	***STMicroelectronics***
APPLICATION NOTE	JULY 7, 2014

Heart Rate Sensor

Application Note

Table of Contents

1 Table of Figures

2 References

[1] "BlueNRG Bluetooth LE stack application command interface (ACI)", User manual UM1755, available from st.com at
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00114498.pdf

[2] "BlueNRG development kits", User manual UM1686, available from st.com at
http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00099259.pdf

[3] "Developer's Guide to create Bluetooth® Low Energy Applications using STM32 Nucleo and BlueNRG", STM32_Nucleo_BlueNRG_Application_Note.pdf

[4] Heart Rate Profile, Bluetooth Profile Specification, HRP_V10.pdf

[5] "STM32L0 series datasheet", available from st.com at
http://www.st.com/web/en/resource/technical/document/datasheet/DM00105960.pdf

3 Acronyms and Abbreviations

Acronym	Description
ACI	Application Controller Interface
ATT	Attribute Protocol
BLE	Bluetooth Low Energy
BLE	Bluetooth low energy
BSP	Board support package
BT	Bluetooth
GAP	Generic access profile
GATT	Generic attribute profile
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
PHY	Physical layer
SIG	Special interest group
SM	Security manager
SPI	Serial Peripheral Interface
UUID	Universally unique identifier

4 Overview

This document describes the Heart Rate Monitor Sensor application running over STM32Nucleo L0 board connected with a BlueNRG expansion board. This application is included as part of the STM32Cube firmware for the X-NUCLEO-IDB04A1 expansion board. It is assumed that the reader has prior knowledge on Bluetooth Low Energy protocol and its application profiles like the Heart Rate Profile. Heart rate monitoring application uses dummy value of heart rate measurement, as real heart rate sensor is not present on the board.

The Heart Rate Sensor application is based on the standard GATT-based Heart Rate Profile (https://developer.bluetooth.org/TechnologyOverview/Pages/HRP.aspx) described in the Bluetooth SIG specifications.

Please refer to [4] for getting started information and for understanding nuances of BLE application development on STM32 Nucleo board equipped with a BlueNRG expansion board.

4.1 Requirements

This section describes the hardware and software components needed to get started.

4.1.1 Hardware Requirements

- STM32Nucleo L0 development board
- X-NUCLEO-IDB04A1, a BlueNRG expansion board for STM32 Nucleo development boards
- An Android Smartphone having:
  - android version equal to or greater than 4.3
  - Bluetooth low energy support

4.1.2 Software Requirements

- Software Application available in the STSW-IDB04A1 package (from version 1.1), available from st.com at http://www.st.com/web/en/catalog/tools/PF260705
- Android app running on phone, included inside the package in the "$BASE_DIR\Utilities\Android_Software\HRM_Central" folder (the .apk file for installation is STM32_BLE_Toolbox.apk).

4.2 Application overview

The STM32 device acts as the GAP peripheral (Heart Rate Sensor) device running the sensor application. It sends heart rate measurement data to the GAP central device (The Collector), which is an android smartphone in this case.

The Heart Rate Monitor (HRM) Application typically runs the Heart Rate Profile as described in the Bluetooth Profile Specifications. It exposes the Heart Rate Service and enables the role

of the Heart Rate Sensor in the STM32 device. The application sends periodic data which is simulated heart rate measurements as heart rate measurement sensor is not present on Nucleo board.

Any central device running the Heart Rate Profile will be able to collect data from the Heart Rate Monitor.

The HRS device initializes the Heart Rate Profile and then starts advertising its address and services. The Collector device will scan for HRS device, and once it is discovered it sends a connection request to the HRS device. After establishing the connection the HRS device will start sending data over the connection link.

5 Software Architecture Overview

5.1 Software blocks

This chapter describes the various software layers present in an application based on a standard BLE profile, running on a STM32 Nucleo development board with the STM32Cube software environment. Important layers are:

- STM32Cube HAL Layer
- Board Support Package (BSP) Layer
- BlueNRG HAL Layer
- Profile Command Interface (PCI) Layer
- GATT Profile
- BLE Profile

The software layers are shown in the following diagram. As shown in the diagram, PCI and GATT both rely on the HCI layer for sending BLE commands to BlueNRG expansion board. Generic Attribute (GATT) profile uses both PCI API and HCI for sending BLE commands to BlueNRG expansion board.

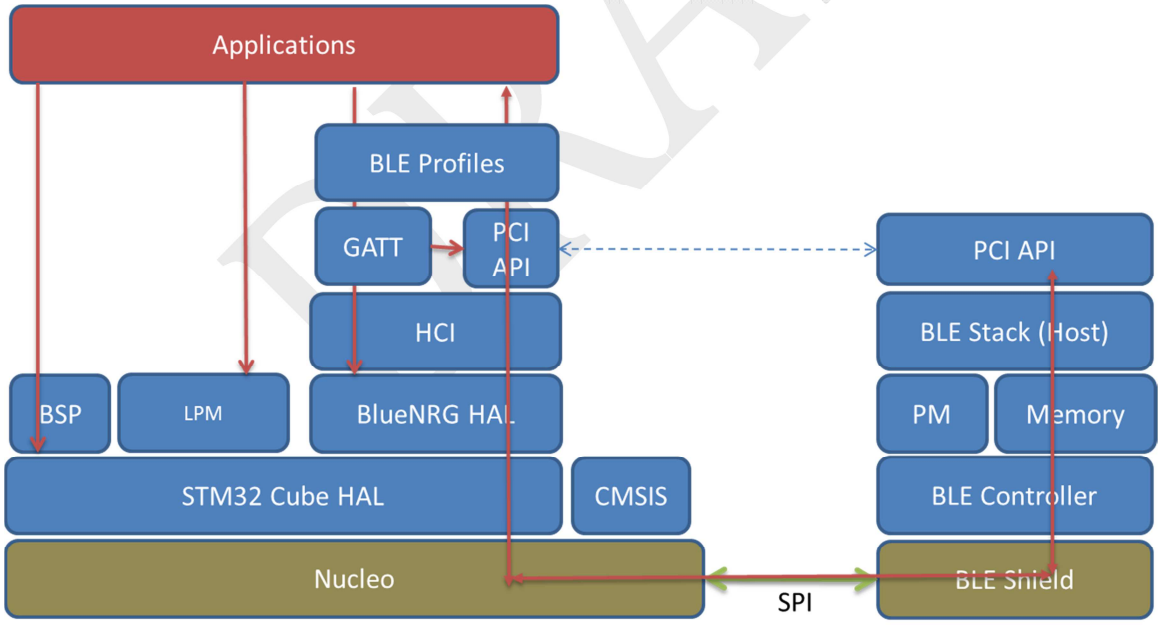


Figure 1 HRM Application SW Architecture

The software layers that are important from the application point of view are described in brief below.

5.1.1 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 expansion board.

5.1.2 Low Power Manager (LPM)

The Low Power Manager provides the possibility for the application to allow the system to go 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 inform the system about the lowest power mode it can go to
- Informing the system that it has no more requirements of power
- Callbacks for 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:

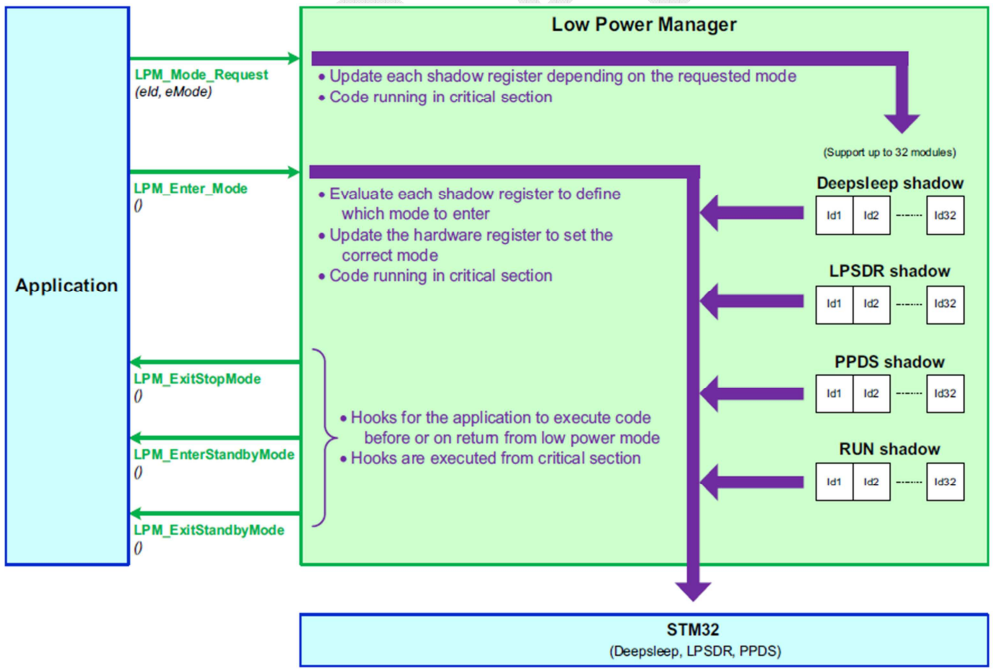


Figure 2: Overview of the Low Power Manager

The low power manager provides a simple interface to receive the input of different users and computes the lowest power mode the system may enter. It provides as well hooks to the application before entering or on exit of low power mode.

5.1.2.1 Low Power Manager API

The Low Power Manager provides the following APIs that can be used by the applications for putting the system in specific low power modes:

- LPM_Mode_Request(): This API can be used by applications to specify the lowest power mode supported by it.
- LPM_Enter_Mode():This API can be used by the application to request the system to enter low power mode.
- LPM_ExitStopMode(): This API is called by the LPM in a critical section to allow the application to implement dedicated code before getting out from the critical section.
- LPM_Enter_StandbyMode(): This API is called by the LPM in a critical section to allow the application to implement dedicated code before entering standby mode.

5.1.3 BLE Profiles

The BLE Profile layer implements the Bluetooth low energy profiles which are used by the applications. The BLE Profile layer uses the PCI for communicating with the controller. It defines the necessary functions and callbacks for the applications to communicate with the profiles.

5.1.4 BSP Layer

The BSP software layer supports the peripherals on the STM32 Nucleo board apart from the MCU. This is a limited set of APIs which provides a programming interface for certain board specific peripherals, e.g. the LED, the user button etc. This interface also helps in identifying the specific board version. Please refer [3] to see how BSP layer is used and initialized.

6 HRS Application Overview

6.1 HRS application files

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

- $BASE_DIR\Projects\STM32L053R8-Nucleo\Applications\Bluetooth_LE\HRM_LowPowerRTC\Project\Srcmain.c: This file contains source code to perform the STM32 Nucleo board initialization and calls to the HRM profile related functionality defined in hrm_profile_application.c
- $BASE_DIR\Projects\STM32L053R8-Nucleo\Applications\Bluetooth_LE\HRM_LowPowerRTC\Project\Src\hrm_profile_application.c: This file contains source code for application event handling, profile initialization, advertisement, and application state machine management.

6.2 Initialization

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

6.2.1 Generic initialization

This refers to initializing various hardware blocks present on the STM32 Nucleo board. Applications must initialize these blocks properly before using them. Please refer to chapter 7.1.1 of [4] for having a detailed discussion on generic initialization.

6.2.2 BLE Profile Initialization

Application must initialize the BLE stack and BLE profile that it intends to use. This initialization is done by using the following two APIs:

- BLE_Profile_Init(): This API performs generic initialization for BLE profiles.
- Profile specific initialization: BLE application must initialize the individual profile that it intends to use, the PCI layer provides API for doing this initialization. In case of HRS application, the Heart rate profile is initialized by using the HRProfile_Init() API.

6.3 HRS Application State machine

The following diagram shows the HRS application state machine. When the Nucleo board is turned on, the HRS application is in APPL_UNINITIALIZED state. In this state the application initializes the hardware and the BLE heart rate profile, as discussed in the previous section. Once the initialization is complete, the HRS application receives the HRS_EVT_INITIALIZED event and enters APPL_INIT_DONE state. Once APPL_INIT_DONE state is reached the application starts advertising its characteristics so that the central device can discover it. After discovering peripheral device, the central device may choose to connect to it. Once connection is complete the peripheral device enters APPL_CONNECTED state. From APPL_CONNECTED state, the peripheral device will enter APPL_SEND_HEART_RATE_MEASUREMENT state when an interrupt (timer or button press) is received.

In APPL_SEND_HEART_RATE_MEASUREMENT state, the heart rate measurement is sent to the central device. After sending the heart rate measurement the peripheral device enters APPL_WAIT state and waits for the interrupt (timer or button press).

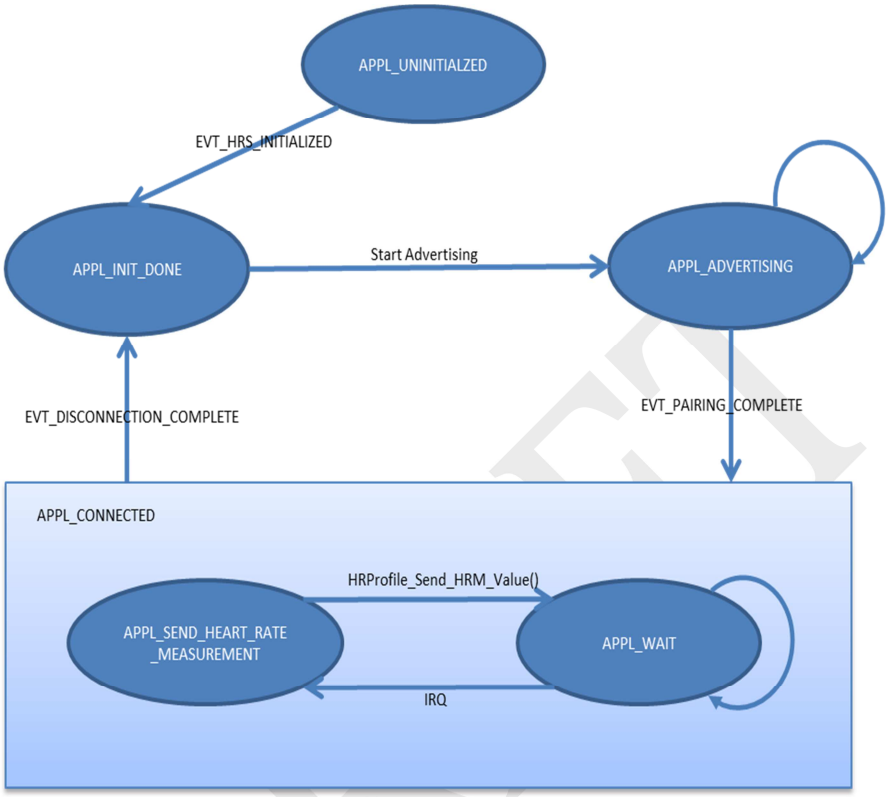


Figure 3: HRM Application State Machine

6.4 Processor Power Modes

The HRS application leverages the advanced STM32L0 processor low-power features.

Two MCU modes are used by the application:

- RUN mode

  This is the standard execution mode in which all the MCU features are available.

- STOP mode with RTC

  This is the mode that 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 the low-power mode. The device can be woken up

from Stop mode by any of the EXTI line, in 3.5 µs, the processor can serve the interrupt or resume the code.

The low power manager is used to exploit low power features of STM32L0 CPU. Please refer to section 5.1.2 for details of low power manager API. For further information about the STM32L0 low power modes, please refer to [4].

6.5 Event Handling

A number of events are generated by the underlying BLE stack during the lifetime of HRS application. This section describes these events in detail.

6.5.1 Generic events

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

- EVT_BLUE_INITIALIZED: This event is given to the application by the main profile when the controller has been initialized.
- EVT_CONNECTION_COMPLETE: This event is given to the application by the main profile when a connection has been successfully established with the peer.
- EVT_PASSKEY_REQUEST: This event is given 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_PAIRING_COMPLETE: This event is given to the application by the main profile when the device is successfully paired with the peer.
- EVT_DISCONNECTION_COMPLETE: This event is given to the application by the main profile to notify the result of a disconnection procedure started either by master or slave.
- EVT_ADVERTISING_TIMEOUT: This event is given by child profiles when the limited discoverable mode times out or the profile specific advertising timeout happens. It will be application's responsibility to restart the advertising.

6.5.2 Heart Rate Profile events

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

- EVT_HRS_INITIALIZED: This event is given 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: This event is given 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: This event is given 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.

6.6 Running the HRS Application

This section demonstrates how to run the HRM application on the STM32 Nucleo connected with a BlueNRG expansion board and then using it from an android application running on a smartphone.

6.6.1 Hardware Setup

In subsequent sections, the Nucleo board equipped with BlueNRG expansion board will be referred to as "peripheral device" and the Smartphone as the "central device".

To setup the hardware please refer to section 4.1.1 for details. Connect the two boards and power the system up using a USB type A to Mini-B USB cable by connecting the Nucleo with a PC USB slot.

6.6.2 Software setup on Nucleo

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

The binary files for HRS application can be found in the following folder:

"$BASE_DIR\Projects\STM32L053R8-Nucleo\Applications\Bluetooth_LE\HRM_LowPowerRTC\Project\\STM32L0xx_EWARM\STM32L053R8_NUCLEO\Exe\"

The IAR project file for HRS application can be found in the following folder:

"$BASE_DIR\Projects\STM32L053R8-Nucleo\Applications\Bluetooth_LE\HRM_LowPowerRTC\Project\\STM32L0xx_EWARM\Profiles.eww"

This project was built using IAR version 7.20.

6.6.3 Starting the Nucleo and app

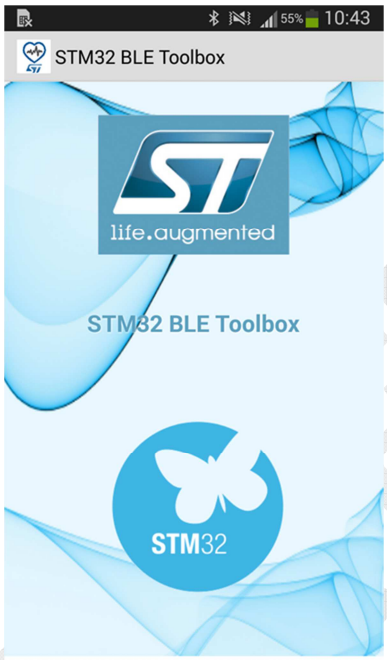
Reset the Nucleo board to start running the heart rate monitoring application. When the application is running, the LED present on Nucleo board will start blinking. Please note that the HRS FW should be loaded on the board before performing this step.

6.6.4 Software setup on the phone

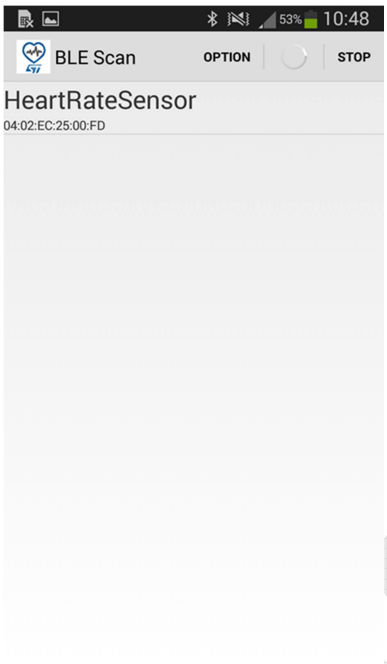
Retrieve the ST HRM app from the software package (see section 4.1.2) and install it on the Android smartphone.

6.6.5 Starting the phone App and connecting to central device

The following diagram shows the main page of the android application, automatic scan procedure can be started by tapping on “STM32” circle on the application. The scanning can be started/stopped by hitting the “Start/Stop” button.

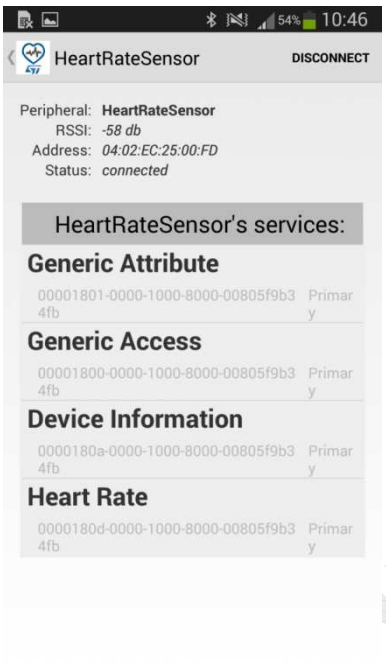


The following diagram shows the application screenshot when the heart rate sensor is detected by the application:



6.6.6 Reading HRM Data on smartphone

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



To obtain heart rate measurements, the user should first select “HeartRateSensor’s Services”, and then enable notifications by pressing “Notif/indic” button. The following diagram shows heart rate measurements being obtained from the heart rate sensor.