CSR

Push every boundary.™

CSR µEnergy®

Heart Rate Sensor

Application Note

Issue 7

Document History

Revision	Date	History
1	15 OCT 13	Original publication of this document
2	16 JUN 12	Table 7.1 corrected
3	13 SEP 12	Figure 4.1 and Figure 4.2 updated
4	11 FEB 13	Updated to reference CSR101x devices and for SDK 2.1
5	22 FEB 13	Updated current consumption values
6	10 MAY 13	Updated for SDK 2.2; static random address, dormant mode and connection parameter update
7	04 FEB 14	Updated to new CSR branding and changes to connection parameters

Contacts

General information	www.csr.com
Information on this product	sales@csr.com
Customer support for this product	www.csrsupport.com
More detail on compliance and standards	product.compliance@csr.com
Help with this document	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.

Use of this document is permissible only in accordance with the applicable CSR licence agreement.

Safety-critical Applications

CSR's products are not designed for use in safety-critical devices or systems such as those relating to: (i) life support; (ii) nuclear power; and/or (iii) civil aviation applications, or other applications where injury or loss of life could be reasonably foreseeable as a result of the failure of a product. The customer agrees not to use CSR's products (or supply CSR's products for use) in such devices or systems.

Performance and Conformance

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

Contents

Tables, Figures and Equations

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.

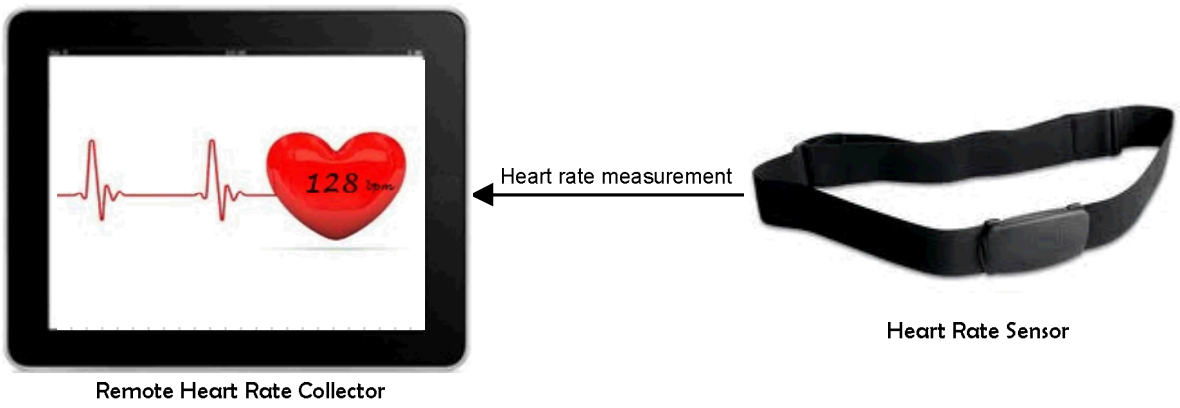

Figure 1.1: Heart Rate Profile Use Case

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 4.1*

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 4.1*. 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 4.1*.

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 4.1* for more information.

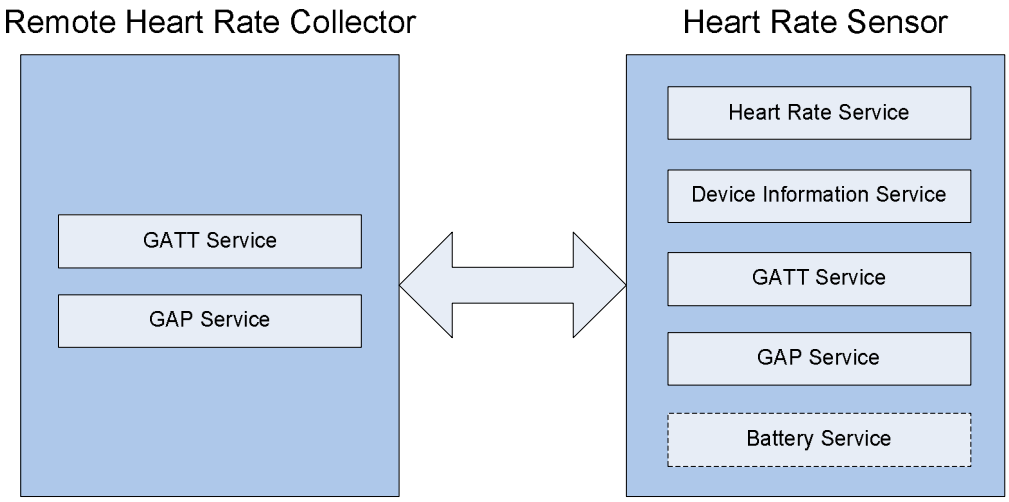

Figure 1.2: Primary Services

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 CSR 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	CSR10x0 Development Board	Heart Rate Sensor Application v1.0
Heart Rate Collector	CSR µEnergy BT4.0 USB dongle	CSR µEnergy Profile Demonstrator Application

Table 2.1: Demonstration Components

Note:

Although the Heart Rate Sensor application primarily targets the CSR10x0 development boards, the CSR10x1 development boards may also be used as an alternative hardware platform.

The CSR µEnergy Profile Demonstrator application, CSR device driver and installation guide are included in the SDK.

2.1.1. Heart Rate Sensor

The SDK is used to build and download the Heart Rate Sensor application to the development board. See the *CSR 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.

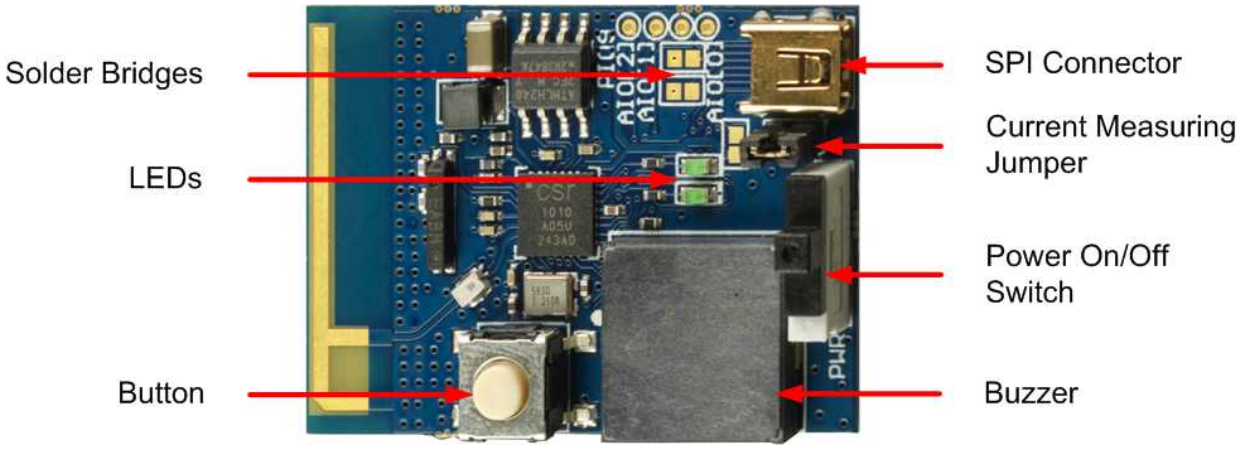

Figure 2.1: CSR10x0 Development Board

2.1.1.1. User Interface

This application makes use of the button and buzzer available on the CSR10x0 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.

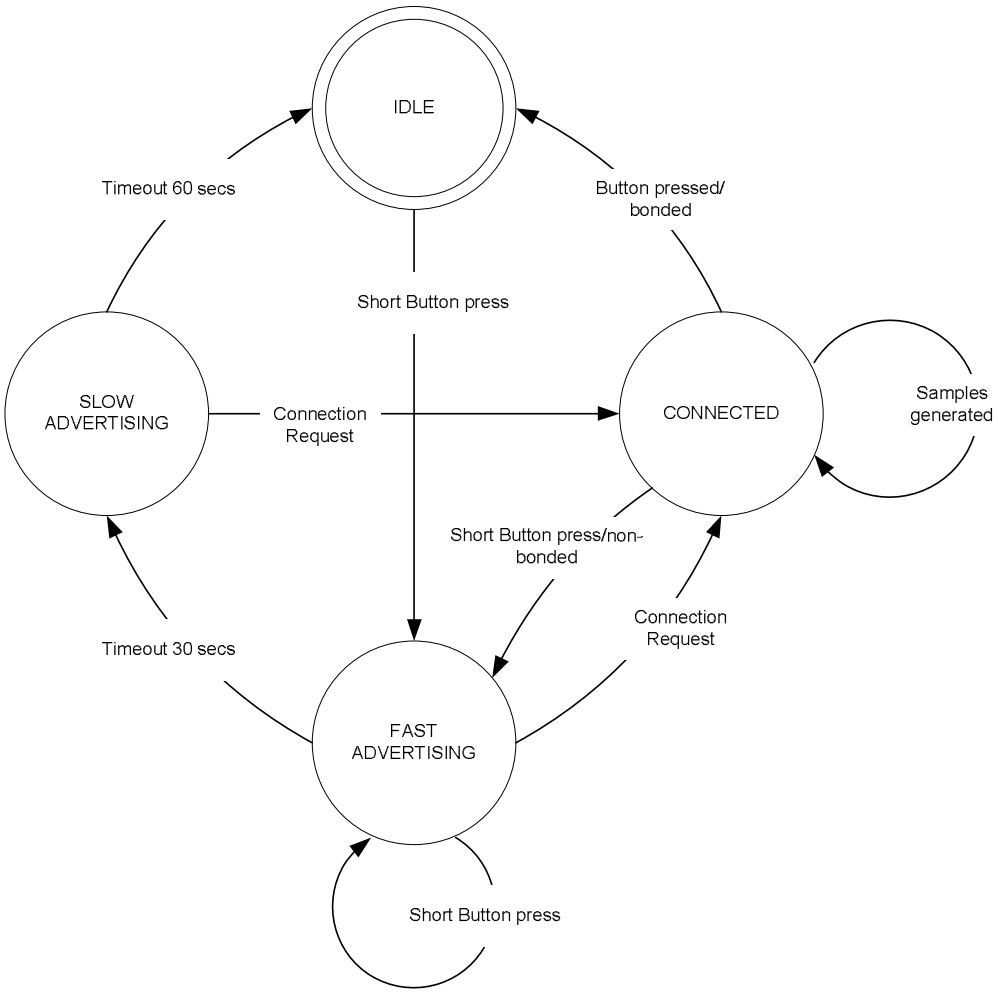

Figure 2.2: Device Behaviour (Sample Measurement Mode)

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**.

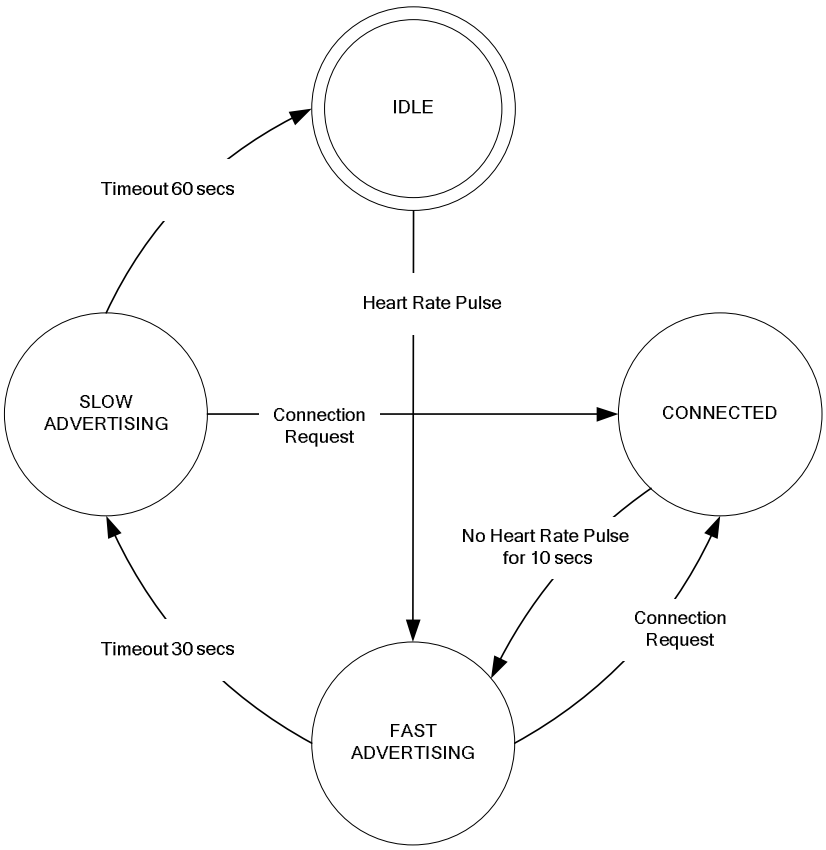

Figure 2.3: Device Behaviour (Actual Measurement Mode)

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. CSR µEnergy BT4.0 USB Dongle

The CSR8510 USB dongle can be used with the CSR 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 driver must be replaced with the CSR BlueCore device driver as described in the *Installing the CSR Driver for the Profile Demonstrator Application* user guide.

The CSR device driver and installation guide is included in the SDK.

Figure 2.4: CSR µEnergy BT4.0 USB Dongle

2.1.2.2. CSR µEnergy Profile Demonstrator Application

The CSR µEnergy Profile Demonstration application is compatible with a PC running Windows XP, Windows 7 (32 bit and 64-bit) or Windows 8 (32-bit and 64-bit). Launch the application once the USB dongle is attached to the PC and the driver has been loaded.

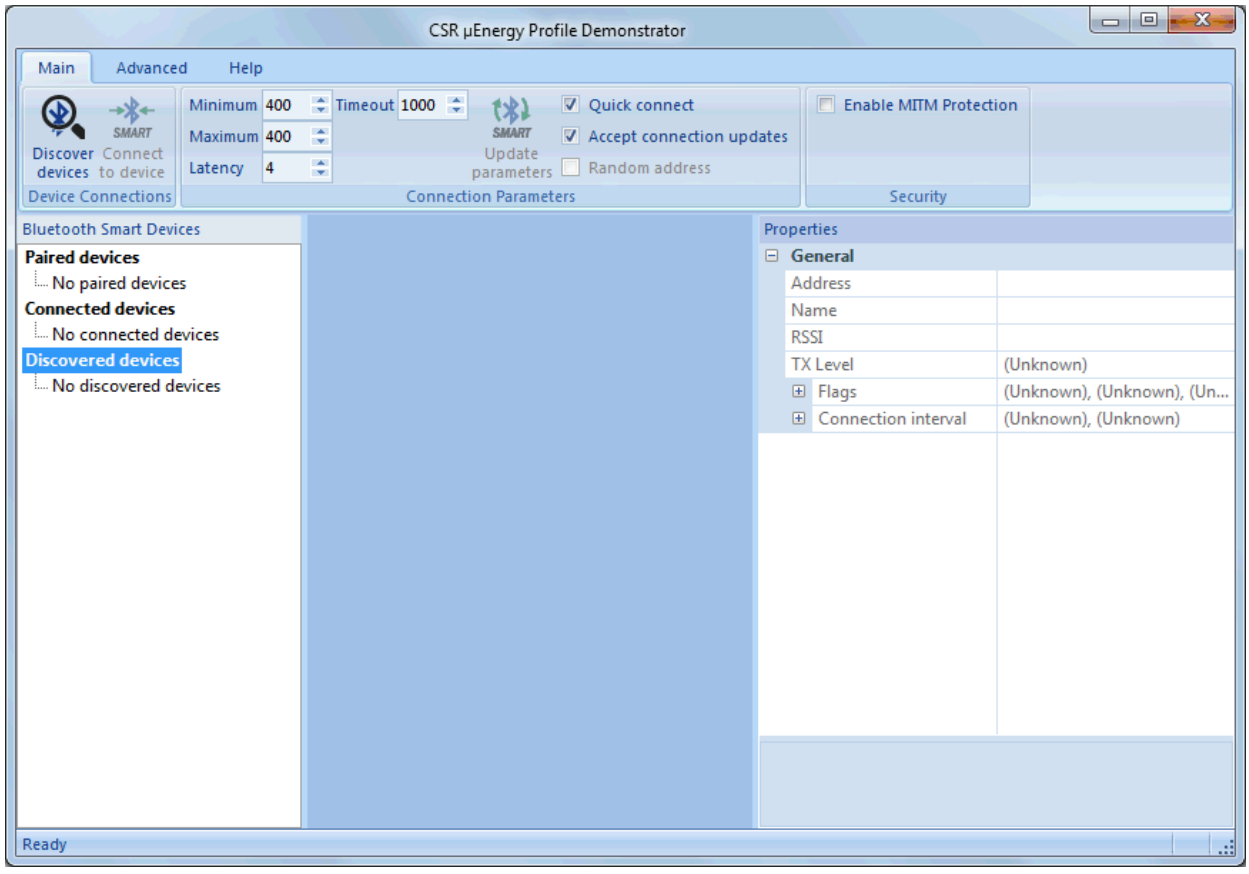

Figure 2.5: CSR µEnergy Profile Demonstrator

2.2. Demonstration Procedure

To demonstrate the Heart Rate Sensor application:

1. Switch on the 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 Smart devices and lists all the discovered devices on the left hand side of the application window, see Figure 2.6.

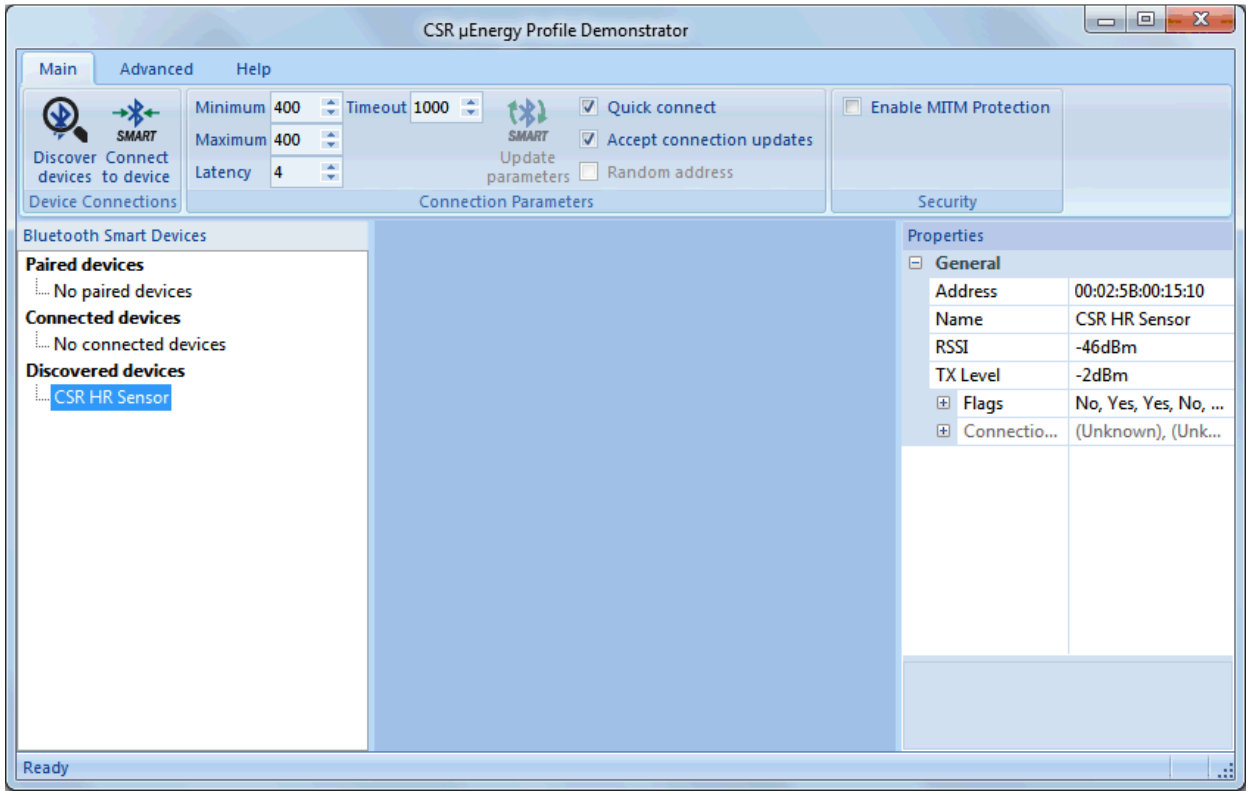

Figure 2.6: Heart Rate Sensor Device Discovered

3. When the device labelled **CSR HR Sensor** appears, select it to display the device address on the right hand side of the screen. Enter the preferred connection parameters from Table 6.4 in the **Connection Parameters** tab. Click on the **Connect to device** button to connect to this device, see. Figure 2.6. The CSR µEnergy Profile Demonstrator application displays a tabbed pane corresponding to different services supported by the device, see Figure 2.7.

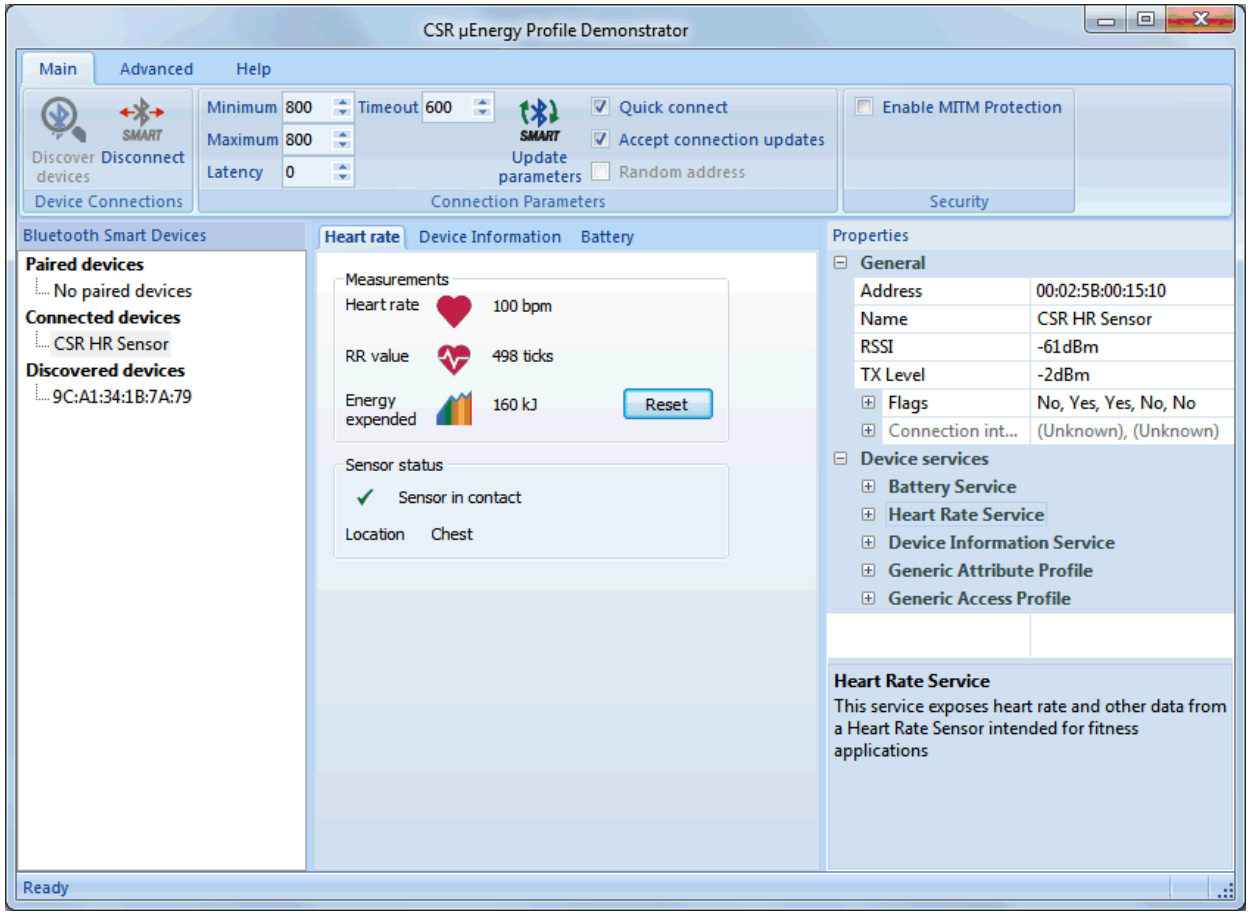

Figure 2.7: Device Connected

4. The **Heart Rate** tab in the **CSR µEnergy Profile Demonstrator** application window, see Figure 2.7 shows the latest received Heart Rate measurement and the status of the sensor.

   - **Measurements**
     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 expended energy values once every 10 seconds.
   - **Sensor Status**
     Displays the location of the Heart Rate Sensor.
   - **Reset Button**
     Clicking this button resets the expended energy value on the Heart Rate Sensor to zero.

5. 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.

Note:

The battery level may fluctuate depending on the connection state.

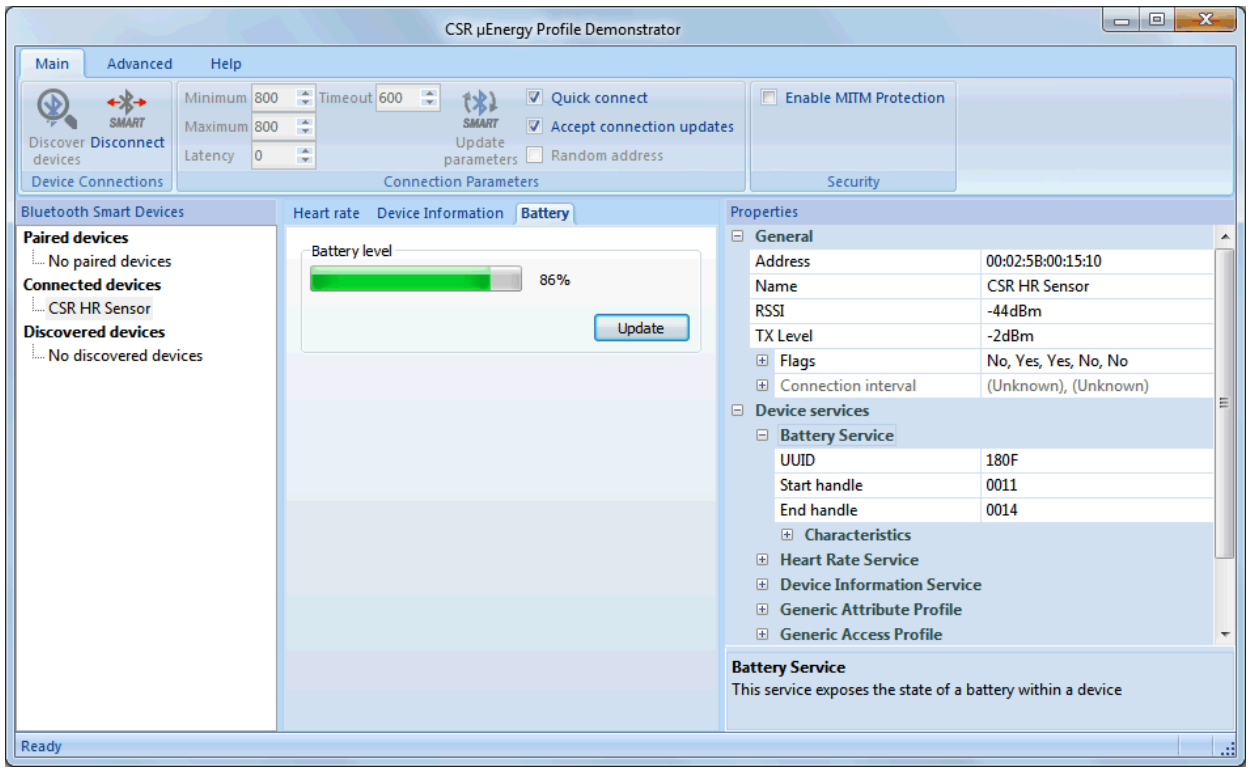

Figure 2.8: Battery Level Tab

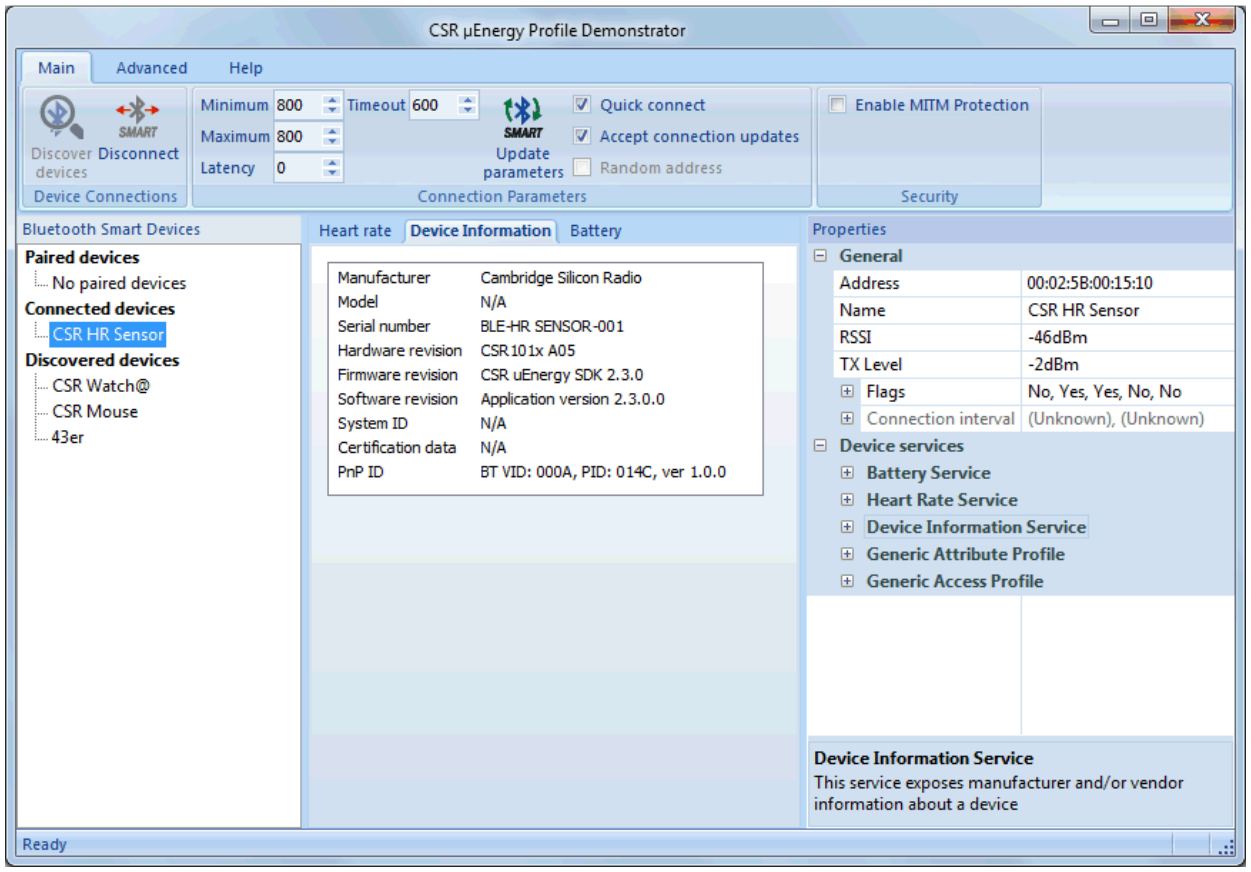

Figure 2.9: Device Information Service Tab

6. Click **Disconnect** to disconnect the Bluetooth Smart 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. AppInit(), AppProcessSystemEvent() and AppProcessLmEvent(). 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 battery_service.c 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.
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.

File Name	Purpose
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.
user_config.h	Contains macros for customising the application.

Table 3.2: Header Files

3.3. Database Files

The 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.
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. AppInit ()

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 use I^2C 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 4.1*.
- `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.
- `LS_RADIO_EVENT_IND`: This radio event indication is received when the chip firmware receives an acknowledgement for the Tx data sent by the application. On receiving this event, the application aligns the timer wakeup (which sends data periodically to the collector) with the latent connection interval.

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 2.4 and Section 3.6 of the *Bluetooth Core Specification Version 4.1*.
- `SM_SIMPLE_PAIRING_COMPLETE_IND`: This indication event indicates that the pairing has completed successfully or otherwise. See Volume 3, Part H, Section 2.4 and Section 3.6 of *Bluetooth Core Specification Version 4.1.* 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_CONNECTION_COMPLETE: This event is received when the connection with the master is considered to be complete and includes the new connection parameters.
- 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.
- LM_EV_CONNECTION_UPDATE: This event indicates that the connection parameters have been updated to a new set of values and is generated when the connection parameter update procedure is either initiated by the master or the slave. These new values are stored by the application for comparison against the preferred connection parameter, see section 6.6.

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

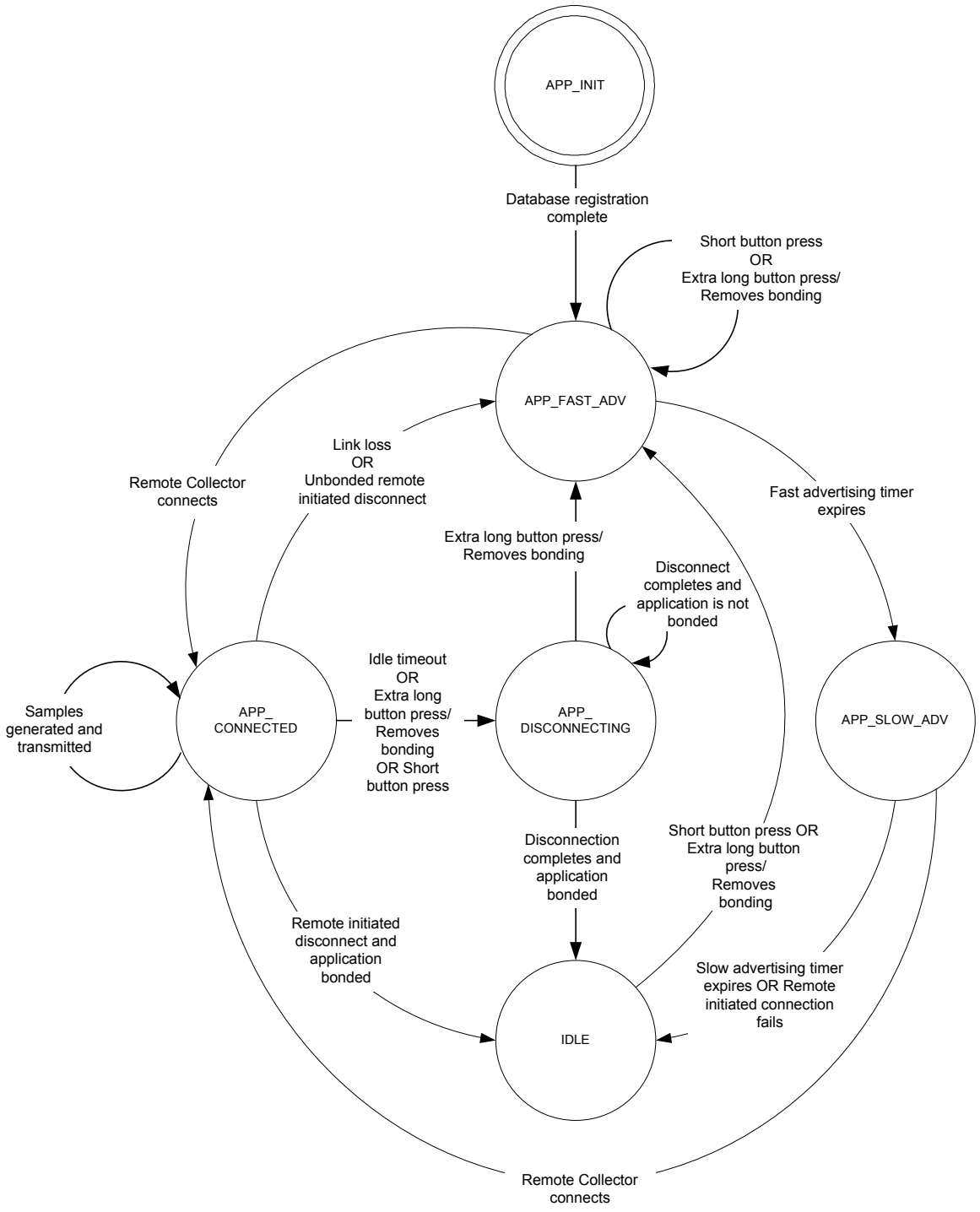

Figure 4.1: Internal State Diagram (Sample Measurement Mode)

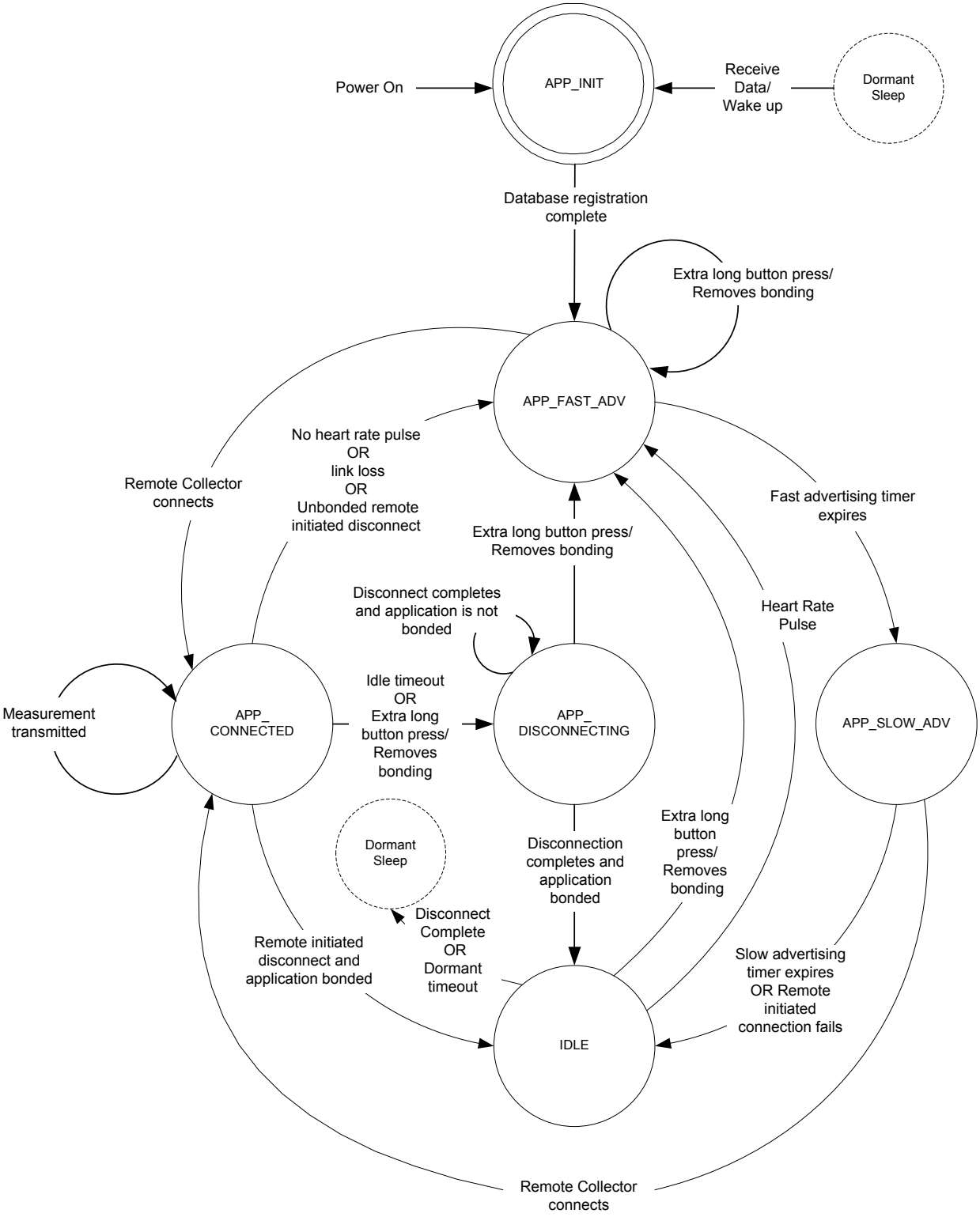

Figure 4.2: Internal State Diagram (Actual Measurement Mode)

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.
- When no signal is received on `HR_INPUT_PIO` PIO in **Actual Measurement** mode, the Heart Rate Sensor application automatically moves to dormant sleep state after a timeout period. See 6.6 for more information on the dormant idle timer.

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 4.1* 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 the idle timeout period in **Actual Measurement** mode, the application disconnects the link and moves to the `APP_DISCONNECTING` state. See section 6.5 for information on the idle timer.
- On an **Extra Long button press**, application disconnects the link, removes the bonding and starts advertising again.
- 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 disconnection was triggered by an idle timeout, then the application will enter a dormant sleep state in **Actual Measurement** mode. The application will wake up from this sleep state whenever the wake pin matches the configured state of the macro `WAKE_SIGNAL_LEVEL` in `user_config.h`.
- 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 (Words)	NVM Offset (Words)
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 (Words)	NVM Offset (Words)
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 (Words)	NVM Offset (Words)
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 (Words)	NVM Offset (Words)
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 the `user_config.h` file, is used to select the measurement mode. To switch to **Actual Measurement** mode, see Table 1.4, the `user_config.h` file 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 4.1* 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 by default, and should be used to configure the Profile Demonstrator before a connection is attempted, see section 2.2 . 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 4.1* for the connection parameter range. See section 6.7 for the connection update procedure.

Parameter Name	Parameter Value	Profile Demonstrator Configuration
Minimum Connection Interval	1000 ms	800
Maximum Connection Interval	1000 ms	800
Slave Latency	0 intervals	0
Supervision Timeout	6000 ms	600

Table 6.4: Connection Parameters (Default)

When connecting to Apple products, the connection parameters listed in Table 6.5 should be used.

Parameter Name	Parameter Value	Profile Demonstrator Configuration
Minimum Connection Interval	980 ms	800
Maximum Connection Interval	1000 ms	800
Slave Latency	0 intervals	0
Supervision Timeout	6000 ms	600

Table 6.5: Preferred Connection Parameters for Apple Products

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 6.6: Idle Connected Timeout

6.6. Idle Timeout

The Heart Rate Sensor application runs a dormant timer after moving to the idle state. If the application does not receive any heart rate measurements during this period, it moves to dormant sleep after the time out. The macro for this timer value is defined in file `user_config.h`. To disable the transition to dormant sleep, `user_config.h` must be modified by commenting out the definition of the `ENABLE_DORMANT_MODE_FUNCTIONALITY` macro.

Parameter Name	Parameter Value
Idle Timeout	1800 s

Table 6.7: Idle timeout

6.7. Connection Parameter Update

The Heart Rate Sensor application can request the remote connected Collector to update the connection parameters according to its power requirements. The application requests a connection parameter update as per the recommendations in *Bluetooth Core Specification Version 4.1* [Vol. 3], Part C Section 9.3.9, or 30 seconds after the remote Collector changes the connection parameters. Upon connection establishment with the Collector, the following procedure is used to send a connection parameter update request:

1. Upon connection, the 5s $T_{GAP(conn\_pause\_peripheral)}$ timer is started.
2. Upon the expiry of $T_{GAP(conn\_pause\_peripheral)}$, the 1s $T_{GAP(conn\_pause\_central)}$ timer is started.
3. During this 1s $T_{GAP(conn\_pause\_central)}$ period, if the application receives a `GATT_ACCESS_IND LM` event, the timer will be deleted and re-created. The receipt of this event means that the service discovery procedure is in progress and the Heart Rate Sensor application should not request a connection parameter update.
4. Upon the expiry of $T_{GAP(conn\_pause\_peripheral)}$, a connection parameter update request will be sent from the Heart Rate Sensor application.

The remote Collector may or may not accept the requested parameters. If the remote Collector rejects the new requested parameters, the application again requests an update after 30 seconds. The macro for this time value is defined in file `hr_sensor.c` and can be modified as required but should be greater than 30 seconds as per the recommendation of the core specification. See *Bluetooth Core Specification Version 4.1 [Vol. 3], Part C Section 9.3.9*. After two failed attempts, the Heart Rate Sensor application tries to update with the Apple preferred connection parameters another two times. Ensure the **Accept connected parameters** option on the CSR µEnergy Profile Demonstrator application is checked to accept the requested parameters, see Figure 6.1.

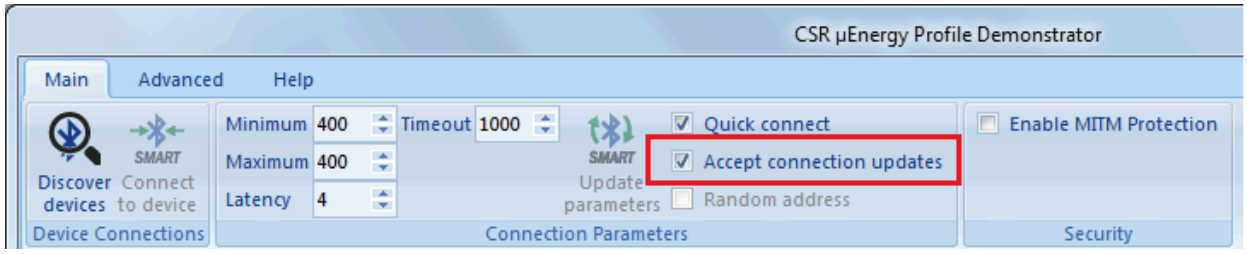

Figure 6.1: Accept Connection Parameters

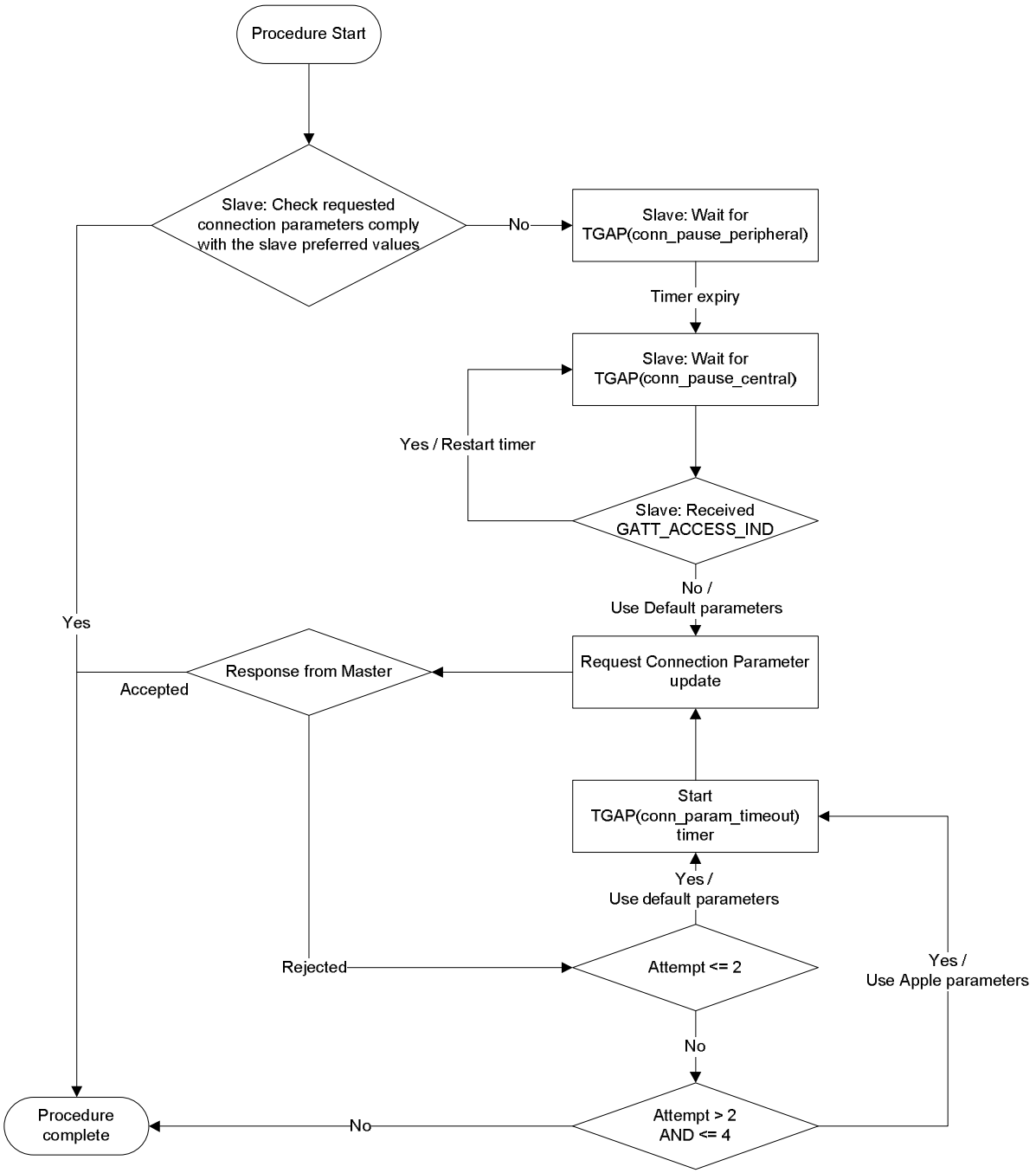

Figure 6.2: Connection Parameter Update Procedure

6.8. 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.

6.9. 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 7. The macro for enabling the buzzer is defined in the file `user_config.h`. To disable the buzzer, comment out the macro `ENABLE_BUZZER` in file `user_config.h`.

6.10. Device Address

The Heart Rate Sensor application uses a public address by default. The `USE_STATIC_RANDOM_ADDRESS` macro in `app_gatt.h` file can be used to enable the support for static random addresses.

6.11. Configuring the Wake Pin

The application is expecting the wake pin to be in shorted state with the input PIO `HR_INPUT_PIO`. Since the application is pulling up this PIO, the wake pin default state will also be HIGH. Therefore configure the wake on Low value. The default value for wake is defined by the macro `WAKE_SIGNAL_LEVEL` in `user_config.h`.

6.12. Body Location

The default location can be changed by setting the `CURRENT_BODY_SENSOR_LOCATION_VALUE` to one of the values defined in `heart_rate_service_uuids.h`. The default location is set to 'chest'.

6.13. Non-Volatile Memory

The Heart Rate Sensor application uses one of the following macros to store and retrieve persistent data in either the EEPROM or Flash-based memory.

- `NVM_TYPE_EEPROM` for I^2C EEPROM.
- `NVM_TYPE_FLASH` for SPI Flash.

Note:

The macros are enabled by selecting the NVM type using the Project Properties in xIDE. This macro is defined during compilation to let the application know which NVM type it is being built for. If EEPROM is selected `NVM_TYPE_EEPROM` will be defined and for SPI Flash the macro `NVM_TYPE_FLASH` will be defined. Follow the comments given in the keyr file as per the selection above.

7. Current Consumption

The current consumed by the application can be measured by removing the Current Measuring 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 CSR µEnergy Profile Demonstrator application must be configured to accept connection parameter update requests, see Figure 2.5.

Table 7.1 shows the typical current consumption values measured during testing under noisy RF conditions with Channel Map Updates disabled, and with typical connection parameter values using the CSR1010 development board. See the Release Notes for the actual current consumption values measured for the application.

Test Scenario	Description	Average Current Consumption	Remarks
Fast Advertisements	1. Switch on the Heart Rate Sensor device 2. Wait for 5 s 3. Take the measurement	441 µA	▪ Advertising Interval: 60 ms ▪ Advertisement Data length: 29 ▪ Measurement Time Duration: 20 s
Slow Advertisements	1. Switch on the Heart Rate Sensor device 2. Wait for 40 s 3. Take the measurement	28 µA	▪ Advertising Interval: 1.28 s ▪ Advertisement Data length: 29 ▪ Measurement Time Duration: 40 s
Connected Active	1. Connect to the Collector 2. Wait for 60 s 3. 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. 4. Take the measurement	21 µA	▪ Connection parameters Minimum Connection Interval: 1000 ms Maximum Connection Interval: 1000 ms Slave Latency: 0 ▪ Measurement Time Duration: 60 s
Notes: ▪ Average current consumption is measured at 3.0 V ▪ Ammeter Used: Fluke 289 ▪ Channel map update has been disabled on the USB dongle by setting the AFH options PS Key to `0x0037` (Default value: `0x0017`) using the PSTool application (included in CSR BlueSuite tools only and available on www.CSRSupport.com)			

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 4.1*.

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	*<Chip Identifier>*
Firmware Revision String	0x001b	Read	Firmware	Security Mode 1 and Security Level 1	*<SDK Version>*
Software Revision String	0x001d	Read	Firmware	Security Mode 1 and Security Level 1	*<Application Version>*
Manufacturer Name String	0x001f	Read	Firmware	Security Mode 1 and Security Level 1	"Cambridge Silicon Radio"

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
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 4.1* 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 - 1000 ms Max connection interval – 1000 ms Slave latency - 0 Connection timeout - 6 s

Table B.3: GAP Service Characteristics

For more information on GAP service and security permissions, see *Bluetooth Core Specification Version 4.1*.

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 4.1* 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`.

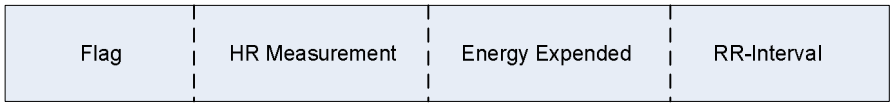

Figure B.1: Heart Rate Measurement Data Format

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 *Bluetooth core Specification Version 4.1* for more information.
Service UUIDs	The Heart Rate sensor application adds 16-bit UUIDs for the Heart Rate service.
Device Appearance	Generic Heart Rate Sensor: `0x0340`
Tx Power	Current Tx power level
Device Name(1)	Device name, Default value: “CSR HR Sensor”.
Note: (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.	

Table C.1: Advertising Data Fields

Appendix D Known Issues or Limitations

See the Heart Rate Sensor application and SDK Release Notes.

Document References

Document	Reference
Bluetooth Core Specification Version 4.1	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Heart Rate Profile Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Heart Rate Service Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Battery Service Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Device Information Service Specification Version 1.1	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Service Characteristics And Descriptions	http://developer.bluetooth.org/gatt/characteristics/Pages/default.aspx
GATT Database Generator	CS-219225-UG
CSR µEnergy xIDE User Guide	CS-212742-UG
Installing the CSR Driver for the Profile Demonstrator Application	CS-235358-UG

Terms and Definitions

ATT	Attribute
Bluetooth®	Set of wireless technologies providing audio and data transfer over short-range radio connections
BLE	Bluetooth Low Energy (now known as Bluetooth Smart)
Bluetooth Smart	Formerly known as 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
i.e.	*Id est*, that is
I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
IDE	Integrated Development Environment
IRK	Identity Resolving Key
kJ	Kilo Joules
LM	Link Manager
NVM	Non Volatile Memory
PC	Personal Computer
PIO	Programmable Input Output
PnP	Plug and Play
PWM	Pulse Width Modulation
RR-interval	R wave to R wave interval (inverse of heart rate)
UUID	Universally Unique Identifier