Push every boundary.™

CSR µEnergy®

Hello Server Example

Application Note

Issue 2

Document History

Revision	Date	History
1	02 NOV 13	Internal Release
2	18 NOV 13	Editorial Updates

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

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 Hello Server Example application supplied as a part of the Bluetooth Quick Start Kit v2 and provides guidance on how to customise the on-chip application. This application demonstrates the "Hello Bluetooth" custom profile specified by the Bluetooth SIG.

1.1. Application Overview

1.1.1. Profiles Supported

The Hello Server Example application supports the Hello Bluetooth profile.

The Hello Bluetooth profile facilitates a Bluetooth Smart enabled registration desk to exchange information, e.g. name, with a Bluetooth Smart enabled business card. See Figure 1.1.

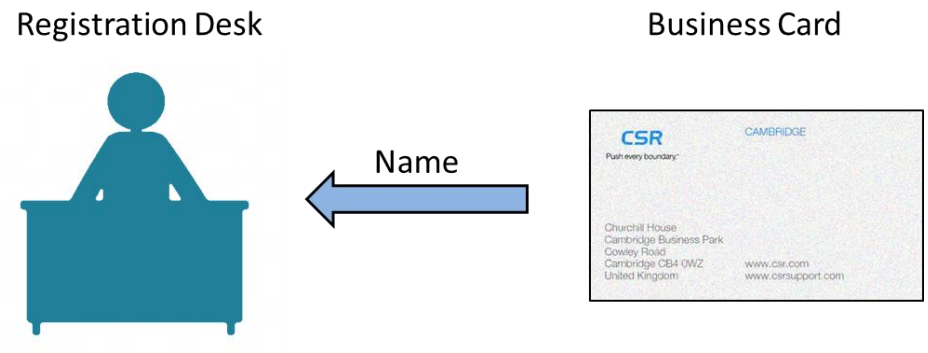

Figure 1.1: Hello Bluetooth Server

The Hello Bluetooth profile defines two roles, see Table 1.1.

Role	Description
Business card	It is the device that acts as a server and stores the name information of the person.
Registration Desk	It is the device that acts as a client. It receives the name information and display a welcome message.

Table 1.1: Hello Bluetooth Profile Roles

1.1.2. Application Topology

The Hello Server Example application implements the Hello Bluetooth profile in Business card role see Table 1.2.

Role	Hello Bluetooth Profile	GAP Service	GATT Service
GATT Role	GATT Server	GATT Server	GATT Server
GAP Role	Peripheral	Peripheral	Peripheral

Table 1.2: Application Topology

Role	Description
GATT Server	It accepts requests from the client and sends responses to it.
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

The Hello Server Example application exposes the following services:

- Hello Server custom service
- GAP
- GATT

The Hello Bluetooth profile mandates only one service, Hello Server. GAP and GATT services are mandated by the *Bluetooth Core Specification Version 4.0*.

For more information on the Hello Server service, see the *Custom Hello Bluetooth Profile.* For more information on GATT and GAP services, see the *Bluetooth Core Specification Version 4.0*.

Note:

The Hello Server Example application does not support any characteristic for GATT service. See the *Bluetooth Core Specification Version 4.0* for more information.

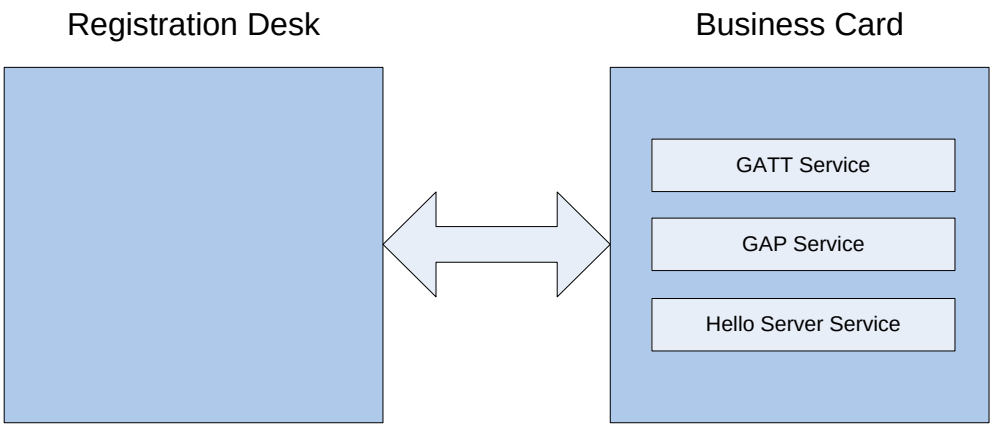

Figure 1.2: Primary Services

2. Software

The CSR µEnergy SDK builds and downloads the on-chip Hello Server Example application on to the CSR1011/CSR1010/CSR1001/CSR1000 development boards. For more information, see the *CSR µEnergy xIDE User Guide.*

2.1. Application Structure

2.1.1. Source Files

Table 2.1 lists the source files and their purpose.

File name	Purpose
app_main.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.
app_gatt.c	Implements routines for triggering advertisement procedures.
gap_service.c	Implements routines for GAP service e.g. handling read access indications on the GAP service characteristics.
app_hw.c	Implements routines for hardware initialization and button press handling.
buzzer.c	Implements routines for sounding buzzer beeps. This functionality can be enabled or disabled during build.
hello_service.c	Implements routines for handling read access on Hello Server service characteristics.

Table 2.1: Source Files

2.1.2. Header Files

Table 2.2 lists the header files and their purpose.

File Name	Purpose
app_gatt.h	Contains macro definitions and function prototypes which are being used across the application.
app_hw.h	Contains prototypes of externally referred routines in the app_hw.c file.
app_main.h	Contains prototypes of externally referred routines in the app_main.c file.
appearance.h	Contains the appearance value macro of the Hello Server Example application.
buzzer.h	Contains prototypes of externally referred functions defined in the buzzer.c file.
gap_conn_params.h	Contains macro definitions for advertising and preferred connection parameters.
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

File Name	Purpose
	characteristics.
gatt_service_uuids.h	Contains macros for UUID values for GATT service.
hello_service.h	Contains prototypes of externally referred functions defined in the hello_service.c file.
hello_service_uuids.h	Contains macro definitions for UUID values of the Hello Server service and related characteristics.
user_config.h	Contains macro for enabling various features e.g. buzzer functionality etc.

Table 2.2: Header Files

2.1.3. 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 2.3 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.
gap_service_db.db	Contains information related to GAP service characteristics, their descriptors and values.
gatt_service_db.db	Contains information related to GATT service characteristics, their descriptors and values.
hello_service_db.db	Contains information related to Hello Server service characteristics, their descriptors and values.

Table 2.3: Database Files

2.2. Building the Application

CSR µEnergy xIDE supplied with CSR µEnergy SDK is used to build and download the Hello Server Example application to the development board.

2.2.1. Prerequisites

xIDE may be installed on a PC running:

- Windows XP
- Windows 7 (32-bit or 64-bit)

  or
- Windows 8 (32-bit or 64-bit).

xIDE requires a USB port or an LPT port to communicate with development boards. The LPT port is only supported when xIDE is running on 32-bit versions of Windows.

Note:

Spaces in folder names of the directory path are not supported i.e. you should not try to install the software in a directory which itself has spaces in its name or is contained within a folder that has spaces in its name e.g. xIDE cannot be successfully installed in the **Program Files** directory.

2.2.2. Installation Procedure

CSR recommends that any applications running on the PC are closed before installing the software.

1. The software is provided on a CD-ROM but can also be downloaded from www.csrsupport.com/uEnergy.
2. Double-click on the `CSR_uEnergy_SDK-<Version>.exe` file to launch the Setup wizard, which guides you through the rest of the installation process.
3. Follow the on-screen instructions, clicking **Next** to continue.
4. Click **Finish** to complete the installation.

Notes:

(1) For a first time installation, CSR recommends that the default settings are used.

(2) If the option to install the SPI LPT device driver was selected, the PC must be restarted to complete the installation.

2.2.3. Launching xIDE

Launch xIDE for CSR µEnergy SDK by double-clicking on the icon on your desktop, the icon in the Quick Launch bar or from the Windows Start menu:

- On Windows 8 open the Start page and click on **CSR µEnergy SDK <version> (xIDE)**.
- On Windows XP and Windows 7 open the Start menu and navigate to **CSR µEnergy SDK <version>/CSR µEnergy SDK <version> (xIDE)**.

The xIDE application window opens, see Figure 2.1.

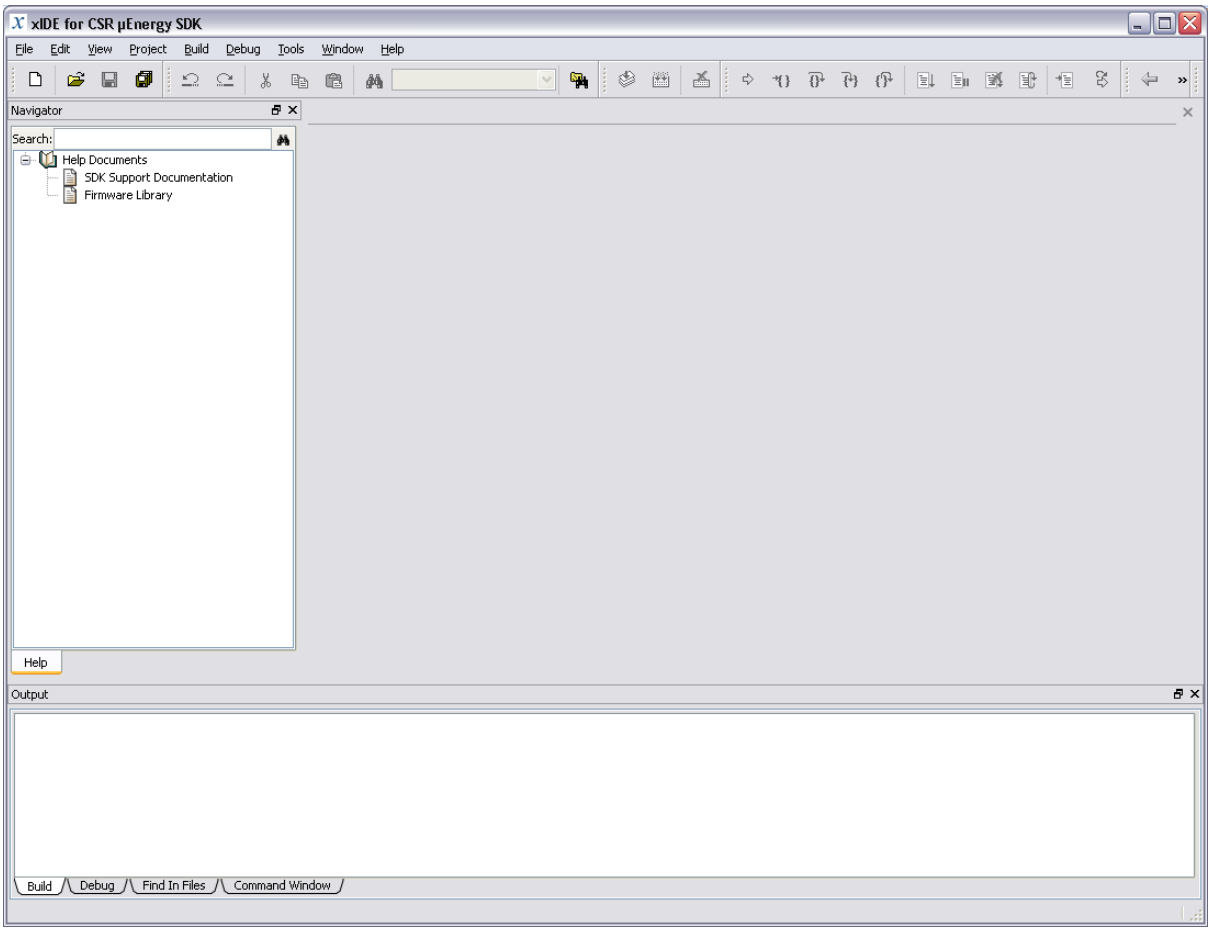

Figure 2.1: xIDE Opening Screen

2.2.4. Working with xIDE

To open a project workspace for the Hello Server Example application:

1. Select **Open Workspace** in the xIDE **Project** menu. An **Open workspace** window appears.
2. Browse to the folder containing the Hello Server Example application.
3. Open the required folder. The `hello_server.xiw` project file is displayed, see Figure 2.2.

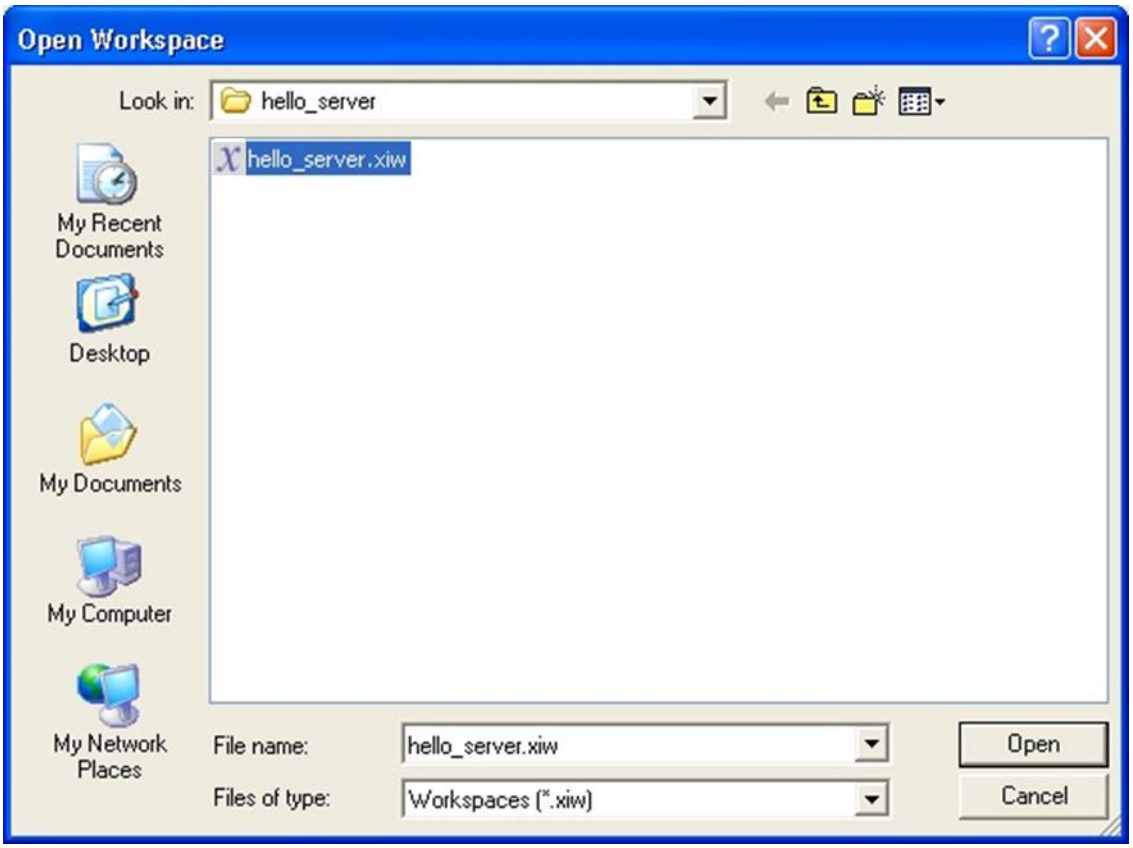

Figure 2.2: Project Workspace Window

To compile and execute the code:

1. Clean the `hello_server` project by selecting **Clean Active Project** in the xIDE **Build** menu. This deletes all output files created by previous build operations.
2. Build all the files included in the `hello_server` project by selecting **Build Active Project** in the xIDE **Build** menu or press the **F7** key. The build is incremental, so the minimum set of builds are performed in order to reflect changes to source files and configurations.
3. Select **Run** from the **Debug** menu or press the **F5** key. This runs the application on the development board. Ensure the development board is switched on using the Power On/Off switch. Figure 2.3 shows the switch in the Off position.

See *CSR µEnergy xIDE User Guide* for further information.

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.

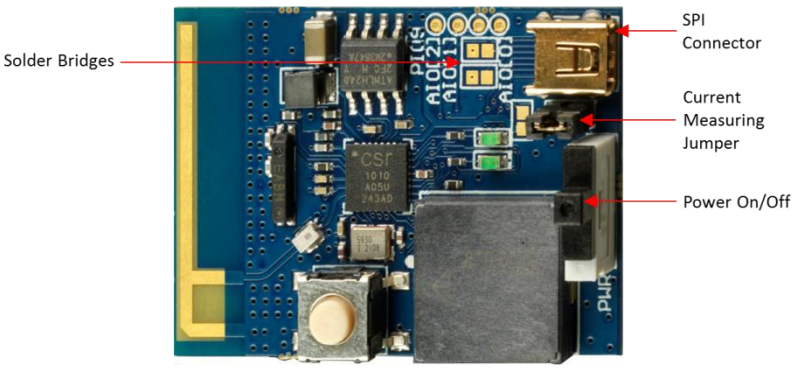

Figure 2.3: CSR1010/CSR1000 Development Board

Note:

Although the Hello Server Example application primarily targets the CSR1010/CSR1000 development boards, the CSR1011/CSR1001 development boards may also be used as an alternative hardware platform.

2.3. Demonstration Procedure

To demonstrate the Hello Server Example application:

1. Switch on the development board to trigger advertisements.
2. Open the Key Fob application on an iOS device, see Figure 2.4 for the initial screen.

Note:

The Key Fob application source code is available as a part of the Bluetooth Quick Start Kit v2, which must be installed on an iOS device using the Xcode development environment. Installation procedure of an application on an iOS device is beyond the scope of this document. See developer.apple.com for more details.

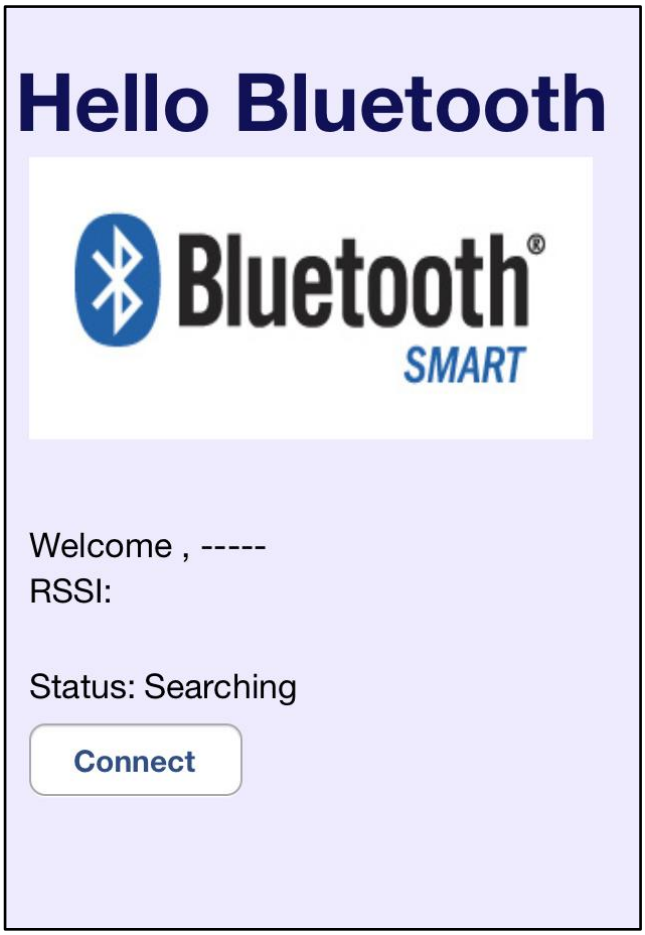

Figure 2.4: Key Fob App Initial Screen

3. Press the **Connect** button on the iOS device application to connect the Hello Server Example application. Figure 2.5 shows a screenshot of the Key Fob application in the CONNECTED state.

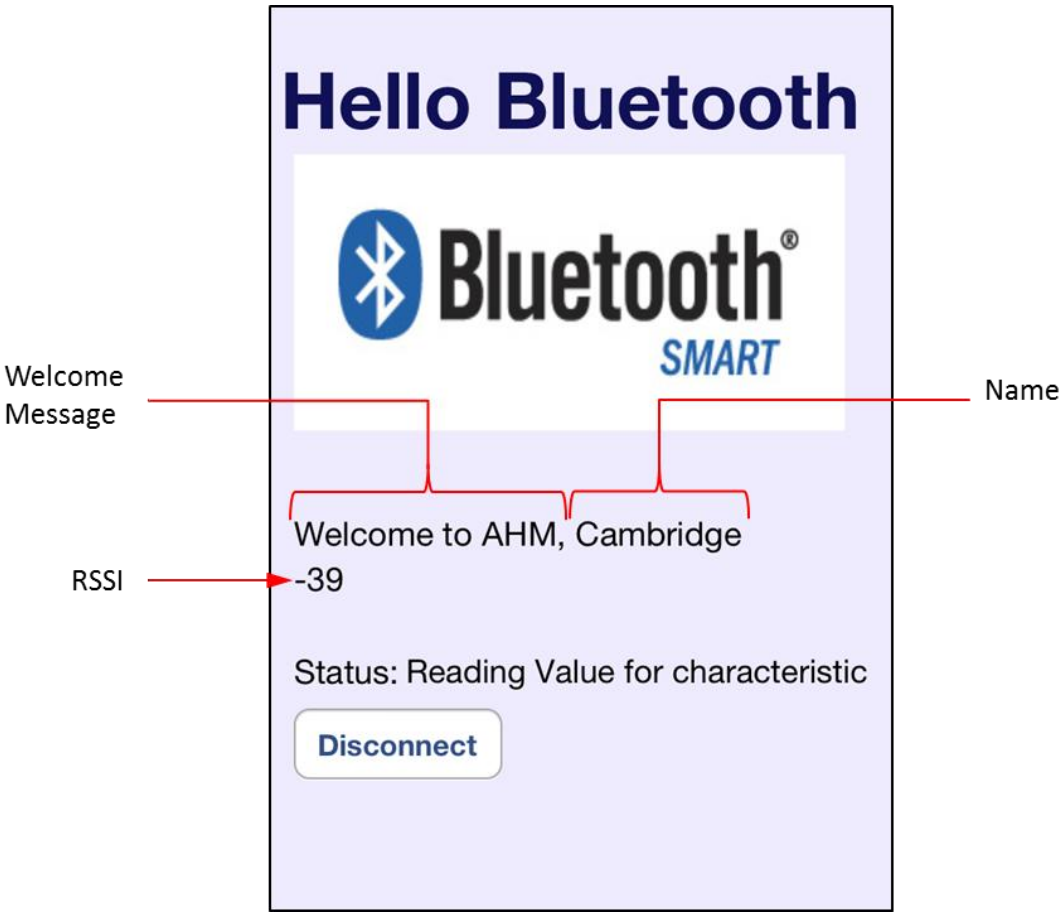

Figure 2.5: Key Fob App Screenshot in the CONNECTED State

Note:

The Key Fob application constantly monitors the RSSI value of the connected device. If the RSSI value is less than the threshold, the application performs a Read operation on the name characteristics and displays **Welcome to AHM, <name>**, see Figure 2.5. If the RSSI value is greater than equal to the threshold value, the application displays **Welcome to AHM, ….**, see Figure 2.6.

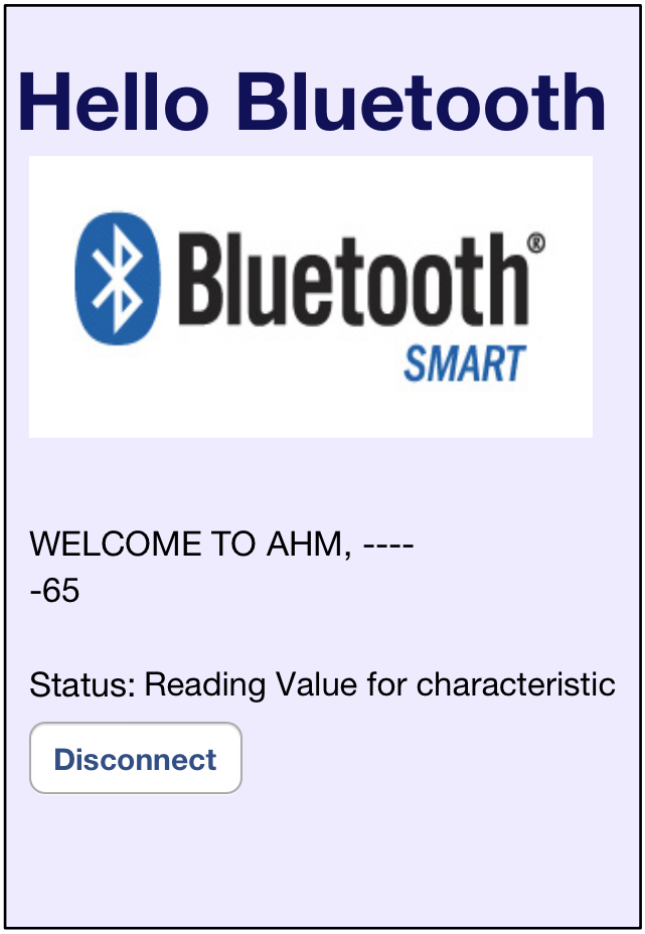

Figure 2.6: Key Fob App Screenshot in the CONNECTED State, for RSSI Values Greater than the Threshold

3. User Interface

The Hello Server Example application makes use of the button and buzzer available on the CSR1010/CSR1000 development board.

A Button Press:

- Disconnects the link if in the CONNECTED state
- Triggers advertisements if in IDLE state
- Is ignored in all other states

Figure 3.1 shows button press handling in various application states.

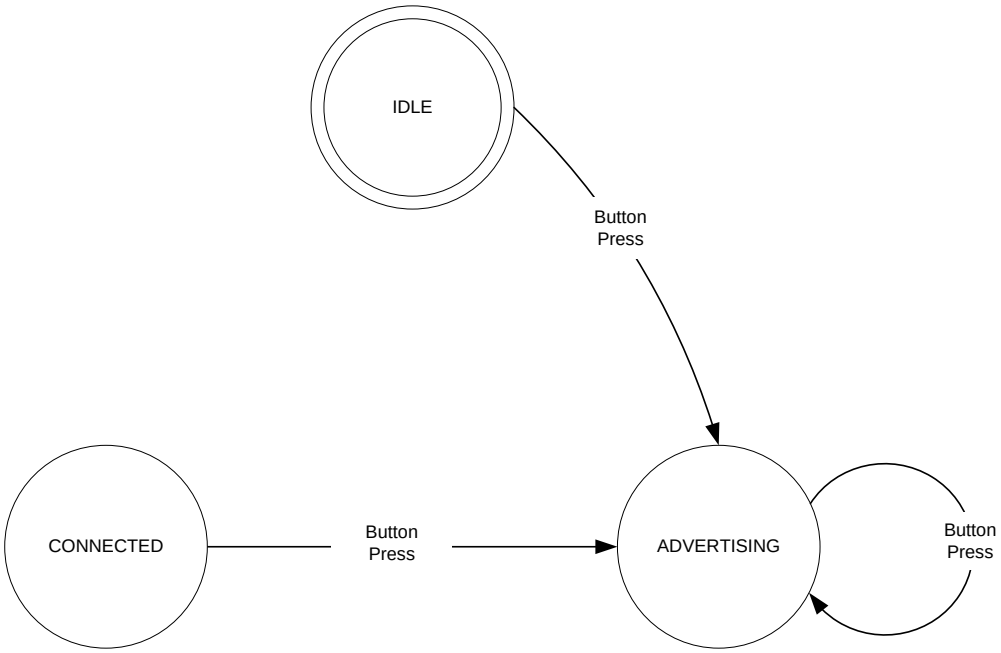

Figure 3.1: Application State Handling on Button Press

Note:

This state machine only shows the button press behaviour. For detailed application states, see Figure 4.1.

4. Code Overview

This section describes the significant functions of the Hello Server example application.

4.1. Application Entry Points

4.1.1. AppInit ()

`AppInit ()` is invoked when the application is powered on or the chip resets and performs the following initialisation functions:

- Initialises the application timers, hardware and application data structures.
- Configures GATT entity for server role.
- Initialises all the services.
- Registers the ATT database with the firmware.

4.1.2. AppProcessLmEvent ()

`AppProcessLmEvent ()` 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 Hello Server Example application starts advertising.
- `GATT_ACCESS_IND`: This indication event is received when the Registration desk tries to access an ATT characteristic managed by the application.

4.1.2.2. LS Events

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

4.1.2.3. 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 fails to establish a connection it 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 Hello Server Example 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.

4.1.3. AppProcessSystemEvent ()

`AppProcessSystemEvent ()` handles system events such as a low battery notification or a PIO change. It currently handles the following system event:

- `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

The Hello Server Example application has five internal states, see Figure 4.1 for details.

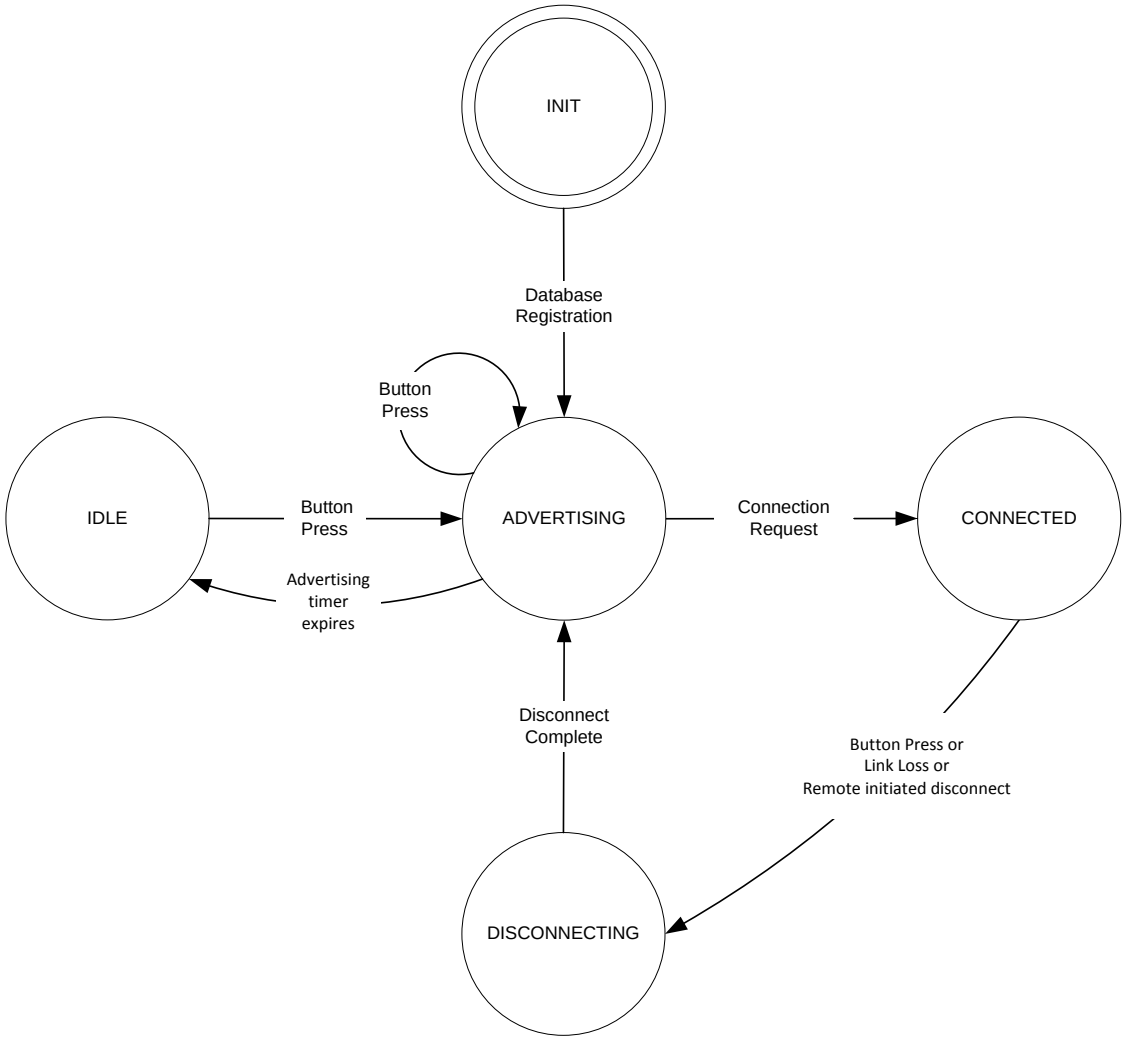

Figure 4.1: Internal State Machine

4.2.1. Init

When the application is powered on or the chip resets it enters `APP_STATE_INIT` state. The Hello Server Example application registers the service database with the firmware and waits for a confirmation. On a successful database registration it starts advertising.

4.2.2. Idle

The application enters APP_STATE_IDLE state when the advertisement timer has expired. In this state, the application is not connected to any remote device. On a button press application triggers advertisements and enters the APP_STATE_ADVERTISING state.

4.2.3. Advertising

In the APP_STATE_ADVERTISING, the application advertises itself and beeps twice to indicate the start of advertisements. If a remote device connects to it, it stops advertisements and enters the APP_STATE_CONNECTED state and if the advertising timer expires before a connection is made, the APP_STATE_IDLE state is entered.

4.2.4. Connected

In the APP_STATE_CONNECTED state, the application is connected to a remote device and sends name information on request to the remote device.

The application disconnects the link and enters APP_STATE_DISCONNECTING, when:

- The remote device triggers a disconnection
- There is a link loss
- A button is pressed on the CSR1010/CSR1000 board
- A connection idle timer expires

4.2.5. Disconnecting

In the APP_STATE_DISCONNECTING state, the application waits for a disconnect confirmation after initiating the disconnection. After receiving the disconnect confirmation, it triggers advertisements.

5. Customising the Application

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

5.1. Advertising Parameters

The application uses the parameters in Table 5.1 for advertisements. The macros for these values are defined in the `gap_conn_params.h` file. These values have been chosen by considering the overall current consumption of the device. See the *Bluetooth Core Specification Version 4.0* for the advertising parameters range.

Parameter Name	Advertisements Time
Minimum Advertising Interval	1280 ms
Maximum Advertising Interval	1280 ms

Table 5.1: Advertising Parameters

5.2. Advertisement Timers

The application enters the IDLE state on expiry of the advertisement timers. See section 4.2 for more information. The macro for the timer value is defined in the `user_config.h` file. To disable the expiry of advertisements the macro `CONNECTION_ADVERT_TIMEOUT_VALUE`, should be defined, i.e:

```
#define CONNECTION_ADVERT_TIMEOUT_VALUE            (0)
```

The default advertisement timer value is listed in Table 5.2.

Timer Name	Timer Value
Advertisement Timer Value	5 min

Table 5.2: Advertisement Timers

5.3. Connection Parameters

The application uses the connection parameters listed in Table 5.3. The macros for these values are defined in the file `gap_conn_params.h`.

Parameter Name	Parameter Value
Minimum Connection Interval	80 ms
Maximum Connection Interval	100 ms
Slave Latency	4 intervals
Supervision Timeout	2 s

Table 5.3: Connection Parameters

5.4. Connection Parameter Update

The Hello Server Example application can request the connected peer device, i.e. a Bluetooth enabled registration desk, to update the connection parameters according to its power requirements.

This application requests a connection parameter update when the connected peer device changes the connection parameters after a duration of 30 seconds.

The connection parameter update is initiated by the Hello Server Example application when the new values are outside the range of the preferred connection parameters. The connected peer device may or may not accept the requested parameters.

If the peer device rejects the newly requested parameters, the application again requests an update after 30 seconds. The macro `GAP_CONN_PARAM_TIMEOUT` for this time value is defined in the file `app_main.c`.

5.5. Device Name

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

5.6. Buzzer

The application uses a buzzer to indicate different states and events to the user. The user can enable or disable the buzzer as required in the `user_config.h` file.

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

Document References

Document	Reference
Bluetooth Core Specification Version 4.0	https://www.bluetooth.org
CSR µEnergy xIDE User Guide	CS-212742-UG
GATT Database Generator	CS-219225-UG
Custom Hello Bluetooth Profile	Available as a part of Bluetooth Quick Start Kit v2 https://developer.bluetooth.org
Installing the CSR Driver for the Profile Demonstrator Application	CS-235358-UG

Terms and Definitions

Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
Bluetooth Smart	Formerly known as Bluetooth low energy
Bluetooth SIG	Bluetooth Special Interest Group
CD	Compact Disc
CSR	Cambridge Silicon Radio
e.g.	*exempli gratia*, for example
etc	*et cetera*, and the rest, and so forth
GAP	Generic Access Profile
GATT	Generic Attribute Profile
i.e.	*id est*, that is
L2CAP	Logical Link Control and Adaptation Protocol
LM	Link Manager
LPT	Line Printer Terminal
PC	Personal Computer
PIO	Programmable Input Output
ROM	Read Only Memory
RSSI	Received Signal Strength Indication
SDK	Software Development Kit
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
UUID	Universally Unique Identifier
Xcode	Apple's Integrated Development Environment
xIDE	CSR's Integrated Development Environment