteristic value to be written or updated.

6.9.8 BLE_Profile_Update_DIS_IEEECertification()

Description

The application calls this to update the IEEE Certification characteristic of device information service.

Parameters

1. length – the Length of the characteristic to be updated.
2. ieeeCert – the Characteristic value

6.9.9 BLE_Profile_Update_DIS_pnpId()

Description

The application calls this to update the pnpID characteristic of device information service.

Parameters

1. length – the Length of the characteristic to be updated.
2. pnpId – the Characteristic value to be written or updated.

6.10 Proximity profile

Proximity monitor

6.10.1 ProximityMonitor_Init()

Description

Initializes the proximity profile in the monitor role. It returns BLE_STATUS_SUCCESS if the procedure is started successfully and notifies the application through the event EVT_PM_INITIALIZED.

Parameters

1. BLE_CALLBACK_FUNCTION_TYPE: the callback function to be called by the profile to notify the application of the profile specific events.

6.10.2 ProximityMonitor_Make_Discoverable()

Description

The command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

1. useBoundedDeviceList – set this to '1' if advertising is to be performed on the devices already bonded.

6.10.3 ProximityMonitorProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

Proximity reporter

6.10.4 ProximityReporter_Init()

Description

Initializes the proximity profile in reporter role. Returns BLE_STATUS_SUCCESS if the procedure is started successfully and notifies the application through the event EVT_PR_INITIALIZED.

Parameters

1. BLE_CALLBACK_FUNCTION_TYPE: the callback function to be called by the profile to notify the application of the profile specific events.
2. immAlertTxPowerSupport: set this to a non-zero value if the TX power level and immediate alert services are to be exposed by the profile, otherwise set it to '0'.

6.10.5 ProximityReporter_Make_Discoverable()

Description

The command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

1. useBoundedDeviceList – set this to '1' if advertising is to be performed on the devices already bonded.

6.10.6 ProximityReporterProfile_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

6.11 Glucose sensor profile

6.11.1 GL_Init ()

Description

Initializes the Glucose Sensor Profile. It returns BLE_STATUS_SUCCESS if the procedure is started successfully and then notifies the application of successful initialization of the profile through the event EVT_GL_INITIALIZED.

Parameters

- sequenceNumber: initial sequence number value (number of stored records on glucose measurement database).
- gl_measurement_db_records : pointer to user glucose measurement database.
- gl_measurement_context_db_records : pointer to user glucose measurement context database
- BLE_CALLBACK_FUNCTION_TYPE: callback function to be called by the profile to notify the application of the events.

6.11.2 GL_Advertize()

Description

This command puts the device into discoverable mode. BLE_STATUS_SUCCESS is returned if the command to put the device into discoverable mode was issued successfully.

Parameters

- useWhitelist: If the useWhiteList is set to TRUE, the device is configured to use the whitelist which is configured with bonded devices at the time of initialization, otherwise the device enters limited discoverable mode to connect to any of the available devices.

6.11.3 GL_ResetFlags ()

This function allows resetting of the initialization flags for the glucose sensor.

Parameters

- sequenceNumber: last stored sequence number on database.

6.11.4 GL_Send_Glucose_Measurement()

Description

This function is used to update the glucose measurement characteristic value. The function is only called as consequence of the reception of a record access control point command requesting the notification of one or more glucose measurements based on the glucose sensor database stored measurements.

Parameters

- glucoseMeasurementVal: the glucose measurement value structure containing the following members:

1. record_status_flag: flag to identify the glucose database record status
2. flags field: these flags define which data fields are present in the Characteristic value
   a) GLUCOSE_MEASUREMENT_FLAGS_TIME_OFFSET_IS_PRESENT (0x01)
   b) GLUCOSE_MEASUREMENT_FLAGS_CONCENTRATION_IS_PRESENT (0x02)
   c) GLUCOSE_MEAUREMENTS_FLAG_MMOL_L_UNITS (0x04)
   d) GLUCOSE_MEASUREMENT_FLAGS_STATUS_ANNUNCIATION_IS_PRESENT (0x08)
3. sequenceNumber field: sequence number of the glucose measurement value
4. tBasetime baseTime: time of the measurement
5. timeOffset field: time component used to define the overall user-facing time
6. glucoseConcentration: glucose concentration field (SFLOAT units of Kg or Liters)
7. typeSampleLocation Field: measurement type and sample location information

   type nibble:
   a) GLUCOSE_TYPE_CAPILLARY_WHOLE_BLOOD (0x1)
   b) GLUCOSE_TYPE_CAPILLARY_PLASMA (0x2)
   c) GLUCOSE_TYPE_VENOUS_WHOLE_BLOOD (0x3)
   d) GLUCOSE_TYPE_VENOUS_PLASMA (0x4)
   e) GLUCOSE_TYPE_ARTERIAL_WHOLE_BLOOD (0x5)
   f) GLUCOSE_TYPE_ARTERIAL_PLASMA (0x6)
   g) GLUCOSE_TYPE_UNDERTERMINED_WHOLE_BLOOD (0x7)
   h) GLUCOSE_TYPE_UNDERTERMINED_PLASMA (0x8)
   i) GLUCOSE_TYPE_INTERSISTIAL_FLUID (0x9)
   j) GLUCOSE_TYPE_CONTROL (0xA)

   sampleLocation nibble:
   a) GLUCOSE_SAMPLE_LOCATION_FINGER (0x10)
   b) GLUCOSE_SAMPLE_LOCATION_AST (0x20)
   c) GLUCOSE_SAMPLE_LOCATION_EARLOBE (0x30)
   d) GLUCOSE_SAMPLE_LOCATION_CONTROL_SOLUTION (0x40)
   e) GLUCOSE_SAMPLE_LOCATION_VALUE_NOT_AVAILABLE (0xF0)
8. sensorStatusAnnunciation field:
   a) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_DEVICE_BATTERY_LOW (0x0001)
   b) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_SENSOR_MALFUNCTION (0x0002)
   c) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_SAMPLE_SIZE (0x0004)
   d) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_STRIP_INSERTION_ERROR (0x0008)
   e) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_STRIP_TYPE_INCORRECT (0x0010)
   f) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_SENSOR_RESULT_TOO_HIGH (0x0020)
   g) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_SENSOR_RESULT_TOO_LOW

(0x0040)

h) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_SENSOR_TEMPERATURE_TOO_HIGH (0x0080)

i) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_SENSOR_TEMPERATURE_TOO_LOW (0x0100)

j) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_SENSOR_READ_INTERRUPTED (0x0200)

k) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_GENERAL_DEVICE_FAULT (0x0400)

l) GLUCOSE_SENSOR_STATUS_ANNUNCIATION_TIME_FAULT (0x0800)

6.11.5 GL_Send_Glucose_Measurement_Context()

Description

This function is used to update the glucose measurement context characteristic value. The function is only called as consequence of the reception of a record access control point command requesting the notification of one or more glucose measurement based on the glucose sensor database stored measurements if the associated glucose measurement characteristic includes contextual information.

Parameters

- glucoseMeasurementContextVal: the glucose measurement context value structure containing the following members

1. flags field: these flags define which data fields are present in the Characteristic value
   a) GLUCOSE_MEASUREMENT_CONTEXT_FLAG_CARBOHYDRATE_IS_PRESENT (0x01)
   b) GLUCOSE_MEASUREMENT_CONTEXT_FLAG_MEAL_IS_PRESENT (0x02)
   c) GLUCOSE_MEASUREMENT_CONTEXT_FLAG_TESTER_HEALTH_IS_PRESENT (0x04)
   d) GLUCOSE_MEASUREMENT_CONTEXT_FLAG_EXERCISE_DURATION_IS_PRESENT (0x08)
   e) GLUCOSE_MEASUREMENT_CONTEXT_FLAG_MEDICATION_ID_IS_PRESENT (0x10)
   f) GLUCOSE_MEASUREMENT_CONTEXT_FLAG_MEDICATION_LITER_UNITS (0x20)
   g) GLUCOSE_MEASUREMENT_CONTEXT_FLAG_HB1A1C_IS_PRESENT (0x40)
   h) GLUCOSE_MEASUREMENT_CONTEXT_FLAG_EXTENDED_IS_PRESENT (0x80)
2. sequenceNumber field: sequence number of the glucose measurement context value (same as the associated glucose measurement value)
3. extendedFlags field: optional field
4. carbohydrateId field:
   a) GLUCOSE_MEASUREMENT_CONTEXT_CARBOHYDRATE_BREAKFAST (0x01)
   b) GLUCOSE_MEASUREMENT_CONTEXT_CARBOHYDRATE_LUNCH (0x02)
   c) GLUCOSE_MEASUREMENT_CONTEXT_CARBOHYDRATE_DINNER (0x03)
   d) GLUCOSE_MEASUREMENT_CONTEXT_CARBOHYDRATE_SNACK (0x04)
   e) GLUCOSE_MEASUREMENT_CONTEXT_CARBOHYDRATE_DRINK (0x05)
   f) GLUCOSE_MEASUREMENT_CONTEXT_CARBOHYDRATE_SUPPER (0x06)
   g) GLUCOSE_MEASUREMENT_CONTEXT_CARBHYDRATEO_BRUNCH (0x07)
5. carbohydrateUnits field: units of carbohydrate
6. meal field:
   a) GLUCOSE_MEASUREMENT_CONTEXT_MEAL_PREPRANDIAL (0x1)
   b) GLUCOSE_MEASUREMENT_CONTEXT_MEAL_POSTPRANDIAL (0x2)
   c) GLUCOSE_MEASUREMENT_CONTEXT_MEAL_FASTING (0x3)
   d) GLUCOSE_MEASUREMENT_CONTEXT_MEAL_CASUAL (0x4)
   e) GLUCOSE_MEASUREMENT_CONTEXT_MEAL_BEDTIME (0x5)
7. testerHealth field

   tester nibble:
   a) GLUCOSE_MEASUREMENT_CONTEXT_TESTER_SELF (0x1)
   b) GLUCOSE_MEASUREMENT_CONTEXT_TESTER_HEALTH_CARE_PROFESSIONAL (0x2)
   c) GLUCOSE_MEASUREMENT_CONTEXT_TESTER_LAB_TEST (0x3)
   d) GLUCOSE_MEASUREMENT_CONTEXT_TESTER_NOT_AVAILABLE (0xF)

   health nibble:
   a) GLUCOSE_MEASUREMENT_CONTEXT_HEALTH_MINOR_ISSUES (0x10)
   b) GLUCOSE_MEASUREMENT_CONTEXT_HEALTH_MAJOR_ISSUES (0x20)

   c) GLUCOSE_MEASUREMENT_CONTEXT_HEALTH_DURING_MENSES (0x30)
   d) GLUCOSE_MEASUREMENT_CONTEXT_HEALTH_UNDER_STRESS (0x40)
   e) GLUCOSE_MEASUREMENT_CONTEXT_HEALTH_NO_ISSUE (0x50)
   f) GLUCOSE_MEASUREMENT_CONTEXT_HEALTH_VALUE_NOT_AVAILABLE (0xF0)
8. exerciseDuration field: exercise duration in seconds
9. exerciseIntensity field: intensity of exercise
10. medicationId field:
    a) GLUCOSE_MEASUREMENT_CONTEXT_MEDICATION_ID_RAPID_ACTING_INSULIN (0x1)
    b) GLUCOSE_MEASUREMENT_CONTEXT_MEDICATION_SHORT_ACTING_INSULIN (0x2)
    c) GLUCOSE_MEASUREMENT_CONTEXT_MEDICATION_INTERMEDIATE_ACTING_INSULIN (0x3)
    d) GLUCOSE_MEASUREMENT_CONTEXT_MEDICATION_LONG_ACTING_INSULIN (0x4)
    e) GLUCOSE_MEASUREMENT_CONTEXT_MEDICATION_PRE_MIXED_INSULIN (0x5)
11. medicationUnits field: units of kilograms or liters;
12. HbA1c field

6.11.6 GL_Set_Glucose_Feature_Value ()

Description

This function is used to set the glucose feature characteristic value.

6.11.7 GL_StateMachine()

Description

The application calls this function for checking current main profile and profile state, substate and performing related actions and consequent states updates.

Parameters

None

Parameters

- value: glucose feature value
  a) GLUCOSE_FEATURE_LOW_BATTERY_DETECTION_IS_SUPPORTED (0x0001)
  b) GLUCOSE_FEATURE_SENSOR_MALFUNCTION_DETECTION_IS_SUPPORTED (0x0002)
  c) GLUCOSE_FEATURE_SAMPLE_SIZE_IS_SUPPORTED (0x0004)
  d) GLUCOSE_FEATURE_SENSOR_STRIP_INSERTION_ERROR_IS_SUPPORTED (0x0008)
  e) GLUCOSE_FEATURE_SENSOR_STRIP_TYPE_ERROR_IS_SUPPORTED (0x0010)
  f) GLUCOSE_FEATURE_SENSOR_RESULT_HIGH_LOW_DETECTION_IS_SUPPORTED (0x0020)
  g) GLUCOSE_FEATURE_SENSOR_TEMPERATURE_HIGH_LOW_DETECTION_IS_SUPPORTED (0x0040)
  h) GLUCOSE_FEATURE_SENSOR_READ_INTERRUPT_DETECTION_IS_SUPPORTED (0x0080)
  i) GLUCOSE_FEATURE_GENERAL_DEVICE_FAULT_IS_SUPPORTED (0x0100)
  j) GLUCOSE_FEATURE_TIME_FAULT_IS_SUPPORTED (0x0200)
  k) GLUCOSE_FEATURE_MULTIPLE_BOND_IS_SUPPORTED (0x0400)

7 List of references

Table 2. References

Name	Title
ACI	Application Command Interface
BLE	Bluetooth Low Energy
Bluetooth Specification specification	Specification of the Bluetooth system V4.0
UM1755	BlueNRG Bluetooth LE stack application command interface (ACI) User Manual
UM1686	BlueNRG Development Kits User Manual
USB	Universal serial bus

8 Revision history

Table 3. Document revision history

Date	Revision	Changes
26-Aug-2014	1	Initial release.
04-Dec-2014	2	Ported profiles to simplified ACI framework (the one provided within the BlueNRG DK SW package).