CSR μEnergy®

Alert Tag

Application Note

Issue 4

Document History

Revision	Date	History
1	11 MAY 12	Original publication of this document
2	11 JUN 12	Minor editorial changes and updates to current consumption section
3	08 FEB 13	Updated to reference CSR101x devices and for SDK 2.1
4	19 FEB 13	Updated current consumption values

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

Contents

Tables, Figures and Equations

1. Introduction

This document describes the Alert Tag 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 Alert Notification and Phone Alert Status profiles which are specified by the Bluetooth SIG.

1.1. Application Overview

1.1.1. Profile Supported

The Alert Tag application supports the two profiles described in sections 1.1.1.1 and 1.1.1.2:

1.1.1.1. Alert Notification Profile

The Alert Notification profile enables a client device to receive information from a cell phone about the incoming calls, missed calls and SMS/MMS messages. The information may include caller ID for an incoming call or sender's ID for email/SMS/MMS but not the actual message contents.

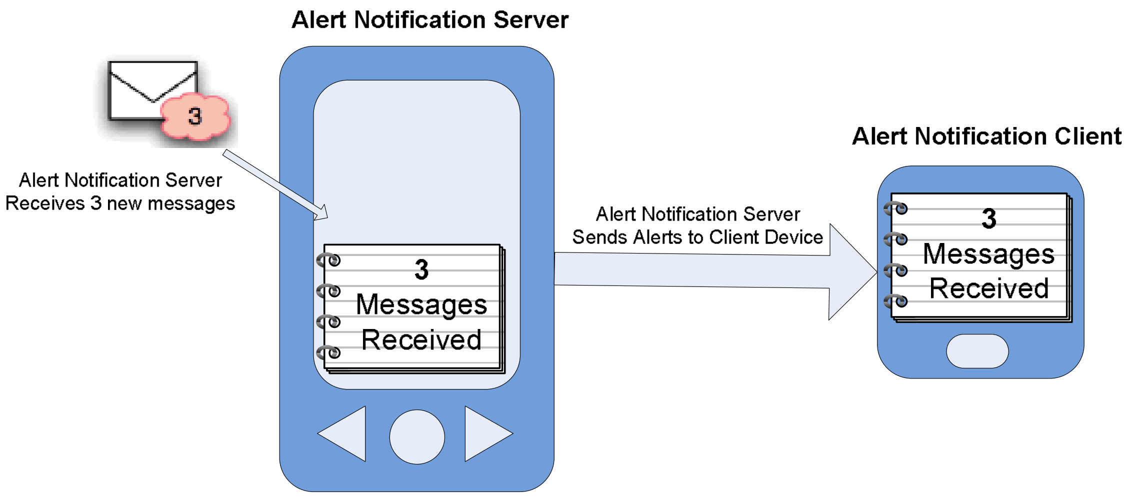

Alert Notification Server
3
Alert Notification Server Receives 3 new messages
3 Messages Received
Alert Notification Server Sends Alerts to Client Device
Alert Notification Client
3 Messages Received

Figure 1.1: Alert Notification Profile

The profile defines two roles, see Table 1.1:

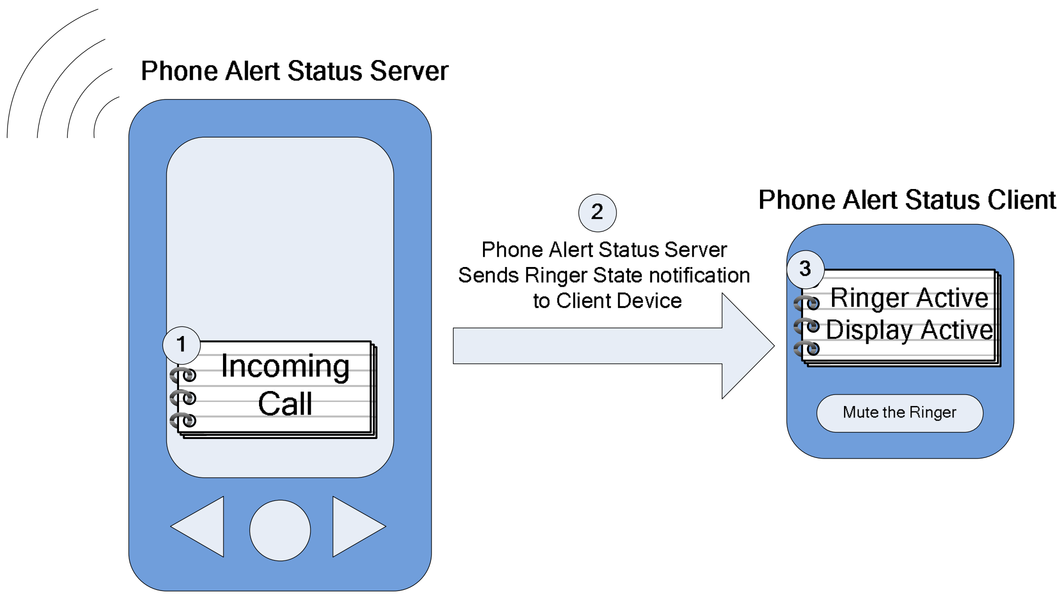
Role	Description
Alert Notification Server	Alert Notification Server is the device which originates the alert notifications about new and unread messages, incoming calls etc. The cell phone acts as the Alert Notification Server.
Alert Notification Client	Alert Notification Client is the device which receives alert notifications from the server device about the incoming calls, missed calls and SMS/MMS messages etc. The Alert Tag application acts as the Alert Notification Client.

Table 1.1: Alert Notification Profile Roles

See *Alert Notification Profile Specification Version 1.0* for more information on the Alert Notification Profile.

1.1.1.2. Phone Alert Status Profile

The Phone Alert Status profile enables a client device to receive information related to the Alert Status and Ringer Setting of a phone. This profile also enables the client device to configure the ringer status from the peer device.



Phone Alert Status Server
1
Incoming Call
2
Phone Alert Status Server Sends Ringer State notification to Client Device
Phone Alert Status Client
3
Ringer Active
Display Active
Mute the Ringer

Figure 1.2: Phone Alert Status Profile

The profile defines two roles, described in Table 1.1:

Role	Description
Phone Alert Server	The Phone Alert Server is the device which originates the alerts about alert status, ringer setting and ringer status. The cell phone acts as the Phone Alert Server.
Phone Alert Client	The Phone Alert Client is the device which receives alerts from the server. The Alert Tag application acts as the Phone Alert Client.

Table 1.2: Phone Alert Status Profile Roles

See *Phone Alert Status Profile Specification Version 1.0* for more information on the Phone Alert Status Profile.

1.1.2. Application Topology

The Alert Tag application implements both the Alert Notification and the Phone Alert Status profiles in client role and uses the following topology:

Role	Alert Notification Profile	Phone Alert Status Profile	GAP service	GATT service	Device Information service
GATT Role	GATT Client	GATT Client	GATT Server	GATT Server	GATT Server
GAP Role	Peripheral	Peripheral	Peripheral	Peripheral	Peripheral

Table 1.3: Application Topology

Role	Responsibility
GATT Client	It initiates commands and requests towards the server and receives responses, notifications and indications sent by the server.
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 1.4: Responsibilities

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

Note:

The Alert Tag application is referred to as the **Alert Tag Client** in this application note as it implements the Client role of the Alert Notification and the Phone Alert Status profiles.

The remote connected device is referred to as the **Alert Tag Server** in this application note as it implements the Server role of the Alert Notification and the Phone Alert Status profiles.

1.1.3. Services

The Alert Tag application exposes the following services:

- Device Information (Version 1.1)
- GAP
- GATT

GAP and GATT services are mandated by *Bluetooth Core Specification Version 4.0*. Device Information Service is optional, see Figure 1.3.

See *Device Information Service Specification Version 1.1* for more information on the Device Information Service.

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

Note:

The Alert Tag application does not support any characteristic for the GATT service.

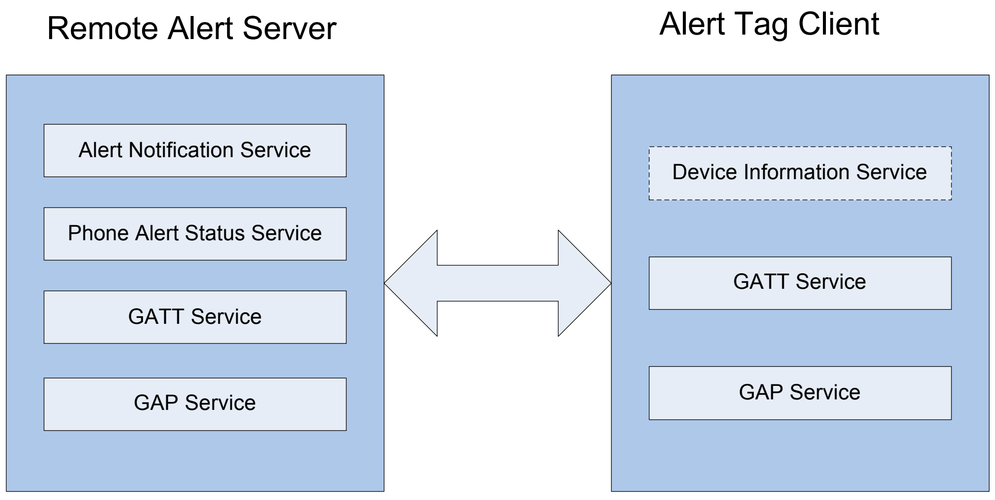


Figure 1.3: Primary Services

2. Using the Application

This section describes how the Alert Tag application can be used with the CSR µEnergy Profile Demonstrator application available from CSR.

2.1. Demonstration Kit

Table 2.1 lists the components of the demonstration Alert Tag application:

Component	Hardware	Application
Alert Tag Client	CSR1010 Development Board	Alert Tag Application
Alert Tag Server	CSR µEnergy BT4.0 USB dongle	CSR µEnergy Profile Demonstrator Application

Table 2.1: Demonstration Components

Notes:

1. Although the Alert Tag application primarily targets the CSR1010 development board, the CSR1011 development board may also be used as an alternative hardware platform.
2. From SDK v2.1.0 the Alert Tag application is not supported on CSR100x devices.

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

2.1.1. Alert Tag Client

The SDK is used to build and download the Alert Tag 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.

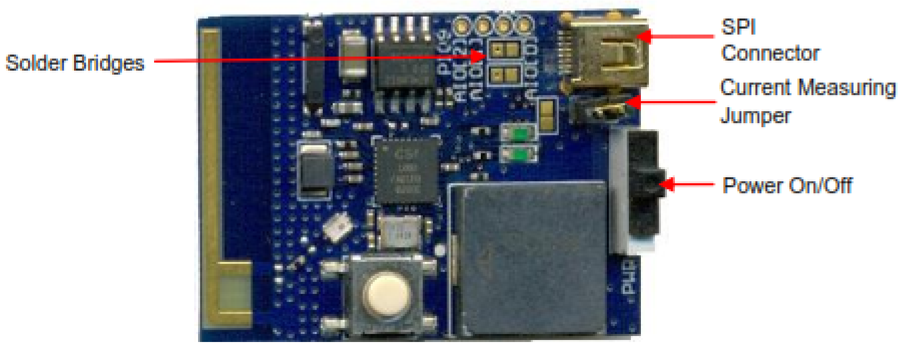


Figure 2.1: CSR1010 Development Board

2.1.1.1. User Interface

The application makes use of the button, LED and buzzer available on the development board.

Button Press Behaviour

- In Idle state, a **Short button press** starts advertising.
- In Advertising state, a **Short button press** and a **Long button press** is ignored.
- In Connected state, a **Short button press** silences the ringer of the remote connected Alert Server if ringing.
- In Connected state, a **Long button press** toggles the ringer setting of the remote connected Alert Server between Normal and Mute.
- In Connected state, an **Extra Long button press** removes the bonding information, disconnects the link and triggers advertisements.
- In all other states, an **Extra Long button press** removes the bonding information and starts advertisements.

Figure 2.2 summarises the Alert Tag application device behaviour.

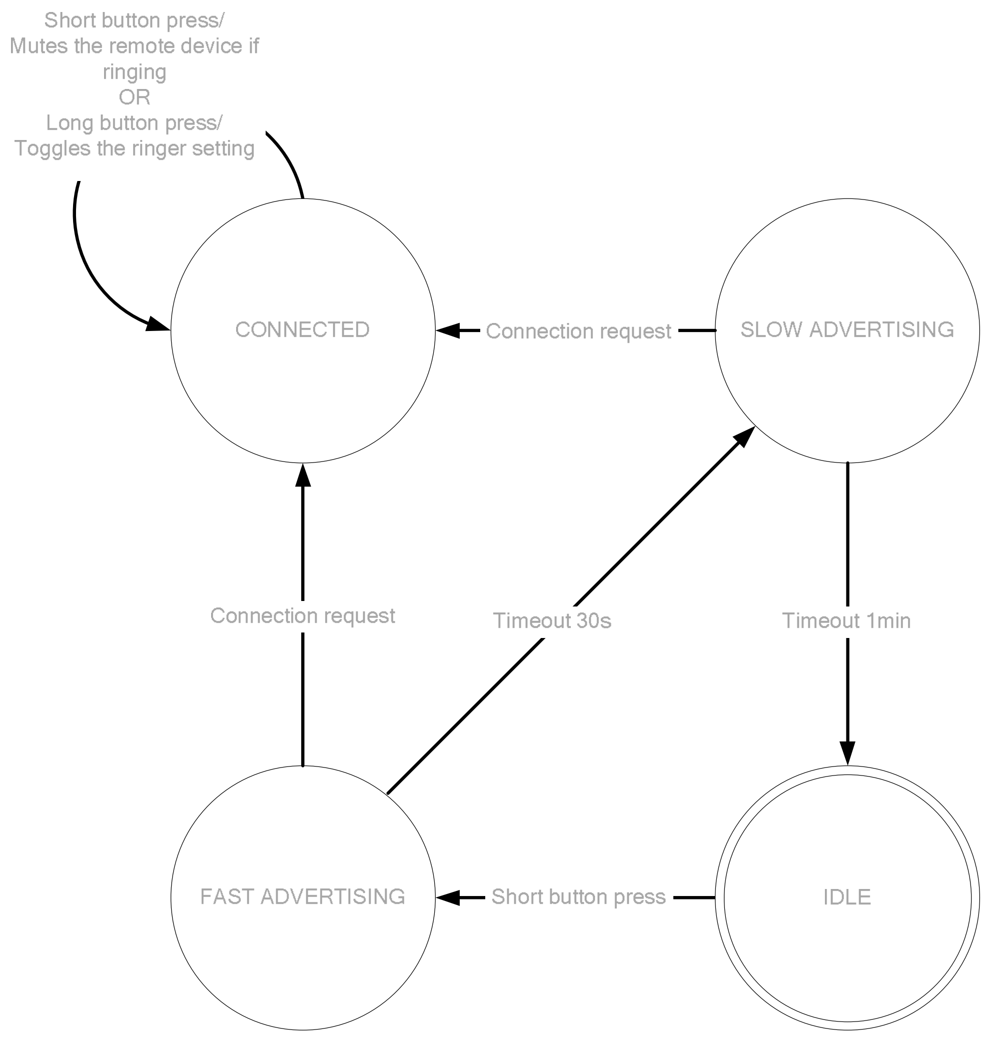


Figure 2.2: Device Behaviour

Buzzer Behaviour

- A single short beep on a **Short button press** indicates that the Alert Tag has silenced the remote connected Alert Server (if ringing).
- A single short beep without a **Short button press** indicates the reception of a New Alert.
- Two short beeps are used to indicate start of advertisements.
- Three short beeps indicate the removal of bonding.
- A long beep on a **Long button press** indicates the toggling of Ringer Setting of remote connected Alert Server between Normal and Mute.
- A long beep without the button being pressed indicates that the Alert Tag application has stopped advertising and has entered the non-connectable mode. This happens when the Alert Tag advertises for a certain time without an Alert server connecting to it.

UART Messages

The Alert Tag application uses the UART connection to list the supported categories for new and unread Alert Notifications. Notifications for Unread and New Alert notifications received from the Alert Server are also sent over this connection. A terminal client application such as HyperTerminal can be used to display these notifications using the following communication configuration:

- 2400 Baud Rate
- 8 Data Bits
- No Parity Bit
- 1 Stop Bit

The UART baud rate is configured using the supplied **CsConfig** tool.

Note:

Access to the UART is enabled by soldering the bridges on the development board. See the *CSR µEnergy Development Kit Quick Start Guide* for more information.

2.1.2. Alert Tag Server

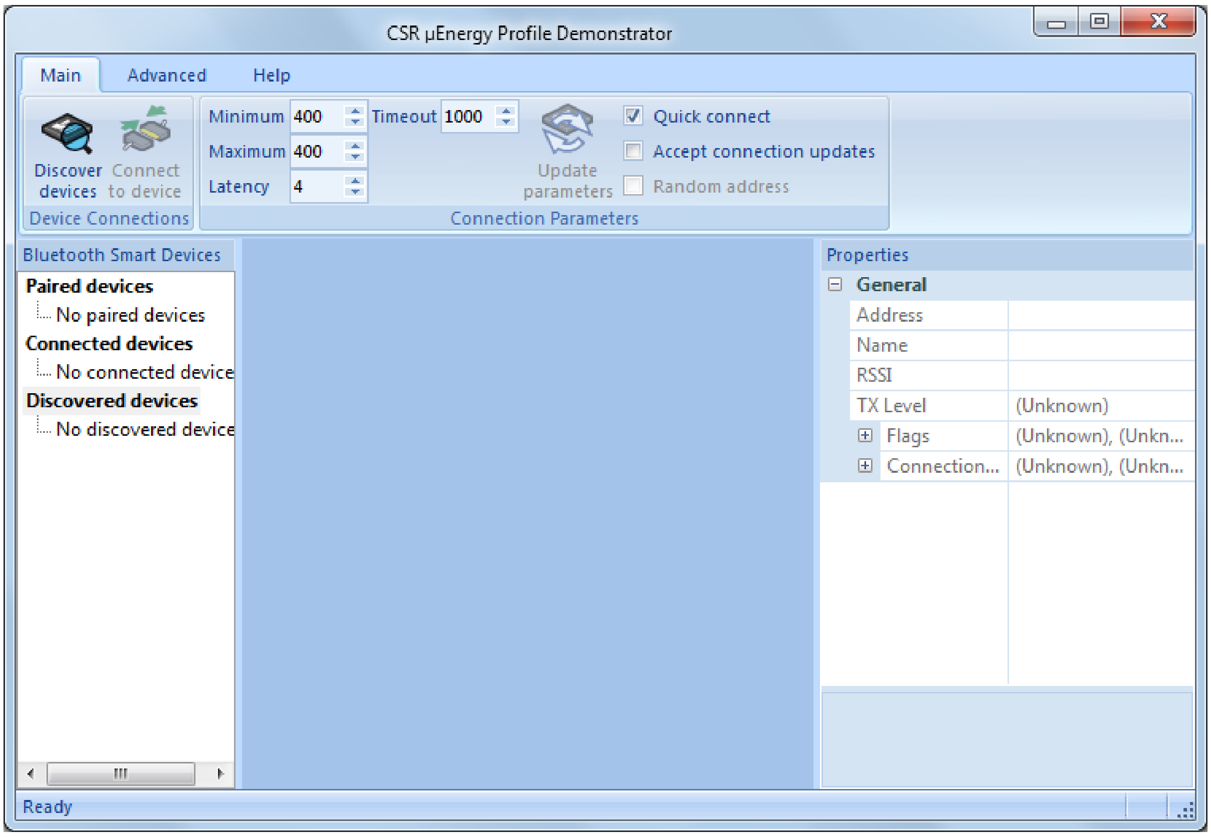
2.1.2.1. CSR µEnergy BT4.0 USB Dongle

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 CSR8510 USB Driver for the Profile Demonstrator Application* user guide.

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 and Windows 7 (32-bit and 64-bit).The application may be launched when the USB dongle is attached to the PC and the driver has been loaded.


CSR µEnergy Profile Demonstrator

Main | Advanced | Help

Minimum 400 | Timeout 1000
Maximum 400
Latency 4

☑ Quick connect
☐ Accept connection updates
☐ Random address

Bluetooth Smart Devices

Paired devices
No paired devices
Connected devices
No connected device
Discovered devices
No discovered device

Properties

General	
Address	
Name	
RSSI	
TX Level	(Unknown)
Flags	(Unknown), (Unkn...
Connection...	(Unknown), (Unkn...

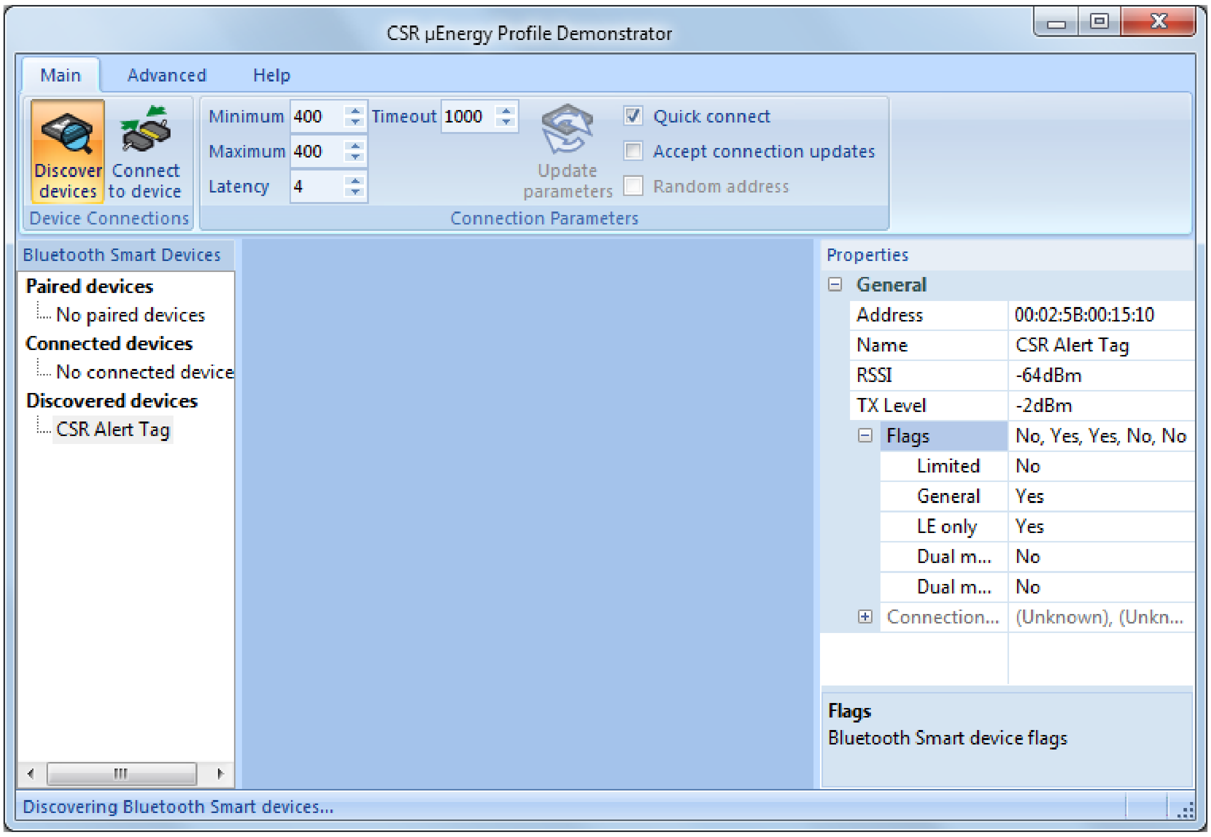
Ready

Figure 2.4: CSR µEnergy Profile Demonstrator

2.2. Demonstration Procedure

To demonstrate the Alert Tag application:

3. Switch on the development board to trigger advertisements.
4. 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.



Minimum 400
Maximum 400
Latency 4
Timeout 1000

☑ Quick connect
☐ Accept connection updates
☐ Random address

Paired devices
No paired devices
Connected devices
No connected device
Discovered devices
CSR Alert Tag

General	
Address	00:02:5B:00:15:10
Name	CSR Alert Tag
RSSI	-64dBm
TX Level	-2dBm
Flags	No, Yes, Yes, No, No
Limited	No
General	Yes
LE only	Yes
Dual m...	No
Dual m...	No
Connection...	(Unknown), (Unkn...

Flags
Bluetooth Smart device flags

Discovering Bluetooth Smart devices...

Figure 2.5: Alert Tag Devices Discovered

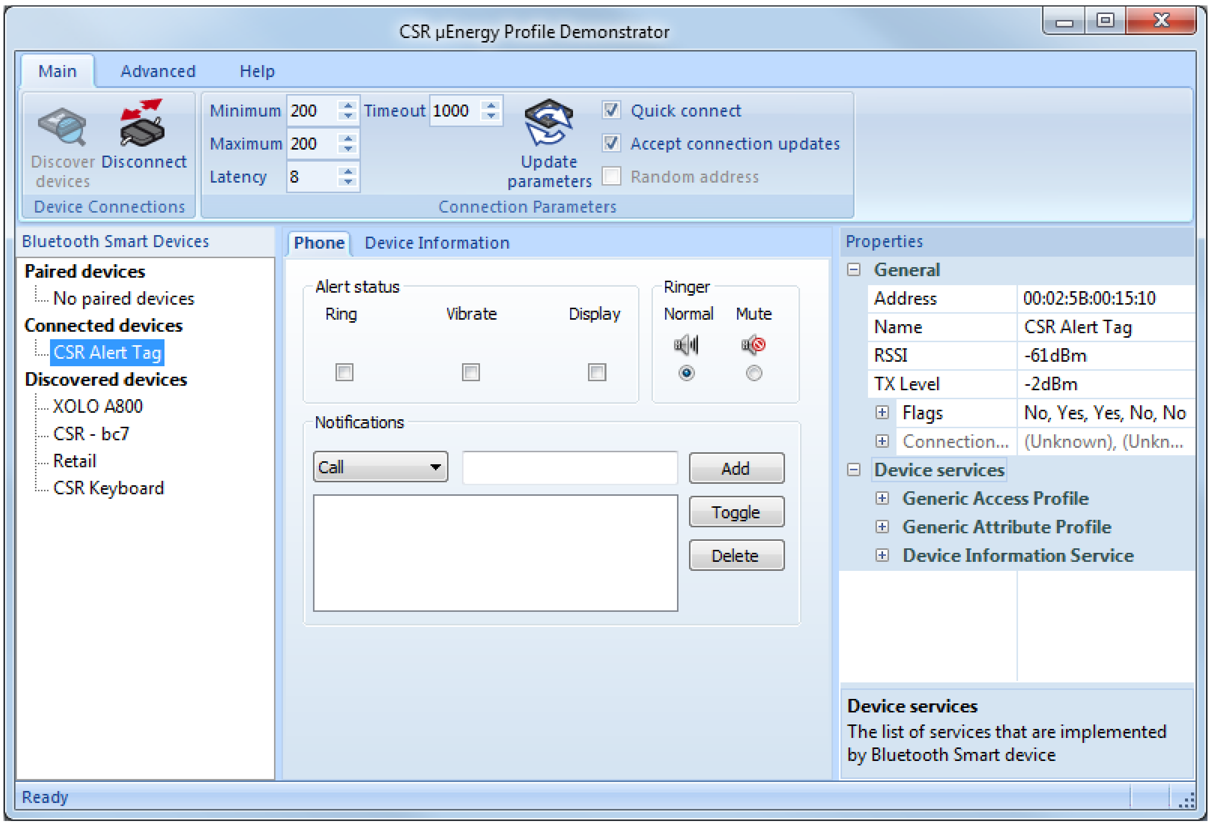
When the device labelled **Alert Tag** appears, select it to display the device address on the right hand side of the screen.

Click on the **Connect to device** button to connect to this device, see Figure 2.5.

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

Note:

> The Alert Tag application disconnects the link if the remote connected device does not support both the Alert Notification and the Phone Alert Status Service.



Minimum 200; Maximum 200; Latency 8; Timeout 1000

☑ Quick connect
☑ Accept connection updates
☐ Random address

Paired devices
- No paired devices

Connected devices
- CSR Alert Tag

Discovered devices
- XOLO A800
- CSR - bc7
- Retail
- CSR Keyboard

Alert status: Ring ☐ Vibrate ☐ Display ☐

Ringer: [✓ Normal] Mute

Notifications: Call

General	
Address	00:02:5B:00:15:10
Name	CSR Alert Tag
RSSI	-61dBm
TX Level	-2dBm
Flags	No, Yes, Yes, No, No
Connection...	(Unknown), (Unkn...
Device services	
Generic Access Profile	
Generic Attribute Profile	
Device Information Service	

Device services
The list of services that are implemented by Bluetooth Smart device

Figure 2.6: Device Connected

5. The **Phone** tab in the CSR µEnergy Profile Demonstrator application window shows three configuration groups:

   5.1. **Alert Status**
   This configuration group corresponds to the Phone Alert Status Profile:

   - **Ring**
     The current Ringer state of the remote connected Alert Server device is displayed. Ticking this check box activates the ringer; a notification is sent to the Alert Tag Client which then outputs the message over UART. When the Ringer Setting is set to Normal and the Ringer State is active i.e. ringing, a **Short button press** on the Alert Tag client mutes the Ringer and the Ringer State is changed to inactive. This is indicated by a short beep.

- **Vibrate**
  The current Vibrator state of the remote connected Alert Server device is displayed. Ticking this check box activates the vibrator; a notification is sent to the Alert Tag Client which then outputs the message over the UART.
- **Display**
  The display state of the remote connected Alert Server device is shown. Ticking this check box activates the display; a notification is sent to the Alert Tag Client which then outputs the message over the UART.

5.2. Ringer

This configuration group corresponds to the Phone Alert Status Profile and shows the current Ringer Setting of the Alert Server device. Selecting **Mute** disables the Ring Alert Status if active. A **Long Button Press** on the Alert Tag device toggles this ringer setting between Normal and Mute.

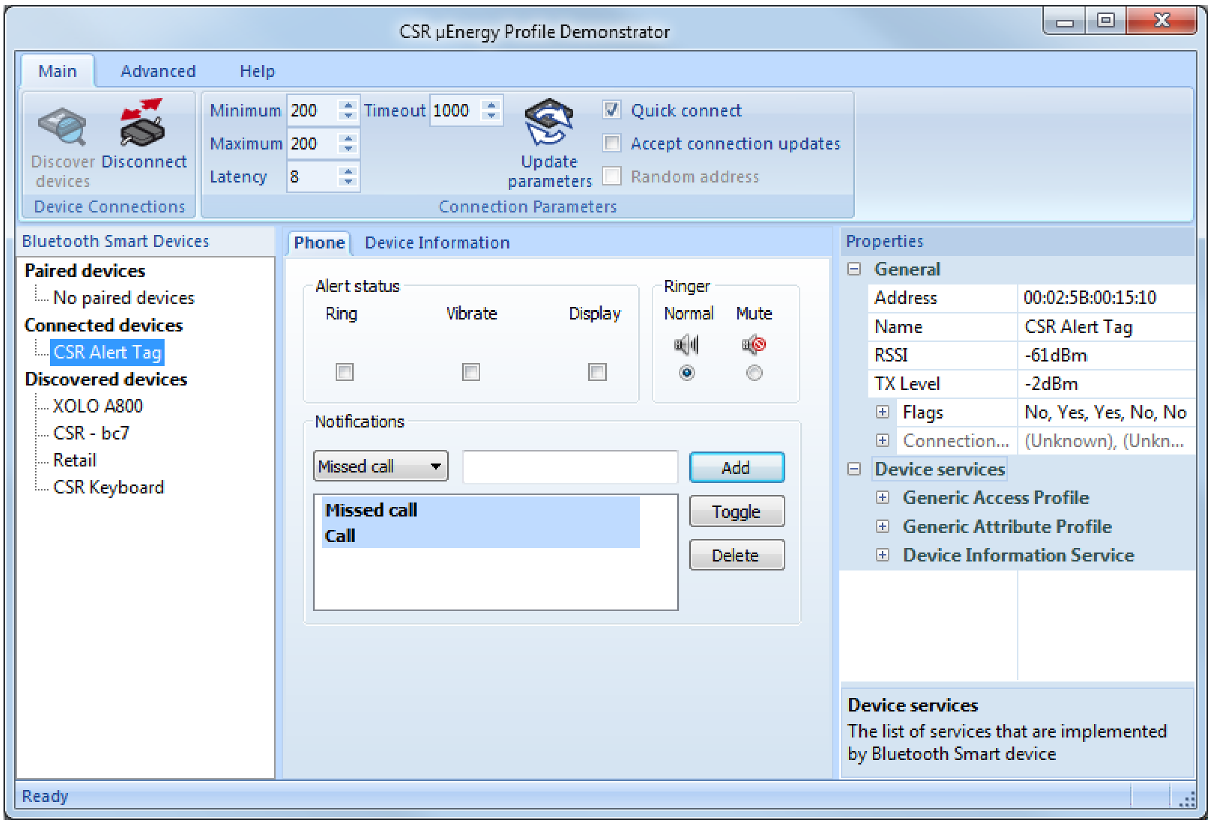
5.3. Notifications

This configuration group corresponds to the Alert Notification Profile and is used to send New Alert and Unread Alert notifications for the Alert category selected from the dropdown box. Optional Text String Information can also be added.

- Pressing the **Add** button generates a notification of the category selected in the dropdown box and is added to the list of notification types; a new alert followed by an unread alert notification is sent to the Alert Tag Client. The CSR µEnergy Profile Demonstrator application immediately acknowledges this new alert and sends a "0 new alerts" notification, see Figure 2.7.

  New Alert Handling: Receiving a New Alert is indicated by a beep on the Alert Tag device. The category and count of New Alerts is displayed over the UART.

  Unread Alert Handling: The category and count of Unread Alerts is displayed over the UART.



Minimum 200 · Maximum 200 · Latency 8 · Timeout 1000

[✓] Quick connect
[ ] Accept connection updates
[ ] Random address

Paired devices
- No paired devices

Connected devices
- CSR Alert Tag

Discovered devices
- XOLO A800
- CSR - bc7
- Retail
- CSR Keyboard

Alert status: Ring [ ] Vibrate [ ] Display [ ]

Ringer: [✓ Normal] Mute

Notifications: Missed call

Missed call
Call

Properties

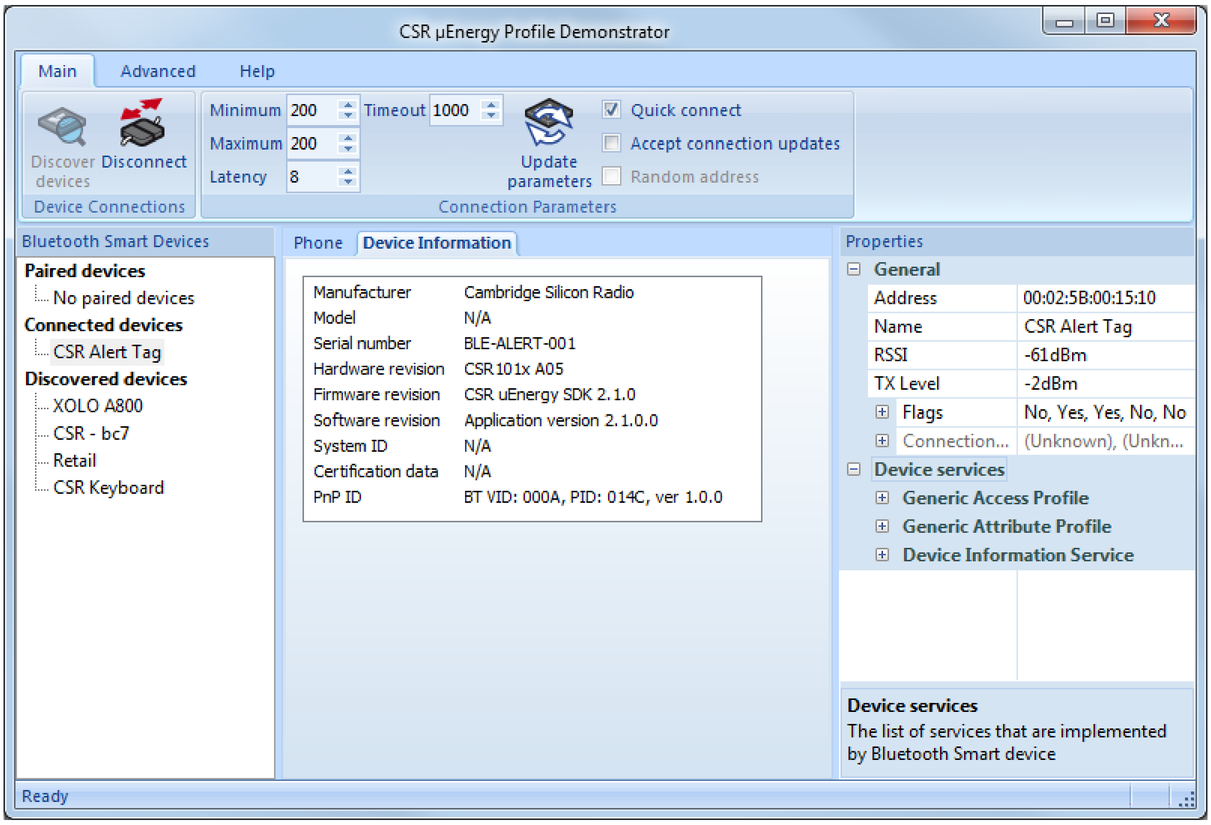
General	
Address	00:02:5B:00:15:10
Name	CSR Alert Tag
RSSI	-61dBm
TX Level	-2dBm
Flags	No, Yes, Yes, No, No
Connection...	(Unknown), (Unkn...
Device services	
Generic Access Profile	
Generic Attribute Profile	
Device Information Service	

Device services
The list of services that are implemented by Bluetooth Smart device

Figure 2.7: Add Button Function

- Pressing the **Toggle** button marks the selected alert in the list as read, sends an Unread Alert to the Alert Tag device and the bold highlight is removed. The category and count of the Unread Alert received is displayed over the UART.
- Pressing the **Delete** button deletes the selected alert in the text box and sends a notification to the Alert Tag Client which is displayed over the UART.

6. The **Device Information** tab displays the device information characteristics of the Alert Tag application, see Figure 2.8.



Manufacturer	Cambridge Silicon Radio
Model	N/A
Serial number	BLE-ALERT-001
Hardware revision	CSR101x A05
Firmware revision	CSR uEnergy SDK 2.1.0
Software revision	Application version 2.1.0.0
System ID	N/A
Certification data	N/A
PnP ID	BT VID: 000A, PID: 014C, ver 1.0.0

General	
Address	00:02:5B:00:15:10
Name	CSR Alert Tag
RSSI	-61dBm
TX Level	-2dBm
Flags	No, Yes, Yes, No, No
Connection...	(Unknown), (Unkn...
Device services	
Generic Access Profile	
Generic Attribute Profile	
Device Information Service	

Device services
The list of services that are implemented by Bluetooth Smart device

Figure 2.8: Device Information Service

7. 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
`alert_client.c`	Implements all the entry functions e.g. `AppInit()`, `AppProcessSystemEvent()` and `AppProcessLmEvent()`. Events received from the hardware or the firmware are first handled here. This file contains handling functions for all the LM and system events.
`alert_client_gatt.c`	Implements routines for triggering advertisement procedures.
`alert_client_hw.c`	Implements routines for hardware initialisation, button press handling, indicating different states using buzzer beeps and printing messages over the UART.
`alert_service_notification_data.c`	Implements routines required for configuring the Alert Notification server, reading different characteristics values of the Alert Notification service and handling their responses.
`phone_alert_serivce_data.c`	Implements routines required for configuring the Phone Alert Status service, reading different characteristic values of the Phone Alert Status service and handling their responses.
`gatt_service_data.c`	Implements routine for handling the service changed characteristic notification. See *Bluetooth Core Specification Version 4.0* for more information on the Service Changed characteristic.
`local_debug.c`	Implements routines for printing Strings and Numbers over the UART.
`nvm_access.c`	Implements the NVM read/write routines.

Table 3.1: Source Files and their Purpose

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 that are used across the application.
`appearance.h`	Contains the appearance value macro of the Alert Tag application.
`alert_client.h`	Contains prototypes of externally referred functions defined in the `alert_client.c` file.
`alert_client_gatt.h`	Contains macro definitions for fast and slow advertisement timers and prototypes of externally referred functions defined in the `alert_client_gatt.c` file.
`alert_client_hw.h`	Contains macro definitions for PWM parameters, different button press timer values and prototypes of the externally referred functions defined in the `alert_client_hw.c` file.
`alert_notification_service_data.h`	Contains user defined data types and UUID macros related to the Alert Notification service and prototypes of externally referred functions defined in the `alert_notification_service_data.c` file.
`dev_info_uuids.h`	Contains UUID macros for the Device Information service.
`gap_conn_params.h`	Contains macro definitions for advertisement and connection parameters.
`gap_uuids.h`	Contains macros for UUID values for the GAP service and related characteristics
`gatt_service_data.h`	Contains user defined data types and UUID macros related to the GATT service and prototypes of externally referred functions defined in the `gatt_service_data.c` file.
`local_debug.h`	Contains prototypes of externally referred functions defined in the `local_debug.c` file.
`nvm_access.h`	Contains prototypes of externally referred functions defined in the `nvm_access.c` file.
`phone_alert_service_data.h`	Contains user defined data types and UUID macros related to the Phone Alert Status service and prototypes of externally referred functions defined in the `phone_alert_service_data.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 list 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.
`dev_info_service_db.db`	Contains information related to the Device Information Service characteristics, their descriptors and values. See Table B.1 for the Device Information service characteristics.
`gap_service_db.db`	Contains information related to the GAP service characteristics, their descriptors and values. See Table B.2 for the GAP service characteristics.
`gatt_service_db.db`	Contains information related to the GATT service characteristics, their descriptors and values.

Table 3.3: Database Files

4. Code Overview

This section describes the 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, tag hardware and application data structure
- Enables debug prints over the UART
- Configures the GATT entity for server role
- Resets the white list to remove any filtering of discovered devices. See *Bluetooth Core Specification Version 4.0* for more information on the white list
- Configures the NVM manager to use I^{2}C EEPROM
- Reads the persistent store
- Registers the ATT database with the firmware

4.1.2. AppProcessLmEvent()

The `AppProcessLmEvent()` function is invoked whenever a LM-specific event is received by the system. The events described in 4.1.2.1 to 4.1.2.4 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. The Alert Tag starts advertising on receiving this event.

4.1.2.2. GATT Events

- `GATT_SERV_INFO_IND`: This indication event is received in response to the Primary Service Discovery initiated by the Alert Tag application. One indication is received for each primary service discovered by the Alert Tag application. The Alert Tag application is interested in only two services; the Alert Notification service and the Phone Alert Status service.
- `GATT_DISC_ALL_PRIM_SERV_CFM`: This confirmation event marks the end of Primary Service Discovery procedure initiated by the Alert Tag application. If the remote connected device does not support both the Alert Notification Service and the Phone Alert Status Service, the alert tag application disconnects the link.
- `GATT_CHAR_DECL_INFO_IND`: This indication event is received in response to the characteristic discovery initiated by the Alert Tag application. One indication is received for each characteristic discovered.
- `GATT_DISC_SERVICE_CHAR_CFM`: This confirmation event marks the end of the characteristic discovery.
- `GATT_CHAR_DESC_INFO_IND`: This indication event is received in response to the characteristic descriptor discovery initiated by the Alert Tag application. One indication is received for each characteristic descriptor discovered.
- `GATT_DISC_ALL_CHAR_DESC_CFM`: This confirmation event marks the end of the characteristic descriptor discovery initiated by the Alert Tag application.

- `GATT_READ_CHAR_VAL_CFM`: This confirmation event is received in response to a Read request by the Alert Tag application.
- `GATT_WRITE_CHAR_VAL_CFM`: This confirmation event is received in response to a Write request by the Alert Tag application.

4.1.2.3. SMP Events

- `SM_KEYS_IND`: This indication event is received on completion of the bonding procedure. It contains the diversifier (DIV), keys and security information used on a connection that has completed short term key generation. The application stores the received diversifier and Identity Resolving Key (IRK) to the non-volatile memory (NVM). See the *Bluetooth Core Specification Version 4.0* for more information.
- `SM_SIMPLE_PAIRING_COMPLETE_IND`: This indication event indicates that the pairing has completed successfully or otherwise. In the case of a successful completion of pairing, the Alert Tag application is bonded with the remote Alert Tag Server and bonding information is stored in the NVM. The bonded remote device address is added to the white list if it is not a resolvable random address. See the *Bluetooth Core Specification Version 4.0* for more information on pairing and resolvable random addresses.
- `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. 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 an Alert Tag server to prevent any new device disguising itself as one previously bonded to the Alert Tag application.

4.1.2.4. Connection Events

- `GATT_CONNECT_CFM`: This confirmation event indicates that the connection procedure has completed. If it has not completed successfully, the application goes to into the idle mode and waits for user activity before starting advertising. If the application is bonded and the connection is established to the remote device that uses resolvable random address, the application tries to resolve the connected device address using the IRK stored. If it fails to resolve the address, it disconnects the link and starts advertising again.
- `GATT_CANCEL_CONNECT_CFM`: This confirmation event confirms the cancellation of connection procedure. When the application stops advertisements to change the advertising parameters or to save power, this signal indicates that advertisements have been successfully stopped.
- `LM_EV_DISCONNECT_COMPLETE`: This event marks the completion of the disconnection procedure. Disconnection could be due to link loss, either locally triggered or triggered by the remote connected device. The Alert Tag application starts advertising on receiving this event.
- `LM_EV_ENCRYPTION_CHANGE`: This event indicates a change in the link encryption.

4.1.3. AppProcessSystemEvent()

The `AppProcessSystemEvent()` function handles the system events such as low battery notification or PIO change:

- `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 a PIO changed event is generated. The application receives this PIO changed event and takes the appropriate action.

4.2. Internal State Machine

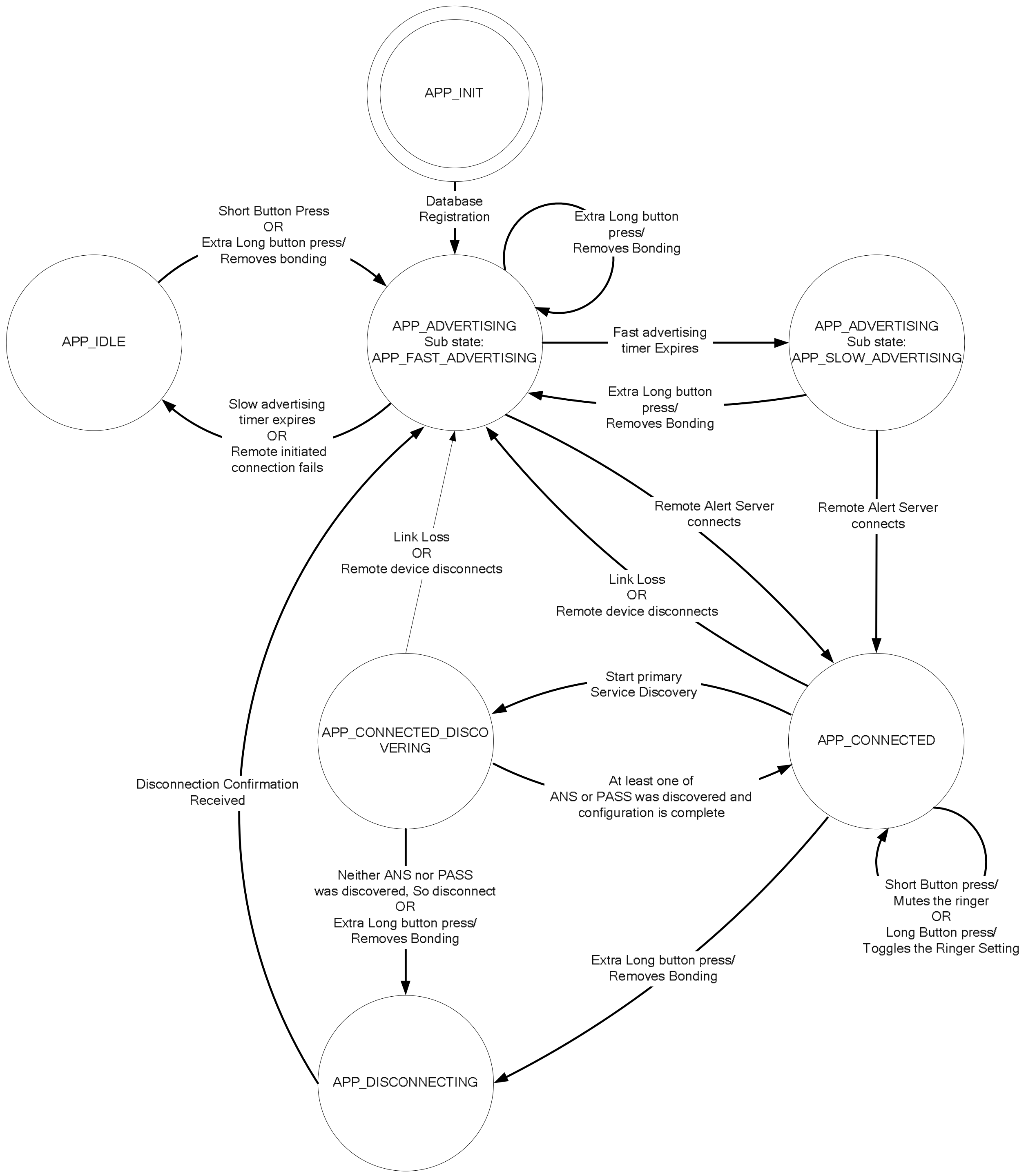

APP_INIT
Database Registration
Extra Long button press/ Removes Bonding
Short Button Press OR Extra Long button press/ Removes bonding
APP_IDLE
APP_ADVERTISING Sub state: APP_FAST_ADVERTISING
Fast advertising timer Expires
APP_ADVERTISING Sub state: APP_SLOW_ADVERTISING
Extra Long button press/ Removes Bonding
Slow advertising timer expires OR Remote initiated connection fails
Remote Alert Server connects
Remote Alert Server connects
Link Loss OR Remote device disconnects
Link Loss OR Remote device disconnects
Start primary Service Discovery
APP_CONNECTED_DISCOVERING
APP_CONNECTED
At least one of ANS or PASS was discovered and configuration is complete
Disconnection Confirmation Received
Neither ANS nor PASS was discovered, So disconnect OR Extra Long button press/ Removes Bonding
Short Button press/ Mutes the ringer OR Long Button press/ Toggles the Ringer Setting
Extra Long button press/ Removes Bonding
APP_DISCONNECTING

Figure 4.1: Internal State Machine

The Alert Tag application has six internal states, see Figure 4.1.

4.2.1. APP_INIT

When the application is powered on or the chip resets, the Alert Tag application initialises 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 Alert Tag application is not connected to any Alert server. On entering this state, a long beep is sounded.

- On a **Short button press**, the application triggers advertisements and enters the `APP_ADVERTISING` state.

4.2.3. APP_ADVERTISING

The application sends advertisements in this state. On entering this state, two short beeps are sounded.

- Sub state `APP_FAST_ADVERTISING`: The application starts in this sub state and uses fast advertising parameters. If the remote device connects to it, it stops advertisements and enters the `APP_CONNECTED` state. If the fast advertising timer expires before the connection is made, the `APP_SLOW_ADVERTISING` sub state is entered.
- Sub state `APP_SLOW_ADVERTISING`: The application uses slow advertising parameters in this Sub state. On connection to the remote device, advertisements cease and the application enters the `APP_CONNECTED` state. If the slow advertising timer expires before the connection is made or the connection fails, the application enters the `APP_IDLE` state.
- When the user performs 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 list.

4.2.4. APP_CONNECTED

The Alert Tag application is connected to the remote alert server.

- When the Alert Tag application starts the primary service discovery, it enters the `APP_CONNECTED_DISCOVERING` state.
- In the case of a link loss or remote triggered disconnection, the application enters the `APP_ADVERTISING` state.
- On an **Extra long button press**, the Alert Tag application disconnects the link, removes the bonding information and enters the `APP_DISCONNECTING` state.
- After service discovery and the Alert Server Configuration is complete;
  - A **Short button press** mutes the Ringer of the remote Alert server.
  - A **Long button press** toggles the Ringer Setting of the remote Alert Server.

4.2.5. APP_CONNECTED_DISCOVERING

When the Alert Tag application starts the service discovery, it enters the `APP_CONNECTED_DISCOVERING` state. After the service discovery and Alert server configuration is complete, the Alert Tag application switches back to the `APP_CONNECTED` state.

- In the case of a link loss or remote triggered disconnection, the application enters the `APP_ADVERTISING` state.
- If the remote device does not support both the Alert Notification service and the Phone Alert Status service, the Alert Tag application disconnects the link and enters the `APP_DISCONNECTING` state.
- On an **Extra Long button press** the Alert Tag application disconnects the link, removes bonding information and enters the `APP_DISCONNECTING` state.

4.2.6. APP_DISCONNECTING

The Alert Tag application waits for a disconnect confirmation for the disconnection initiated by it. On receiving a disconnection confirmation, the application enters the `APP_ADVERTISING` state.

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 Alert Tag application uses the memory map for NVM described in Table 5.1.

Entity Name	Description	Type	Size of Entity (Words)	NVM Offset (Words)
Sanity Word	Starting Word in NVM which gets written on First Power up	uint16	1	0
Bonded Flag	Flag indicating if application is bonded or not	Boolean	1	1
Bonded Device Address	Device Address of the bonded remote device	Structure	5	2
Diversifier	Parameter for security	uint16	1	7
IRK	Identity Resolving Key	uint16 array	8	8

Table 5.1: NVM Memory Map for Application

See *Bluetooth Core Specification Version 4.0* for more information on the Diversifier and the IRK.

Note:

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

6. Customising the Application

The developer can customise the application by modifying the parameter values described in sections 6.1 to 6.9.

6.1. Advertising Parameters

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 *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 Alert Tag application enters the appropriate state on expiry, see section 4.2 for more information. The macros for these timer values are defined in the `alert_client_gatt.h` file.

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 macros for these values are defined in the `gap_conn_params.h` file. These values have been chosen considering the overall current consumption of the device. See *Bluetooth Core Specification Version 4.0* for connection parameter range.

Parameter Name	Parameter Value
Minimum Connection Interval	250 ms
Maximum Connection Interval	250 ms
Slave Latency	8 intervals
Supervision Timeout	10 s

Table 6.3: Connection Parameters

6.4. Connection Parameter Update

The Alert Tag application can request the remote Alert server to update the connection parameters according to its power requirements. The application waits for 30 seconds after connection formation and then requests for a connection parameter update or when the remote Alert Server changes the connection parameters and the updated connection parameters does not comply with the Alert Tag application requirements. The remote Alert Server may or may not accept the requested parameters. If the remote Alert Server rejects the new requested parameters, the application requests for an update after 30 seconds. The macro for this time value is defined in file `app_gatt.c` and can be modified as required.

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

6.5. Device Name

By default, the device name is set to “CSR Alert Tag" in the `gap_uuids.h` file.

6.6. Manufacturer Nam

The developer can change the Manufacturer Name of the application. By default, it is set to "Cambridge Silicon Radio" in `dev_info_service_db.db`.

6.7. LED and Buzzer

The Alert Tag application uses the LED and buzzer to indicate the different states and events to the user. The user can enable or disable the LED and Buzzer as required. The macros for enabling the LED and buzzer code are defined in `alert_client_hw.h`.

6.8. UART Messages

The application prints messages over the UART to display the different Alert categories supported by the Alert Server. It also prints messages for Alert status, New alert or Unread alert received. The user can disable UART message printing by disabling the macro `DEBUG_THRU_UART` defined in `app_gatt.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 Alert Tag application uses the first CS key with index zero of the 8 keys provided for use by the application. See *CSR µEnergy xIDE User Guide* for more information on configuration keys.

- Bit[0] of the configuration key should be set to `1` to pass the following PTS test cases:
  - Alert Notification Profile PTS test cases:
    - TC_ANPCF_ANPC_BC_02_C
    - TC_ANPCF_ANPC_BC_04_C
  - Phone Alert Status Profile
    - TC_PPCF_PPC_BV_02_C
    - TC_PPCF_PPC_BV_04_C

These PTS test cases require the Alert Tag application to disable notifications on Client configuration descriptors of characteristics. The Alert Tag application default behaviour is to always enable notifications on Client configuration descriptors. Setting Bit[0] changes the application behaviour to disable notifications.

- Bit[1] of the configuration key should be set to `1` to pass the Phone Alert Status profile PTS test case:

- TC_PPWF_PPC_BV_01_C

This PTS test case requires the Alert Tag application to change the Ringer Setting to “Silent” mode irrespective of the current Ringer Setting. The Alert Tag application toggles the Ringer Setting on **Long button press**. Setting `bit[1]` changes the **Long button press** behaviour by changing the Ringer Setting to “Silent” mode.

For more information on PTS test cases, see *Alert Notification Profile Test Specification Version 1.0.0* and *Phone Alert Status Profile Test Specification Version 1.0.0*.

Note

Bit[0] and Bit[1] should be set to zero for normal operation of the Alert Tag application. These bits should only be set for these specific test cases and must not be set simultaneously.

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, sounding the buzzer or utilising the UART should be disabled before measuring the actual current consumed. See section 6 for more information on how to disable the LED, buzzer and UART print output.

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

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

Test Scenario	Description	Average Current Consumption	Remarks
Fast Advertisements	1. Switch on the Alert Tag Client 2. Wait for 5 s 3. Take the measurement	355 µA	▪ Advertisement Interval: 60 ms ▪ Advertisement data length: 7 octets ▪ Measurement Time Duration: 20 s
Slow Advertisements	1. Switch on the Alert Tag Client 2. Wait for 40 s 3. Take the measurement	23 µA	▪ Advertisement Interval: 1.28 s ▪ Advertisement data length: 7 octets ▪ Measurement Time Duration: 60 s
Connected Idle	1. Connect to the Alert Tag Server 2. Wait for 60 s 3. Take the measurement	13 µA	▪ Connection Parameters Minimum connection interval: 250 ms Maximum connection interval: 250 ms Slave latency: 8 ▪ Measurement Time Duration: 60 s
Connected Active	1. Connect to the Alert Tag Server 2. Wait for 60 s 3. Send new alert notification to the Alert Tag Client every 5 s from the Alert Tag Server	15 µA	▪ Connection Parameters Minimum connection interval: 250 ms Maximum connection interval: 250 ms Slave latency: 8 ▪ Measurement Time Duration: 60 s

Notes:

- Average current consumption is measured at 3.2 V
- Ammeter used: Agilent 34411A
- Buzzer, LED and UART messages 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

Table A.1: Definitions

Appendix B Service Characteristics

B.1.1 Device Information Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Static Value (If any)
Serial Number String	`0x000a`	Read	Firmware	Security Mode 1 and Security Level 2	"BLE-ALERT-001"
Hardware Revision String	`0x000c`	Read	Firmware	Security Mode 1 and Security Level 2	*<Chip Identifier>*
Firmware Revision String	`0x000e`	Read	Firmware	Security Mode 1 and Security Level 2	*<SDK Version>*
Software Revision String	`0x0010`	Read	Firmware	Security Mode 1 and Security Level 2	*<Application version>*
Manufacturer Name String	`0x0012`	Read	Firmware	Security Mode 1 and Security Level 2	"Cambridge Silicon Radio"
PnP ID	`0x0014`	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.1: Device Information Service Database

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

See *Device Information Service Specification Version 1.1* for more information on Device Information Service.

B.1.2 GAP Service Database

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Device Name	`0x0003`	Read	Firmware	Security Mode 1 and Security Level 1	Device name Default: "CSR Alert Tag"
Appearance	`0x0005`	Read	Firmware	Security Mode 1 and Security Level 1	Generic Tag "`0x0200`"

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Peripheral preferred connection parameters	`0x0007`	Read	Firmware	Security Mode 1 and Security Level 1	Min connection interval - 250 ms Max connection interval – 250 ms Slave latency - 8 Connection timeout - 10 s

Table B.2: Gap Service Database

See *Bluetooth Core Specification Version 4.0* for more information on GAP service and security aspects.

Appendix C Advertising and Scan Response Data

Table C.1 lists the fields the Alert Tag application adds to the Advertising data:

Advertising Data Field	Contents
Flags	The Alert Tag 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.
Device Appearance	Generic Tag : `0x0200`
Device Name[1]	Device name (Default value : “CSR Alert Tag”)
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

Table C.2 lists the fields the Alert Tag application adds 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 Alert Tag application and SDK Release Notes.

The application includes functionality specifically to support PTS testing, see section 6.9 for details.

Document References

Document	**Reference**
Bluetooth Core Specification Version 4.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Alert Notification Profile Specification Version 1.0	https://www.bluetooth.org/Technical/Specifications/adopted.htm
Phone Alert Status Profile 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
Alert Notification Profile Test Specification Version 1.0.0	https://www.bluetooth.org/Technical/Qualification/requirements.htm
Phone Alert Status Profile Test Specification Version 1.0.0	https://www.bluetooth.org/Technical/Qualification/requirements.htm
GATT Database Generator User Guide	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

ANS	Alert Notification Service
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
ID	Identifier
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
PASS	Phone Alert Status Service
PC	Personal Computer
PIO	Programmable Input Output
PnP	Plug and Play
PTS	Profile Testing Suite
PWM	Pulse Width Modulation

SIG	Special Interest Group
Tx	Transmit
UUID	Universally Unique Identifier