AN4642
Application note

Overview of the BLE Profiles application for X-CUBE-BLE1, expansion for STM32Cube

Introduction

This document describes the BLE Profiles application for X-CUBE-BLE1.

X-CUBE-BLE1 provides the complete STM32 middleware to build applications based on Bluetooth Low Energy. It is highly portable across different MCU families thanks to STM32Cube.

The software:

- provides implementation examples for STM32 Nucleo platforms equipped with the X-NUCLEO-IDB04A1 STM32 expansion board
- is based on STM32Cube technology and extends STM32Cube-based packages

Information regarding STM32Cube is available on st.com at:

http://www.st.com/stm32cube.

Contents

1 BLE Profiles application for X-CUBE-BLE1, expansion for STM32Cube

1.1 Profiles application overview

This chapter describes the BLE Profiles example application running on STM32 Nucleo board connected with an X-NUCLEO-IDB04A1 STM32 expansion board. This application is included as part of the STM32Cube firmware for the X-CUBE-BLE1.

It is assumed that the reader has prior knowledge of the Bluetooth Low Energy protocol and its application profiles.

The current software release supports the following BLE profiles:

- Heart Rate
- Health Thermometer
- Glucose
- Blood Pressure
- Find Me
- Alert Notification
- Time
- Proximity

The application is based on the standard GATT-based profiles (https://developer.bluetooth.org/TechnologyOverview/Pages/Profiles.aspx) described in the Bluetooth SIG specifications.

Please refer to [3] for information on getting started and for details regarding BLE application development for STM32 Nucleo boards equipped with an X-NUCLEO-IDB04A1 STM32 expansion board.

The STM32 Nucleo device acts as the GAP peripheral device running the BLE Profiles application. It sends profile-related data to the GAP central device (the collector), which is an Android smartphone in this case. Please refer to *Section 2* for details about system setup and configuration.

The following sub-sections briefly describe the interaction between the peripheral and the central devices for each BLE Profile.

1.1.1 Heart rate monitor

The Heart Rate Monitor (HRM) application runs the Heart Rate Profile described in the Bluetooth Profile Specifications. It exposes the Heart Rate Service and enables the role of the Heart Rate Sensor (HRS) in the STM32 Nucleo device.

The application sends periodic data to simulate heart rate measurements as the actual sensor is not present on the STM32 Nucleo board.

Any central device running the Heart Rate Profile can collect data from the Heart Rate Monitor.

1. The HRS initializes the Heart Rate Profile and starts advertising its address and services.
2. The Collector scans for a HRS device and sends a connection request if it detects one.
3. Once connection is established, the HRS sends data over the connection link.

1.1.2 Health thermometer monitor

The Health Thermometer Monitor (HTM) application runs the Health Thermometer Profile described in the Bluetooth Profile Specifications. It exposes the Health Thermometer Service and enables the role of the Health Thermometer Sensor (HTS) in the STM32 Nucleo device.

The application sends periodic data to simulate body temperature readings as an actual temperature sensor is not present on the STM32 Nucleo board.

Any central device running the Health Thermometer Profile can collect data from the Health Thermometer Monitor.

1. The HTS device initializes the Health Thermometer Profile and starts advertising its address and services.
2. The Collector scans for an HTS and sends a connection request if it detects one.
3. Once connection is established, the HTS sends data over the connection link.

1.1.3 Glucose monitor

The Glucose Monitor (GLM) application runs the Glucose Profile as described in the Bluetooth Profile Specifications. It exposes the Glucose Service and enables the role of the Glucose Sensor (GLS) in the STM32 Nucleo device.

The application sends periodic data to simulate glucose concentration readings as an actual glucose sensor is not present on the STM32 Nucleo board. Other information describing the sensor position and its supported features is also transmitted.

Any central device running the Glucose Profile can collect data from the Glucose Monitor.

1. The GLS initializes the Glucose Profile and then starts advertising its address and services.
2. The Collector scans for a GLS and sends a connection request if it detects one.
3. Once connection is established, the GLS sends data over the connection link.

1.1.4 Blood pressure monitor

The Blood Pressure Monitor (BLP) application runs the Blood Pressure Profile as described in the Bluetooth Profile Specifications. It exposes the Blood Pressure Service and enables the role of the Blood Pressure Sensor (BLS) in the STM32 Nucleo device.

The application sends periodic data to simulate blood pressure readings (and related data) as an actual blood pressure sensor is not present on the STM32 Nucleo board.

Any central device running the Blood Pressure Profile can collect data from the Blood Pressure Monitor.

1. The BLS initializes the Blood Pressure Profile and begins advertising its address and services.
2. The Collector scans for a BLS and sends a connection request if it detects one.
3. Once connection is established, the BLS sends data over the connection link.

1.1.5 Find me

The Find Me (FMT) application runs the Find Me Profile described in the Bluetooth Profile Specifications. It exposes and enables the role of the Find Me Target (FMT) in the STM32 Nucleo device. It causes the STM32 Nucleo board to produce an alert signal (blinking LED) when a button is pressed on a Find Me Locator device (when acting as the central device).

1. The FMT device initializes the profile and starts advertising its address and services.
2. The Locator scans for an FMT and sends a connection request if it detects one.

1.1.6 Alert notification

The Alert Notification (ANS) application runs the Alert Notification Profile described in the Bluetooth Profile Specifications. It exposes and enables the role of the Alert Notification Server (ANS) in the STM32 Nucleo device. A client device can receive different types of alerts and event information, as well as information on the count of new alerts and unread items in the server device (the STM32 Nucleo board).

To demonstrate the function, the ANS exposes the "New email" alert with a sample text message. Any central device running the Alert Notification Client is notified when a new email is received.

1. The ANS device initializes the profile and then starts advertising its address and services.
2. The Client device scans for an ANS device and sends a connection request if it detects one, so it can enable alert notification.

1.1.7 Time

The Time (TIP) application runs the Time Profile described in the Bluetooth Profile Specifications. It exposes and enables the role of the Time Server (TS) in the STM32 Nucleo device. A client device (when acting as the central device) can obtain date and time, and related information exposed by the Current Time Service in the STM32 Nucleo device.

1. The TS device initializes the profile and then starts advertising its address and services.
2. The Time Client scans for a TS and sends a connection request if it detects one, so it can obtain date and time data.

1.1.8 Proximity

The Proximity (PXP) application runs the Proximity Profile described in the Bluetooth Profile Specifications. It exposes and enables the role of the Proximity Reporter in the STM32 Nucleo device.

A device running the Proximity Monitor can immediately signal when a peer device moves away and the connection drops or when the path loss exceeds a pre-defined threshold.

1. The Proximity Reporter initializes the profile and then starts advertising its address and services.
2. The Monitor scans for a PXR and sends a connection request if it detects one, to enable alert notifications and/or set the path loss level.

1.2 Software blocks

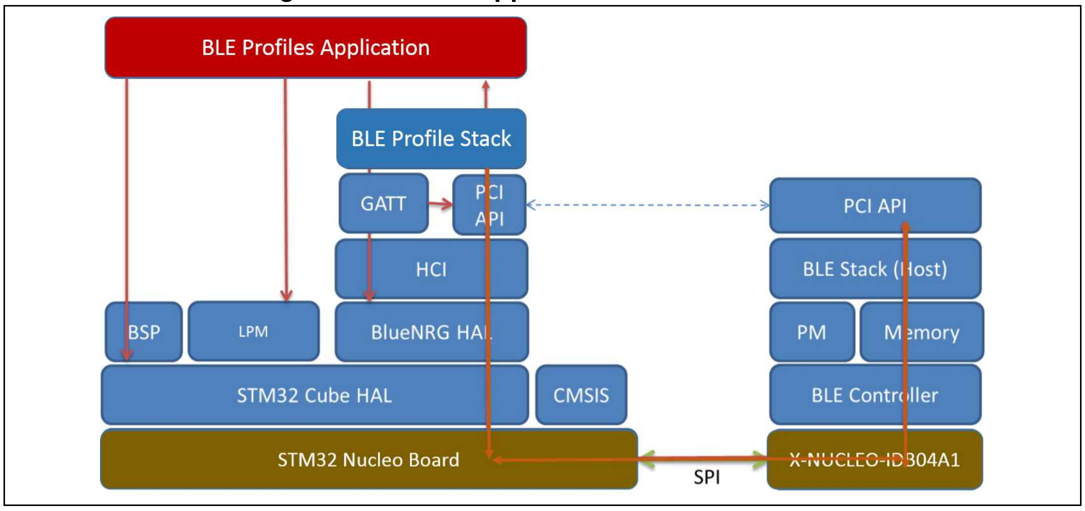
This section describes the software layers (shown in the diagram below) present in the BLE Profiles application, running on an STM32 Nucleo development board with the STM32Cube software environment.

The most important layers are:

- the STM32Cube HAL layer
- the Board Support Package (BSP) layer
- the BlueNRG HAL layer
- the Profile Command Interface (PCI) layer
- the GATT profile
- the BLE profile stack
- the BLE Profiles application

The PCI and GATT rely on the HCI layer to send BLE commands to the BlueNRG STM32 expansion board. The Generic Attribute (GATT) profile uses both PCI API and HCI to send BLE commands to the BlueNRG STM32 expansion board.

Figure 1. Profiles application SW architecture



The most important software layers for the application are briefly described below.

1.2.1 Profile command interface (PCI)

The Profile Command Interface (PCI) is the main API for all applications that use the GATT-based profiles on the BlueNRG device. The PCI exposes the APIs used to send commands to the BlueNRG STM32 expansion board.

1.2.2 Low Power Manager (LPM)

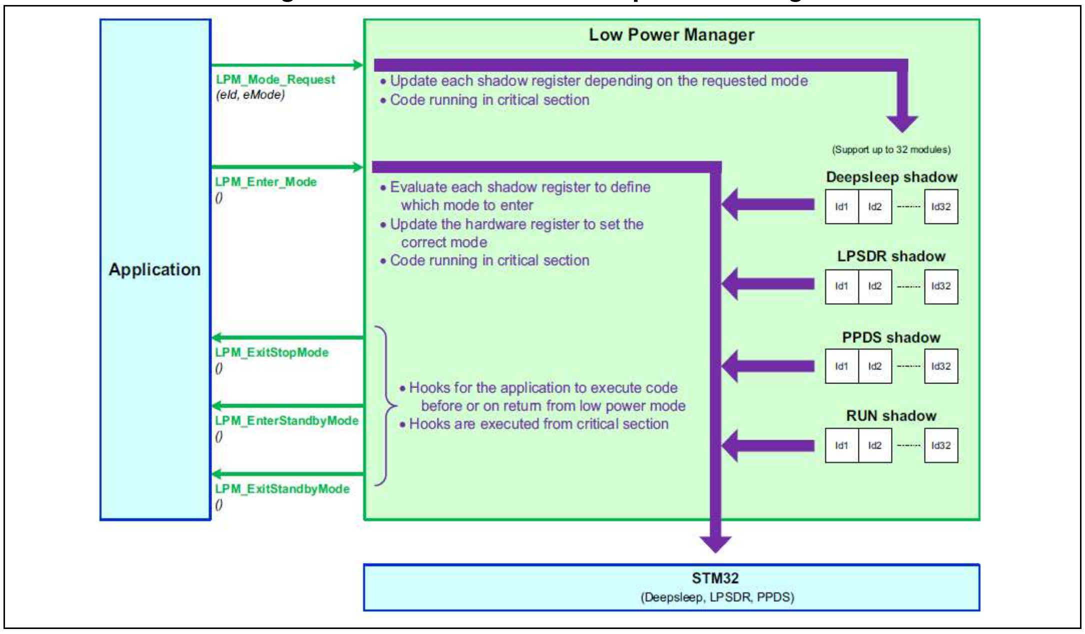
The Low Power Manager lets the application allow the system drop to low-power states whenever necessary conditions are met. The Low Power Manager provides a set of APIs that enable the application to support active power management.

It provides support for:

- the application to establish the lowest allowable power mode
- informing the system that no more power is required
- callbacks for the implementation of specific routines when coming out of, or going into low power modes (Stop mode or Standby mode)

The following figure provides an overview of LPM:

Figure 2. Overview of the low power manager


Application
Low Power Manager
LPM_Mode_Request
(eId, eMode)
• Update each shadow register depending on the requested mode
• Code running in critical section
(Support up to 32 modules)
LPM_Enter_Mode
()
• Evaluate each shadow register to define which mode to enter
• Update the hardware register to set the correct mode
• Code running in critical section
Deepsleep shadow
LPSDR shadow
PPDS shadow
RUN shadow
Id1
Id2
Id32
LPM_ExitStopMode
()
LPM_EnterStandbyMode
()
LPM_ExitStandbyMode
()
• Hooks for the application to execute code before or on return from low power mode
• Hooks are executed from critical section
STM32
(Deepsleep, LPSDR, PPDS)

The Low Power Manager provides a simple interface to receive the input of different users and compute the lowest allowable system power mode. It also provides hooks to the application before entering or exiting low power mode.

1.2.3 Low power manager API

The Low Power Manager provides the following APIs that applications can use to place the system in specific low power modes:

- LPM_Mode_Request(): specify the lowest supported power mode.
- LPM_Enter_Mode(): request the system to enter low power mode.
- LPM_ExitStopMode(): called by the LPM in a critical section to allow the application to implement dedicated code before exiting the critical section.
- LPM_Enter_StandbyMode(): called by the LPM in a critical section to allow the application to implement dedicated code before entering standby mode.

1.2.4 BLE profile stack

The BLE profile stack layer implements the Bluetooth Low Energy profiles which are used by the applications. The BLE profile stack layer uses the PCI for communicating with the controller. It defines the necessary functions and callbacks for the applications to communicate with the profiles.

1.2.5 BSP layer

The BSP software layer supports the peripherals on the STM32 Nucleo board, excluding the MCU. This is a limited set of APIs which provides a programming interface for certain board-specific peripherals like the LED, the user button, etc. This interface also helps in identifying the specific board version. For information on using and initializing the BSP layer, refer to [3].

1.3 BLE profiles application files

The directory structure of the Profiles application is shown below. Application-specific source code is present in the following files:

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\main.c:
  – performs STM32 Nucleo board initialization and calls the generic profile layer initialization defined in the profile_application.c source file
  – defines the loop handling the main processes. Three processes are handled: HCI_Process(), generic for all profiles; profileStateMachineFunc() which is specific for each profile state machine; profileApplicationProcessFunc(), implements a specific profile according to the standard specifications described in *Section 1.1*.
- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\profile_application.c:
  – contains the source code to initialize the BlueNRG Profile stack, set TX power and security parameters common to all profiles
  – defines the callback for handling generic BLE events

According to the list in *Section 1.1*, the following files contains source code for profile context initialization, application event handling, profile initialization, advertisement and application state machine management.

1.3.1 Heart rate monitor

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\hrm_profile_application.c

1.3.2 Health thermometer monitor

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\htm_profile_application.c

1.3.3 Glucose monitor

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\gs_profile_application.c

1.3.4 Blood pressure monitor

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\bps_profile_application.c

1.3.5 Find me target

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\fmt_profile_application.c

1.3.6 Alert notification server

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\ans_profile_application.c

1.3.7 Time server

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\ts_profile_application.c

1.3.8 Proximity reporter

- $BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Project\Src\pr_profile_application.c

1.4 BLE profile initialization

The application must initialize various hardware blocks and data structures before using them. The following sections describe two types of initialization which are needed.

1.4.1 Generic initialization

The application must correctly initialize various hardware blocks on the STM32 Nucleo board before using them. Refer to chapter 7.1.1 of [3] for details on generic initialization.

1.4.2 BLE profile specific initialization

The application must initialize the BLE stack and BLE profile that it intends to use. This initialization is done via the following APIs:

- BLE_Profile_Init(): performs generic initialization for BLE profiles (see profile_application.c file).
- Profile specific initialization: the application must initialize the individual profile that it intends to use, the PCI layer provides the API for performing this initialization.
  - HRM application: the Heart Rate profile is initialized via HRProfile_Init() API (see hrm_profile_application.c)
  - HTM application: the Health Thermometer profile is initialized via HT_Init() API (see htm_profile_application.c)
  - GLM application: the Glucose profile is initialized via GL_Init() API (see gs_profile_application.c)
  - BLP application: the Blood Pressure profile is initialized via BPS_Init() API (see bps_profile_application.c)
  - FMT application: the Find Me Target is initialized via FindMeTarget_Init() API (see fmt_profile_application.c)
  - ANS application: the Alert Notification Server is initialized via ANS_Init() API (see ans_profile_application.c)
  - TIP application: the Time Server is initialized via the TimeServer_Init() API (see ts_profile_application.c)
  - PXP application: the Proximity Reporter is initialized via the ProximityReporter_Init() API (see pr_profile_application.c)

1.5 BLE profiles application state machine

This section describes the state machine specific for each profile.

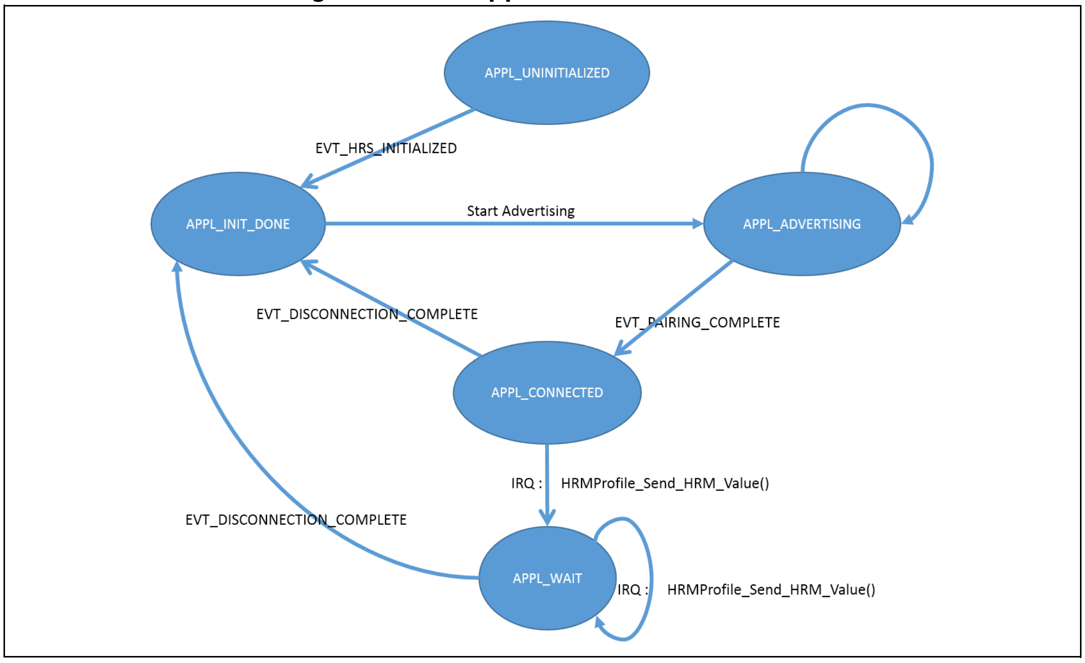
1.5.1 HRM application state machine

When the STM32 Nucleo board is turned on, the HRM application is in the APPL_UNINITIALIZED state, and the application initializes the hardware and the BLE heart rate profile (*Section 1.4*).

Once this is complete, the HRS application receives the EVT_HRS_INITIALIZED event and enters the APPL_INIT_DONE state, and the application starts advertising its characteristics so that the central device can detect it and choose whether to connect to it, upon which the peripheral device enters the APPL_CONNECTED state.

In APPL_CONNECTED state, the Heart Rate measurement is sent to the central device when an interrupt (timer) is received, after which the peripheral device enters the APPL_WAIT state and waits for the interrupt (timer).

Figure 3. HRM application state machine



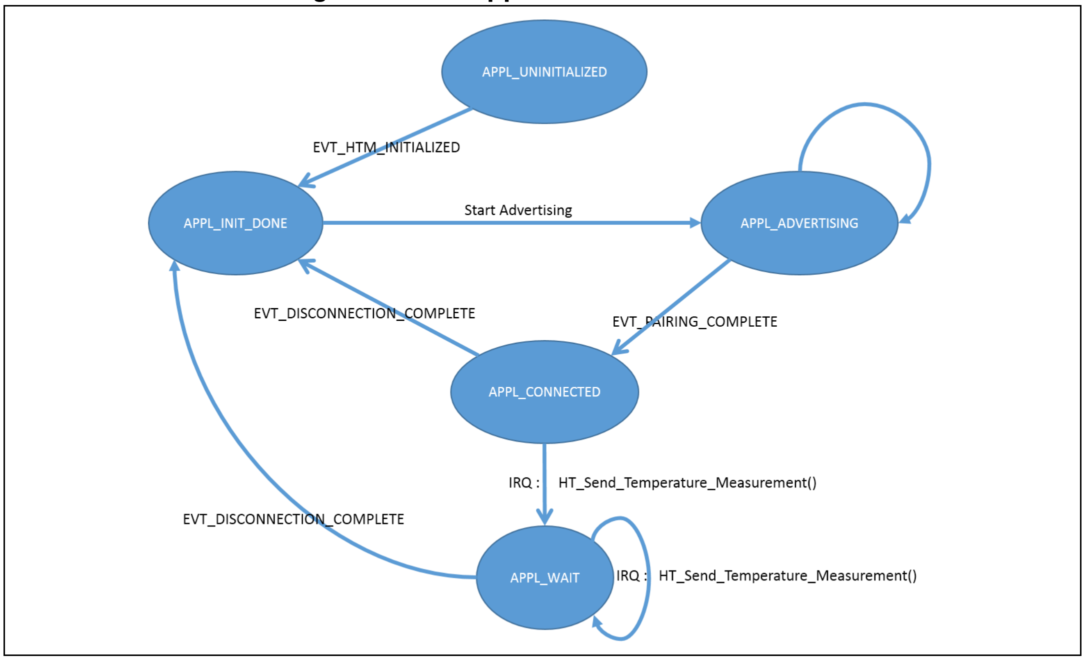
1.5.2 HTM application state machine

When the STM32 Nucleo board is turned on, the HTM application is in APPL_UNINITIALIZED state and initializes the hardware and the BLE Health Thermometer profile (*Section 1.4*).

On completion, the HTM application receives the EVT_HT_INITIALIZED event and enters APPL_INIT_DONE state. The application then starts advertising its characteristics so that the central device can discover it and choose whether to connect to it, upon which the peripheral device enters APPL_CONNECTED state.

In the APPL_CONNECTED state, the Measurement Interval is set and the Health Thermometer measurement is sent to the central device when an interrupt (timer) is received, after which the peripheral device enters APPL_WAIT state and waits for the interrupt (timer).

Figure 4. HTM application state machine


APPL_UNINITIALIZED
EVT_HTM_INITIALIZED
APPL_INIT_DONE
Start Advertising
APPL_ADVERTISING
EVT_DISCONNECTION_COMPLETE
EVT_PAIRING_COMPLETE
APPL_CONNECTED
IRQ : HT_Send_Temperature_Measurement()
EVT_DISCONNECTION_COMPLETE
APPL_WAIT
IRQ : HT_Send_Temperature_Measurement()

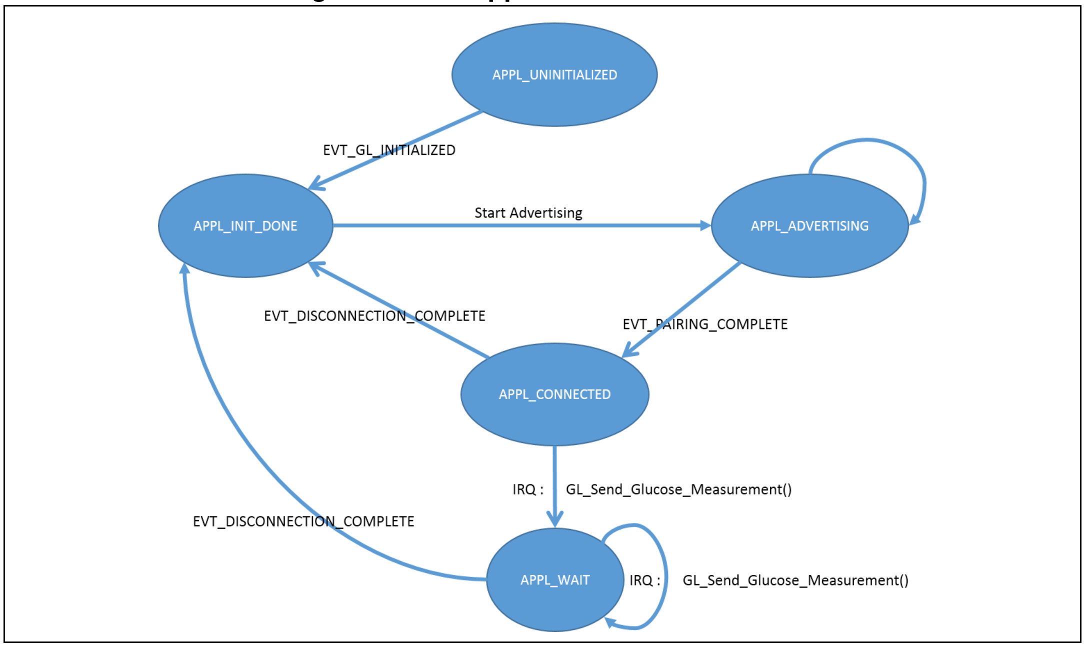
1.5.3 GLM application state machine

When the STM32 Nucleo board is turned on, the GLM application is in APPL_UNINITIALIZED state and initializes the hardware and the BLE Glucose profile (*Section 1.4*).

Once initialization is complete, the GLM application receives the EVT_GL_INITIALIZED event and enters APPL_INIT_DONE state. The application then starts advertising its characteristics so that the central device can discover it and choose whether to connect to it, upon which the peripheral enters the APPL_CONNECTED state.

In the APPL_CONNECTED state, the Glucose measurement is sent to the central device when an interrupt (timer) is received, after which the peripheral device enters APPL_WAIT state and waits for the interrupt (timer).

Figure 5. GLM application state machine



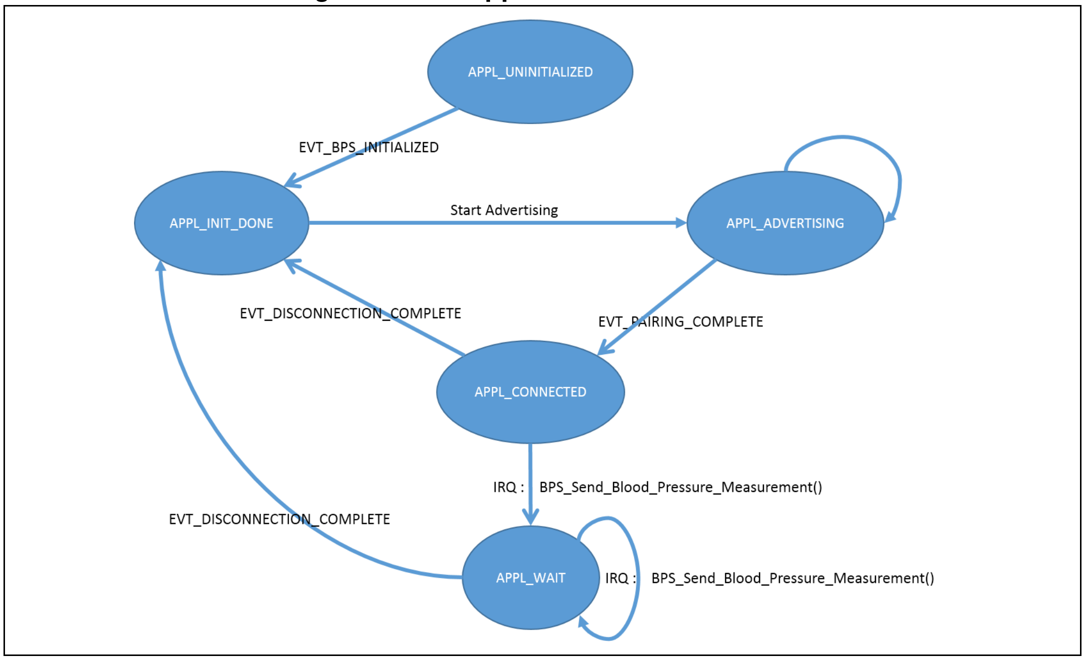
1.5.4 BLP application state machine

When the STM32 Nucleo board is turned on, the BLP application is in the APPL_UNINITIALIZED state and initializes the hardware and the BLE Blood Pressure profile (*Section 1.4*).

Once the initialization is complete, the BLP application receives the EVT_BPS_INITIALIZED event and enters APPL_INIT_DONE state. The application then starts advertising its characteristics so that the central device can detect and choose whether to connect to it, upon which the peripheral enters APPL_CONNECTED state.

In the APPL_CONNECTED state, the Blood Pressure measurement is sent to the central device when an interrupt (timer) is received, after which the peripheral device enters the APPL_WAIT state and waits for the interrupt (timer).

Figure 6. BLP application state machine


APPL_UNINITIALIZED
EVT_BPS_INITIALIZED
APPL_INIT_DONE
Start Advertising
APPL_ADVERTISING
EVT_DISCONNECTION_COMPLETE
EVT_PAIRING_COMPLETE
APPL_CONNECTED
IRQ : BPS_Send_Blood_Pressure_Measurement()
EVT_DISCONNECTION_COMPLETE
APPL_WAIT
IRQ : BPS_Send_Blood_Pressure_Measurement()

1.5.5 FMT application state machine

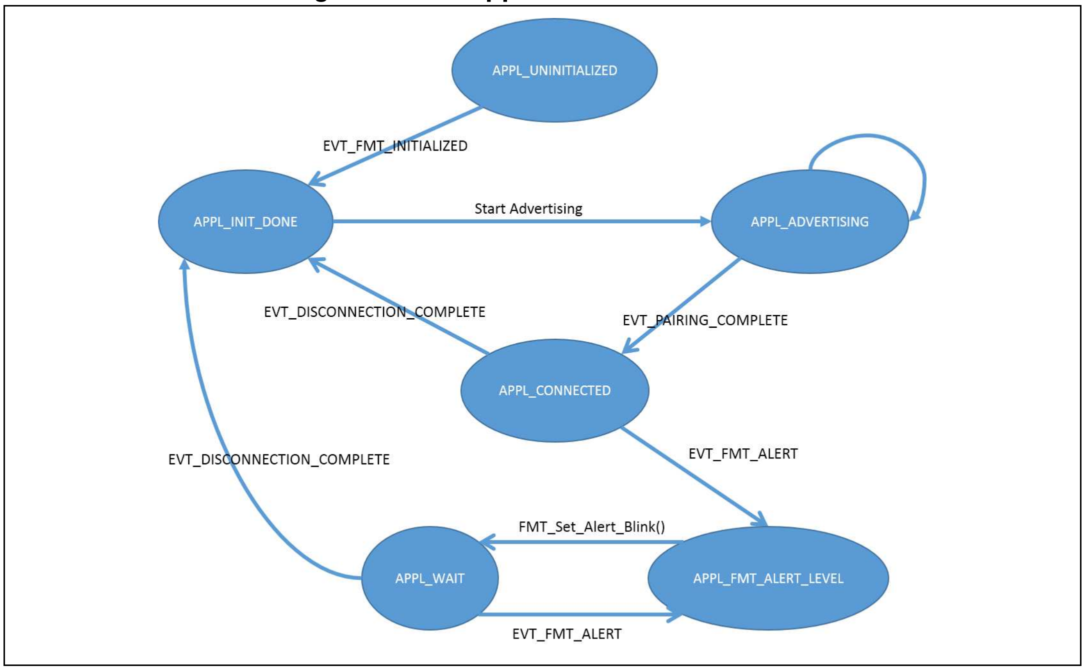
When the STM32 Nucleo board is turned on, the FM application is in APPL_UNINITIALIZED state and initializes the hardware and the BLE Find Me Target (*Section 1.4*).

Once the initialization is complete, the FMT application receives the EVT_FMT_INITIALIZED event and enters APPL_INIT_DONE state. The application then starts advertising its characteristics so that the central device can detect and choose whether to connect to it, upon which the peripheral enters the APPL_CONNECTED state.

From the APPL_CONNECTED state, the application enters the APPL_FMT_ALERT_LEVEL state when an EVT_FMT_ALERT event is received, which is indicated by a blinking yellow LED on the STM32 Nucleo board at a rate determined by the alert level set by the central device. If the central device sets the "No Alert" option, the LED is turned off.

From the APPL_FMT_ALERT_LEVEL state, the application enters the APPL_WAIT state and waits for another alert event to occur.

Figure 7. FMT application state machine



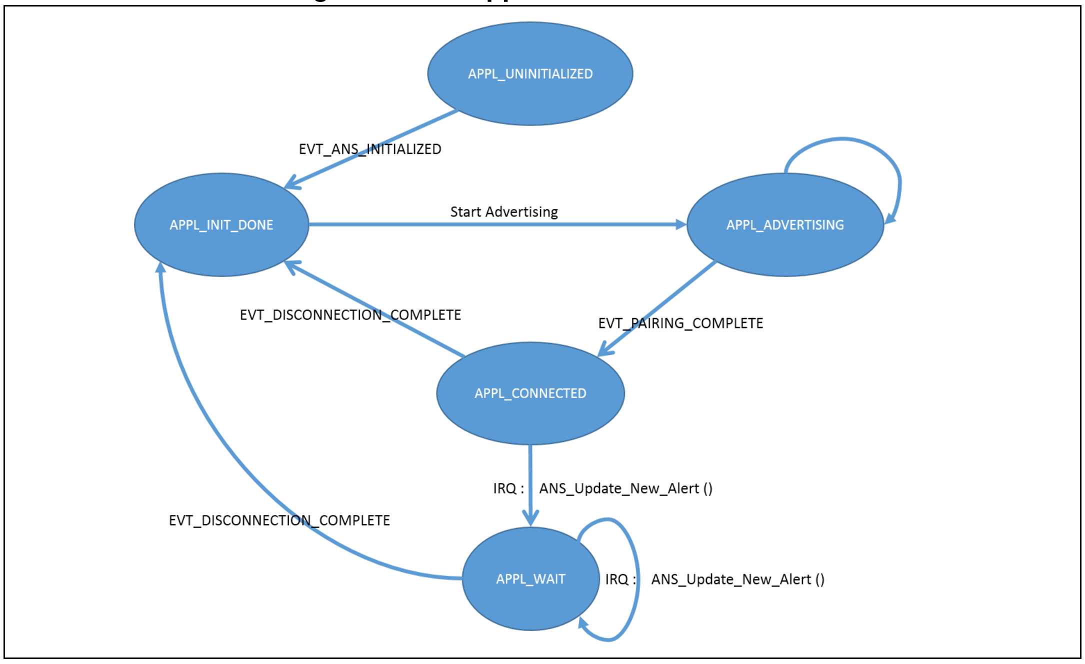
1.5.6 ANS application state machine

When the STM32 Nucleo board is turned on, the ANS application is in APPL_UNINITIALIZED state and initializes the hardware and the BLE Alert Notification Server (*Section 1.4*).

Once the initialization is complete, the ANS application receives the EVT_ANS_INITIALIZED event and enters APPL_INIT_DONE state. The application then starts advertising its characteristics so that the central device can discover and choose whether to connect to it, upon which the peripheral enters the APPL_CONNECTED state.

In APPL_CONNECTED state, the new alert (e.g., a new email alert) is sent to the central device when an interrupt (timer) is received, after which the peripheral enters APPL_WAIT state and waits for the interrupt (timer).

Figure 8. ANS application state machine



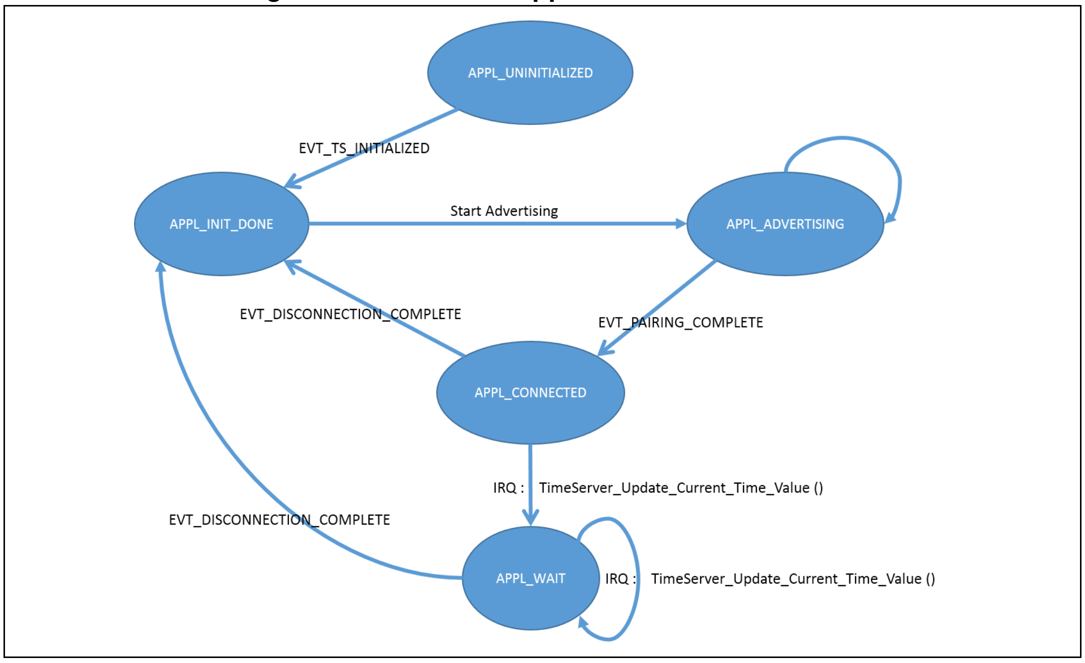
1.5.7 Time server application state machine

When the STM32 Nucleo board is turned on, the Time Server application is in APPL_UNINITIALIZED state and initializes the hardware and the BLE Time Server (*Section 1.4*).

Once the initialization is complete, the Time Server application receives the EVT_TS_INITIALIZED event and enters APPL_INIT_DONE state. The application then starts advertising its characteristics so that the central device can detect and choose whether to connect to it, upon which the peripheral enters APPL_CONNECTED state.

In the APPL_CONNECTED state, the current time is updated and sent to the central device when an interrupt (timer) is received, after which the peripheral device enters the APPL_WAIT state and waits for the interrupt (timer).

Figure 9. Time server application state machine


APPL_UNINITIALIZED
EVT_TS_INITIALIZED
APPL_INIT_DONE
Start Advertising
APPL_ADVERTISING
EVT_DISCONNECTION_COMPLETE
EVT_PAIRING_COMPLETE
APPL_CONNECTED
IRQ : TimeServer_Update_Current_Time_Value ()
EVT_DISCONNECTION_COMPLETE
APPL_WAIT
IRQ : TimeServer_Update_Current_Time_Value ()

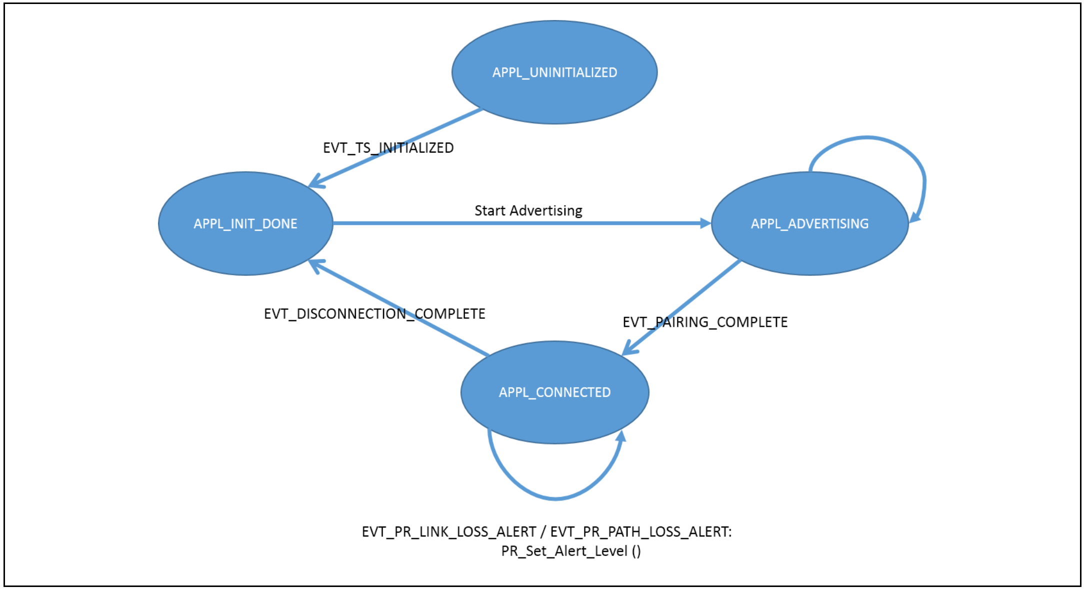
1.5.8 PXP application state machine

When the STM32 Nucleo board is turned on, the Proximity Reporter application is in APPL_UNINITIALIZED state. and initializes the hardware and the BLE Proximity Reporter (*Section 1.4*).

Once the initialization is complete, the PXP application receives the EVT_PR_INITIALIZED event and enters APPL_INIT_DONE state. The application then starts advertising its characteristics so that the central device can detect and choose whether to connect to it, upon which the peripheral enters the APPL_CONNECTED state.

In the APPL_CONNECTED state, the application waits for an EVT_PR_PATH_LOSS_ALERT or an EVT_PR_LINK_LOSS_ALERT event to occur, which is indicated by a blinking yellow LED on the STM32 Nucleo board at a rate which is determined by the alert level set by the central device.

Figure 10. PXP application state machine



1.6 MCU power modes

The Profiles application leverages the advanced STM32 MCU low-power features.

Two MCU modes are used by the application:

- RUN mode
  - The standard execution mode in which all the MCU features are available.
- STOP mode with RTC
  - This mode achieves the lowest power consumption while retaining the RAM and register contents and real time clock.
  - All clocks in the VCORE domain are stopped: the PLL, MSI RC, HSE crystal and HSI RC oscillators are disabled.
  - The LSE or LSI is still running.
  - The voltage regulator is in low-power mode.
  - The device can be woken up from Stop mode by any of the EXTI lines, in 3.5 µs, the MCU can serve the interrupt or resume the code.

The low power manager is used to exploit low power features of the STM32 MCU. Refer to *Section 1.2.3* for details regarding the low power manager API. For further information about the STM32 MCU low power modes, please refer to [5].

1.7 BLE profiles event handling

The following events are generated by the underlying BLE stack during the lifetime of the profile application.

1.7.1 Generic events

Generic BLE events are not specific to any profile. The following generic events are used by Profiles application:

- EVT_MP_BLUE_INITIALIZED: sent to the application by the main profile when the controller has been initialized.
- EVT_MP_CONNECTION_COMPLETE: sent to the application by the main profile when a connection has been successfully established with the peer.
- EVT_MP_PASSKEY_REQUEST: sent to the application by the main profile when there is a request for passkey during pairing process from the controller. This event has no parameters. The application has to call the function BLE_Profile_Send_Pass_Key(), and pass the passkey to the controller.
- EVT_MP_PAIRING_COMPLETE: sent to the application by the main profile when the device is successfully paired with the peer.
- EVT_MP_DISCONNECTION_COMPLETE: sent to the application by the main profile to notify the result of a disconnection procedure initiated either by master or slave.
- EVT_MP_ADVERTISING_TIMEOUT: sent by child profiles when the limited discoverable mode times out or the profile-specific advertising timeout occurs. It is the application's responsibility to restart the advertising.

1.7.2 Heart rate profile events

These events are specific to the heart rate profile. The following heart rate profile events are used by the HRM application:

- EVT_HRS_INITIALIZED: sent to the application when the heart rate profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.
- EVT_HRS_CHAR_UPDATE_CMPLT: sent to the application whenever it has started the characteristic update procedure to update heart rate measurement or body sensor location.
- EVT_HRS_RESET_ENERGY_EXPENDED: sent to the application when the peer writes a value of 0x01 to the control point characteristic. This event has no parameters. The application has to restart accumulating the energy expended value from 0.

1.7.3 Health thermometer profile events

These events are specific to health thermometer profile. The following health thermometer profile events are used by the HTM application:

- EVT_HT_INITIALIZED: sent to the application when the health thermometer profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.
- EVT_HT_TEMPERATURE_CHAR_UPDATE_CMPLT: sent to the application whenever it has completed the characteristic update procedure to update the temperature measurement.
- EVT_HT_INTERMEDIATE_TEMP_CHAR_UPDATE_CMPLT: sent to the application whenever it has completed the characteristic update procedure to update the intermediate temperature measurement.

- EVT_HT_MEASUREMENT_INTERVAL_RECEIVED: sent to the application whenever it has started the characteristic update procedure to update the temperature measurement interval according to the value received from the collector.
- EVT_HT_MEASUREMENT_INTERVAL_UPDATE_CMPLT: sent to the application whenever it has completed the characteristic update procedure to update the temperature measurement interval.

1.7.4 Glucose profile events

These events are specific to glucose profile. The following glucose profile events are used by the GLM application:

- EVT_GL_INITIALIZED: sent to the application when the glucose profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.

1.7.5 Blood pressure profile events

These events are specific to blood pressure profile. The following blood pressure profile events are used by the BLP application:

- EVT_BPS_INITIALIZED: sent to the application when the blood pressure profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.
- EVT_BPS_BPM_CHAR_UPDATE_CMPLT: sent to the application whenever it has completed the characteristic update procedure to update blood pressure measurement.
- EVT_BPS_ICP_CHAR_UPDATE_CMPLT: sent to the application whenever it has completed the characteristic update procedure to update intermediate cuff pressure.

1.7.6 Find me profile events

These events are specific to Find Me profile (target role). The following find me profile (target role) events are used by the FMT application:

- EVT_FMT_INITIALIZED: sent to the application when the find me target has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.
- EVT_FMT_ALERT: sent to the application whenever an alert signaling request has been received from the Locator.

1.7.7 Alert notification profile events

These events are specific to Alert Notification profile (server role). The following alert notification profile (server role) events are used by the ANS application:

- EVT_ANS_INITIALIZED: sent to the application when the alert notification server has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.

1.7.8 Time profile events

These events are specific to Time profile (server role). The following time profile (server role) events are used by the TIP application:

- EVT_TS_INITIALIZED: sent to the application when the time profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.
- EVT_TS_START_REFTIME_UPDATE: sent to the application whenever it has requested to update its reference time.
- EVT_TS_CHAR_UPDATE_CMPLT: sent to the application whenever it has completed the characteristic update procedure to update any time server characteristic.
- EVT_TS_CURTIME_READ_REQ: sent to the application whenever that the current time is being requested for read.

1.7.9 Proximity profile events

These events are specific to Proximity profile (reporter role). The following proximity profile (reporter role) events are used by the PXP application:

- EVT_PR_INITIALIZED: sent to the application when the proximity profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.
- EVT_PR_PATH_LOSS_ALERT: sent to the application whenever an alert signaling is requested due to path loss increasing over a preset level.
- EVT_PR_LINK_LOSS_ALERT: sent to the application whenever an alert signaling is requested due link loss.

1.8 Running the BLE profiles application

This section demonstrates how to run the profiles application on the STM32 Nucleo connected with a BlueNRG STM32 expansion board and then use it from an Android application running on a smartphone. The user can select the specific profile by unchecking the respective profile macro within the file:

$BASE_DIR\Middlewares\ST\STM32_BlueNRG\Profile_Framework\includes\host_config.h

1.8.1 Software setup on STM32 Nucleo

Download the firmware onto the STM32 Nucleo by "drag and drop" or using the ST-Link utility.

The binary files for the profile application (Profiles.bin) can be found in the following folder:

“$BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC\Binary"

The IAR project file for profile application (e.g., for STM32 Nucleo L0 board) can be found in the following folder:

“$BASE_DIR\Projects\Multi\Applications\Profiles_LP_RTC \EWARM\L0\Profiles.eww"

This project was built using IAR version 7.20.

1.8.2 Starting the STM32 Nucleo and app

Reset the STM32 Nucleo board to start running the profiles application. When the application is running, the LED on the STM32 Nucleo board will start blinking.

Note: The application FW should be loaded on the board before performing this step.

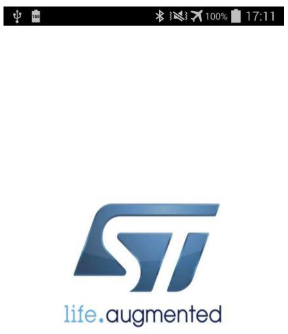
1.8.3 Software setup on the phone

Retrieve the ST BLE Profiles app from the software package (see *Section 2.2*) and install it on the Android smartphone.

1.8.4 Starting the phone Profiles App and connecting to central device

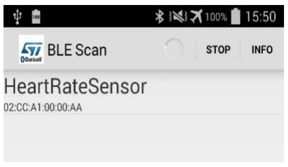
The following diagram shows the main page of the android application; an automatic background scan is initiated.

Figure 11. Splash screen of the profiles Android app



The scan can be started/stopped by hitting the "Start/Stop" button. The following diagram shows a screenshot when the peripheral is detected by the application (in this case, a Heart Rate Sensor has been detected):

Figure 12. Screenshot of profiles app showing scanned HRS device



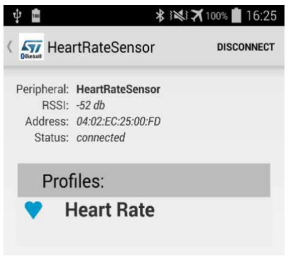
HeartRateSensor
02:CC:A1:00:00:AA

A brief description about running the app and the link to the ST product web page can be accessed through the "Info" button.

1.8.5 Reading HRM data on a smartphone

Once the heart rate sensor is detected, the user can select this device by tapping on it. The Android BLE pairing procedure will initiate the pairing with heart rate sensor if it is being used for the first time. The following screenshot shows the successful pairing process:

Figure 13. Screenshot of profiles app paired with HRS device


16:25

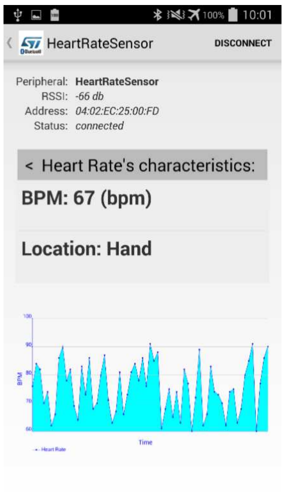
HeartRateSensor DISCONNECT

Peripheral: HeartRateSensor
RSSI: -52 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Heart Rate

To obtain heart rate measurements, select "Heart Rate" Service. The following diagram shows heart rate characteristics obtained from the heart rate sensor. The user can read the instantaneous heart rate value, the sensor location and the graph displaying the latest heart rate values.

Figure 14. Screenshot of profiles app paired with HRS device and being notified


10:01

HeartRateSensor DISCONNECT

Peripheral: HeartRateSensor
RSSI: -66 db
Address: 04:02:EC:25:00:FD
Status: connected

< Heart Rate's characteristics:

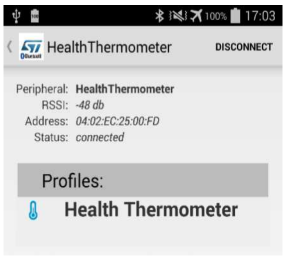
BPM: 67 (bpm)

Location: Hand

1.8.6 Reading HTM Data on smartphone

The following screenshot shows the successful pairing with a health thermometer sensor:

Figure 15. Screenshot of profiles app paired with HTS device


17:03

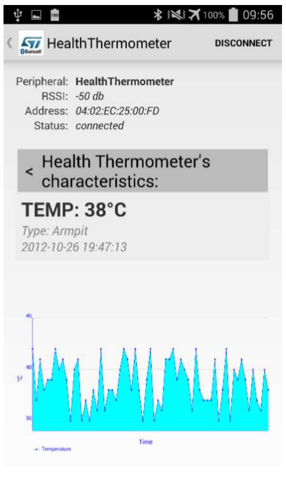
HealthThermometer DISCONNECT

Peripheral: HealthThermometer
RSSI: -48 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Health Thermometer

To obtain temperature measurements, the user should select "Health Thermometer" Service. The following diagram shows health thermometer characteristics being obtained from the HTM sensor. The user can read the instantaneous temperature value and the graph displaying the latest temperature values.

Figure 16. Screenshot of profiles app paired with HTS device and being notified



Peripheral: HealthThermometer
RSSI: -50 db
Address: 04:02:EC:25:00:FD
Status: connected

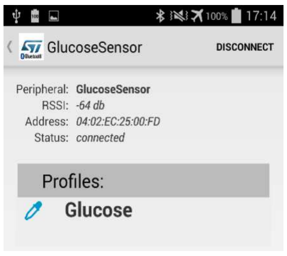
Health Thermometer's characteristics:

TEMP: 38°C
Type: Armpit
2012-10-26 19:47:13

1.8.7 Reading GL Data on smartphone

The following screenshot shows the successful pairing with a glucose sensor:

Figure 17. Screenshot of profiles app paired with GL device

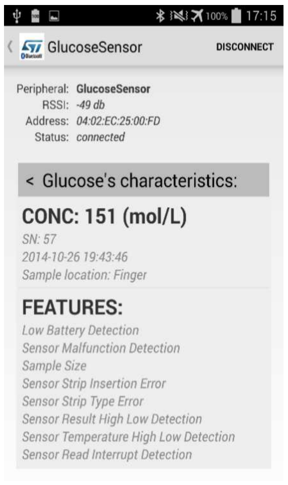


Peripheral: GlucoseSensor
RSSI: -64 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Glucose

To obtain glucose related measurements, select the "Glucose" Service. The following diagram shows glucose characteristics (e.g., glucose concentration, timestamp, sensor location, etc…) being obtained from the GL sensor.

Figure 18. Screenshot of profiles app paired with GL device and being notified



Peripheral: GlucoseSensor
RSSI: -49 db
Address: 04:02:EC:25:00:FD
Status: connected

< Glucose's characteristics:

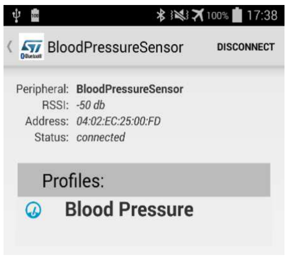
CONC: 151 (mol/L)
SN: 57
2014-10-26 19:43:46
Sample location: Finger

FEATURES:
Low Battery Detection
Sensor Malfunction Detection
Sample Size
Sensor Strip Insertion Error
Sensor Strip Type Error
Sensor Result High Low Detection
Sensor Temperature High Low Detection
Sensor Read Interrupt Detection

1.8.8 Reading BLP Data on smartphone

The following screenshot shows the successful pairing with a blood pressure sensor:

Figure 19. Screenshot of profiles app paired with BLP device

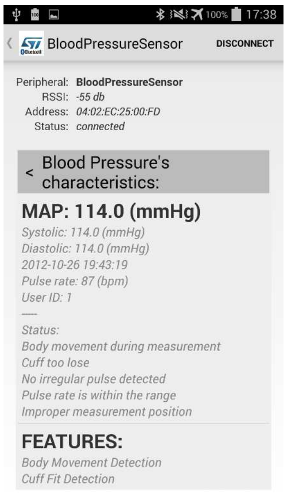


Peripheral: BloodPressureSensor
RSSI: -50 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Blood Pressure

To obtain blood pressure data, select the "Blood Pressure" Service. The following diagram shows blood pressure characteristics (e.g., mean arterial pressure, systolic pressure, diastolic pressure, timestamp, pulse rate, etc…) being obtained from the BLP sensor.

Figure 20. Screenshot of profiles app paired with BLP device and being notified


17:38
BloodPressureSensor DISCONNECT

Peripheral: BloodPressureSensor
RSSI: -55 db
Address: 04:02:EC:25:00:FD
Status: connected

< Blood Pressure's characteristics:

MAP: 114.0 (mmHg)
Systolic: 114.0 (mmHg)
Diastolic: 114.0 (mmHg)
2012-10-26 19:43:19
Pulse rate: 87 (bpm)
User ID: 1

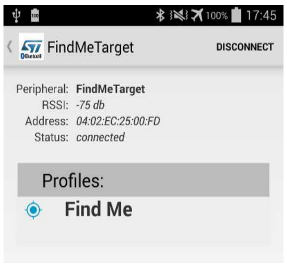
Status:
Body movement during measurement
Cuff too lose
No irregular pulse detected
Pulse rate is within the range
Improper measurement position

FEATURES:
Body Movement Detection
Cuff Fit Detection

1.8.9 Reading FMT Data on smartphone

The following screenshot shows the successful pairing with a Find Me Target device:

Figure 21. Screenshot of profiles app paired with FMT device

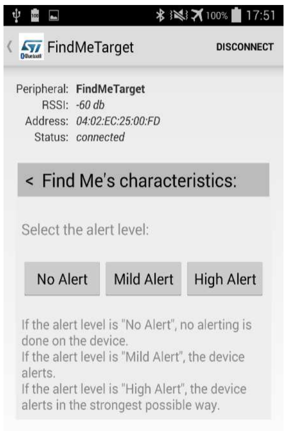

17:45
FindMeTarget DISCONNECT

Peripheral: FindMeTarget
RSSI: -75 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Find Me

After selecting the "Find Me" Service, the user can generate an alert signal on the remote device by pressing the respective button shown in the diagram below.

Figure 22. Screenshot of profiles app paired with FMT device and respective actions


17:51
FindMeTarget DISCONNECT

Peripheral: FindMeTarget
RSSI: -60 db
Address: 04:02:EC:25:00:FD
Status: connected

< Find Me's characteristics:

Select the alert level:

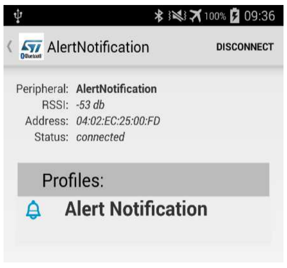
No Alert | Mild Alert | High Alert

If the alert level is "No Alert", no alerting is done on the device.
If the alert level is "Mild Alert", the device alerts.
If the alert level is "High Alert", the device alerts in the strongest possible way.

1.8.10 Reading ANS Data on smartphone

The following screenshot shows the successful pairing with an Alert Notification Server device:

Figure 23. Screenshot of profiles app paired with ANS device

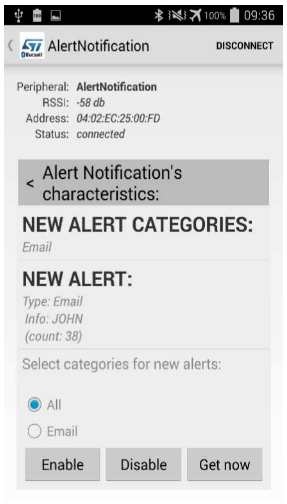


Peripheral: AlertNotification
RSSI: -53 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Alert Notification

To access the actions that can be performed on the ANS device, select the "Alert Notification" Service. The following diagram shows the list of alert categories provided by the ANS, the alert details and the options that can be selected for notification regarding a specific set of alert categories. In this case, the ANS provides the Email alert category.

Figure 24. Screenshot of profiles app paired with ANS device and respective actions



Peripheral: AlertNotification
RSSI: -58 db
Address: 04:02:EC:25:00:FD
Status: connected

Alert Notification's characteristics:

NEW ALERT CATEGORIES:
Email

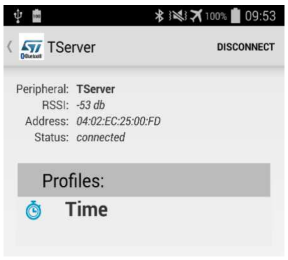
NEW ALERT:
Type: Email
Info: JOHN
(count: 38)

Select categories for new alerts:
[✓ All]
Email

1.8.11 Reading TS Data on smartphone

The following screenshot shows the successful pairing with a Time Server device:

Figure 25. Screenshot of profiles app paired with TS device

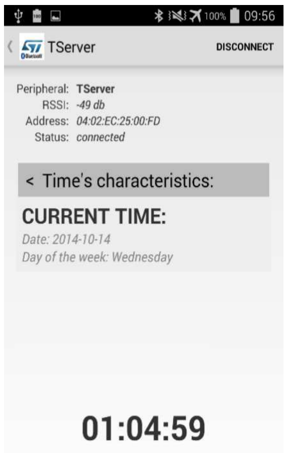


Peripheral: TServer
RSSI: -53 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Time

To obtain the current time, select "Time" Service. The following diagram shows time characteristics being obtained from the Time Server.

Figure 26. Screenshot of profiles app paired with TS device and being notified



Peripheral: TServer
RSSI: -49 db
Address: 04:02:EC:25:00:FD
Status: connected

< Time's characteristics:

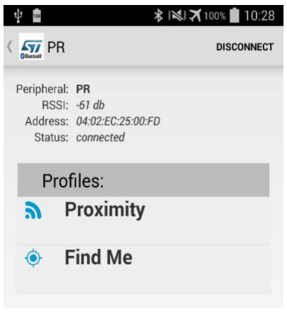
CURRENT TIME:
Date: 2014-10-14
Day of the week: Wednesday

01:04:59

1.8.12 Reading PXP Data on smartphone

The following screenshot shows the successful pairing with a Proximity Reporter device:

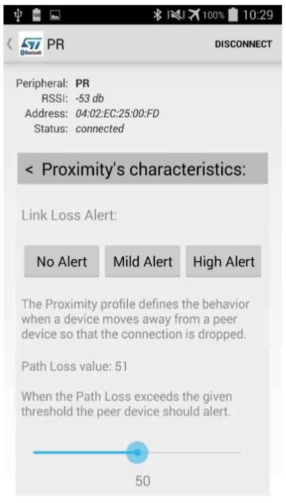
Figure 27. Screenshot of profiles app paired with PXR device



Peripheral: PR
RSSI: -61 db
Address: 04:02:EC:25:00:FD
Status: connected

In this case, two services are exposed: "Find Me" (see *Section 1.8.9*) and "Proximity". After selecting "Proximity" Service, the user can set the alert level to be generated by the remote in case of disconnection or link loss events. Furthermore, the user can set the path loss threshold: if the path loss exceeds this level, an alert signal is generated by the remote.

Figure 28. Screenshot of profiles app paired with PXR device and respective actions


PR DISCONNECT

Peripheral: PR
RSSI: -53 db
Address: 04:02:EC:25:00:FD
Status: connected

< Proximity's characteristics:

Link Loss Alert:

No Alert | Mild Alert | High Alert

The Proximity profile defines the behavior when a device moves away from a peer device so that the connection is dropped.

Path Loss value: 51

When the Path Loss exceeds the given threshold the peer device should alert.

50

2 System setup guide

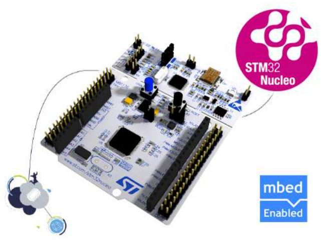
2.1 Hardware requirements

This chapter describes the hardware components needed for developing a sensor-based application. The following sub-sections describe the individual components.

2.1.1 STM32 Nucleo platform

The STM32 Nucleo boards provide an affordable and flexible way for users to try out new ideas and build prototypes with any of the STM32 microcontroller lines. The Arduino™ connectivity support and ST Morpho headers make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards. The STM32 Nucleo board does not require any separate probes as it integrates the ST-LINK/V2-1 debugger/programmer. The STM32 Nucleo board comes with the STM32 comprehensive software HAL library, together with various packaged software examples.

Figure 29. STM32 nucleo board



2.1.2 X-NUCLEO-IDB04A1 STM32 expansion board

The X-NUCLEO-IDB04A1 is a Bluetooth Low Energy evaluation board to allow expansion of the STM32 Nucleo boards. It is compatible with the Arduino UNO R3 connector layout and is designed around BlueNRG, a Bluetooth Low Energy, low power network coprocessor compliant with BTLE 4.0. The X-NUCLEO-IDB04A1 interfaces with the STM32 MCU via an SPI pin and the user can change the default SPI clock, SPI chip select and SPI IRQ by changing a resistor on the evaluation board

Figure 30. X-NUCLEO-IDB04A1 STM32 expansion board

2.1.3 Android Smartphone

- Android version equal to or greater than 4.3
- Bluetooth Low Energy support

2.2 Software requirements

2.2.1 Software application

The BLE Profiles application can be found in the OSXSmartConnPS package (version 1.0), available from st.com at
http://www.st.com/web/catalog/tools/FM147/CL2116/SC2023/PF261620

2.2.2 Android app

The Android app "STM32 BLE Toolbox" running on phone is included inside the package in the "$BASE_DIR\Utilities\Android_Software\Profiles" folder (the .apk file for installation is STM32_BLE_Toolbox.apk).

3 Acronyms and abbreviations

Table 1. Acronyms

Acronym	Description
ACI	Application Controller Interface
ANS	Alert Notification Server
ATT	Attribute Protocol
BLE	Bluetooth Low Energy
BLS	Blood Pressure Sensor
BSP	Board Support Package
BT	Bluetooth
FMT	Find Me Target
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GLS	Glucose Sensor
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
HCI	Host Controller Interface
HRS	Heart Rate Sensor
IDE	Integrated Development Environment
L2CAP	Logical Link Control and Adaptation Protocol
LED	Light Emitting Diode
LL	Link Layer
LPM	Low Power Manager
MCU	Micro Controller Unit
PCI	Profile Command Interface
PXP	Proximity Profile
PXR	Proximity Reporter
PHY	Physical Layer
SIG	Special Interest Group
SM	Security Manager
SPI	Serial Peripheral Interface
TIP	Time Profile
TS	Time Server
UUID	Universally Unique Identifier

4 References

[1] UM1755: BlueNRG Bluetooth LE stack application command interface (ACI).

[2] UM1686: BlueNRG development kits

[3] AN4559: Developer's Guide to create Bluetooth® Low Energy Applications using STM32 Nucleo and BlueNRG

[4] HRP_SPEC: Heart Rate Profile, Bluetooth profile specification, HRP_V10.pdf, available from https://developer.bluetooth.org/TechnologyOverview/Pages/HRP.aspx

[5] DM00105960: STM32L053C6 STM32L053C8, STM32L053R6 STM32L053R8 Datasheet

5 Revision history

Table 2. Document revision history

Date	Revision	Changes
18-Mar-2015	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2015 STMicroelectronics – All rights reserved