

A decorative wavy line that spans the width of the page, starting with a red-to-orange gradient on the left and transitioning through yellow, green, and cyan to a blue-to-purple gradient on the right.

μ Energy™

Heart Rate Sensor

Application Note

Issue 3

Document History

Revision	Date	History
1	15 JUN 12	Original publication of this document
2	16 JUN 12	Corruption to Table 7.1 corrected
3	13 SEP 12	Figure 4.1 and Figure 4.2 updated

Contacts

General information
 Information on this product
 Customer support for this product
 More detail on compliance and standards
 Help with this document

www.csr.com
sales@csr.com
www.csrsupport.com
product.compliance@csr.com
comments@csr.com

Trademarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc and/or its affiliates.

Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR.

Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc or its affiliates.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.

Contents

Document History.....	2
Contacts.....	2
Trademarks, Patents and Licences.....	2
Life Support Policy and Use in Safety-critical Compliance.....	2
Performance and Conformance.....	2
Contents.....	3
Tables, Figures and Equations.....	3
1. Introduction.....	5
1.1. Application Overview.....	5
2. Using the Application.....	8
2.1. Demonstration Kit.....	8
2.2. Demonstration Procedure.....	11
3. Application Structure.....	16
3.1. Source Files.....	16
3.2. Header Files.....	16
3.3. Database Files.....	17
4. Code Overview.....	19
4.1. Application Entry points.....	19
4.2. Internal State Machine.....	21
5. NVM Memory Map.....	25
6. Customising the Application.....	26
6.1. Actual Measurement Mode.....	26
6.2. Advertising Parameters.....	26
6.3. Advertisement Timers.....	26
6.4. Connection Parameters.....	27
6.5. Idle Connected Timeout.....	27
6.6. Connection Parameter Update.....	27
6.7. Device Name.....	27
6.8. Buzzer.....	28
7. Current Consumption.....	29
Appendix A Definitions.....	30
Appendix B GATT Database.....	30
Appendix C Advertising/Scan Response Data.....	33
Appendix D Known Issues or Limitations.....	34
Document References.....	35
Terms and Definitions.....	36

Tables, Figures and Equations

Table 1.1: Heart Rate Profile Roles.....	5
Table 1.2: Application Topology.....	6
Table 1.3: Responsibilities.....	6
Table 1.4: Measurement Modes.....	7

Table 2.1: Demonstration Components.....	8
Table 3.1: Source Files	16
Table 3.2: Header Files	17
Table 3.3: Database Files	18
Table 5.1: NVM Memory Map for Application.....	25
Table 5.2: NVM Memory Map for GAP Service.....	25
Table 5.3: NVM Memory Map for Heart Rate Service	25
Table 5.4: NVM Memory Map for Battery Service	25
Table 6.1: PIO for External Heart Rate Input	26
Table 6.2: Advertising parameters	26
Table 6.3: Advertisement Timers	26
Table 6.4: Connection Parameters.....	27
Table 6.5: Idle Connected Timeout	27
Table 7.1: Current Consumption Values	29
Table A.1: Definitions	30
Table B.1: Battery service characteristics	30
Table B.2: Device information service characteristics	31
Table B.3: GAP Service Characteristics.....	31
Table B.4: Heart Rate Service Characteristics	32
Table C.1: Advertising Data Fields.....	33
Figure 1.1: Heart Rate Profile Use Case	5
Figure 1.2: Primary Services	7
Figure 2.1: CSR1000 Development Board	8
Figure 2.2: Device Behaviour (Sample Measurement Mode).....	9
Figure 2.3: Device Behaviour (Actual Measurement Mode).....	10
Figure 2.4: CSR8510 USB Dongle.....	11
Figure 2.5: μ Energy Profile Demonstrator	11
Figure 2.6: Heart Rate Sensor Device Discovered.....	12
Figure 2.7: Device Connected.....	13
Figure 2.8: Battery Level Tab	14
Figure 2.9: Device Information Service Tab	15
Figure 4.1: Internal State Diagram (Sample Measurement Mode)	21
Figure 4.2: Internal State Diagram (Actual Measurement Mode)	22
Figure B.1: Heart Rate Measurement Data Format.....	32

1. Introduction

This document describes the Heart Rate Sensor application supplied with the CSR μEnergy™ Software Development kit (SDK) and provides guidance on how to customise the on-chip application. This application demonstrates the Heart Rate profile which is specified by the Bluetooth SIG.

1.1. Application Overview

1.1.1. Profiles Supported

The Heart Rate Sensor application supports the following profile:

1.1.1.1. Heart Rate Profile

The Heart Rate profile is used to enable a data collection device to obtain Heart Rate measurements from a Heart Rate Sensor that exposes the Heart Rate service.

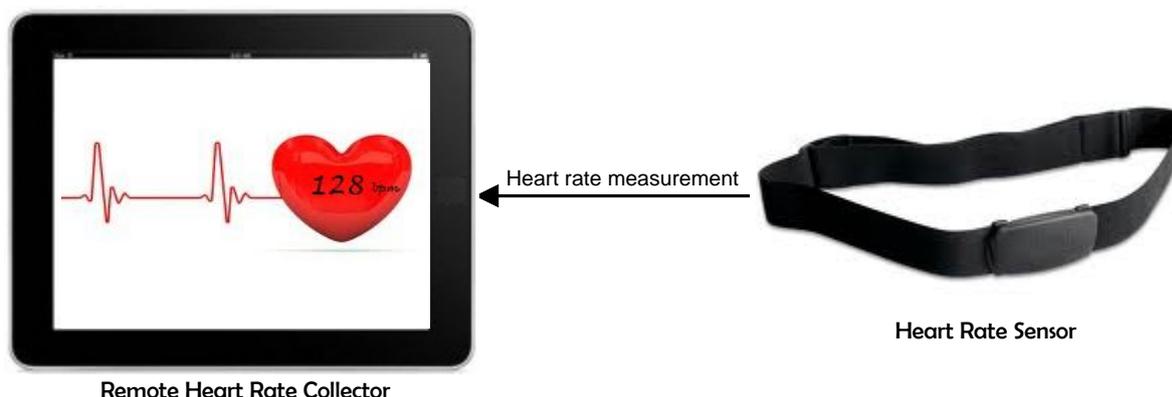


Figure 1.1: Heart Rate Profile Use Case

The Heart Rate profile defines two roles, see Table 1.1:

Role	Description
Heart Rate Sensor	Heart Rate Sensor is the device that measures heart rate and related information.
Heart Rate Collector	Heart Rate Collector is the device that receives heart rate measurement and related data from the sensor.

Table 1.1: Heart Rate Profile Roles

For more information about the Heart Rate profile, see *Heart Rate Profile Specification Version 1.0*.

1.1.2. Application Topology

The Heart Rate Sensor application implements the Heart Rate profile in Heart Rate Sensor role, see Table 1.2:

Role	Heart Rate Profile	GAP Service	GATT Service	Device Information Service	Battery Service
GATT Role	GATT Server	GATT Server	GATT Server	GATT Server	GATT Server
GAP Role	Peripheral	Peripheral	Peripheral	Peripheral	Peripheral

Table 1.2: Application Topology

Role	Description
GATT Server	It accepts incoming commands and requests from the client and sends responses, indications and notifications to the client.
GAP Peripheral	It accepts connection request from the remote device and acts as a slave in the connection.

Table 1.3: Responsibilities

For more information about GATT server and GAP peripheral, see *Bluetooth Core Specification Version 4.0*

1.1.3. Services

This application exposes the following services:

- Heart Rate (Version 1.0)
- Device Information (Version 1.1)
- Battery (Version 1.0)
- GAP
- GATT

The Heart Rate profile mandates only two services: Heart Rate and Device Information. GAP and GATT services are mandated by *Bluetooth Core Specification Version 4.0*. Battery service is optional as shown in Figure 1.2.

For more information on Heart Rate, Device information and Battery services, see the *Heart Rate Service Specification Version 1.0*, *Device Information Specification Version 1.1* and *Battery Service Specification Version 1.0* respectively. For more information on GATT and GAP services, see *Bluetooth Core Specification Version 4.0*.

See Appendix B for information on characteristics supported by each service.

Note:

The Heart Rate Sensor application does not support any characteristic for GATT service. See *Bluetooth Core Specification Version 4.0* for more information.

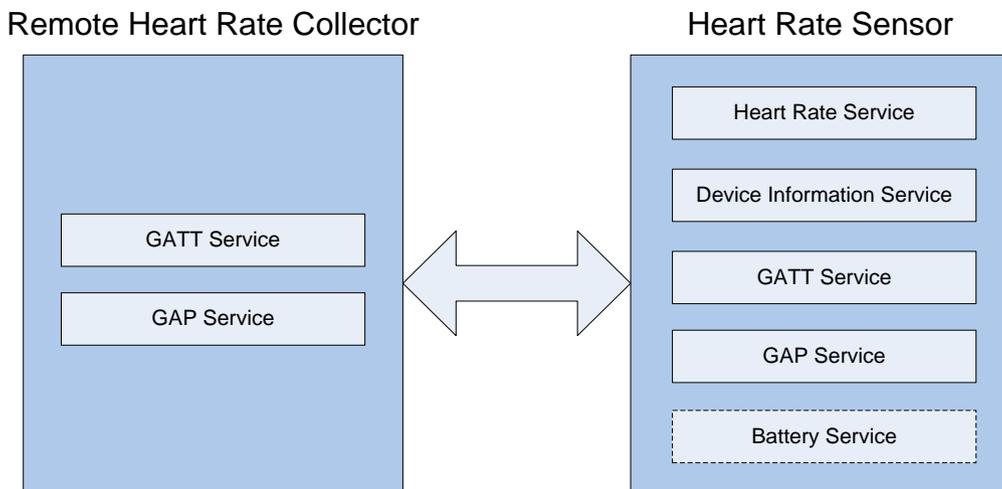


Figure 1.2: Primary Services

1.1.4. Measurement Modes

The Heart Rate Sensor application operates in two measurement modes, see Table 1.4:

Measurement Mode	Description
Sample Measurement Mode (Default Mode)	The Heart Rate Sensor application generates sample heart rate measurements at one second intervals in this mode.
Actual Measurement Mode	The Heart Rate Sensor application takes actual heart rate measurement as input via a PIO from an external Heart Rate device.

Table 1.4: Measurement Modes

To switch between these two modes, see section 6.1.

2. Using the Application

This section describes how the Heart Rate Sensor application can be used with the μEnergy Profile Demonstrator application available from CSR.

2.1. Demonstration Kit

The application can be demonstrated using the following components:

Component	Hardware	Application
Heart Rate Sensor	CSR1000 Development Board	Heart Rate Sensor Application v1.0
Heart Rate Collector	CSR8510 USB dongle	μEnergy Profile Demonstrator Application

Table 2.1: Demonstration Components

Note:

Although the Heart Rate Sensor application primarily targets the CSR1000 development board, the CSR1001 development board may also be used as an alternative hardware platform.

The μEnergy Profile Demonstrator application, CSR device drivers and user guides are available to download from www.csrsupport.com/uEnergy.

2.1.1. Heart Rate Sensor

The SDK is used to build and download the Heart Rate Sensor application to the CSR1000 development board. See the *μEnergy xIDE User Guide* for further information.

Ensure the development board is switched on using the Power On/Off switch. Figure 2.1 shows the switch in the Off position.

Note:

When disconnected from the USB to SPI Adapter, wait at least 1 minute before switching the board on. This allows any residual charge received from the SPI connector to be dissipated.



Figure 2.1: CSR1000 Development Board

2.1.1.1. User Interface

This application makes use of the button and buzzer available on the CSR1000 development board.

Button Press Behaviour

- In **Sample Measurement** mode, a **Short button press** disconnects the link if connected, otherwise the application starts advertising.
- In **Actual Measurement** mode, a **Short button press** is ignored by the application.
- An **Extra Long button press** disconnects the link if present, removes bonding and starts advertising.

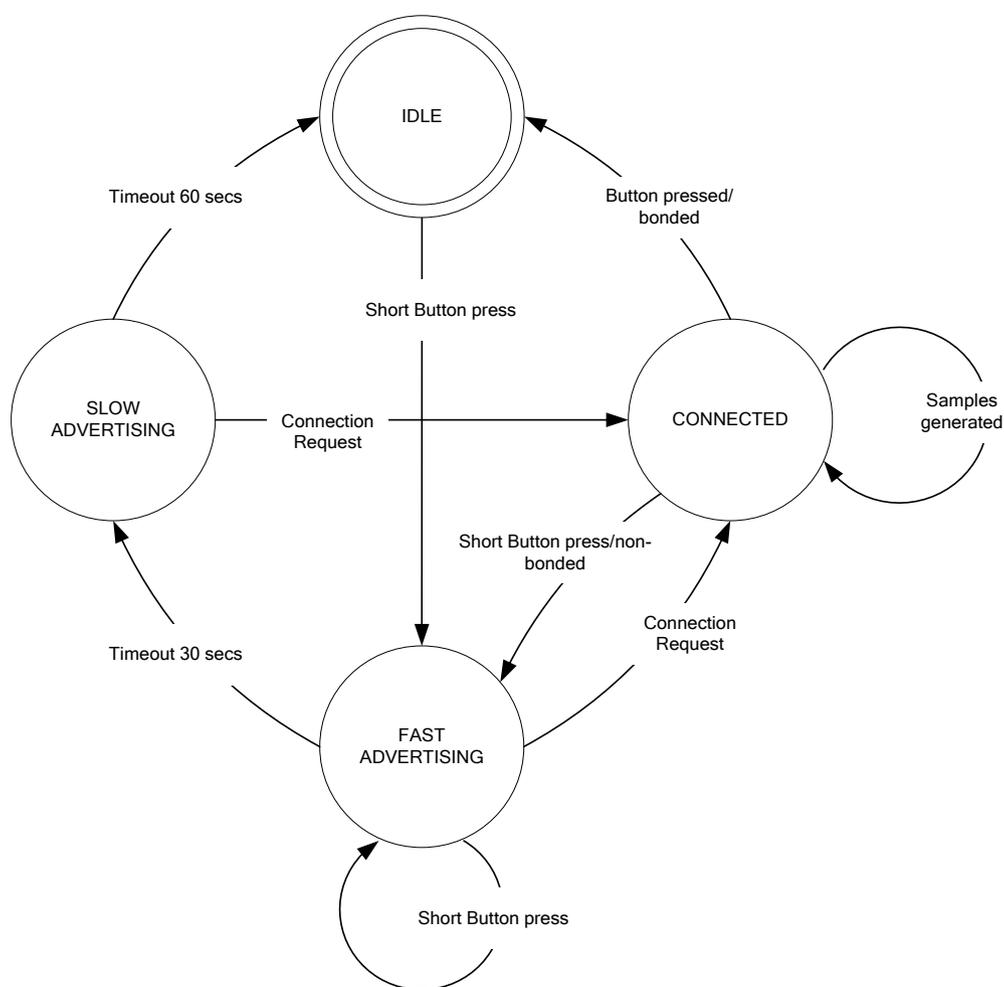


Figure 2.2: Device Behaviour (Sample Measurement Mode)

Note:

As the Heart Rate Sensor application does not perform any specific operation on a Long button press, it is handled in a similar way to a Short button press.

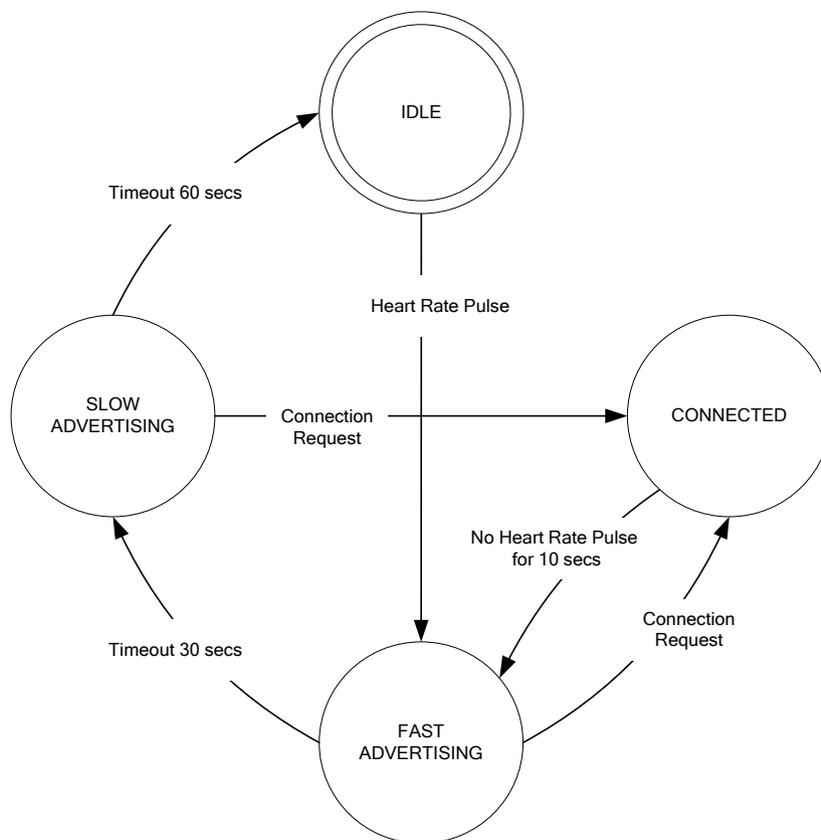


Figure 2.3: Device Behaviour (Actual Measurement Mode)

Buzzer Behaviour

- A single short beep on a **Short button press** indicates the execution of the **Short button press** operation as described earlier. This is applicable to the **Sample Measurement** mode only.
- Two short beeps indicate the start of advertisements.
- Three short beeps indicate the removal of bonding.
- A long beep without the button being pressed indicates that the application has entered idle mode.

2.1.2. Heart Rate Collector

2.1.2.1. CSR8510 USB Dongle

The CSR8510 USB dongle can be used with the µEnergy Profile Demonstrator application to complete the Bluetooth low energy link between two devices. To use the USB dongle shown in Figure 2.4, the default USB Bluetooth Windows device drivers must be replaced with the CSR BlueCore device drivers as described in the *USB Device Driver User Guide*.

CSR device drivers and user guides are available to download from www.csr.com/support.



Figure 2.4: CSR8510 USB Dongle

2.1.2.2. μEnergy Profile Demonstrator Application

The μEnergy Profile Demonstration application is compatible with a PC running Windows XP and Windows 7 (32 bit and 64-bit). Launch the application once the CSR8510 USB dongle is attached to the PC and drivers have been loaded.

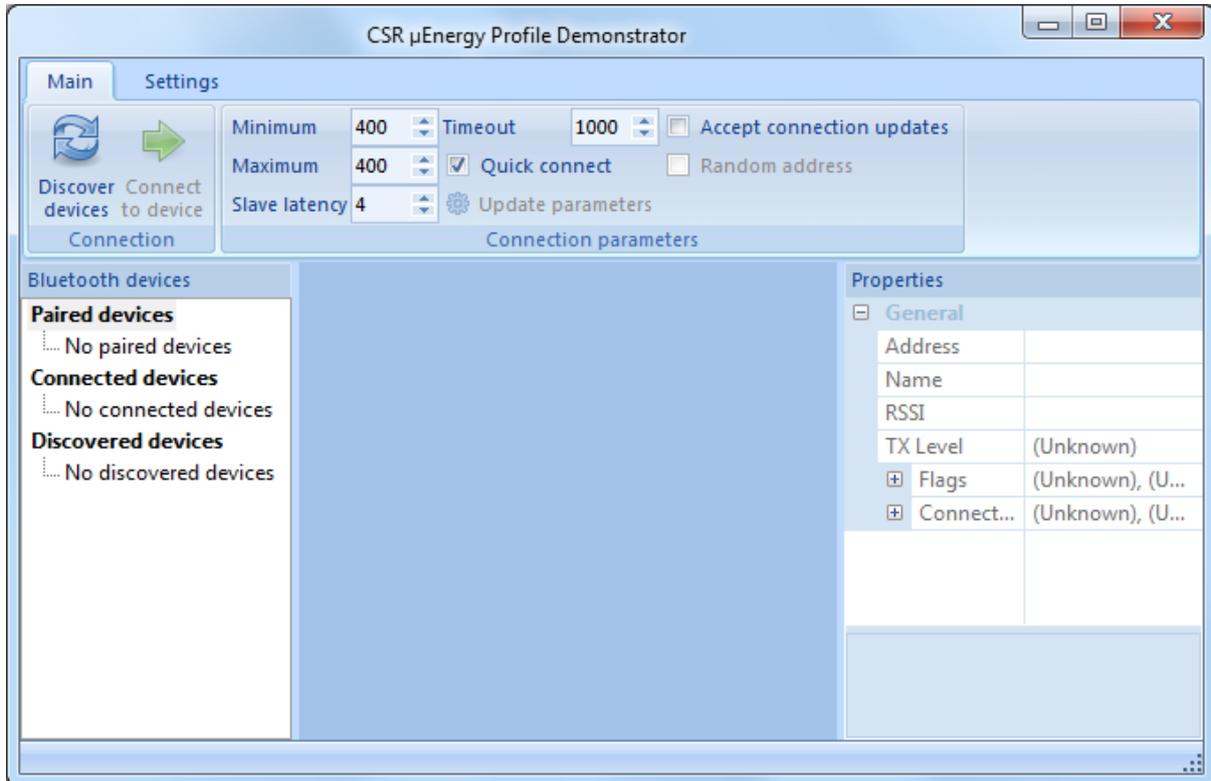


Figure 2.5: μEnergy Profile Demonstrator

2.2. Demonstration Procedure

1. Switch on the CSR1000 development board to trigger advertisements
2. Click on the **Discover devices** button in the **CSR μEnergy Profile Demonstrator** application window. The application searches for Bluetooth Low energy devices and lists all the discovered devices on the left hand side of the application window, see Figure 2.6.

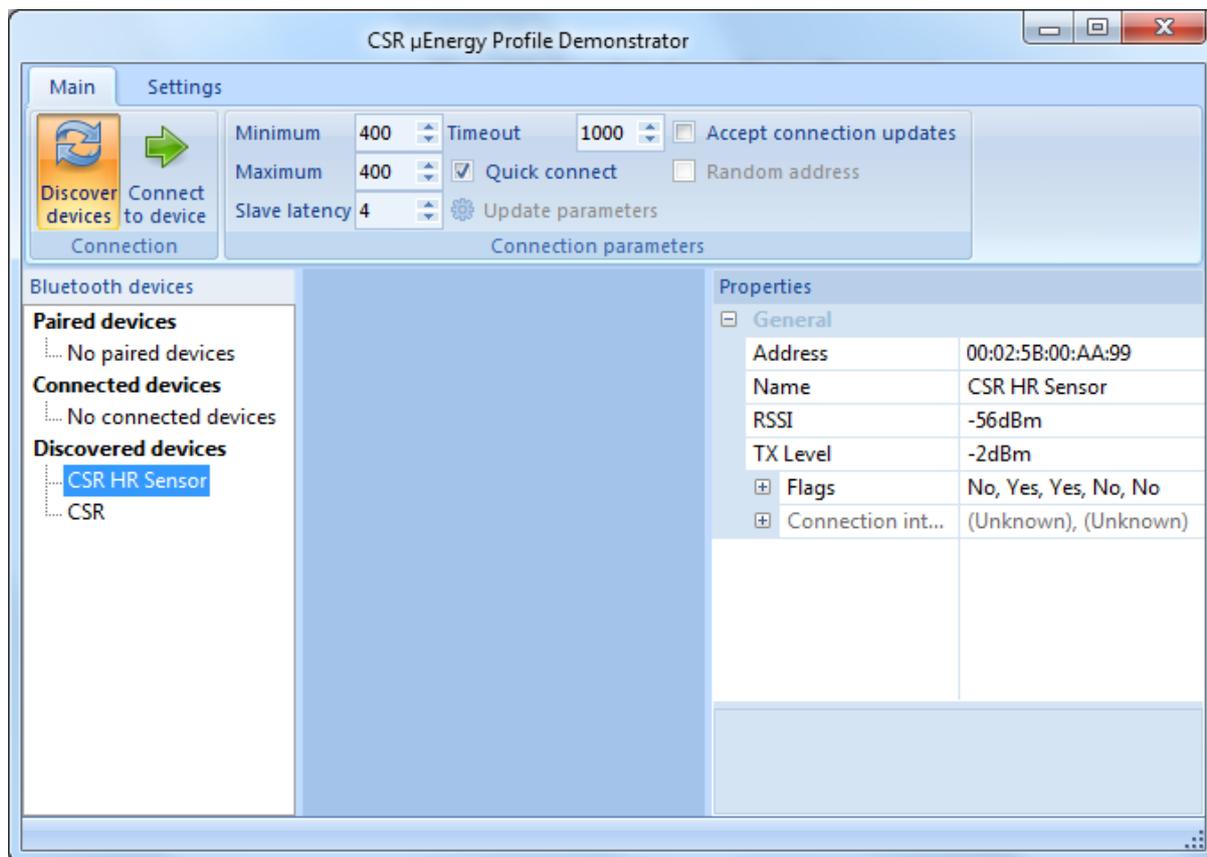


Figure 2.6: Heart Rate Sensor Device Discovered

1. When the device labelled **CSR HR Sensor** appears, select it to display the device address on the right hand side of the screen. Click on the **Connect to device** button to connect to this device, see. Figure 2.6. The μ Energy Profile Demonstrator application displays a tabbed pane corresponding to different services supported by the CSR1000 application, see Figure 2.7.

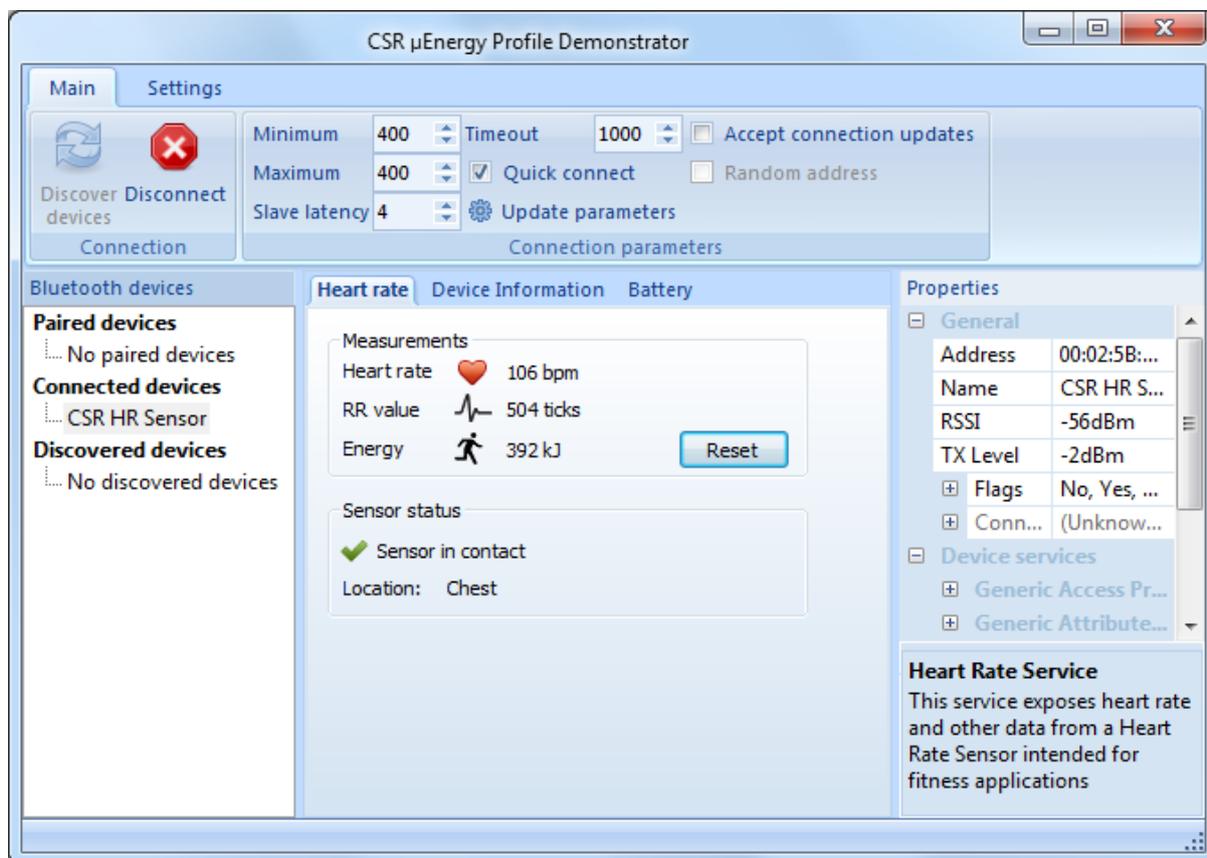


Figure 2.7: Device Connected

3. The **Heart Rate** tab in the **CSR μ Energy Profile Demonstrator** application window, see Figure 2.7 shows the latest received Heart Rate measurement and the status of the sensor.
 - **Measurements**
This section displays the received Heart Rate measurement value in BPM, RR-interval value in ticks and Energy expended in Kilo Joules. The Heart Rate Sensor application sends accumulated Energy expended values once every 10 seconds at a regular interval.
 - **Sensor Status**
Displays the location of the Heart Rate Sensor.
 - **Reset Button**
Clicking this button resets the energy expended value on the Heart Rate Sensor to zero.

4. The **Battery** tab, see Figure 2.8, displays the current state of battery and the **Device Information** tab, see Figure 2.9, displays the Device Information characteristics of the application.

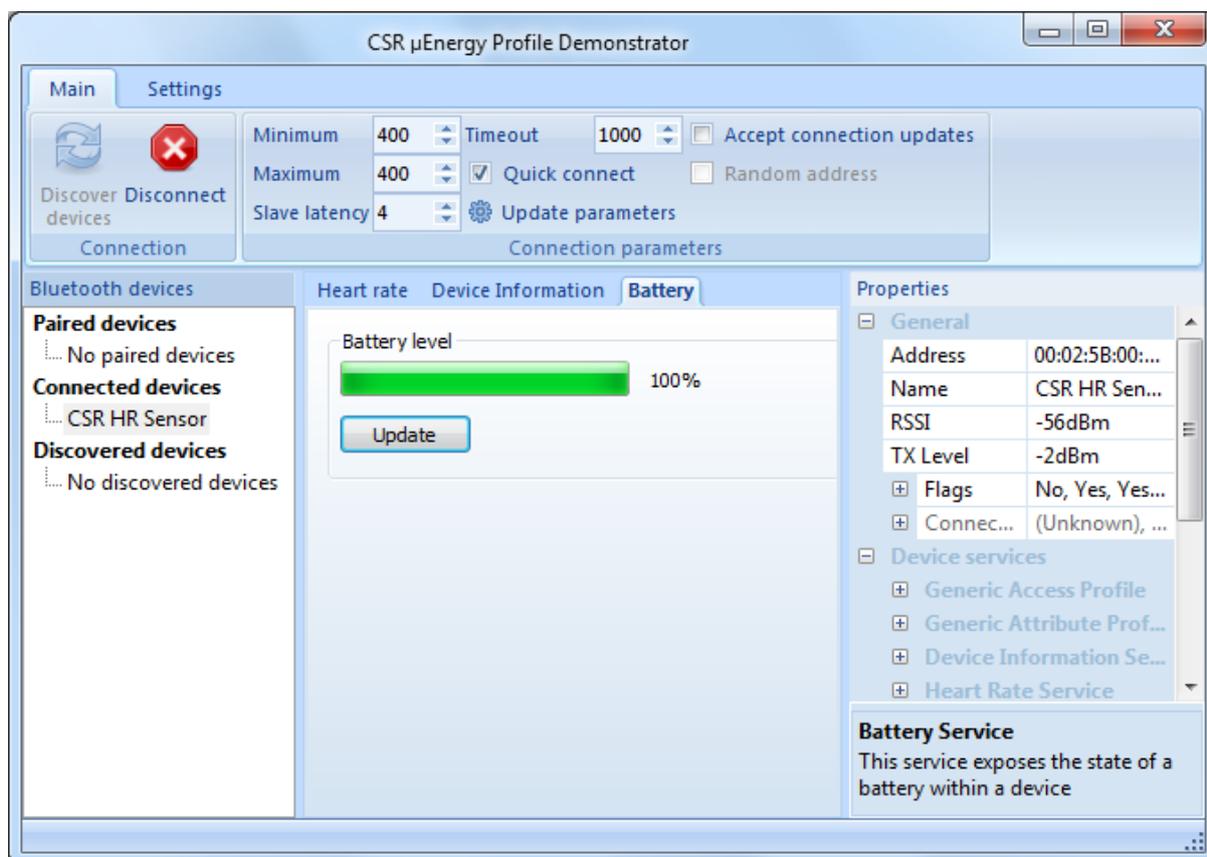


Figure 2.8: Battery Level Tab

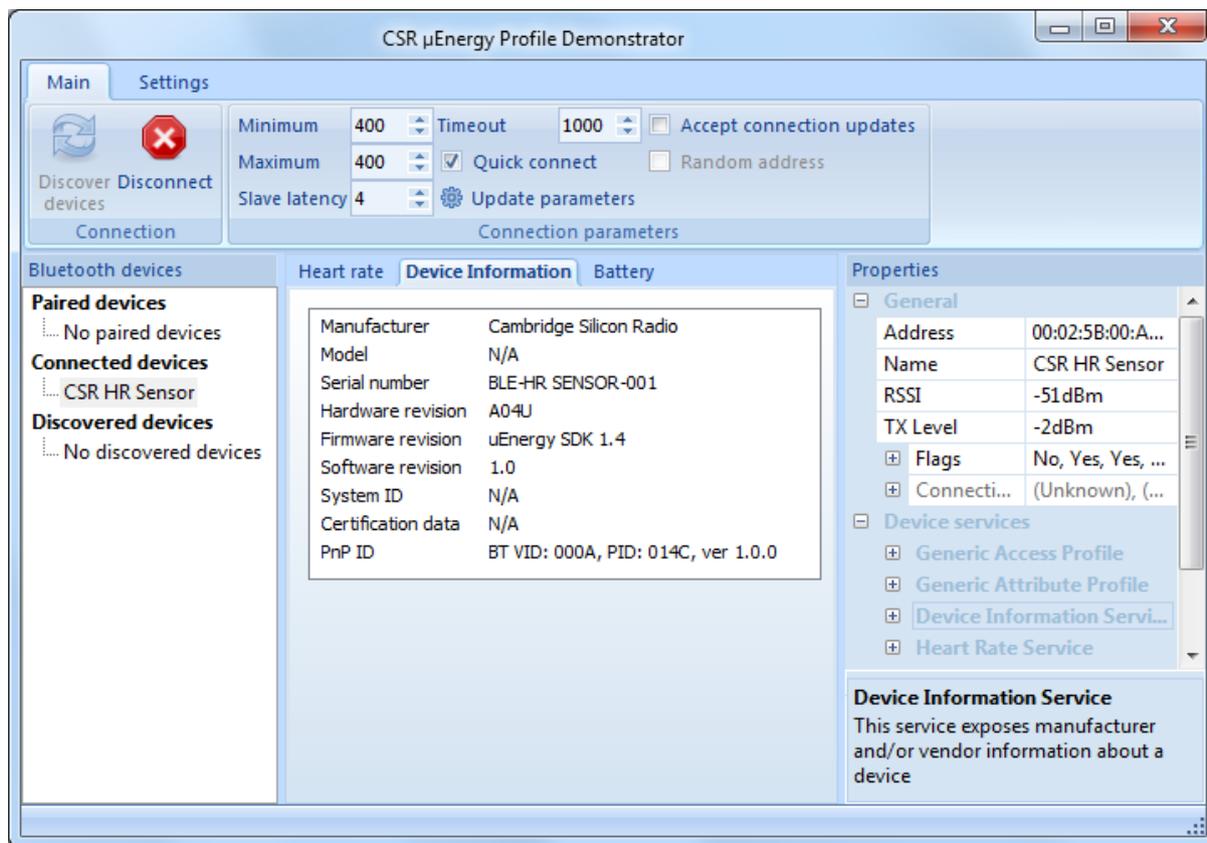


Figure 2.9: Device Information Service Tab

5. Click **Disconnect** to disconnect the Bluetooth low energy link to the application.

3. Application Structure

3.1. Source Files

The table below lists the source files and their purpose.

File name	Purpose
hr_sensor.c	Implements all the entry functions e.g. <code>AppInit()</code> , <code>AppProcessSystemEvent()</code> and <code>AppProcessLmEvent()</code> . Events received from the hardware and firmware are first handled here. This file contains handling functions for all the LM and system events.
hr_sensor_gatt.c	Implements routines for triggering advertisement procedures.
battery_service.c	Implements routines required for the Battery service e.g. reading Battery Level and notifying it to the remote device, and handling access indications on the Battery service specific ATT attributes.
gap_service.c	Implements routines for GAP service e.g. handling read/write access indications on the GAP service characteristics, reading/writing device name on NVM etc.
hr_sensor_hw.c	Implements routines for hardware initialization, button press handling and sounding the buzzer.
nvm_access.c	Implements the NVM read/write routines.
heart_rate_service.c	Implements routines for handling read/write access on Heart Rate service characteristics and for sending notifications of Heart Rate measurements.

Table 3.1: Source Files

3.2. Header Files

The table below lists the header files and their purpose.

File name	Purpose
app_gatt.h	Contains macro definitions, user defined data type definitions and function prototypes which are being used across the application.
appearance.h	Contains the appearance value macro of the Heart Rate Sensor application.
battery_service.h	Contains prototypes of externally referred functions defined in the <code>battery_service.c</code> file.
battery_uuids.h	Contains macro definitions for UUIDs of Battery service and related characteristics.
gap_conn_params.h	Contains macro definitions for fast/slow advertising, preferred connection parameters, idle connection timeout values etc.

File name	Purpose
gap_service.h	Contains prototypes of the externally referred functions defined in the gap_service.c file.
gap_uuids.h	Contains macros for UUID values of the GAP service and related characteristics.
gatt_service_uuids.h	Contains macros for UUID values for GATT service.
heart_rate_service.h	Contains prototypes of externally referred functions defined in the heart_rate_service.c file.
heart_rate_service_uuids.h	Contains macro definitions for UUID values of the Heart Rate service and related characteristics.
hr_sensor.h	Contains timeout values for fast/slow advertising and prototypes of externally referred functions defined in the hr_sensor.c file
hr_sensor_gatt.h	Contains macro definitions for advertising timer values and prototypes of externally referred routines in the hr_sensor_gatt.c file
hr_sensor_hw.h	Contains prototypes of externally referred routines in the hr_sensor_hw.c file
nvm_access.h	Contains prototypes of externally referred NVM read/write functions defined in the nvm_access.c file.

Table 3.2: Header Files

3.3. Database Files

SDK uses database files to generate attribute database for the application. For more information on how to write database files, see *GATT Database Generator User Guide*.

Table 3.3 lists the database files and their purpose.

File name	Purpose
app_gatt_db.db	Master database file which includes all service specific database files. This file is imported by the GATT Database Generator.
battery_service_db.db	Contains information related to Battery service characteristics, their descriptors and values. See Table B.1 for more information on Battery service characteristics.
dev_info_service_db.db	Contains information related to Device Information service characteristics, their descriptors and values. See Table B.2 for Device Information service characteristics.
gap_service_db.db	Contains information related to GAP service characteristics, their descriptors and values. See Table B.3 for GAP characteristics.
gatt_service_db.db	Contains information related to GATT service characteristics, their descriptors and values.

File name	Purpose
Heart_rate_service_db.db	Contains information related to Heart Rate service characteristics, their descriptors and values. See Table B.4 for Heart Rate service characteristics.

Table 3.3: Database Files

4. Code Overview

The following sections describe significant functions of this application.

4.1. Application Entry points

4.1.1. `ApplInit ()`

This function is invoked when the application is powered on or the chip resets and performs the following initialisation functions:

- Initialises the application timers, hardware and application data structures.
- Configures GATT entity for server role.
- Configures the NVM manager to user I2C EEPROM.
- Initialises all the services.
- Reads the persistent store.
- Registers the ATT database with the firmware.

4.1.2. `AppProcessLmEvent ()`

This function is invoked whenever a LM-specific event is received by the system. The following events are being handled in this function:

4.1.2.1. Database Access

- `GATT_ADD_DB_CFM`: This confirmation event marks the completion of database registration with the firmware. On receiving this event, the Heart Rate sensor application starts advertising.
- `GATT_ACCESS_IND`: This indication event is received when the remote Heart Rate Collector tries to access an ATT characteristic managed by the application.

4.1.2.2. LS Events

- `LS_CONNECTION_PARAM_UPDATE_CFM`: This confirmation event is received in response to the connection parameter update request by the application. The connection parameter update request from the application triggers the L2CAP connection parameter update signalling procedure. See Volume 3, Part A, Section 4.20 of *Bluetooth Core Specification Version 4.0*.
- `LS_CONNECTION_PARAM_UPDATE_IND`: This indication event is received when the remote central device updates the connection parameters. On receiving this event, the application validates the new connection parameters against the preferred connection parameters and triggers a connection parameter update request if the new connection parameters do not comply with the preferred connection parameters.

4.1.2.3. SMP Events

- `SM_KEYS_IND`: This indication event is received on completion of the bonding procedure. It contains keys and security information used on a connection that has completed the short term key generation. The application stores the received diversifier (DIV) and Identity Resolving Key (IRK) (if the collector device uses resolvable random address) to NVM. See Volume 3, Part H, section 2.1 of the *Bluetooth Core Specification Version 4.0*.
- `SM_SIMPLE_PAIRING_COMPLETE_IND`: This indication event indicates that the pairing has completed successfully or otherwise. See Volume 3, Part H, Section 2.3 of *Bluetooth Core*

Specification Version 4.0. In the case of a successful completion of the pairing procedure, the Heart Rate sensor application is bonded with the collector and bonding information is stored in the NVM. The bonded device address will be added to the white list, if it is not a resolvable random address.

- `SM_DIV_APPROVE_IND`: This indication event is received when the remote connected device re-encrypts the link or triggers encryption at the time of reconnection. The firmware sends the diversifier in this event and waits for the application to approve or disapprove the encryption. The application shall disapprove the encryption if the bond has been removed by the user. The firmware compares this diversifier with the one it had received in `SM_KEYS_IND` at the time of the first encryption. If similar, the application approves the encryption, otherwise it disapproves it.

4.1.2.4. Connection Events

- `GATT_CONNECT_CFM`: This confirmation event indicates that the connection procedure has completed. If it has not successfully completed, the application goes to idle state and waits for user action. See section 4.2 for more information on application states. If the application is bonded to a device with resolvable random address and connection is established, the application tries to resolve the connected device address using the IRK stored in NVM. If the application fails to resolve the address, it disconnects the link and restarts advertising.
- `GATT_CANCEL_CONNECT_CFM`: This confirmation event confirms the cancellation of connection procedure. When the application stops advertisements to change advertising parameters or to save power, this signal confirms the successful stopping of advertisements by the Heart Rate sensor application.
- `LM_EV_DISCONNECT_COMPLETE`: This event is received on link disconnection. Disconnection could be due to link loss, locally triggered or triggered by the remote connected device.
- `LM_EV_ENCRYPTION_CHANGE`: This event indicates a change in the link encryption.

4.1.3. `AppProcessSystemEvent ()`

This function handles the system events such as a low battery notification or a PIO change. It currently handles the following two system events:

- `sys_event_battery_low`: This event is received whenever the battery voltage crosses the low battery voltage threshold. If connected and notifications are configured, the Heart Rate Sensor application notifies the battery level to the collector device.
- `sys_event_pio_changed`: This event indicates a change in PIO value. Whenever the user presses or releases the button, the corresponding PIO value changes and the application receives a PIO changed event and takes the appropriate action.

4.2. Internal State Machine

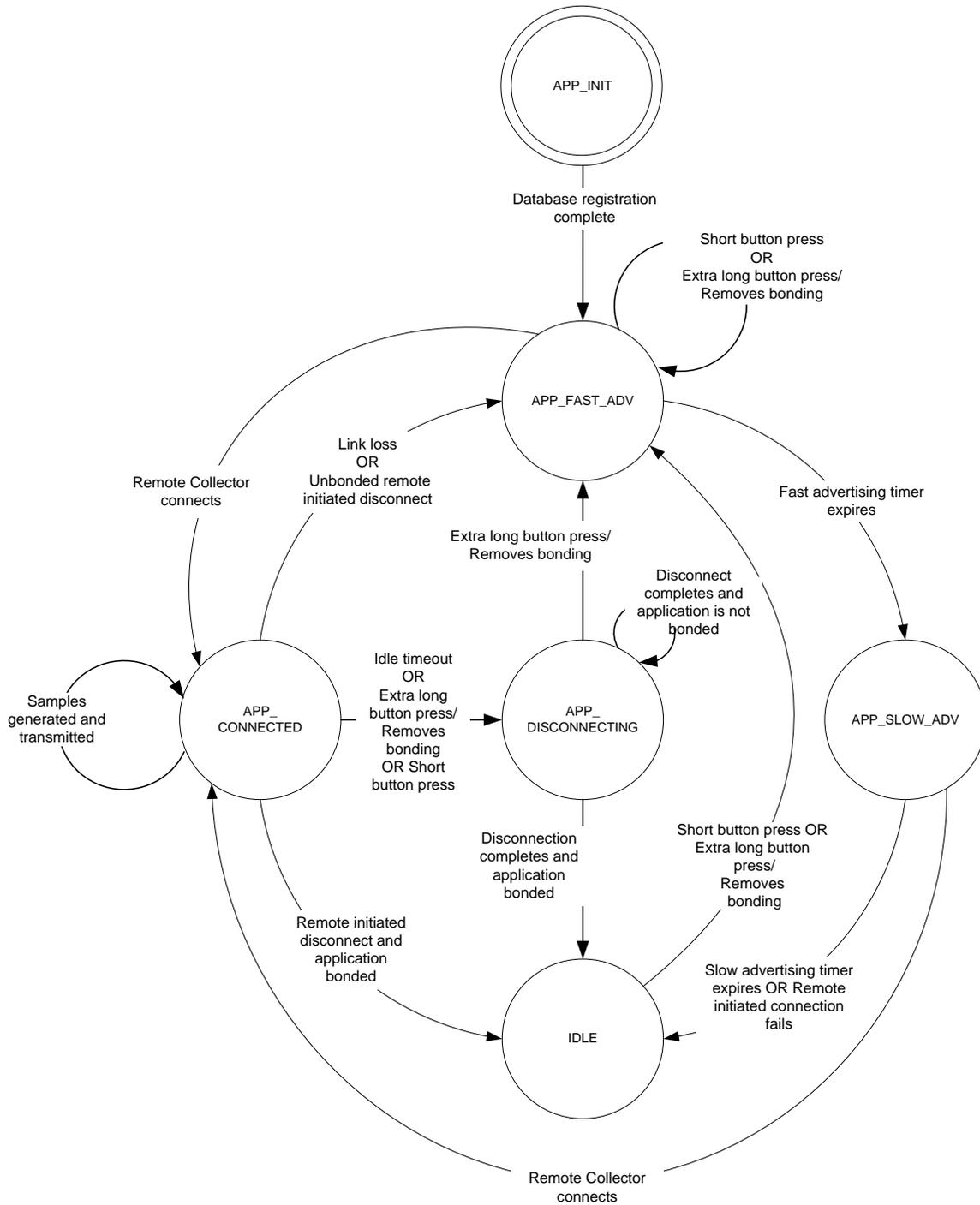


Figure 4.1: Internal State Diagram (Sample Measurement Mode)

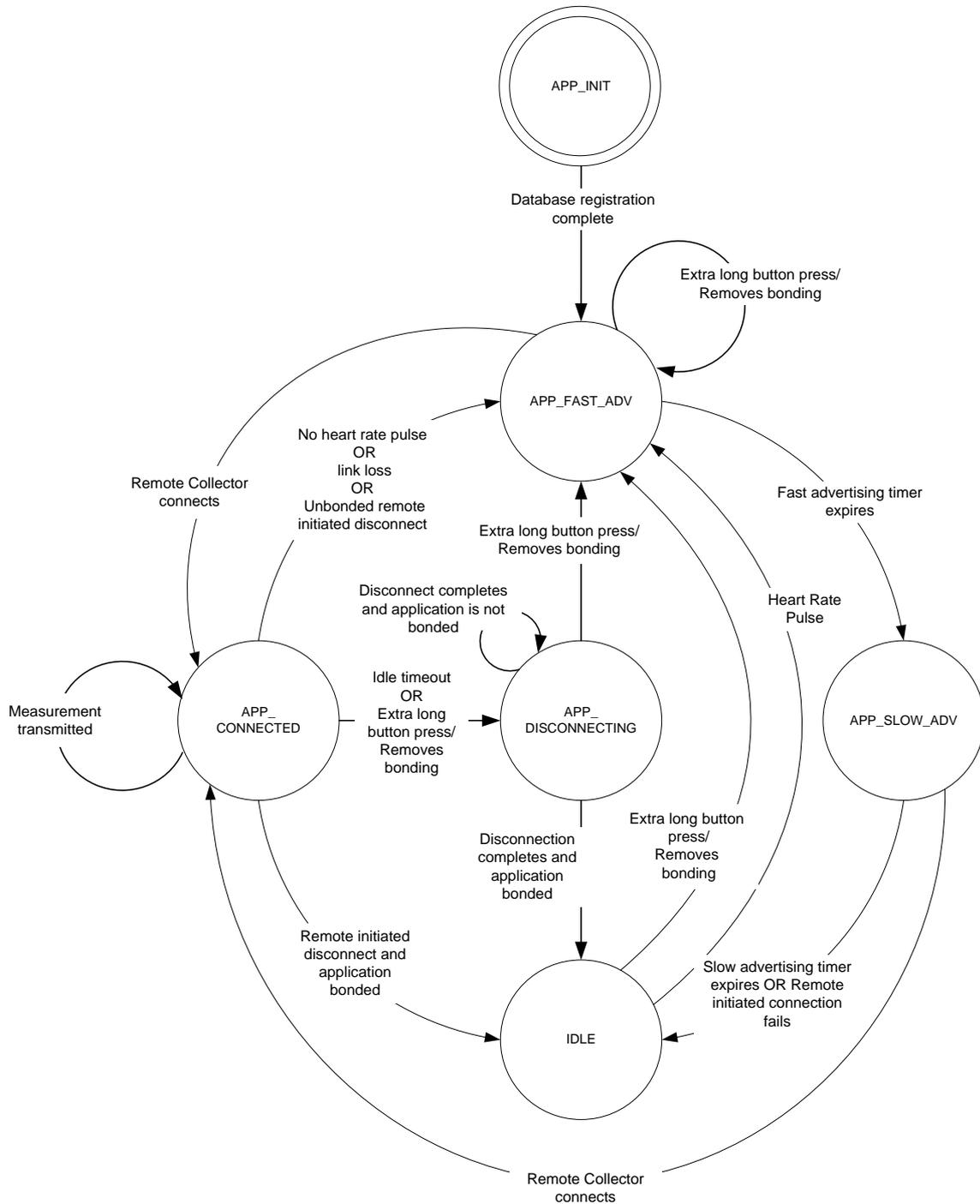


Figure 4.2: Internal State Diagram (Actual Measurement Mode)

The Heart Rate Sensor application has five internal states, see Figure 4.1 and Figure 4.2.

Note:

Short button press does not alter the behaviour of the device in **Actual Measurement** mode.

4.2.1. APP_INIT

When the application is powered on or the chip resets it enters this state. The Heart Rate Sensor application registers the service database with the firmware and waits for a confirmation. On a successful database registration it starts advertising.

4.2.2. APP_IDLE

In this state, the Heart Rate Sensor application is not connected to any Heart Rate Collector.

- On an **Extra long button press**, the application removes the bonding information and starts advertising.
- On a **Short button press** in **Sample Measurement** mode, the application triggers advertisements and enters the `APP_ADVERTISING` state.
- When a Heart Rate Pulse is available in **Actual Measurement** mode, the application triggers advertisements and enters the `APP_ADVERTISING` state.

4.2.3. APP_ADVERTISING

In this state, the Heart Rate Sensor application advertises itself and beeps twice to indicate the start of advertisements.

- Sub state `APP_FAST_ADVERTISING`: The application starts in this sub state and uses fast advertising parameters in this state. If a remote collector connects to it, it stops advertisements and enters the `APP_CONNECTED` state. If the fast advertising timer expires before connection is made, the `APP_SLOW_ADVERTISING` sub state is entered. See section 6.2 for more information on advertisement timers.
- Sub state `APP_SLOW_ADVERTISING`: The application uses slow advertising parameters in this sub state. If a remote device connects to it, it stops advertisements and enters the `APP_CONNECTED` state. If the slow advertising timer expires before a connection is made, the `APP_IDLE` state is entered.

While in any of the above two states:

- If the application is bonded to some remote Heart Rate Collector, it will add the bonded device's address to its white list. This means that it will accept connections only from this bonded device. While triggering advertisements, it starts a Bonded Device Advertisement Timer. If the bonded remote device connects to it within this interval, it stops advertising and enters the `APP_CONNECTED` state.
- If the Bonded Device Advertisement Timer expires before the remote bonded Heart Rate connects to it, it stops advertising, disables the white list and again starts fast advertising for a certain interval period. If the remote collector connects to it, it stops advertisements and enters the `APP_CONNECTED` state.
- On an **Extra Long button press**, the application stops advertisements, removes bonding and restarts advertising without any white list.

See the *Bluetooth Core specification Version 4.0* for more information on white list.

4.2.4. APP_CONNECTED

In this state, the Heart Rate sensor application is connected to a remote Heart Rate Collector and sends Heart Rate measurements at intervals specified in Table 1.4. See Figure B.1 for measurement data format.

- On a **Short button press** in **Sample Measurement** mode, the application disconnects the link and moves to the `APP_DISCONNECTING` state.
- When the Heart Rate Pulse is not available for 10 seconds in **Actual Measurement** mode, the application disconnects the link and moves to the `APP_DISCONNECTING` state.
- On an **Extra Long button press**, application disconnects the link, removes the bonding and starts advertising again.
- The application can disconnect the link if kept idle for some time and enters the `APP_DISCONNECTING` state. See section 6.5 for information on the idle timer.
- If link loss occurs, the application switches to the `APP_ADVERTISING` state.
- In the case of a Remote triggered disconnection, if the application is not bonded to any remote device, it again starts advertising and enters the `APP_ADVERTISING` state else it enters the `APP_IDLE` state.

4.2.5. APP_DISCONNECTING

In this state, the Heart Rate Sensor application waits for a disconnect confirmation after initiating the disconnection. After receiving the disconnect confirmation, it checks if it is bonded to any Heart Rate Collector.

- If the application is bonded, it enters the `APP_IDLE` state and waits for user activity.
- If the application is not bonded, it starts advertising and enters the `APP_ADVERTISING` state.
- On an **Extra long button press**, the application removes the bonding information.

5. NVM Memory Map

The applications can store data in NVM to prevent data loss in the event of a power off or chip panic.

Entity Name	Type	Size of Entity	NVM Offset
Sanity Word	uint16	1	0
Bonded Flag	Boolean	1	1
Bonded Device Address	structure	5	2
Diversifier	uint16	1	7
IRK	uint16 array	8	8

Table 5.1: NVM Memory Map for Application

Entity Name	Type	Size of Entity	NVM Offset
GAP Device Name Length	uint16	1	16
GAP Device Name	uint8 array	20	17

Table 5.2: NVM Memory Map for GAP Service

Entity Name	Type	Size of Entity	NVM Offset
Heart Rate Measurement Characteristic Client Configuration Descriptor	uint16	1	37
Heart Rate Energy Expended	uint16	1	38

Table 5.3: NVM Memory Map for Heart Rate Service

Entity Name	Type	Size of Entity	NVM Offset
Battery Level characteristic Client Configuration Descriptor	uint16	1	39

Table 5.4: NVM Memory Map for Battery Service

Note:

The Application does not pack the data before writing it to the NVM. This means that writing a `uint8` takes one word of NVM memory.

6. Customising the Application

The developer can easily customise the application by modifying the following parameter values.

6.1. Actual Measurement Mode

The Heart Rate Sensor application, by default, operates in **Sample Measurement** mode, see Table 1.4. The macro `NO_ACTUAL_MEASUREMENT`, defined in `hr_sensor.h` file, is used to select the measurement mode. To switch to **Actual Measurement** mode, see Table 1.4, the `hr_sensor.h` must be modified by commenting out the definition of the `NO_ACTUAL_MEASUREMENT` macro.

In **Actual Measurement** mode, the heart rate measurements are fed from an external Heart Rate device to the application via PIO `HR_INPUT_PIO` defined in the `hr_sensor_hw.h` file. The application calculates heart rate measurements from the signal received on `HR_INPUT_PIO` PIO and sends these measurements to the connected Heart Rate Collector.

PIO Name	PIO Number
HR_INPUT_PIO	9

Table 6.1: PIO for External Heart Rate Input

6.2. Advertising Parameters

The Heart Rate Sensor application uses the parameters in Table 6.2 for fast and slow advertisements. The macros for these values are defined in file `gap_conn_params.h`. These values have been chosen by considering the overall current consumption of the device. See *Bluetooth Core Specification Version 4.0* for the advertising parameters range.

Parameter Name	Slow Advertisements	Fast Advertisements
Minimum Advertising Interval	1280 ms	60 ms
Maximum Advertising Interval	1280 ms	60 ms

Table 6.2: Advertising parameters

6.3. Advertisement Timers

The Heart Rate Sensor application enters the appropriate state on expiry of the advertisement timers. See section 4.2 for more information. The macros for these timer values are defined in the file `hr_sensor_gatt.h`.

Timer Name	Timer Values
Bonded Device Advertisement Timer Value	10 s
Fast Advertisement Timer Value	30 s
Slow Advertisement Timer Value	1 min

Table 6.3: Advertisement Timers

6.4. Connection Parameters

The Heart Rate Sensor application uses the connection parameters listed in Table 6.4. The macros for these values are defined in the file `gap_conn_params.h`. These values have been chosen by considering the overall current consumption of the device. See *Bluetooth Core Specification Version 4.0* for the connection parameter range.

Parameter Name	Parameter Value
Minimum Connection Interval	12.5 ms
Maximum Connection Interval	20.0 ms
Slave Latency	100 intervals
Supervision Timeout	12.5 s

Table 6.4: Connection Parameters

6.5. Idle Connected Timeout

While in **Actual Measurement** mode, the Heart Rate Sensor application runs an idle timer in the connected state. If the application does not receive any heart rate measurements during this period, it disconnects the link. The macro for this timer value is defined in file `hr_sensor.c`.

Parameter Name	Parameter Value
Idle Connected Timeout	10 s

Table 6.5: Idle Connected Timeout

6.6. Connection Parameter Update

The Heart Rate Sensor application can request the remote connected Collector to update the connection parameters according to its power requirements. This application requests for connection parameter update on an encryption change or when the remote collector changes the connection parameters. The remote collector may or may not accept the requested parameters. If the remote collector rejects the newly requested parameters, the application again requests for an update after 30 seconds. The macro for this time value is defined in the file `gap_conn_params.h` and can be modified as required.

The μEnergy Profile Demonstrator application by default rejects the Connection Parameter Update request received from the connected Heart Rate Sensor application. Ticking the **Accept connected parameters** option on the μEnergy Profile Demonstrator application accepts the requested parameters, see Figure 2.5.

6.7. Device Name

The user can change the device name for the application. By default, it is set to "CSR HR Sensor" in the file `gap_service.c`. The maximum length of the device name is 20 octets.

6.8. Buzzer

The Heart Rate Sensor application uses the Buzzer to indicate different states and events to the user. The user can enable or disable the buzzer as required by the application. The buzzer should be disabled while taking current consumption readings, see section 0. The macro for enabling the buzzer is defined in the file `hr_sensor_hw.h`. To disable the buzzer, comment out the macro `ENABLE_BUZZER` in file `hr_sensor_hw.h`.

7. Current Consumption

The current consumed by the application can be measured by removing the black 0.1" jumper, see Figure 2.1, and installing an ammeter in its place. The ammeter should be set to DC, measuring current from μA to mA. Any code that sounds the buzzer should be disabled before measuring the actual current consumed.

The setup used while measuring current consumption is described in section 2.1. The μEnergy Profile Demonstrator application must be configured to accept connection parameter update requests, see Figure 2.5.

Table 7.1 shows the current consumption values measured during testing under noisy RF conditions.

Test Scenario	Description	Average Current Consumption	Remarks
Fast Advertisements	<ol style="list-style-type: none"> Switch on the Heart Rate Sensor device Wait for 5 s Take the measurement 	478.40 μA	<ul style="list-style-type: none"> Advertising Interval: 60 ms Advertisement Data length: 29 Measurement Time Duration: 20 s
Slow Advertisements	<ol style="list-style-type: none"> Switch on the Heart Rate Sensor device Wait for 40 s Take the measurement 	27.70 μA	<ul style="list-style-type: none"> Advertising Interval: 1.28 s Advertisement Data length: 29 Measurement Time Duration: 40 s
Connected Active	<ol style="list-style-type: none"> Connect to the Collector Wait for 60 s In sample measurement mode, Heart Rate Sensor device will start sending heart rate measurements at 1 second interval. See section 1.1.4 for details. Take the measurement 	29.09 μA	<ul style="list-style-type: none"> Connection parameters <ul style="list-style-type: none"> Minimum Connection Interval: 12.5 ms Maximum Connection Interval: 20 ms Slave Latency: 100 Measurement Time Duration: 60 s
Notes: <ul style="list-style-type: none"> Average current consumption is measured at 3.2 V Ammeter Used: Agilent 34411A Channel map update has been disabled on CSR8510 USB dongle by setting the AFH options PS Key to 0x0037 (Default value: 0x0017) 			

Table 7.1: Current Consumption Values

Appendix A Definitions

A.1 User interface definitions

Term	Meaning
Short button press	Button press for less than 2 seconds
Long button press	Button press for greater than or equal to 2 seconds but less than 4 seconds
Extra Long button press	Button press for greater than or equal to 4 seconds
Short beep	Beep for 100 ms
Long beep	Beep for 500 ms

Table A.1: Definitions

Appendix B GATT Database

B.1 Battery Service Characteristics

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Battery Level	0x0013	Read, Notify	Application	Security Mode 1 and Security Level 1	Current battery level
Battery Level-Client Configuration Descriptor	0x0014	Read, Write	Application	Security mode 1 and Security level 2	Current client configuration for "Battery Level" characteristic

Table B.1: Battery service characteristics

For more information on Battery service, see *Battery Service Specification Version 1.0*. For information related to security permissions, see *Bluetooth Core Specification Version 4.0*.

B.2 Device Information Service Characteristics

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Serial Number String	0x0017	Read	Firmware	Security Mode 1 and Security Level 1	"BLE-HR SENSOR-001"
Hardware Revision String	0x0019	Read	Firmware	Security Mode 1 and Security Level 1	"A04U"
Firmware Revision String	0x001b	Read	Firmware	Security Mode 1 and Security Level 1	"uEnergy SDK 1.4"
Software Revision String	0x001d	Read	Firmware	Security Mode 1 and Security Level 1	<Application Version>

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Manufacturer Name String	0x001f	Read	Firmware	Security Mode 1 and Security Level 1	"Cambridge Silicon Radio"
PnP ID	0x0021	Read	Firmware	Security Mode 1 and Security Level 1	Vendor Id source - BT Vendor Id - 0x000a Product Id - 0x014c Product Version - 1.0.0

Table B.2: Device information service characteristics

See *Bluetooth Core Specification Version 4.0* for more information on security permissions. For information on Device Information Service see *Device Information Service specification version 1.1*.

B.3 GAP Service Characteristics

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Device Name	0x0003	Read,Write	Application	Security Mode 1 and Security Level 1	Device name Default value : "CSR HR Sensor"
Appearance	0x0005	Read	Firmware	Security Mode 1 and Security Level 1	Generic Heart Rate Sensor : 0x0340
Peripheral preferred connection parameters	0x0007	Read	Firmware	Security Mode 1 and Security Level 1	Min connection interval - 12.5 ms Max connection interval – 20 ms Slave latency - 100 Connection timeout - 12.5 s

Table B.3: GAP Service Characteristics

For more information on GAP service and security permissions, see *Bluetooth Core Specification Version 4.0*.

B.4 Heart Rate Service Characteristics

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Heart Rate Measurement	0x000b	Notify	Application	Security Mode 1 and Security Level 1	Heart Rate measurement value
Heart Rate Measurement - Client Characteristic Configuration descriptor	0x000c	Read, Write	Application	Security Mode 1 and Security Level 1	Current client configuration for "Heart Rate measurement" characteristic
Body Sensor Location	0x000e	Read	Firmware	Security Mode 1 and Security Level 1	0x01 - Default sensor location is chest
Heart Rate Control Point	0x0010	Write	Application	Security Mode 1 and Security Level 1	Supported control points for Heart Rate sensor

Table B.4: Heart Rate Service Characteristics

See *Bluetooth Core Specification Version 4.0* for more information on security permissions. For information on Heart rate service see *Heart Rate Service Specification version 1.0*.

Figure B.1 defines the data format for the Heart Rate Measurement characteristic as used in the application. For more information on Heart Rate Measurement characteristic see the *Bluetooth SIG Developer Portal*.

The Energy Expended field is sent once for every 10 measurements to the collector and the type of the HR Measurement field is a `uint8`.



Figure B.1: Heart Rate Measurement Data Format

Notes:

- (1) This application is usable in public environments, for example a gymnasium, where fitness equipment such as treadmills or steppers do not have the ability to bond. Hence the security permissions for all the characteristics of this application have been set to Security Mode 1 and Security Level 1 i.e. No Security. See *Heart Rate Profile Specification Version 1.0* for more information on public environment requirements and security aspects.
- (2) Characteristics are managed by either the firmware or the application. The characteristics managed by the application have flags set to `FLAG_IRQ` in the corresponding database file. When the remote connected device accesses that characteristic, the application receives an `GATT_ACCESS_IND` LM event that is handled in the `AppProcessLmEvent()` function defined in the `hr_sensor.c` file. See section 4.1.2.1 for more information on the handling of the `GATT_ACCESS_IND` LM event. For more information on flags, see the *GATT Database Generator User Guide*.

Appendix C Advertising/Scan Response Data

The Heart Rate Sensor application adds the following fields to the Advertising Data:

AD Field	Contents
Flags	The Heart Rate Sensor application sets the General Discoverable Mode bit. See section 11, Part C of Volume 3 in <i>Bluetooth core Specification Version 4.0</i> for more information.
Service UUIDs	The Heart Rate sensor application adds 16-bit UUIDs for the following service: <ul style="list-style-type: none"> Heart Rate
Device Appearance	Generic Heart Rate Sensor: 0x0340
Tx Power	Current Tx power level
Device Name(1)	Device name (Default value : "CSR HR Sensor")
<p>Note:</p> <p>⁽¹⁾ If the Device Name length is greater than the space left in the Advertising Data field then the application adds it to Scan Response data.</p>	

Table C.1: Advertising Data Fields



Appendix D Known Issues or Limitations

There are no known issues with this application.

Document References

Document	Reference
<i>Bluetooth Core Specification Version 4.0</i>	https://www.bluetooth.org/Technical/Specifications/adopted.htm
<i>Heart Rate Profile Specification Version 1.0</i>	https://www.bluetooth.org/Technical/Specifications/adopted.htm
<i>Heart Rate Service Specification Version 1.0</i>	https://www.bluetooth.org/Technical/Specifications/adopted.htm
<i>Battery Service Specification Version 1.0</i>	https://www.bluetooth.org/Technical/Specifications/adopted.htm
<i>Device Information Service Specification Version 1.1</i>	https://www.bluetooth.org/Technical/Specifications/adopted.htm
<i>Service Characteristics And Descriptions</i>	http://developer.bluetooth.org/gatt/characteristics/Pages/default.aspx
<i>USB Device Driver User Guide</i>	CS-208306-UG
<i>GATT Database Generator</i>	CS-219225-UG
<i>μEnergy xIDE User Guide</i>	CS-212742-UG

Terms and Definitions

ATT	Attribute
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
BLE	Bluetooth Low Energy
BPM	Beats Per Minute
CS	Configuration Store
CSR	Cambridge Silicon Radio
DIV	Diversifier
EEPROM	Electrically Erasable Programmable Read Only Memory
GAP	Generic Access Profile
GATT	Generic Attribute Profile
IDE	Integrated Development Environment
IRK	Identity Resolving Key
LM	Link Manager
NVM	Non Volatile Memory
PC	Personal Computer
PIO	Programmable Input Output
PnP	Plug and Play
PWM	Pulse Width Modulation
UUID	Universally Unique Identifier