CSR

Push every boundary.™

CSR µEnergy®

Health Thermometer

Application Note

Issue 5

Document History

Revision	Date	History
1	14 JUN 12	Original publication of this document
2	08 FEB 13	Updated to reference CSR101x devices and for SDK 2.1
3	19 FEB 13	Updated current consumption values
4	10 MAY 13	Updated for SDK 2.2; connection parameter update
5	03 FEB 14	Updated to new CSR branding and changes to connection parameter update

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 Health Thermometer application supplied with the CSR µEnergy® Software Development kit (SDK) and provides guidance to developers on how to customise the on-chip application.

The application demonstrates the Health Thermometer profile which is specified by the Bluetooth SIG.

1.1. Application Overview

1.1.1. Profiles Supported

The Health Thermometer application supports the Health Thermometer profile.

1.1.1.1. Health Thermometer Profile

The Health Thermometer profile is used to enable a data collection device to obtain temperature measurement and related data from a thermometer sensor that exposes the Health Thermometer service.

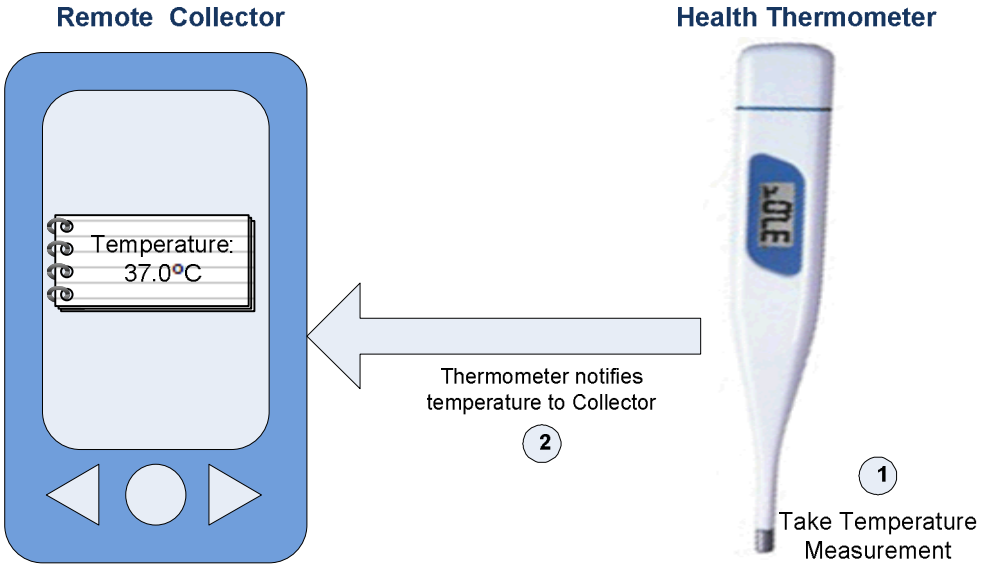


Figure 1.1: Health Thermometer Profile

The Health Thermometer profile defines two roles, described in Table 1.1:

Role	Description
Thermometer	The Thermometer is a device that measures the temperature.
Collector	The Collector is a device that receives the temperature measurement and related data from a Thermometer.

Table 1.1: Health Thermometer Profile Roles

For more information about the Health Thermometer profile, see *Health Thermometer Profile Specification Version 1.0*.

1.1.2. Application Topology

The Health Thermometer application implements the Health Thermometer profile in the Thermometer role, see Table 1.2:

Role	Health Thermometer 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:

- Health Thermometer (Version 1.0)
- Device Information (Version 1.1)
- Battery (Version 1.0)
- GAP
- GATT

The Health Thermometer profile mandates two services: Health Thermometer and Device Information. GAP and GATT services are mandated by *Bluetooth Core Specification Version 4.1*. Battery service is optional, see Figure 1.2.

For more information on Health Thermometer, Device information and Battery services, see *Health Thermometer 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 Health Thermometer application does not support any characteristic for GATT service. See *Bluetooth Core Specification Version 4.1* for more information.

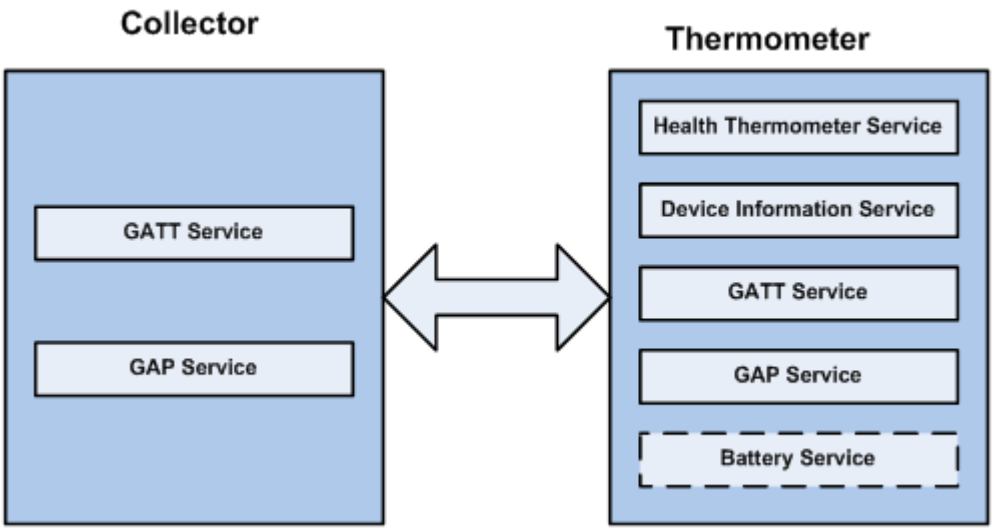


Figure 1.2: Primary Services

2. Using the Application

This section describes how the Health Thermometer application can be used with the CSR µEnergy Profile Demonstrator application.

2.1. Demonstration Kit

The application can be demonstrated using the following components:

Component	Hardware	Application
Thermometer	CSR10x0 Development Board	Health Thermometer Application
Collector	CSR µEnergy BT4.0 USB dongle	CSR µEnergy Profile Demonstrator Application

Table 2.1: Demonstration Components

Note:

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

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

2.1.1. Thermometer

The SDK is used to build and download the Health Thermometer 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.

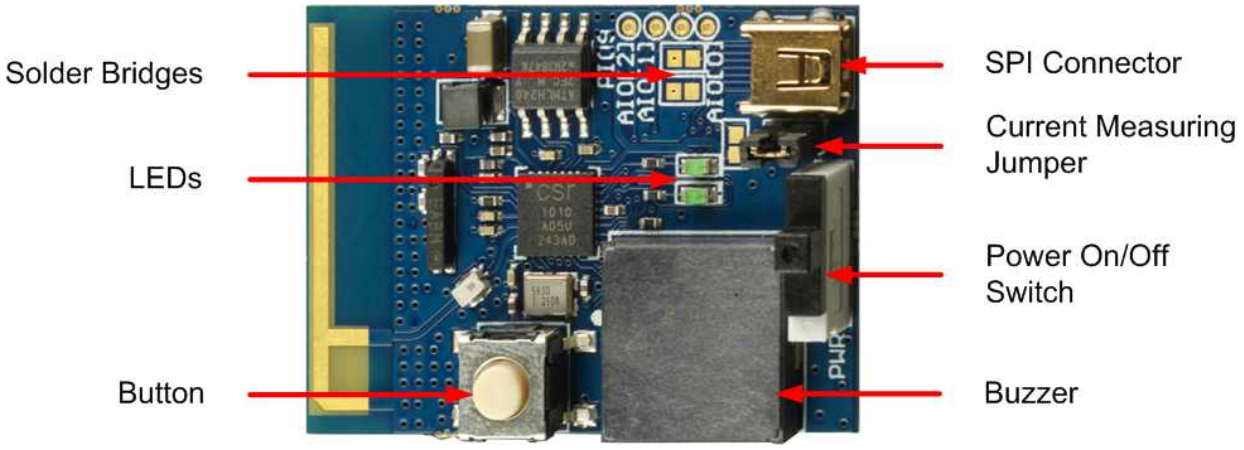


Figure 2.1: CSR10x0 Development Board

2.1.1.1. User Interface

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

Button Behaviour

- In Idle state, a **Short button press** starts advertising.
- In Connected state, a **Short button press** disconnects the connection.
- An **Extra Long button press** disconnects the link if present, removes bonding and starts advertising.

See Figure 2.2 for device behaviour.

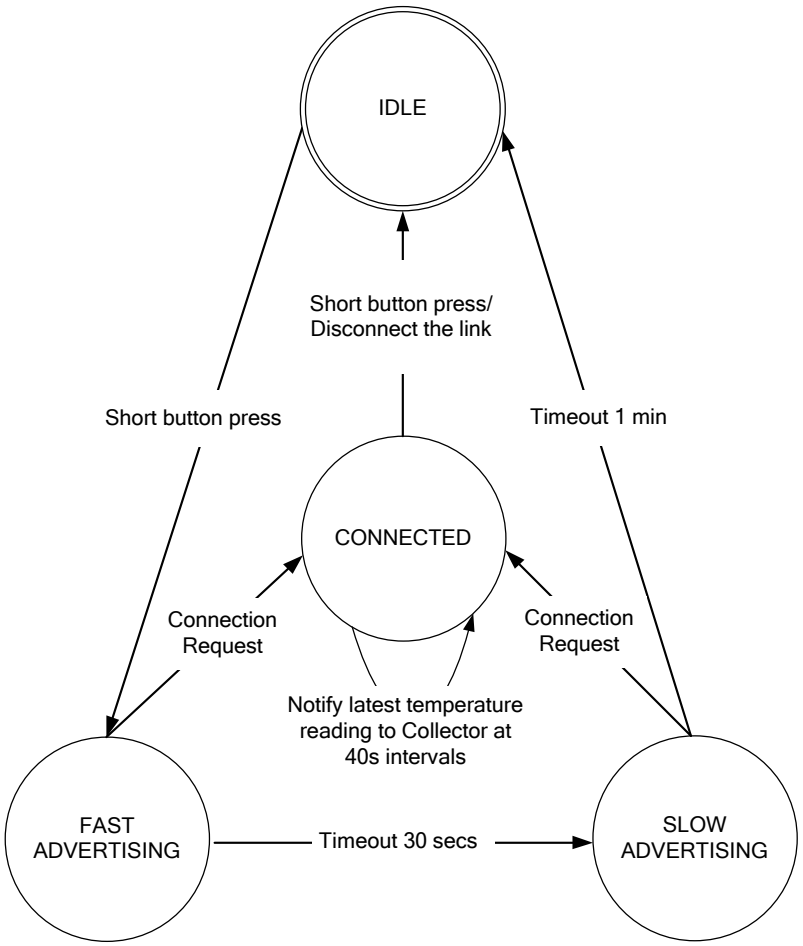


Figure 2.2: Device Behaviour

Note:

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

Buzzer Behaviour

- A single short beep on a **Short button press** indicates the execution of the short button press operation as described earlier.
- 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. Collector

2.1.2.1. CSR µEnergy BT4.0 USB Dongle

The CSR µEnergy BT4.0 USB dongle must be used with the CSR µEnergy Profile Demonstrator application to complete the Bluetooth Smart link between two devices. To use the USB dongle, 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 2.3: 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.

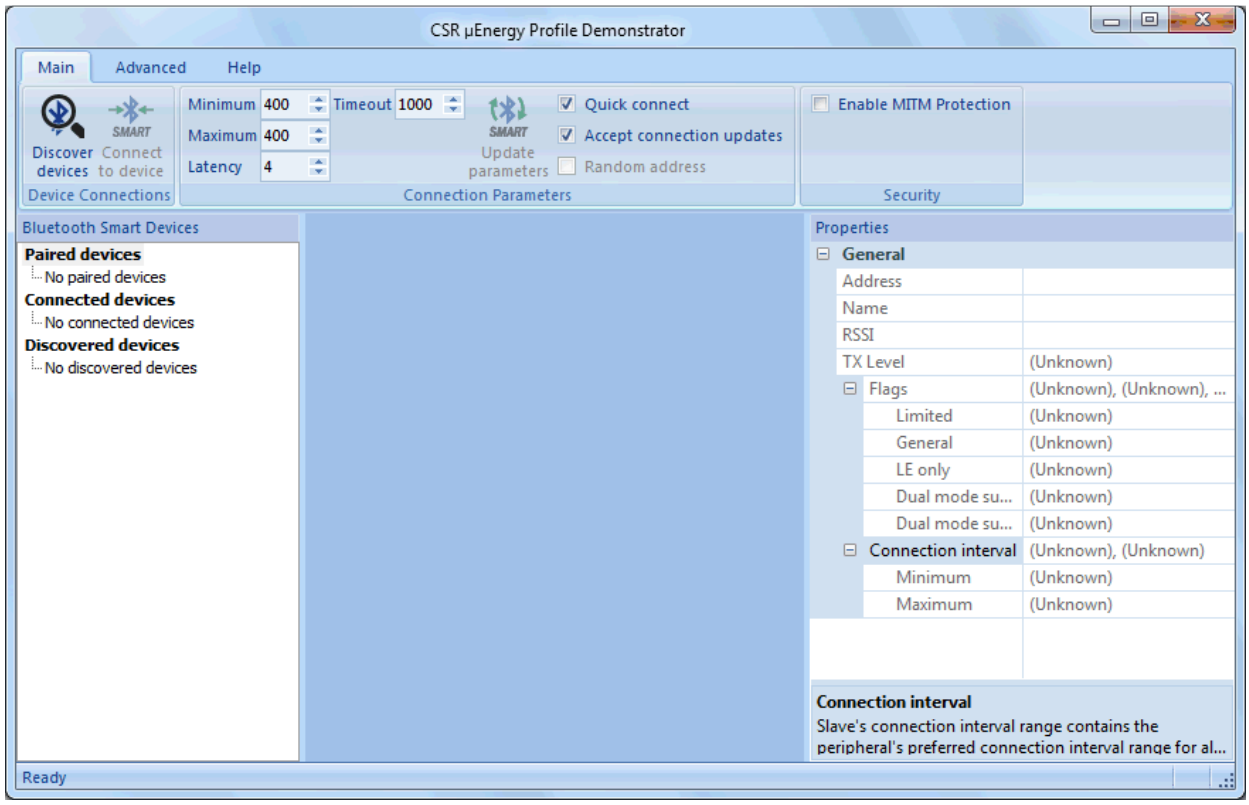


Figure 2.4: CSR µEnergy Profile Demonstrator

2.2. Demonstration Procedure

1. Switch on the development board to start advertising.
2. Click on the **Discover devices** button in the CSR µEnergy Profile Demonstrator application window. The software searches for Bluetooth Smart devices and lists all the discovered devices on the left hand side of the application window, see Figure 2.5.

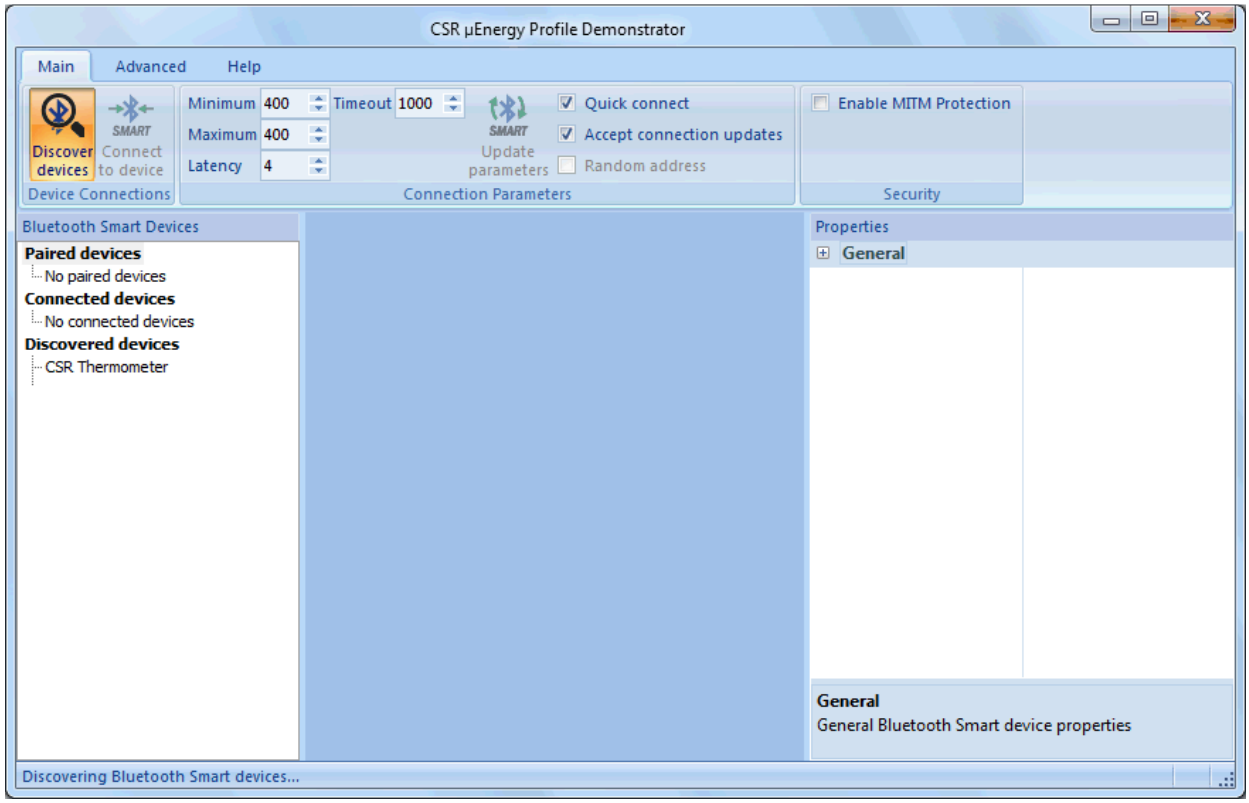


Figure 2.5: Health Thermometer Device Discovered

3. When the device labelled **CSR Thermometer** appears, select it to display the device address on the right hand side of the screen.
4. Enter the preferred connection parameters from Table 6.3 in the **Connection Parameters** tab. Click on the **Connect to device button to connect to this device**, see Figure 2.5.
5. The CSR µEnergy Profile Demonstrator application displays a tabbed pane corresponding to different services supported by the application, see Figure 2.6.

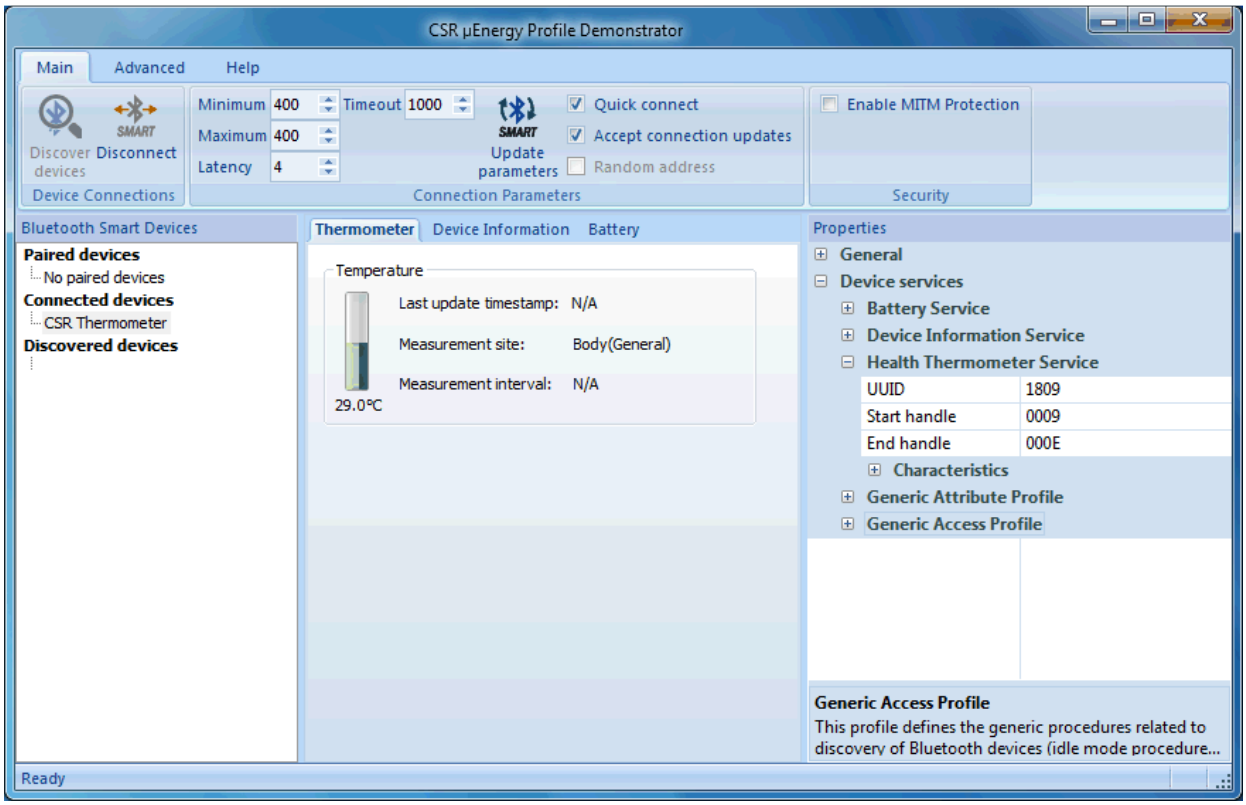


Figure 2.6: Device Connected

6. The **Thermometer** tab in the **CSR µEnergy Profile Demonstrator** window shows the received temperature reading, see Figure 2.6.

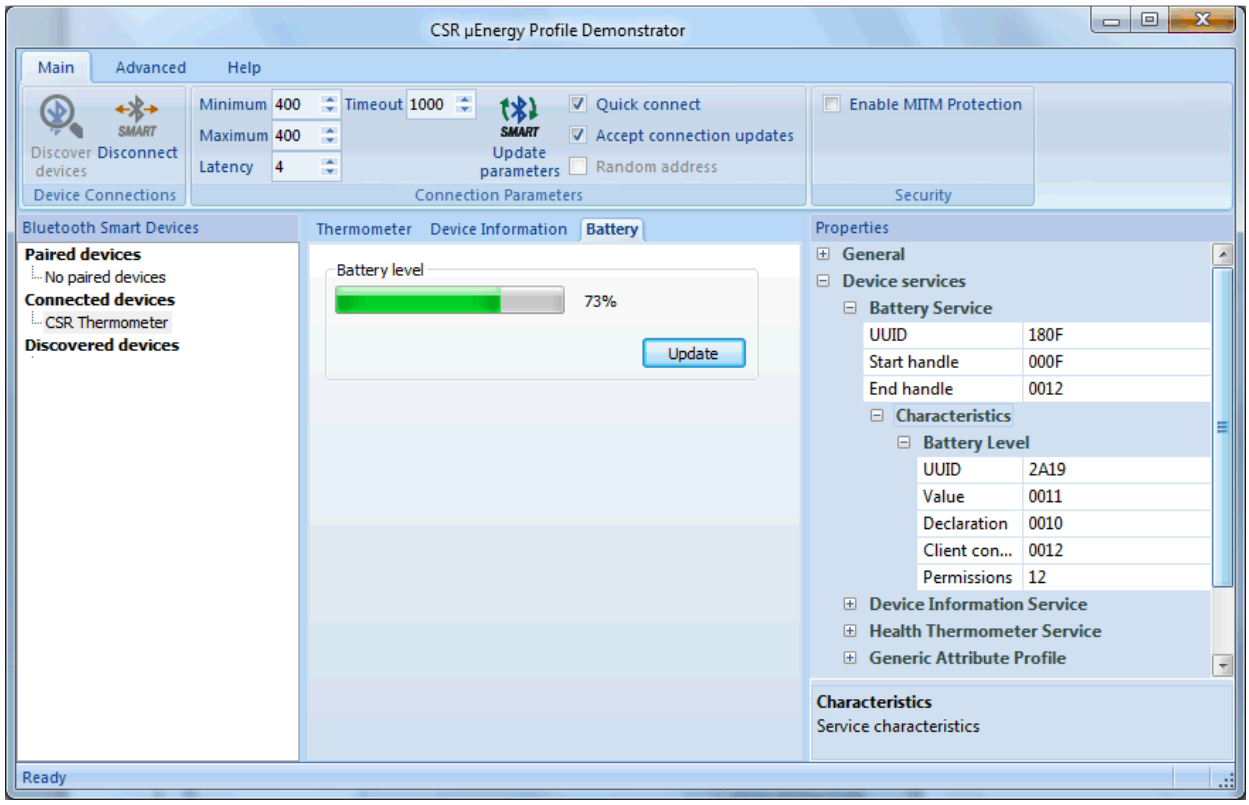
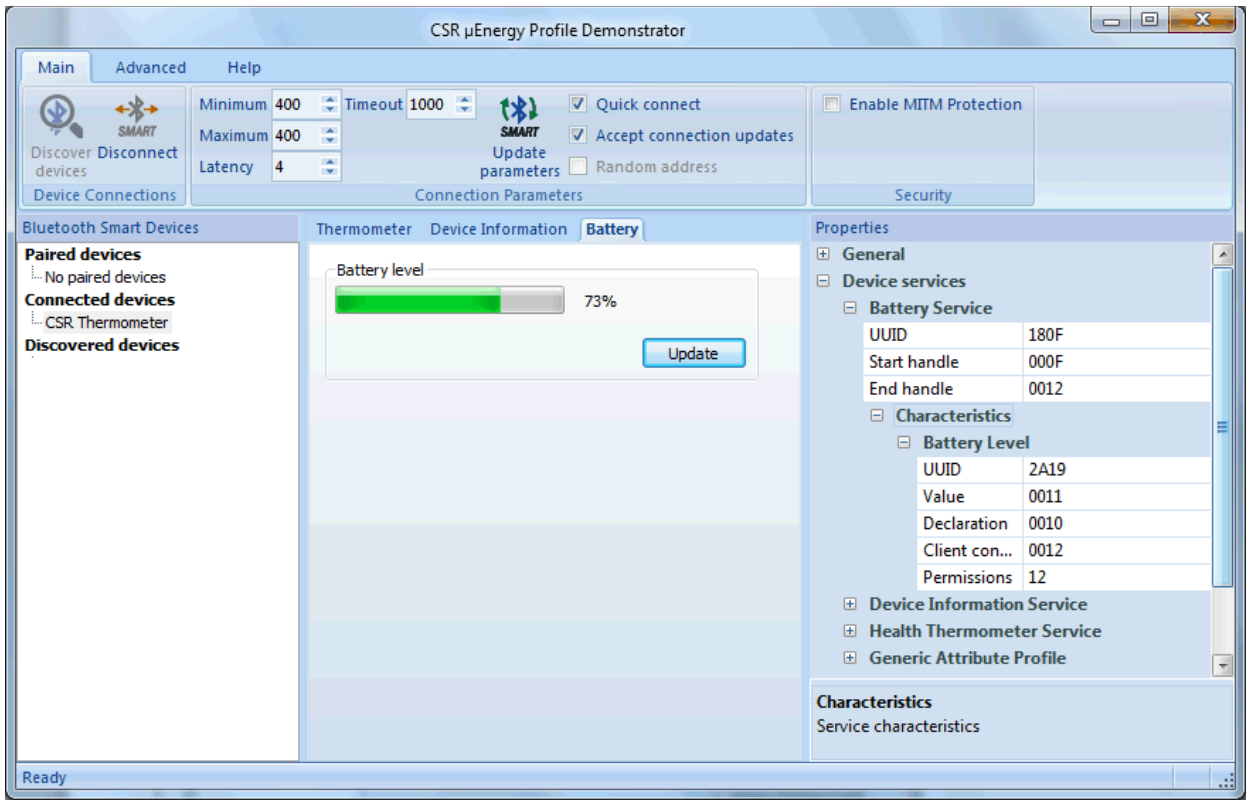


Figure 2.7: Battery Level



7. Figure 2.7 displays the battery level and the **Device Information** tab, see Figure 2.8, displays the device information characteristics of the application.

Note:

The battery level may fluctuate depending on the connection state.

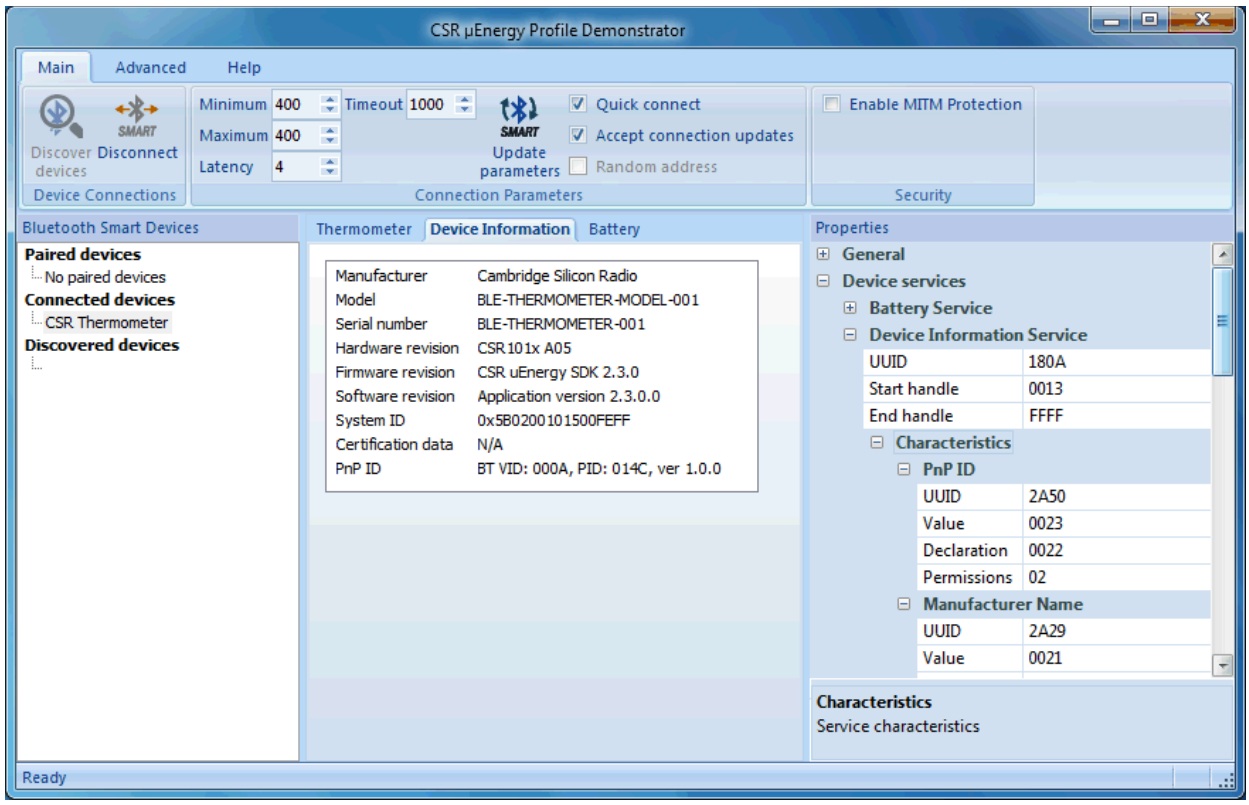


Figure 2.8: Device Information Service

8. Click the **Disconnect** button to disconnect the Bluetooth low energy link between the development board and the USB dongle.

3. Application Structure

3.1. Source Files

Table 3.1 lists the source files and their purpose.

File name	Purpose
health_thermometer.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.
ht_gatt.c	Implements routines for triggering advertisement procedures.
health_thermo_service.c	Implements routines for handling read/write access indications and other procedures of the Health Thermometer service.
battery_service.c	Implements routines required for the Battery service e.g. reading Battery Level, notifying it to the remote device and handling access indications on the Battery service specific ATT attributes.
dev_info_service.c	Implements routines required for the Device Information service e.g. handling read/write access indications on the Device Information service specific ATT attributes.
gap_service.c	Implements routines for the GAP service e.g. handling read/write access indication on the GAP service characteristics, reading/writing device name on NVM etc.
ht_hw.c	Implements routines for hardware initialisation, button press handling and generating beeps of different types like short beep, long beep etc.
nvm_access.c	Implements the NVM read/write routines.

Table 3.1: Source Files

3.2. Header Files

Table 3.2 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 Health Thermometer 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 the Battery service and related characteristics.
dev_info_service.h	Contains prototypes of the externally referred functions defined in the dev_info_service.c file.
dev_info_uuids.h	Contains macros for UUID values of the Device Information service.

File name	Purpose
gap_conn_params.h	Contains macro definitions for fast/slow advertising, preferred connection parameters, maximum number of connection parameter update requests 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 of the GATT service.
health_thermometer.h	Contains application data structure definition and prototypes of externally referred functions defined in the health_thermometer.c file.
ht_gatt.h	Contains timeout values for fast/slow advertising and prototypes of externally referred functions defined in the ht_gatt.c file.
ht_hw.h	Contains prototypes of externally referred hardware routines of the ht_hw.c file.
health_thermo_service.h	Contains prototypes of externally referred functions defined in health_thermo_service.c.
health_thermo_uuids.h	Contains macro definitions for UUID values of the Health Thermometer service and related characteristics.
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 the *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 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
health_thermo_service_db.db	Contains information related to Health Thermometer service characteristics, their descriptors and values. See Table B.4 for Health Thermometer service characteristics.

Table 3.3: Database Files

4. Code Overview

This section describes 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, application data structures and hardware
- 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 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 Health Thermometer application starts advertising.
- `GATT_ACCESS_IND`: This indication event is received when the remote Collector tries to access an ATT characteristic managed by the application.
- `GATT_CHAR_VAL_IND_CFM`: This confirmation event confirms the successful reception of temperature reading by the remote Collector device.

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 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 are not compliant 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 2.4 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 of *Bluetooth Core Specification Version 4.1*. In the case of a successful completion of the pairing procedure, the Health Thermometer application is bonded with the collector and the 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.

- SM_PAIRING_AUTH_IND: This indication is received when the remote connected device initiates pairing. The application can either accept or reject the pairing request from the peer device. The application shall reject the pairing request if it is already bonded to a collector device to prevent any new device disguising itself as one previously bonded to the Health Thermometer application.

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 starts advertising. 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 the connection procedure. When the application stops advertisements to change the advertising parameters or to save power, this signal confirms the successful cessation of advertisements by the Health Thermometer 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 parameter.
- LM_EV_DISCONNECT_COMPLETE: This event is received on a 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 6.4.

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 battery voltage threshold. If connected and notifications are configured, the Health Thermometer 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

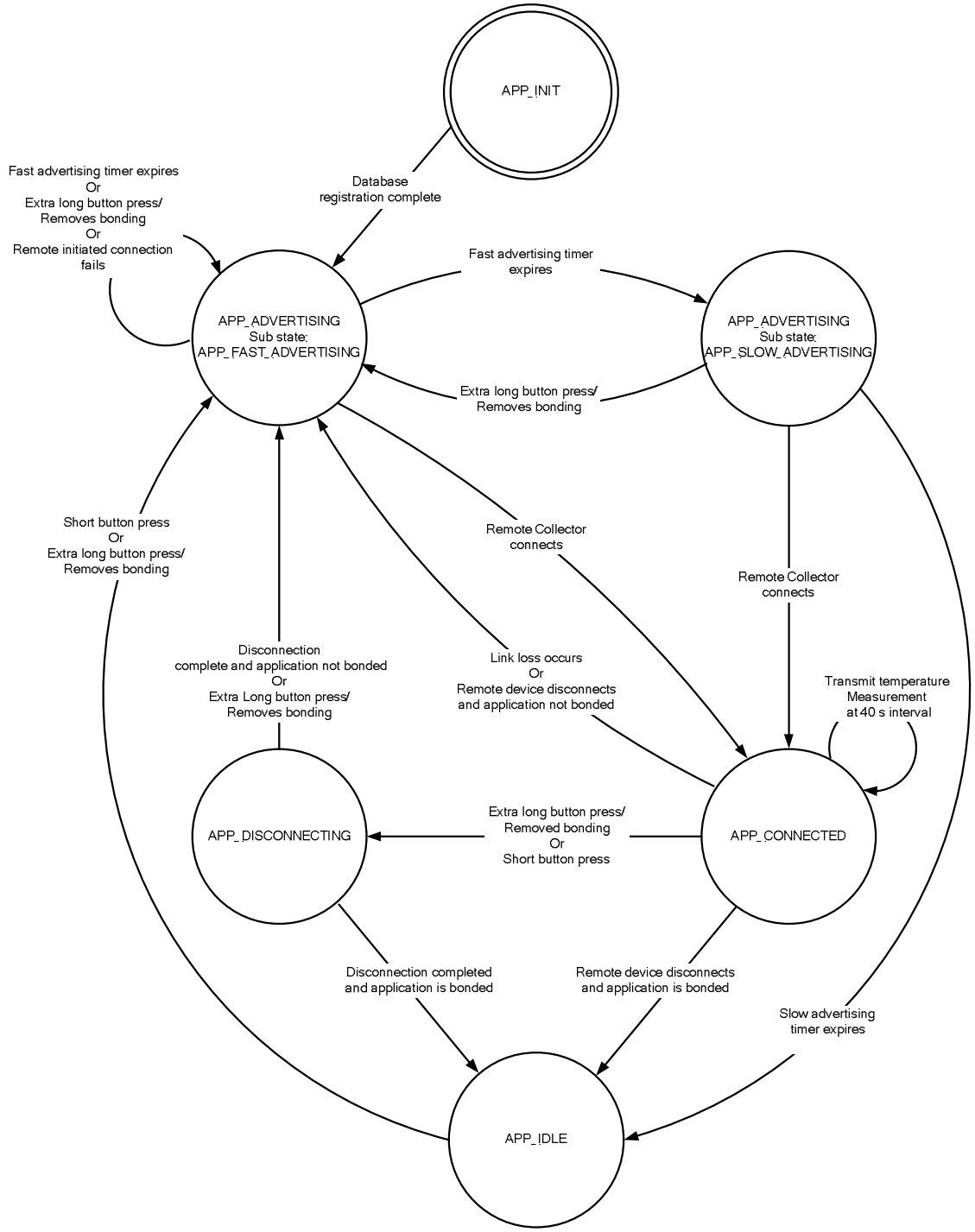


Figure 4.1: Internal State Machine Diagram

The Health Thermometer application has five internal states, see Figure 4.1:

4.2.1. APP_INIT

When the application is powered on or the chip resets, it enters this state. The application initialises different modules in this state and registers the database with the firmware. When the application receives a successful confirmation signal for the database registration it enters the APP_ADVERTISING state.

4.2.2. APP_IDLE

The Health Thermometer application is not connected to any Collector. The application sounds a long beep while entering this state.

- On a **Short button press**, the application triggers advertisements and enters the APP_ADVERTISING state.
- On an **Extra Long button press**, the application removes the bonding information, clears the white list and enters the APP_ADVERTISING state. See *Bluetooth Core specification Version 4.1* for more information on white list.

4.2.3. APP_ADVERTISING

The Health Thermometer application sends advertisements in this state 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 a connection is made, the application enters the APP_SLOW_ADVERTISING sub state. 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 application enters the APP_IDLE state.
- The application ignores a **Short button press** in this state.
- On an **Extra Long button press**, the application stops advertisements, removes bonding and restarts advertising without any white list. See *Bluetooth Core specification Version 4.1* for more information on white list.

4.2.4. APP_CONNECTED

The Health Thermometer application is connected to the remote Collector and sends a Temperature reading (see Figure B.1 for the data format) to the bonded Collector at a regular interval, see Table 6.5.

- On a **Short button press**, the application disconnects the link and enters the APP_DISCONNECTING state. The application indicates this by sounding a single short beep.
- On an **Extra Long button press**, the application removes bonding information, clears white list, disconnects the link and enters the APP_DISCONNECTING state. See *Bluetooth Core specification Version 4.1* for more information on white lists.
- In the case of a link loss, the application enters the APP_ADVERTISING state.
- In the case of remote triggered disconnection, if the Health Thermometer application is bonded to a Collector device, the application enters the APP_ADVERTISING state; otherwise it enters the APP_IDLE state.

4.2.5. APP_DISCONNECTING

The Health Thermometer application waits for a disconnect confirmation for the disconnection initiated by it.

- When it receives the disconnect confirmation, it checks if it is bonded to any Collector.
  - If the application is bonded, it enters the APP_IDLE state and waits for user activity.
  - If the application is not bonded, it enters the APP_ADVERTISING state.

- On an **Extra Long button press**, the application removes the bonding information, clears the white list and remains in the same state.

5. NVM Memory Map

The applications can store data in the NVM to prevent data loss in the event of a power off or chip panic. The Health Thermometer application uses the following memory map for NVM.

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)
Temperature Characteristic Client Configuration Descriptor	uint16	1	37

Table 5.3: NVM Memory Map for Health Thermometer Service

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
Battery Level characteristic Client Configuration Descriptor	uint16	1	38

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. Advertising Parameters

The Health Thermometer application uses the parameters in Table 6.1 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 advertising parameter range.

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

Table 6.1: Advertisement Parameters

6.2. Advertisement Timers

The Health Thermometer 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 file `ht_gatt.h`.

Timer Name	Timer Values
Fast Advertisement Timer Value	30 s
Slow Advertisement Timer Value	1 min

Table 6.2: Advertisement Timers

6.3. Connection Parameters

The Health Thermometer application uses connection parameters listed in Table 6.3 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 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 6.4 for the connection update procedure.

Parameter Name	Parameter Value	Profile Demonstrator Configuration
Minimum Connection Interval	500 ms	400
Maximum Connection Interval	500 ms	400
Slave Latency	4 intervals	4
Supervision Timeout	10 s	1000

Table 6.3: Connection Parameters (Default)

When connecting to Apple products, the connection parameters listed in Table 6.4 should be used.

Parameter Name	Parameter Value	Profile Demonstrator Configuration
Minimum Connection Interval	380 ms	304
Maximum Connection Interval	400 ms	320
Slave Latency	4 intervals	4
Supervision Timeout	6000 ms	600

Table 6.4: Preferred Connection Parameters for Apple Products

6.4. Connection Parameter Update

The Health Thermometer application can request the remote 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 Server changes the connection parameters. Upon connection establishment with the Server, 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 TGAP(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 Health Thermometer 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 Health Thermometer application.

The remote Collector may or may not accept the requested parameters. If the remote server rejects the new requested parameters, the application again requests an update after 30 seconds. The macro for this time value is defined in file `health_thermometer.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 remote Collector 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.

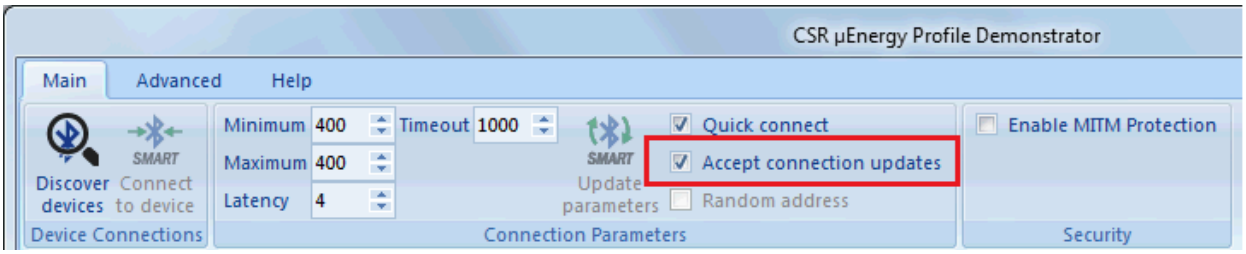


Figure 6.1: Accept Connection Parameters

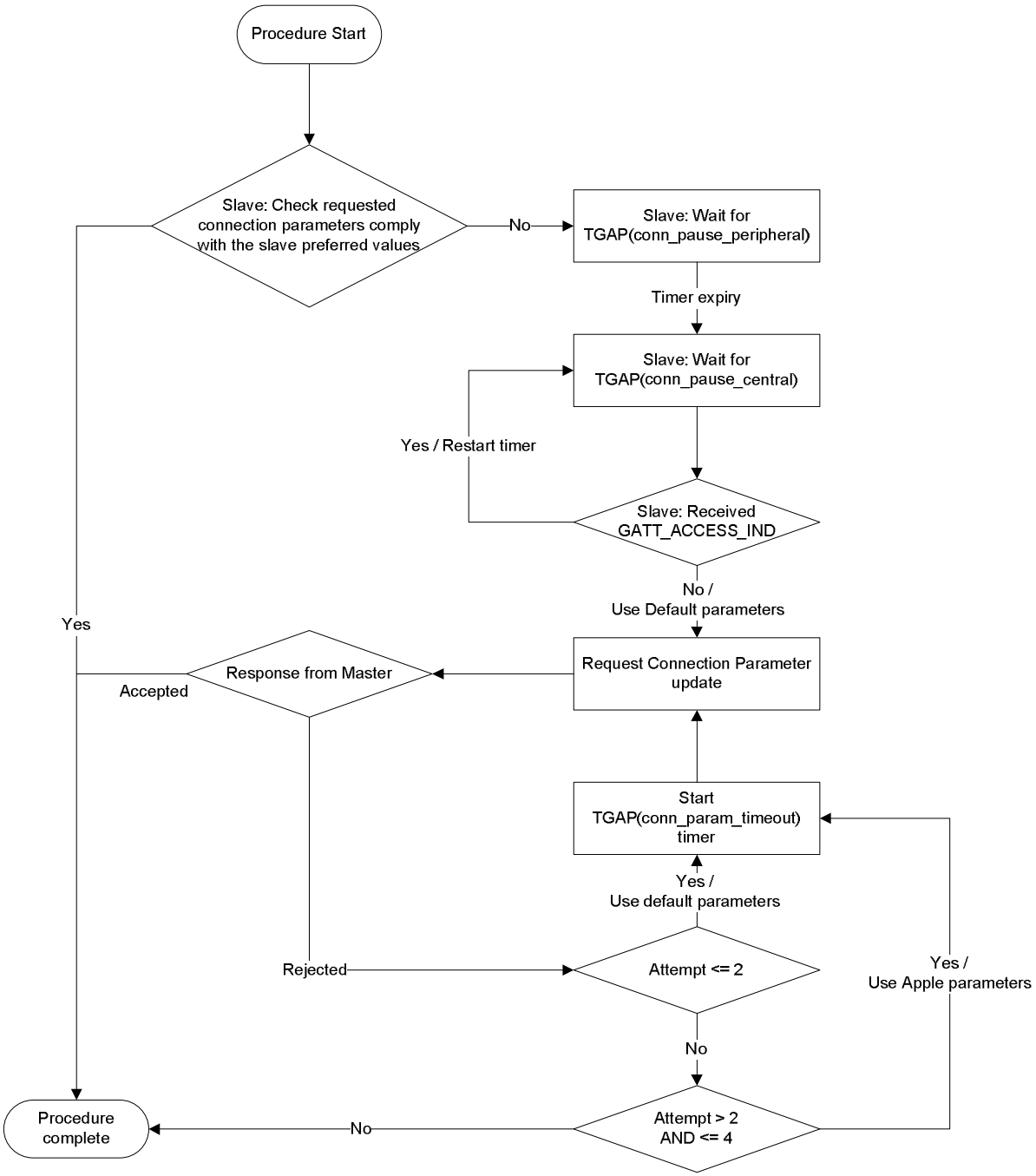


Figure 6.2: Connection Parameter Update Procedure

6.5. Device Name

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

6.6. Buzzer

The Health Thermometer application uses the buzzer to indicate different states and events to the user. The user can enable or disable this buzzer as required by the application. The buzzer should be disabled while taking current consumption readings; see section 7. The macro `ENABLE_BUZZER`, defined in file `ht_hw.h`, is used for enabling the buzzer. To disable the buzzer, comment out the macro `ENABLE_BUZZER` in file `ht_hw.h`.

6.7. Measurement Transmission Interval

The Health Thermometer application transmits temperature measurement at regular intervals. The macro for this interval is defined in `health_thermometer.c` file.

Macro Name	Value
`HT_TEMP_MEAS_TIME`	40 seconds

Table 6.5: Measurement Transmission Interval

6.8. Non-Volatile Memory

The Health Thermometer 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.

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 exercises the LEDs or sounds the buzzer should be disabled before measuring the actual current consumed.

The setup used while measuring current consumption is described in 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 and with Channel Map Updates disabled, 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 Health Thermometer 2. Wait for 5 s 3. Take the measurement	471 µA	▪ Advertising Interval: 60 ms ▪ Advertisement Data length: 31 ▪ Measurement Time Duration: 20 s
Slow Advertisements	1. Switch on the Health Thermometer 2. Wait for 40 s 3. Take the measurement	28 µA	▪ Advertising Interval: 1.28 s ▪ Advertisement Data length: 31 ▪ Measurement Time Duration: 40 s
Connected Active	1. Connect to the Collector 2. Wait for 60 s 3. Health Thermometer sends temperature measurement once every 40 s 4. Take the measurement	14 µA	▪ Connection parameters Minimum Connection Interval: 500 ms Maximum Connection Interval: 500 ms Slave Latency: 4 ▪ Measurement Time Duration: 60 s
Notes: ▪ Average current consumption is measured at 3.0 V ▪ Ammeter used: Agilent 34411A ▪ LED and buzzer disabled ▪ 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

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 and 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 Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Battery Level	0x0011	Read, Notify	Application	Security Mode 1 and Security Level 2	Current battery level
Battery Level-Client Configuration Descriptor	0x0012	Read, Write	Application	Security Mode 1 and Security Level 2	Current client configuration for “Battery level” characteristic

Table B.1: Battery Service Database

For more information on Battery Service and Security Permissions, see *Bluetooth Core Specification Version 4.1*.

B.2 Device Information Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Serial Number String	0x0015	Read	Firmware	Security Mode 1 and Security Level 2	"BLE-THERMOMETER-001"
Model Number String	0x0017	Read	Firmware	Security Mode 1 and Security Level 2	"BLE-THERMOMETER-MODEL-001"
System ID	0x0019	Read	Application	Security Mode 1 and Security Level 2	Organisationally Unique identifier is 0x00025b Manufacturer Identifier depends on the device address
Hardware Revision String	0x001b	Read	Firmware	Security Mode 1 and Security Level 2	<Chip Identifier>
Firmware Revision String	0x001d	Read	Firmware	Security Mode 1 and Security Level 2	<SDK Version>
Software Revision String	0x001f	Read	Firmware	Security Mode 1 and Security Level 2	<Application Version>
Manufacturer Name String	0x0021	Read	Firmware	Security Mode 1 and Security Level 2	"Cambridge Silicon Radio"

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
PnP ID	0x0023	Read	Firmware	Security Mode 1 and Security Level 2	Vendor Id source is BT Vendor Id is 0x000a Product Id is 0x014c Product Version is 1.0.0

Table B.2: Device Information Service Database

For more information on Device Information Service, see *Device Information Service Specification Version 1.1*.

For more information on Security Permissions, see *Bluetooth Core Specification Version 4.1*.

B.3 GAP Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Device Name	0x0003	Read, Write	Application	Security Mode 1 and Security Level 2	Device name Default : "CSR Thermometer"
Appearance	0x0005	Read	Firmware	Security Mode 1 and Security Level 1	Generic Thermometer – 0x0300
Peripheral preferred connection parameters	0x0007	Read	Firmware	Security Mode 1 and Security Level 1	Min connection interval - 500 ms Max connection interval – 500 ms Slave latency - 4 Connection timeout - 10 s

Table B.3: GAP Service Database

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

B.4 Health Thermometer Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Temperature Measurement	0x000b	indicate	Application	Security Mode 1 and Security Level 2	Current temperature, see Figure B.1 for data format
Temperature - Client Characteristic Configuration descriptor	0x000c	Read, Write	Application	Security Mode 1 and Security Level 2	Current client configuration for the "Temperature Measurement" characteristic
Temperature Type	0x000e	Read	Firmware	Security Mode 1 and Security Level 2	Body – 0x02

Table B.4: Health Thermometer Service Database

See Health Thermometer Service Specification Version 1.0 for more information on Health Thermometer service. For more information on Security Permissions, see *Bluetooth Core Specification Version 4.1*.

Figure B.1 shows the data format for the Temperature Measurement characteristic. For more information on the Temperature Measurement characteristic see the Bluetooth SIG Developer Portal.

Figure B.1: Temperature Measurement Data Format

Note:

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 `health_thermometer.c` file. See section 4.1.2.2 for more information on 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 Health Thermometer application adds the following fields in the Advertising Data:

Advertising Data Field	Contents
Flags	The Health Thermometer 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 Health Thermometer application adds 16-bit UUID of the following service: ▪ Health Thermometer
Device Appearance	Thermometer : `0x0300`
Tx Power	Current Tx power level
Device Name(1)	Device name (Default value : “CSR Thermometer”)
Note: (1) If the Device Name length is greater than the space left in the Advertising data field then the application adds it to the Scan Response data.	

Table C.1: Advertising Data Field

Appendix D Known Issues or Limitations

See the Health Thermometer application and SDK Release Notes.

Document References

Document	Reference
Bluetooth Core Specification Version 4.1	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Health Thermometer Profile Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Health Thermometer 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
Bluetooth SIG Developer Portal	http://developer.bluetooth.org/gatt/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 User Guide	CS-235358-UG

Terms and Definitions

ATT	Attribute
BLE	Bluetooth Low Energy (now known as Bluetooth Smart)
Bluetooth®	Set of wireless technologies providing audio and data transfer over short-range radio connections
Bluetooth Smart	Formerly known as Bluetooth Low Energy
CS	Configuration Store
CSR	Cambridge Silicon Radio
DIV	Diversifier
e.g.	exempli gratia, for example
EEPROM	Electrically Erasable Programmable Read Only Memory
etc	et cetera, and the rest, and so forth
GAP	Generic Access Profile
GATT	Generic Attribute Profile
IDE	Integrated Development Environment
i.e.	Id est, that is
I2C	Inter-Integrated Circuit
IRK	Identity Resolving Key
LED	Light Emitting Diode
LM	Link Manager
NVM	Non Volatile Memory
PC	Personal Computer
PIO	Programmable Input Output
PnP	Plug and Play
PTS	Profile Testing Suite
PWM	Pulse Width Modulation
Tx	Transmit
USB	Universal Serial Bus
UUID	Universally Unique Identifier