CSR μEnergy®

Glucose Sensor

Application Note

Issue 5

Document History

Revision	Date	History
1	11 MAY12	Original publication of this document
2	11 JUN 12	Minor editorial changes and updates to section 7
3	09 FEB 13	Updated to reference CSR101x devices, added information on Limited Discoverable mode and updated for SDK 2.1
4	20 FEB 13	Updated current consumption values
5	10 MAY 13	Updated for SDK 2.2; connection parameter update and flow control support

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.

Life Support Policy and Use in Safety-critical Compliance

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

Performance and Conformance

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

Contents

Tables, Figures and Equations

1. Introduction

This document describes the Glucose Sensor 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 Glucose profile which is specified by the Bluetooth SIG.

1.1. Application Overview

1.1.1. Profiles Supported

The Glucose Sensor application supports the Bluetooth Glucose profile.

1.1.1.1. Glucose Profile

The Glucose profile is used to obtain the glucose measurement and related data from a Glucose Sensor that exposes the Glucose service.

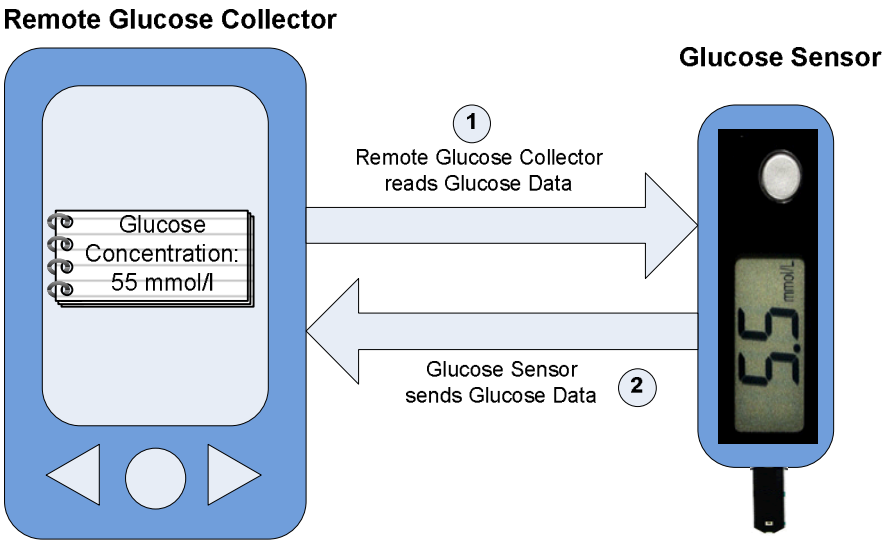

Figure 1.1: Glucose Profile

The Glucose profile defines two roles, described in Table 1.1:

Role	Description
Glucose Sensor	Glucose Sensor is the device that measures the glucose concentration level.
Glucose Collector	Glucose Collector is the device that receives glucose measurement and related data from the Glucose Sensor.

Table 1.1: Glucose Profile Roles

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

1.1.2. Application Topology

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

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

Table 1.3: Responsibilities

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

1.1.3. Services

This application exposes the following services:

- Glucose (Version 1.0)
- Device Information (Version 1.1)
- Battery (Version 1.0)
- GAP
- GATT

The Glucose profile mandates two services:

- Glucose
- Device Information

GAP and GATT services are mandated *by Bluetooth Core Specification Version 4.0*. Battery service is optional see Figure 1.2.

For more information on Glucose, Device information and Battery Services, see *Glucose Service Specification Version 1.0*, *Device Information Specification Version 1.1* and *Battery Service Specification Version 1.0*. For more information on GATT and GAP services, see *Bluetooth Core Specification Version 4.0*.

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

Note:

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

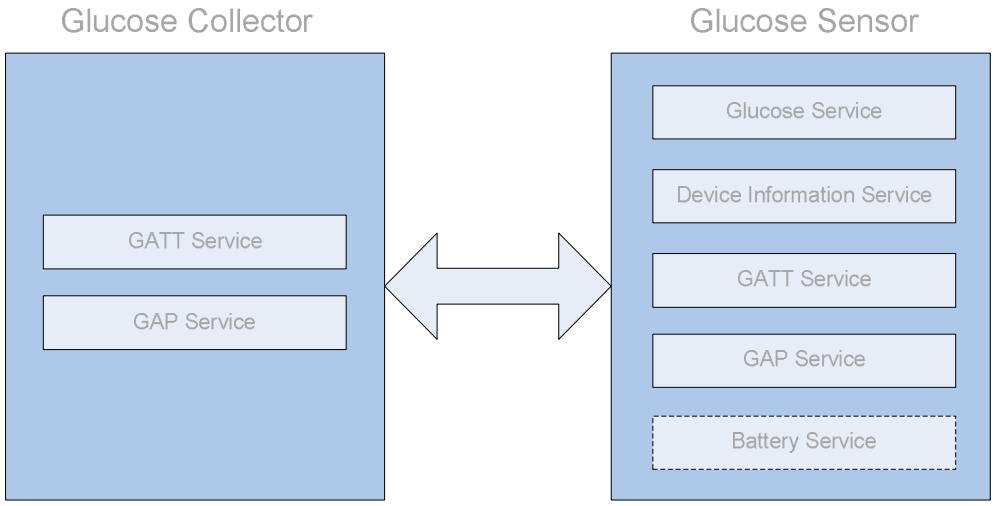

Figure 1.2: Primary Services

2. Using the Application

This section describes how the Glucose Sensor application can be used with the CSR µEnergy Profile Demonstrator application.

2.1. Demonstration Kit

Table 1.1 lists the components of the Glucose Sensor.

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

Table 2.1: Demonstration Components

Note:

Although the Glucose 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. Glucose Sensor

The SDK is used to build and download the Glucose Sensor application to the development board. See the *CSR µEnergy xIDE User Guide*.

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.

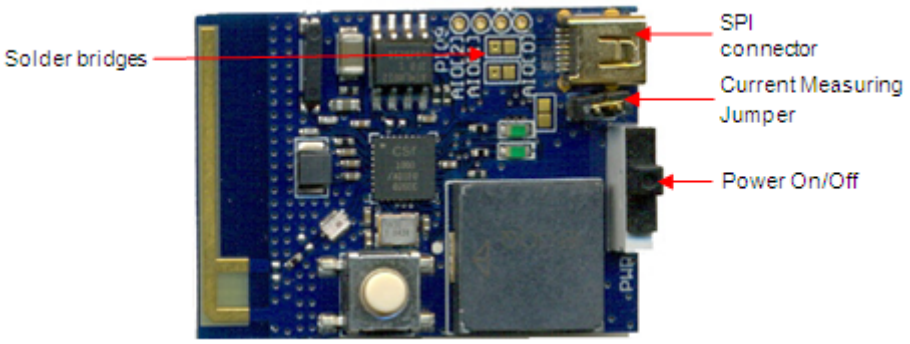

Figure 2.1: CSR10x0 Development Board

2.1.2. User Interface

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

Button Press Behaviour

- A **Short button press** generates a sample Glucose Measurement record and puts the Glucose sensor in limited discoverable mode if it is neither connected nor advertising. In limited discoverable mode, the Glucose sensor sends advertisements with the limited discoverable bit set in advertisement data and is discoverable to all the devices.
- An **Extra Long button press** disconnects the link if present, removes bonding and starts advertising.

Note:

A Glucose Measurement record consists of one Glucose measurement characteristic and an optional Glucose Measurement Context characteristic. See B.4 and *Glucose Service Specification Version 1.0* for more information.

For more information on limited discoverable mode, see *Bluetooth Core Specification v4.0.*

Figure 2.2 summarises the Glucose Sensor application device behaviour

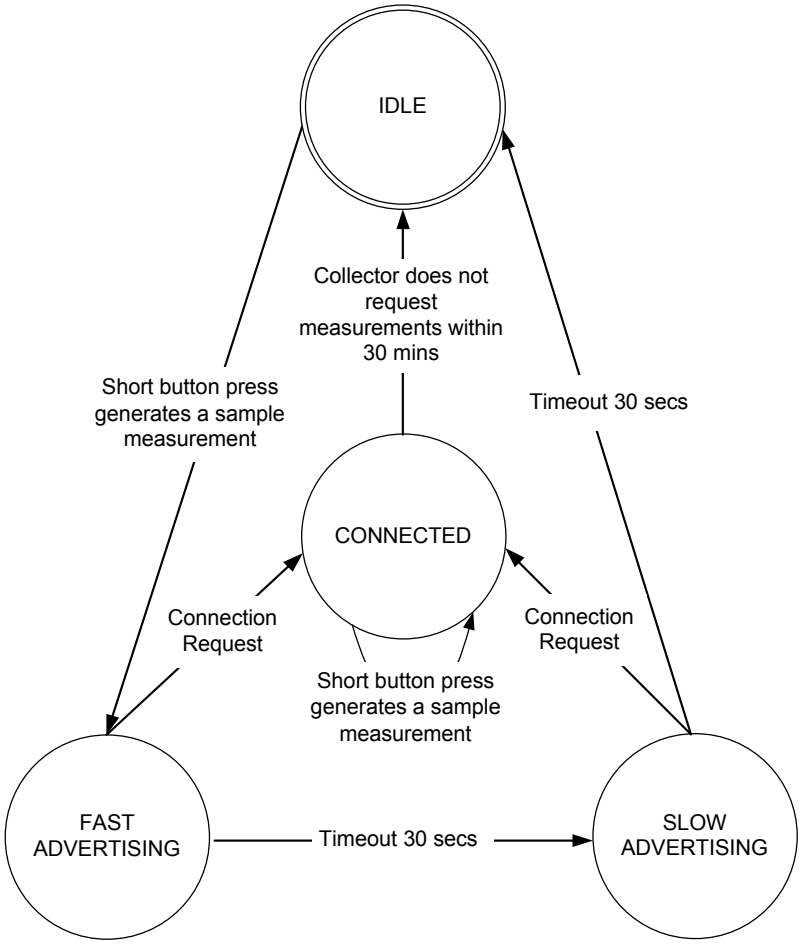

Figure 2.2: Device Behaviour

Note:

As the Glucose 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**.

Buzzer Behaviour

- A single short beep on a **Short button press** indicates the generation of a Glucose Measurement record.
- 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 non-connectable mode. This happens when the Glucose Sensor advertises for a certain time without a Glucose Collector connecting to it.

LED Blinking

The Glucose Sensor application indicates different states by blinking the LED. By default, LED blinking is disabled; see section 6.7 on how to enable the LED.

- Fast LED blinking indicates fast advertising.
- Slow LED blinking indicates the application is in the connected state.

2.1.3. Glucose Collector

2.1.3.1. CSR µEnergy BT4.0 USB dongle

The CSR µEnergy BT4.0 USB dongle can 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.

Figure 2.3: CSR µEnergy BT4.0 USB dongle

2.1.3.2. CSR µEnergy Profile Demonstrator Application

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

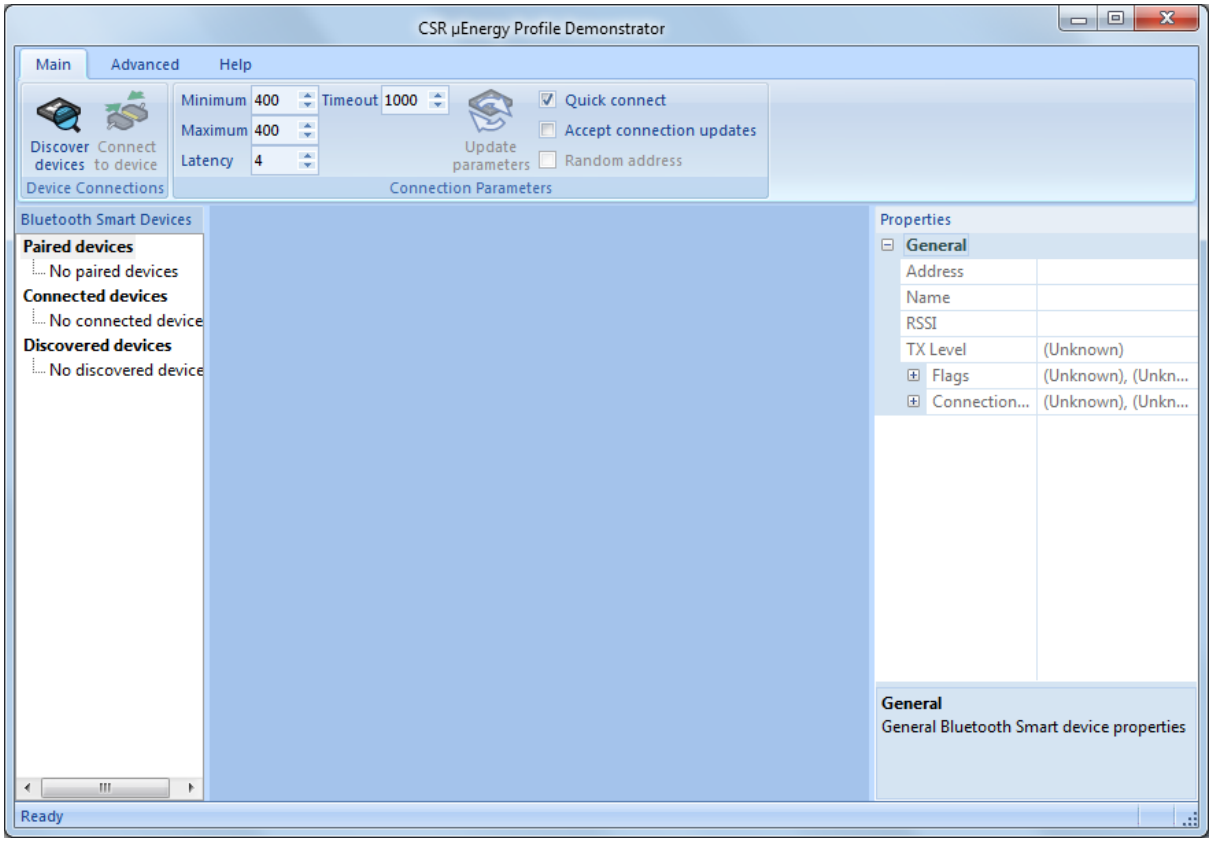

Figure 2.4: CSR µEnergy Profile Demonstrator

2.2. Demonstration Procedure

1. Switch on the development board. Press the button to generate the sample Glucose Measurement record and to trigger advertisements.
2. Click on the **Discover devices** button in the **CSR µEnergy Profile Demonstrator** 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.

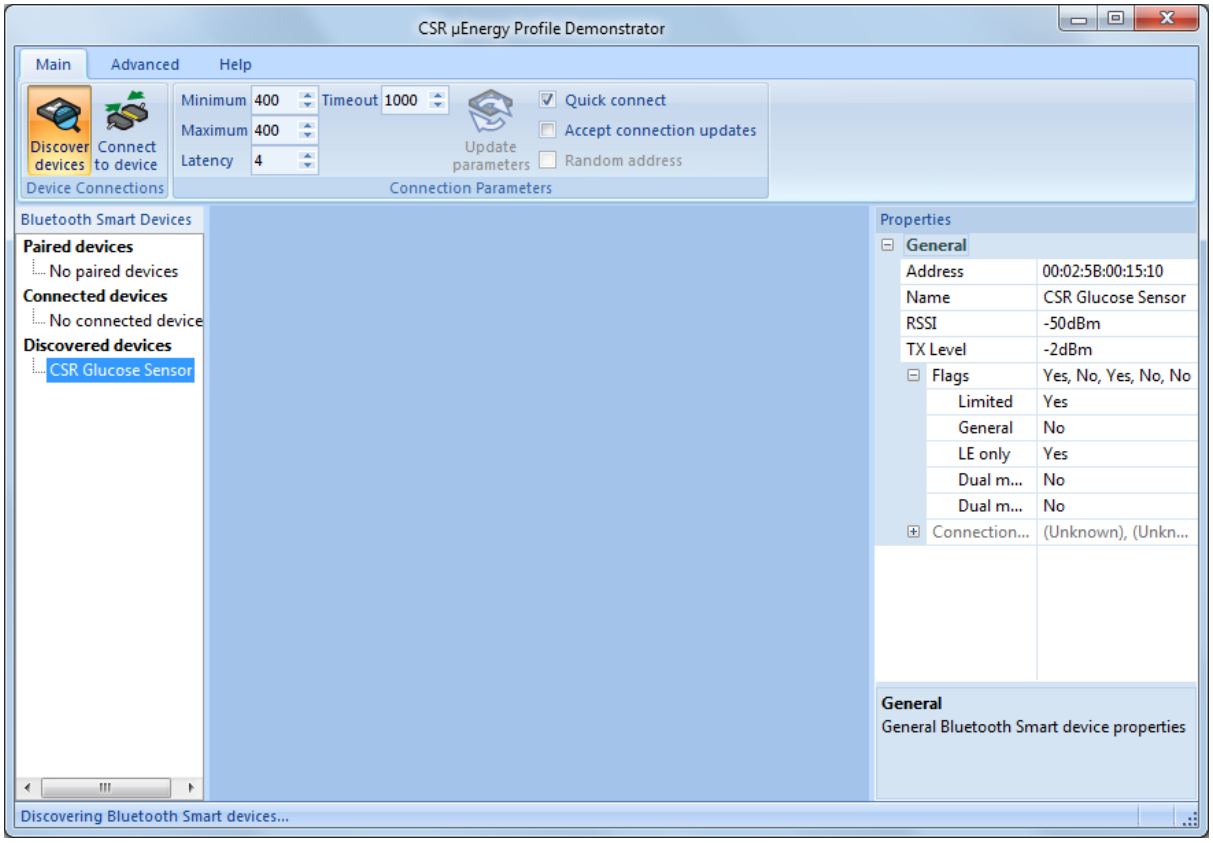

Figure 2.5: Glucose Sensor Device Discovered

When the device named **Glucose Sensor** appears, select it to display the device address on the right hand side of the screen.

Click on the **Connect to device** button to connect to this device.

The CSR µEnergy Profile Demonstrator application displays a tabbed pane corresponding to different services supported by the device, see Figure 2.6.

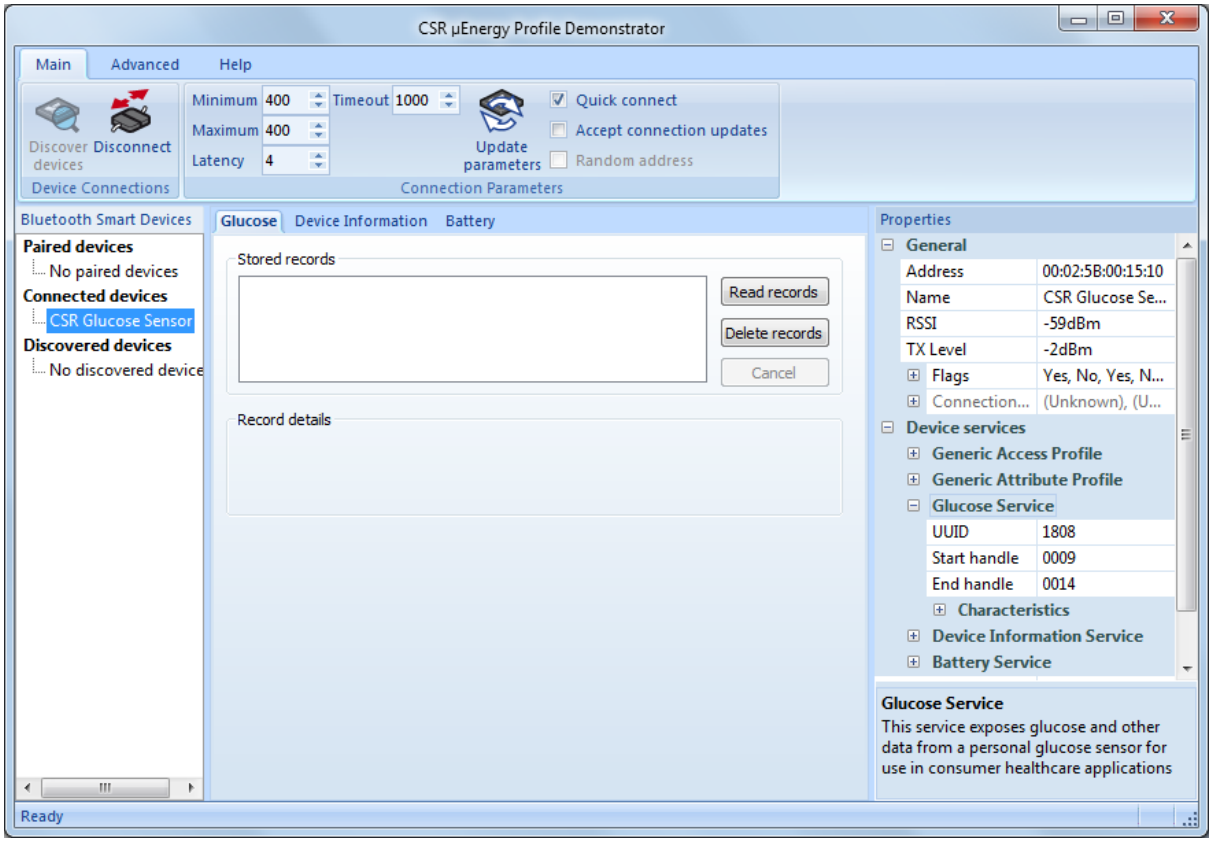

Figure 2.6: Device Connected

3. The **Glucose** tab in the **CSR µEnergy Profile Demonstrator** window has three buttons, see Figure 2.6.

   - **Read records**
     Clicking this button reads all the Glucose Measurement records present in the Glucose sensor, see Figure 2.7.
   - **Delete record**
     Clicking this button deletes all the Glucose Measurement records present in the Glucose sensor.
   - **Cancel**
     This button is enabled when the **Read records** or **Delete records** buttons are clicked and remains active while records are being read or deleted.

     Clicking **Cancel** aborts the ongoing reading or deletion operation.

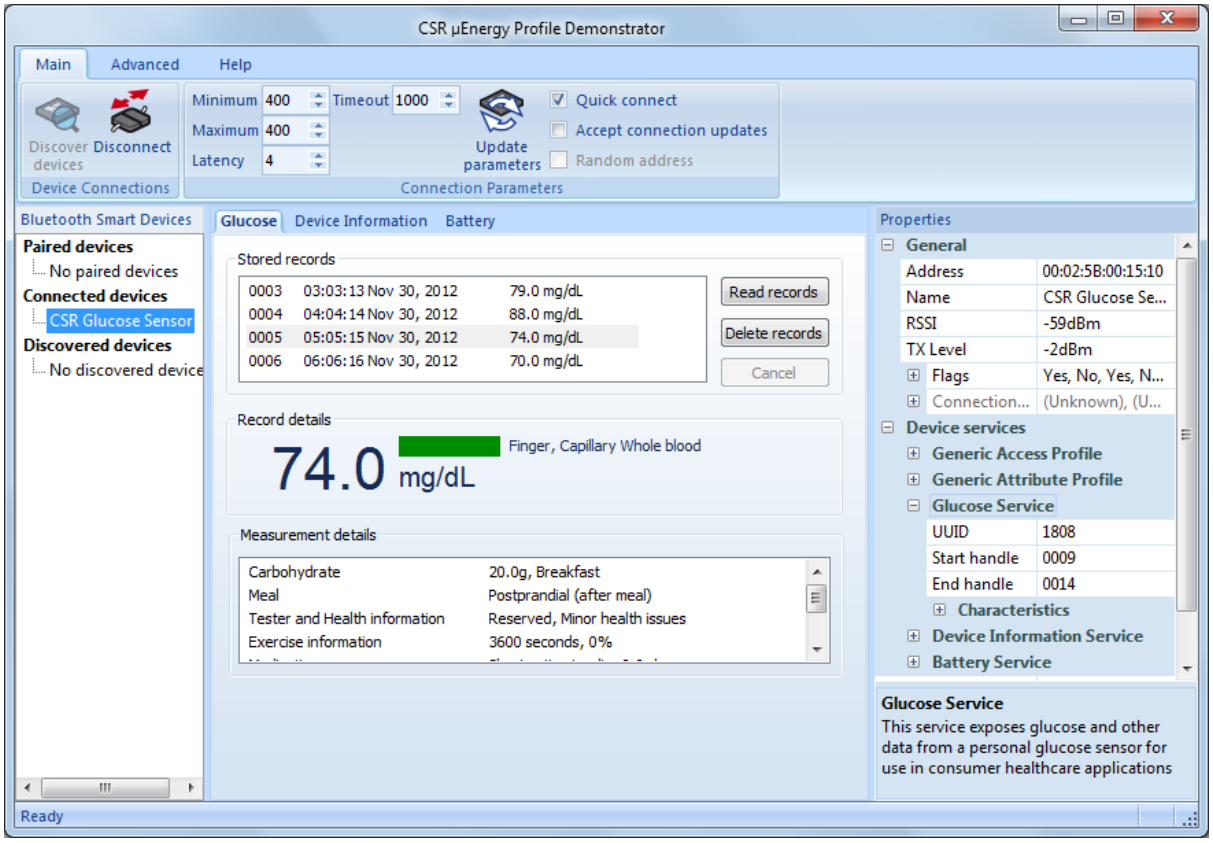

Figure 2.7: Read Glucose Measurements

4. The **Battery** tab displays the current battery state, see Figure 2.8.

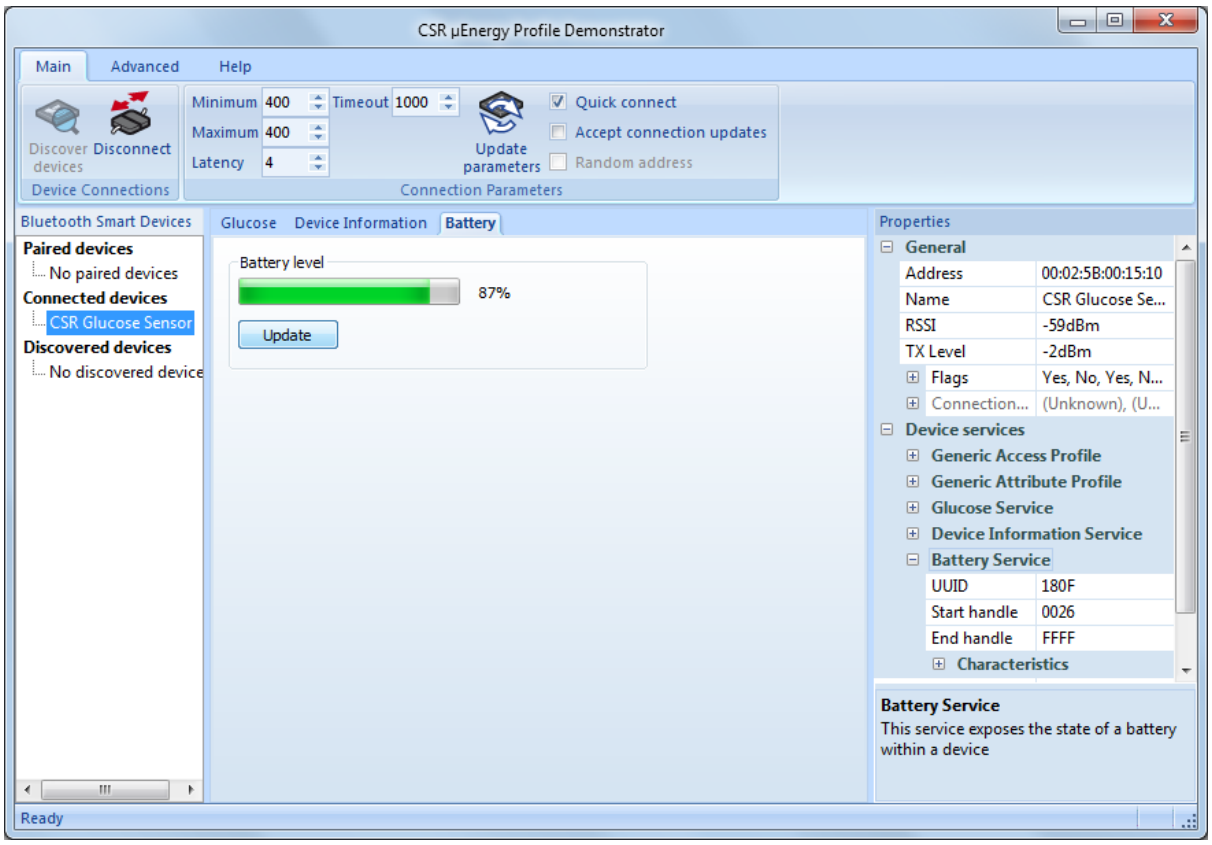

Figure 2.8: Battery Level

5. The **Device Information** tab, see Figure 2.9, displays the device information characteristics of the application.

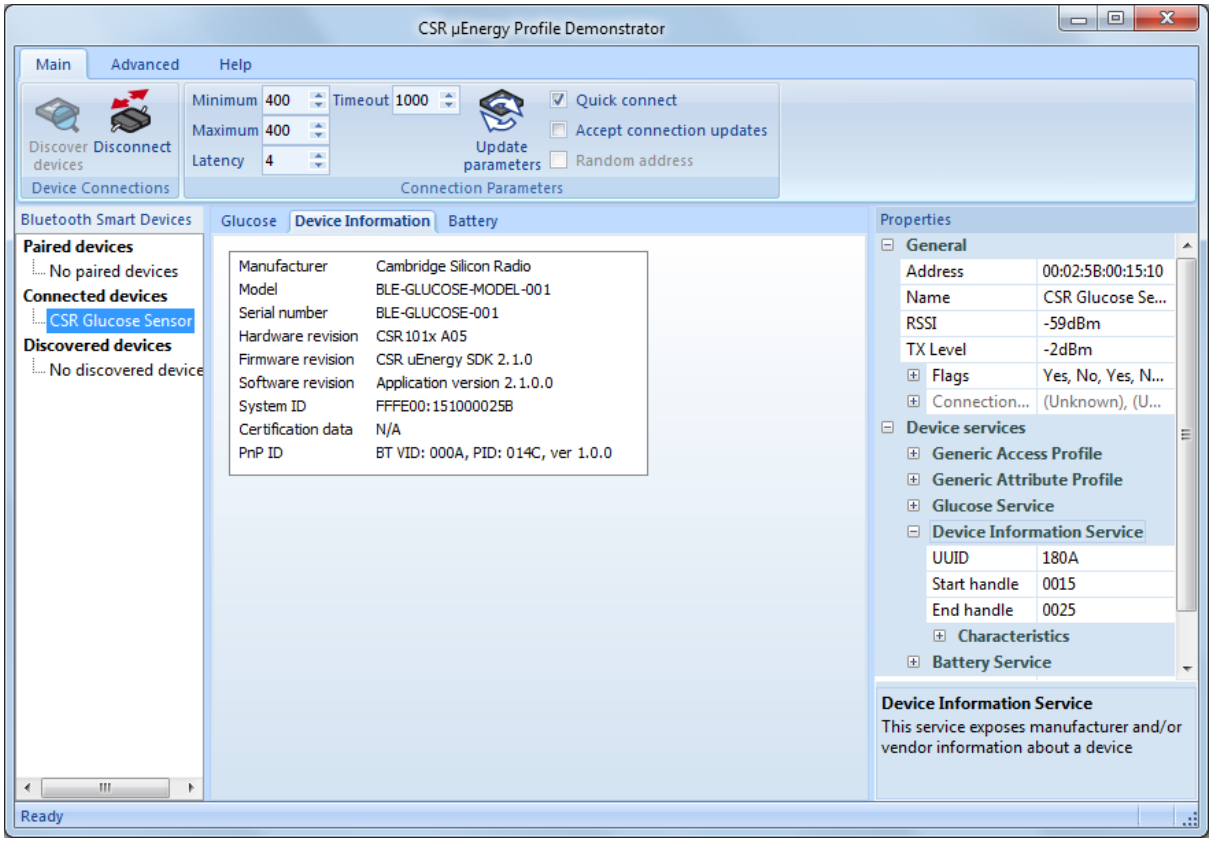

Figure 2.9: Device Information Service

6. Click **Disconnect** to disconnect the Bluetooth Smart link between the device and the USB dongle.

3. Application Structure

3.1. Source Files

Table 3.1 lists the source files and their purpose.

File Name	Purpose
`glucose_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.
`glucose_sensor_gatt.c`	Implements routines for triggering advertisement procedures.
`glucose_service.c`	Implements routines for handling read/write access indications and RACP procedures of Glucose 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.
`glucose_sensor_hw.c`	Implements routines for hardware initialisation, button press handling, generating sample Glucose Measurements, and indicating different states by blinking the LED.
`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 Glucose Sensor application.
`battery_service.h`	Contains prototypes of externally referred functions defined in `battery_service.c` file.
`battery_uuids.h`	Contains macro definitions for UUIDs of the Battery service and related characteristics.

File Name	Purpose
dev_info_service.h	Contains prototypes of the externally referred functions defined in dev_info_service.c file.
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 gap_service.c file.
gap_uuids.h	Contains macros for UUID values of the GAP service and related characteristics.
glucose_sensor.h	Contains prototypes of externally referred functions defined in glucose_sensor.c file.
glucose_sensor_gatt.h	Contains timeout values for fast/slow advertising and prototypes of externally referred functions defined in glucose_sensor_gatt.c file.
glucose_sensor_hw.h	Contains macro definitions for PWM parameters and prototypes of externally referred hardware routines of glucose_sensor_hw.c file.
glucose_service.h	Contains macro definitions for Glucose Measurement value which will be used in formulating sample glucose measurement data.
glucose_service_uuids.h	Contains macro definitions for UUID values of the Glucose service and related characteristics.
nvm_access.h	Contains prototypes of externally referred NVM read/write functions defined in nvm_access.c file.

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 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 B.2 for Device Information service characteristics.
gap_service_db.db	Contains information related to GAP service characteristics, their descriptors and values. See B.3 for GAP characteristics.

File name	Purpose
gatt_service_db.db	Contains information related to GATT service characteristics, their descriptors and values.
glucose_service_db.db	Contains information related to Glucose service characteristics, their descriptors and values. See B.4 for Glucose 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, application data structures and hardware
- Configures GATT entity for server role
- Resets the white list to remove any filtering of discovered devices. See *Bluetooth Core Specification Version 4.0 f*or more information on white list
- 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 Glucose Sensor application waits for user activity to start advertising.
- GATT_ACCESS_IND: This indication event is received when the remote Glucose Collector tries to access an ATT characteristic managed by the application.
- GATT_CHAR_VAL_NOT_CFM: This confirmation event is received in response to a notification sent by the application. If the result is a failure, the application configures radio events for Tx data, disables handling of notification confirmation events and waits for a radio event. The application re-enables the notification confirmation event handling only after the current RACP procedure is complete. If the notification confirmation was successful, the application checks whether the notification sent was a Glucose Measurement notification or a Glucose Context Information notification. If it was a Glucose Measurement notification and if the context information is present

for the same Glucose record, the application sends the Glucose Context Information in a notification. Otherwise, it sends the next Glucose Measurement notification.

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 4.0*.
- `LS_CONNECTION_PARAM_UPDATE_IND`: This indication event is received when the remote central device updates the connection parameters. On receiving this event, the application validates the new connection parameters against the preferred connection parameters and triggers a connection parameter update request if the new connection parameters do not comply with the preferred connection parameters.
- `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. If the last notification sent was a Glucose Measurement notification for a Glucose Record and if the context information is present for the same record, the application sends the Glucose Context Information in a notification, otherwise it sends the next Glucose Measurement notification.

4.1.2.3. SMP Events

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

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 the 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 Glucose 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 initiated by either the master or the slave. These new values are stored by the application for comparison against the preferred connection parameter (see section 6.5).

4.1.3. AppProcessSystemEvent()

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

- `sys_event_battery_low`: This event is received whenever the battery voltage crosses the threshold battery voltage. If connected and notifications are configured, the Glucose 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

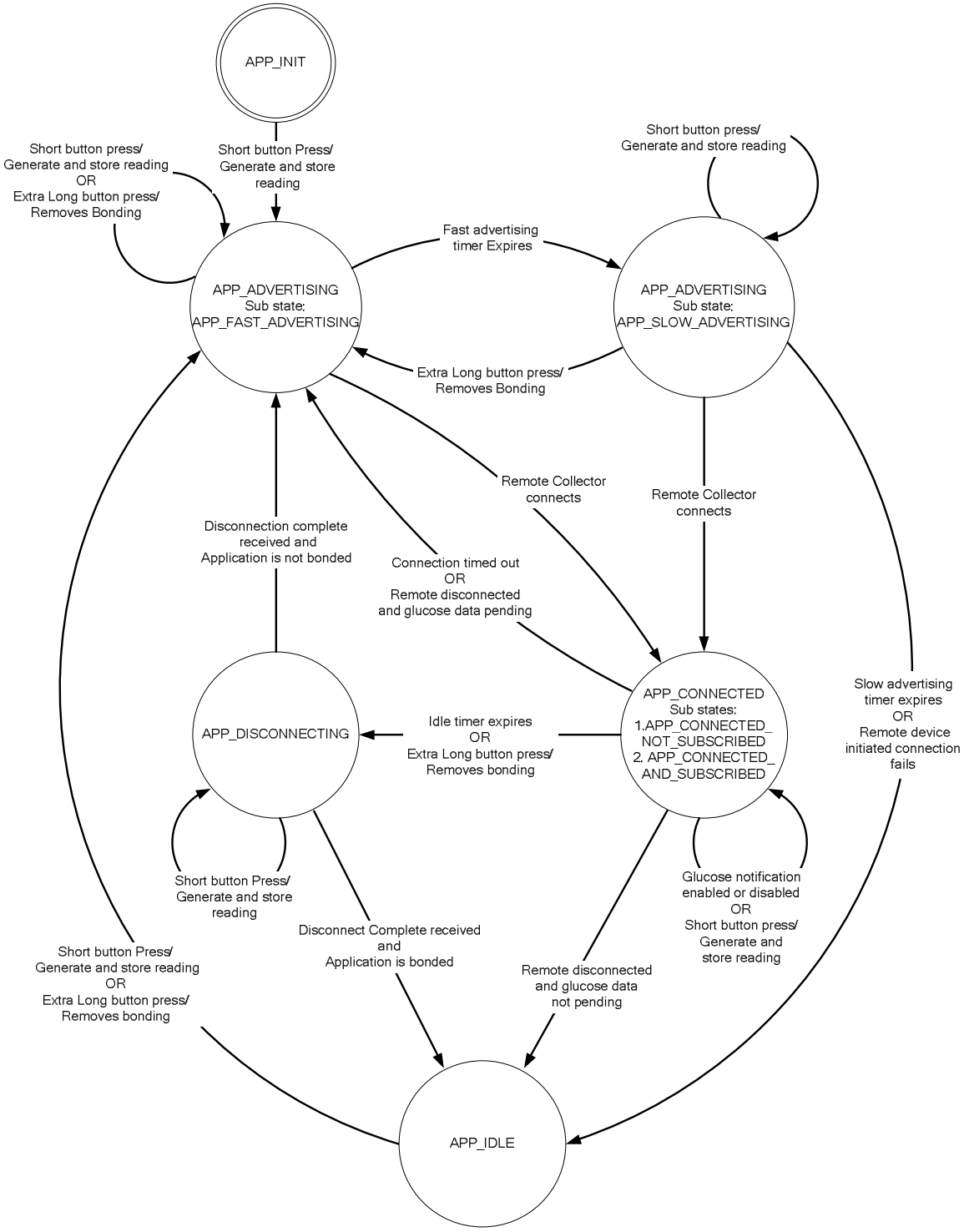

Figure 4.1: Internal State Machine Diagram

The Glucose Sensor application has five internal states described in sections 4.2.1 to 4.2.5.

4.2.1. APP_INIT

When the application is powered on or the chip resets, it enters this state. The Glucose Sensor application does not advertise in this state and responds to button presses only.

- On a **Short button press**, the application triggers advertisements and enters the `APP_ADVERTISING` state. A **Short button press** also generates a sample Glucose Measurement record and the application indicates this by sounding a single short beep.

4.2.2. APP_IDLE

The Glucose Sensor application is not connected to any Glucose Collector.

- On a **Short button press**, the application triggers advertisements and enters the `APP_ADVERTISING` state. A **Short button press** also generates sample Glucose Measurement record and stores it in internal records. The application indicates this Glucose Measurement record generation by sounding a single short beep.
- On an **Extra Long button press**, the application removes bonding information, clears the white list and enters the `APP_ADVERTISING` state.

4.2.3. APP_ADVERTISING

The Glucose Sensor application enters limited discoverable mode 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. 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 `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.
- On a **Short button press**, the application generates a sample Glucose Measurement Record and stores it in internal records. The application indicates this glucose measurement generation by sounding a single short beep.
- On an **Extra Long button press**, the application stops advertisements, removes bonding and restarts advertising without any white list. See *Bluetooth Core specification Version 4.0* for more information on white lists.

Note:

See *Bluetooth Core Specification v4.0* for more information on limited discoverable mode.

4.2.4. APP_CONNECTED

The Glucose Sensor application is connected to the remote Glucose Collector.

- When the application connects to a remote Glucose Collector, it enters the sub state `APP_CONNECTED_NOT_SUBSCRIBED`. When the remote collector subscribes to receiving glucose

measurement notifications, the application moves to sub state APP_CONNECTED_AND_SUBSCRIBED. When the remote collector unsubscribes from receiving glucose measurement notifications, the application moves to sub state APP_CONNECTED_NOT_SUBSCRIBED.

- On a **Short button press**, the application generates a sample Glucose Measurement record (see Figure B.1and Figure B.2 for data format) and stores it in internal records. The application indicates this glucose measurement generation by sounding a single short beep.
- On an **Extra Long button press**, the application removes the bonding information, clears the white list, disconnects the link and enters the APP_ADVERTISING state.
- The application can disconnect the link if kept idle for some time and enters the APP_DISCONNECTING state. See section 6.4 for more information on the idle timer.
- If link loss occurs, the application switches to the APP_ADVERTISING state.
- In the case of a Remote triggered disconnection and if glucose measurement data is present in internal storage, the application again starts advertising and enters the APP_ADVERTISING state, otherwise it enters the APP_IDLE state.

4.2.5. APP_DISCONNECTING

The Glucose Sensor 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 Glucose Collector.

- If the application is bonded to the Glucose Collector, 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. The Glucose Sensor 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)
Glucose Measurement Sequence Number	uint16	1	37
Glucose Measurement Characteristic Client Configuration Descriptor	uint16	1	38
Glucose Measurement Context Characteristic Client Configuration Descriptor	uint16	1	39
RACP Characteristic Client Configuration Descriptor	uint16	1	40

Table 5.3: NVM Memory Map for Glucose Service

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
Battery Level characteristic Client Configuration Descriptor	uint16	1	41

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 Glucose Sensor 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 4.0* 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: Advertising Parameters

6.2. Advertisement Timers

The Glucose 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 file `glucose_sensor_gatt.h`.

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

Table 6.2: Advertisement Timers

6.3. Connection Parameters

Table 6.3 lists the connection parameters used by the Glucose Sensor application. The macros for these values are defined in file `gap_conn_params.h`. These values have been chosen by considering the overall current consumption of the device. See *Bluetooth Core Specification Version 4.0* for the connection parameter range. See section 6.5 for connection update procedure.

Parameter Name	Parameter Value
Minimum Connection Interval	500 ms
Maximum Connection Interval	500 ms
Slave Latency	4 intervals

Parameter Name	Parameter Value
Supervision Timeout	10 s

Table 6.3: Connection Parameters

6.4. Idle Connected Timeout

The Glucose Sensor application disconnects the link if the remote Glucose Collector is not reading glucose measurements and the link is kept idle for some time. The macro for this idle connected timeout is defined in file `gap_conn_params.h`.

Parameter Name	Parameter Value
Idle Connected Timeout	30 min

Table 6.4: Idle Connected Timeout

6.5. Connection Parameter Update

The Glucose Sensor application can request the remote Glucose Collector to update connection parameters according to its power requirements. The application requests a connection parameter update on encryption change or when the remote Glucose collector changes the connection parameters after a period of 30 seconds. The connection parameter update is initiated by the Glucose Sensor application when the new values are outside the range of the preferred connection parameters. The remote Glucose Collector may or may not accept the requested parameters. If the remote Glucose Collector rejects the new requested parameters, the application again requests for an update after 30 seconds. The macro `GAP_CONN_PARAM_TIMEOUT` for this time value is defined in `app_gatt.h`.

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

6.6. Device Name

By default, the device name is set to "CSR Glucose Sensor" in file `gap_service.c`. The maximum length of the device name is 20 octets.

6.7. LED and Buzzer

The Glucose Sensor application uses the buzzer and the LED to indicate different states and events to the user. The user can enable or disable the buzzer and LED as required. Both buzzer and LED should be disabled while taking current consumption readings, see section 7. The macros for enabling the buzzer and LED are defined in file `glucose_sensor_hw.h`.

6.8. Glucose Measurement Records

The Glucose Sensor application stores four Glucose Measurement records. The macro for this number is defined in file `glucose_service.h`.

6.9. PTS Specific Code

Several PTS qualification test cases require applications to behave in a particular way. For the purpose of qualification, this code is enabled by a Configuration (CS) key. The Glucose Sensor application uses the first CS key with index zero of the 8 keys provided for use by the application.

- Bit[0] of the CS key should be set to 1 to pass the following PTS test cases:
  - TC_SPA_BV_01_C
  - TC_SPE_BI_07_C

  These PTS test cases require the Glucose Sensor application to send glucose measurements continuously for some time. Since the Glucose Sensor application stores only four Glucose Measurement records, the transmission of these four records takes less time than the PTS test cases expect. Setting Bit[0] of the configuration key introduces a 1 second delay between every two Glucose Measurements being sent to the remote Glucose Collector device.
- Bit[1] of the CS key should be set to pass the following PTS test cases for the Glucose Service:
  - TC_CN_BV_06_C
  - TC_CN_BV_07_C
  - TC_CN_BV_08_C
  - TC_CN_BV_09_C
  - TC_CN_BV_10_C
  - TC_CN_BV_13_C

  These PTS test cases require the Glucose Measurement Context information to be present in every Glucose Measurement Record. Enabling Bit[1] of the CS key generates the context information for every Glucose Measurement Record. See Appendix B.4 and *Glucose Service Specification Version 1.0* for more information on Glucose Measurement Context.

Note:

For more info on configuration keys, see *CSR µEnergy xIDE User Guide*. For more information on PTS test cases, see the *Glucose Service PTS Test Specification Version 1.0.0* and *Battery Service PTS Test Specification Version 1.0.0* documents.

6.10. Non-Volatile Memory

The Glucose 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 I2C 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 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 and with Channel Map Updates disabled, and 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 Glucose Sensor device 2. Wait for 5 s 3. Take the measurement	490 µA	▪ Advertisement Interval: 60 ms ▪ Advertisement data length: 27 octets ▪ Measurement Time Duration: 20 s
Slow Advertisements	1. Switch on the Glucose Sensor device 2. Wait for 35 s 3. Take the measurement	18 µA	▪ Advertisement Interval: 1.28 s ▪ Advertisement data length: 27 octets ▪ Measurement Time Duration: 20 s
Connected Idle	1. Connect to the Glucose Collector 2. Wait for 60 s 3. Take the measurement	13 µA	▪ Connection Parameters Minimum connection interval: 500 ms Maximum connection interval: 500 ms Slave latency: 4 ▪ Measurement Time Duration: 60 s
Connective Active	1. Connect to the Glucose Collector 2. Wait for 60 s 3. Perform four **Short button press** on Glucose Sensor to generate four glucose measurement records (see section 2.1.2 for **Short button press** behaviour) 4. Wait for 30 s 5. Read all records from Glucose Collector every 5 seconds 6. Take the measurement	18 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`)

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
Glucose Measurement record[(1)]	A Glucose Measurement records consists of one Glucose Measurement characteristic and an optional Glucose Measurement context characteristic.
Note [(1)] See *Glucose Service Specification Version 1.*0 for more information.	

Table A.1: Definitions

Appendix B Service Characteristics

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 `glucose_sensor.c` file. See section 4.1.2.2 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*.

B.1 Battery Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Battery Level	`0x0026`	Read, Notify	Application	Security Mode 1 and Security Level 2	Current battery level
Battery Level-Client Configuration Descriptor	`0x0027`	Read	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 4.0*.

B.2 Device Information Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Serial Number String	0x0017	Read	Firmware	Security Mode 1 and Security Level 2	"BLE-GLUCOSE-001"
Model Number String	0x0019	Read	Firmware	Security Mode 1 and Security Level 2	"BLE-GLUCOSE-MODEL-001"
PnP ID	0x001b	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
Hardware Revision String	0x001d	Read	Firmware	Security Mode 1 and Security Level 2	*<Chip Identifier>*
Firmware Revision String	0x001f	Read	Firmware	Security Mode 1 and Security Level 2	*<SDK version>*
Software Revision String	0x0021	Read	Firmware	Security Mode 1 and Security Level 2	*<Application Version>*
Manufacturer Name String	0x0023	Read	Firmware	Security Mode 1 and Security Level 2	"Cambridge Silicon Radio"
System ID	0x0025	Read	Application	Security Mode 1 and Security Level 2	Organisationally Unique identifier is 0x00025b Manufacturer Identifier depends on the device address

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 4.0*.

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 name: "CSR Glucose Sensor"
Appearance	0x0005	Read	Firmware	Security Mode 1 and Security Level 1	Glucose Meter : "0x0400"
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 4.0.*

B.4 Glucose Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Glucose Measurement	0x000b	Notify	Application	Security Mode 1 and Security Level 2	Glucose measurements. See Figure B.1 for glucose measurement format
Glucose Measurement - Client Characteristic Configuration descriptor	0x000c	Read, Write	Application	Security Mode 1 and Security Level 2	Current client configuration for “Glucose Measurement” characteristic
Glucose Measurement Context	0x000e	Notify	Application	Security Mode 1 and Security Level 2	Glucose measurement context information. See Figure B.2 for format.

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Glucose Measurement Context - Client Characteristic Configuration descriptor	0x000f	Read, Write	Application	Security Mode 1 and Security Level 2	Current client configuration for "Glucose Measurement context" characteristic
Glucose Feature	0x0011	Read	Firmware	Security Mode 1 and Security Level 2	Feature Value: "0x3FF"(1)
Record Access Control Point	0x0013	Indicate, Write	Application	Security Mode 1 and Security Level 2	Response value for last executed RACP procedure. See *Glucose Service Specification Version 1.0* for more information.
Record Access Control Point - Client Characteristic Configuration descriptor	0x0014	Read, Write	Application	Security Mode 1 and Security Level 2	Current client configuration for "Record Access Control point" characteristic

Note

(1) Glucose feature value has been calculated from the following "OR" operation:

```
GLUCOSE_FEATURE_VALUE : [LOW_BATTERY_DETECTION |
                         SENSOR_MALFUNCTION_DETECTION |
                         SENSOR_SAMPLE_SIZE_SUPPORT |
                         STRIP_INSERTION_ERROR_DETECTION |
                         STRIP_TYPE_ERROR_DETECTION |
                         RESULT_HIGH_LOW_DETECTION |
                         TEMPERATURE_HIGH_LOW_DETECTION |
                         SENSOR_READ_INTERRUPT_DETECTION |
                         GENERAL_DEVICE_FAULT_SUPPORT |
                         TIME_FAULT_SUPPORT]
```

See file `glucose_service_uuids.h` for definitions of these macros.

Table B.4: Glucose Service Database

See *Bluetooth Core Specification Version 4.0* for more information on security permissions.

B.4.1 RACP Features

Feature Name	Supported values
RACP Procedure	▪ Read stored records ▪ Delete stored records ▪ Abort operation ▪ Report number of stored records ▪ Number of stored records response ▪ Response code
Procedure Operator	▪ Null ▪ All Records ▪ Less than or equal to ▪ Greater than or equal to ▪ Within range of ▪ First record ▪ Last Record
Filter operand	▪ Sequence Number

Table B.5: RACP Features

B.4.2 Data Format

Figure B.1 and Figure B.2 define the data format for Glucose Measurement and Glucose Measurement Context characteristic respectively.

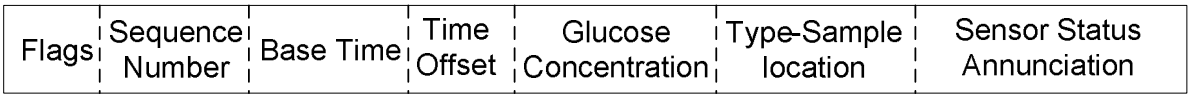

Figure B.1: Glucose Measurement Data Format

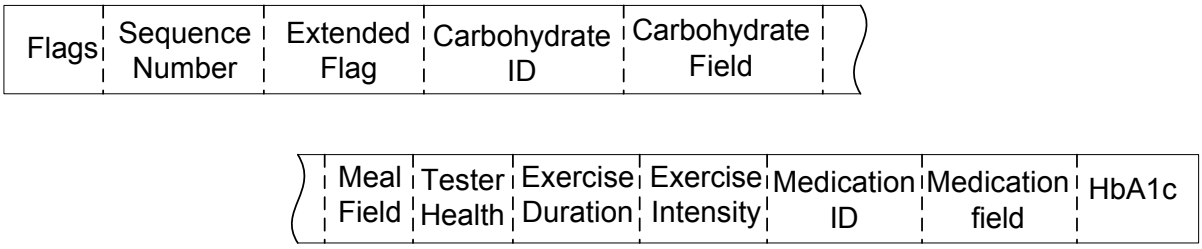

Figure B.2: Glucose Measurement Context Data Format

For more information on Glucose Service characteristics and RACP Procedure features, see *Glucose Service Specification Version 1.0 and Bluetooth SIG Developer Portal.*

Appendix C Advertising and Scan Response Data

The Glucose Sensor application adds the following fields to the Advertising data:

Advertising Data Field	Contents
Flags	The Glucose Sensor application sets the General Discoverable Mode bit. See Section 11, Part C of Volume 3 in *Bluetooth core Specification Version 4.0* for more information.
Service UUIDs	The Glucose Sensor application adds 16-bit UUID of following service: ▪ Glucose
Device Appearance	Glucose Meter : "0x0400"
Device Name (1)	Device name (Default value : "CSR Glucose 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 the Scan Response data.	

Table C.1: Advertising Data Field

The Glucose Sensor application adds the following field to the Scan Response data:

Scan Response Data Field	Contents
Tx Power	Current Tx power level

Table C.2: Scan Response Data Field

Appendix D Known Issues or Limitations

See the Glucose Sensor application and SDK Release Notes.

Document References

Document	Reference
Bluetooth Core Specification Version 4.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Glucose Profile Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Glucose 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
Glucose Service PTS Test Specification Version 1.0.0	https://www.bluetooth.org/Technical/Qualification/requirements.htm
Battery Service PTS Test Specification Version 1.0.0	https://www.bluetooth.org/Technical/Qualification/requirements.htm
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

AFH	Adaptive Frequency Hopping
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
I^2C	Inter-Integrated Circuit
IRK	Identity Resolving Key
LED	Light Emitting Diode
LM	Link Manager
mg/dL	micrograms per decilitre
NVM	Non Volatile Memory
PC	Personal Computer
PIO	Programmable Input Output
PnP	Plug and Play
PS	Persistent Store
PTS	Profile Testing Suite
PWM	Pulse Width Modulation

RACP	Record Access Control Point
Tx	Transmit
UUID	Universally Unique Identifier