Push every boundary.™

CSR µEnergy®™

CSRmesh™ 1.0 Switch Application

Application Note

Issue 1

Document History

Revision	Date	History
1	25 JUL 14	Original publication of this document

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.

CSRmesh™ is a trademark owned by CSR plc

Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc or its affiliates.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

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

Safety-critical Applications

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

Performance and Conformance

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

Contents

Tables, Figures and Equations

1. Introduction

This document describes the CSRmesh™ Switch on-chip application built using the CSR µEnergy® Software Development kit (SDK).

The application demonstrates the following use cases:

- **Light Control**: The Switch application implements the CSRmesh messages related to the Light Model and the power model.
- **CSRmesh Bridge**: It also implements the CSR custom Mesh Control Service. This service allows a Bluetooth Smart enabled Phone to send and receive CSRmesh commands to many devices with the light application acting as a bridge

The CSRmesh Switch Application is provided as part of the CSRmesh release to demonstrate CSRmesh light control using an associated switch.

The application uses the CSRmesh library which is provided as part of the CSRmesh release. See *CSRmesh Library API* documentation for details.

The CSRmesh Switch Application is not compatible with the previous releases of CSRmesh

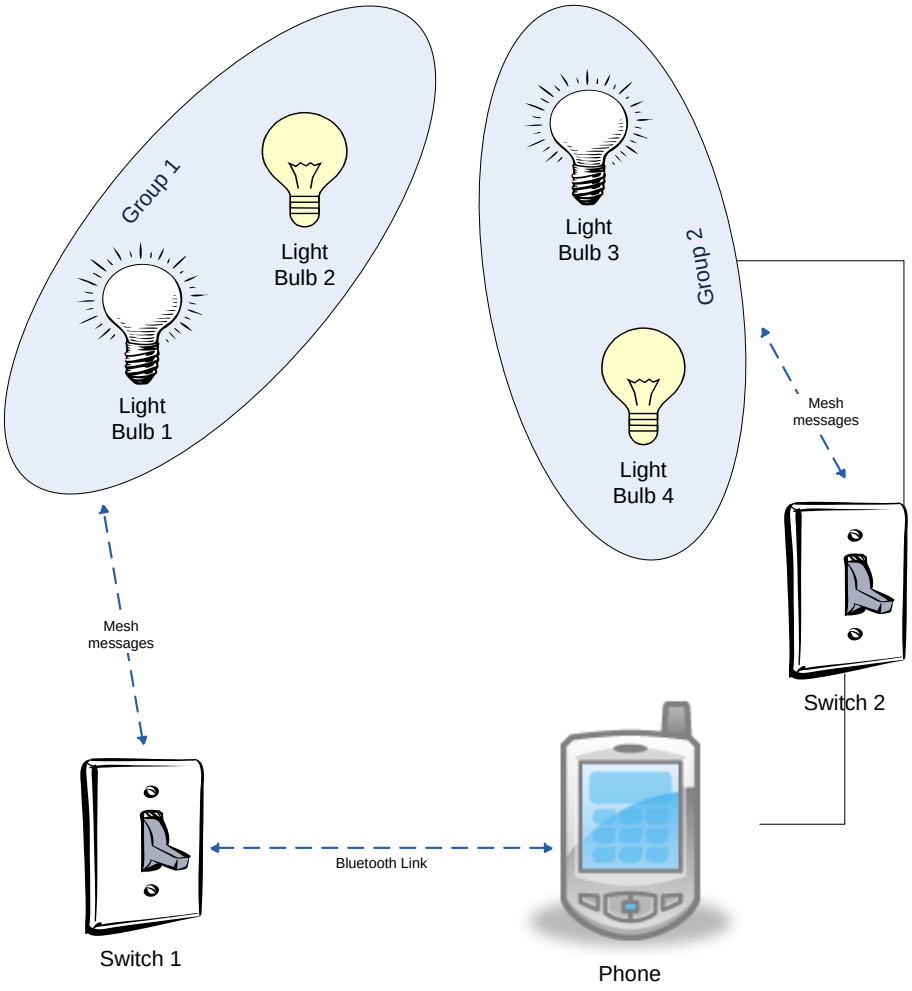

Figure 1.1: CSRmesh Use Case

1.1. Application Overview

The CSRmesh Switch application uses the *CSRmesh Library API* to communicate with other devices that are associated within the CSRmesh network. Additionally it supports a custom profile to allow control of the CSRmesh from a Bluetooth Smart enabled device.

1.1.1. Profiles Supported

The CSRmesh applications referred to in section 1 implement the following CSR custom profile to support the use cases described in section 1.

1.1.1.1. CSRmesh Control Profile

The CSRmesh Control Profile defines the behaviour when

- A network of devices (lights, switches, sensors etc.) needs to be created.
- Controlling the device after a network has been created. For example, Switching on/off, Intensity or the Colour of a light.
- Read the status of a device in network, i.e. on/off state, Colour or Intensity of a Light device.

The Profile defines the following two roles:

Role	Description
CSRmesh Bridge Device	It receives commands from host and sends them over the CSRmesh network. It receives responses from associated devices over the CSRmesh and forwards them onto the host over a BLE connection.
CSRmesh Control Device	The CSRmesh Control Device provides the interface to create a network of devices and to control the associated devices. The control commands are sent over a Bluetooth connection to the CSRmesh Devices.

Table 1.1: CSRmesh Control Profile Roles

The CSRmesh Bridge Device role is implemented on the CSRmesh Switch Application. The CSRmesh Control Device is implemented on a Bluetooth Smart enabled phone or a tablet.

1.1.2. Application Topology

The CSRmesh Bridge uses the following topology:

Role	Mesh Control Service	GAP Service	GATT Service	Device Information Service
GATT Role	GATT Server	GATT Server	GATT Server	GATT Server
GAP Role	Peripheral	Peripheral	Peripheral	Peripheral

Table 1.2: Application Topology

Role	Responsibility
GATT Server	It accepts incoming commands and requests from a client and sends responses, indications

Role	Responsibility
	and notifications to the client.
GAP Peripheral	It accepts connection from remote device and acts as a slave in the connection.

Table 1.3: Role and Responsibilities

For more information about GATT server and GAP peripheral, see *Bluetooth Core Specification Version 4.*1.

1.1.3. Services supported in GATT Server Role

The application exposes the following services

- Mesh Control Service (Version 2.0)
- GATT Service
- GAP Service
- CSR OTA Update Application Service (Version 2.0)

The Mesh Control Service is mandated by the CSRmesh Control Profile. The GATT and GAP Services are mandated by the *Bluetooth Core Specification Version 4.1*.

See Figure 1.2 for the services supported in the GATT Server Role.

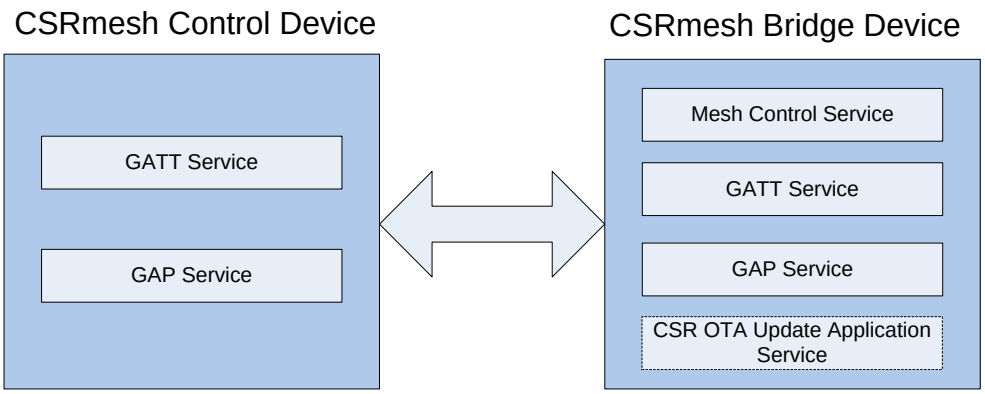

Figure 1.2: Primary Services

1.1.3.1. CSR OTA Update Application Service

The CSR OTA Update Application Service allows the application software to be updated without the need of a wired connection between the device being updated and the device providing the update. A PC or mobile phone application provided by the device manufacturer can allow the end user to keep their device up-to-date with the latest features and bug fixes. See OTA Update System Application Note for more information.

To enable a device for future OTA updates, application needs to:

- Add OTA Update functionality to the on-chip application
- Support for the CSR OTA Update Application Service and GATT Services must be added to an application. See *Modifying an Application to Support OTA Update Application Note*.
- Configure the on-chip bootloader
- The CSR OTA Update bootloader image needs to be present on the device and configured to contain the correct device name and (optional) authentication key, see *OTA Update System Application Note*.

When the device is enabled for OTA Update, the CSR µEnergy Over-the-Air Updater host application included in the SDK can be used to update the device, see *OTA Update System Application Note*.

2. Using the Application

This section describes how the CSRmesh application can be used with CSRmesh Android Application to control devices.

2.1. Demonstration Kit

This application can be demonstrated using the following components.

Component	Hardware	Application
Switch Device	IOT Lighting Board DB-CSR1010-10185-1A	CSRmesh Switch Application v1.0
CSRmesh Control Device	Android Bluetooth LE Device	CSRmesh Android Application v1.0

Table 2.1: CSRmesh Components

2.1.1. IOT Lighting Board

The µEnergy SDK is used to download the CSRmesh Switch application on the development boards. 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.

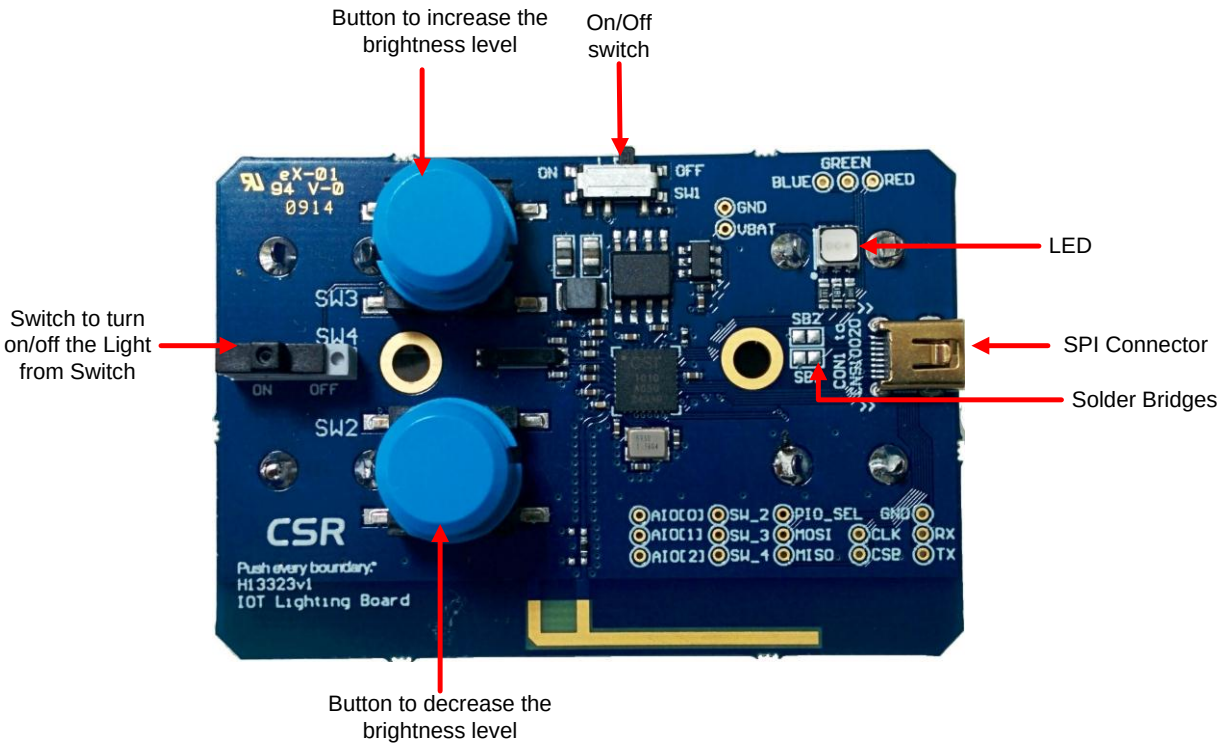

Figure 2.1: CSRmesh Development Board

Note 1:

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.

Note 2:

Shorting the solder bridges allows the UART to be brought out via the CSR SPI connector. The PIOs connected to SW2 and SW3 are also mapped to UART Rx and Tx lines. Pressing any of these buttons will short the UART lines to ground and corrupt the data on the UART. It is recommended not to press these buttons during UART communication.

2.1.1.1. User Interface

This application makes use of the buttons available on the CSRmesh Development Board to switching the Power On/Off and brightness of a CSRmesh Light or a group of lights

SW1 Switch

- This switch allows the user to power On/Off the board.

SW4 Switch

- This switch is used to change the Power state of the assigned CSRmesh Light or a group of Lights.

SW2 Button Press Behaviour

- Pressing the button once decreases the brightness by 1 level.
- Keeping the button pressed continuously will result in the following behaviour
  - The destination light brightness will decrease by 1 level immediately.
  - The brightness will decrease by 5 levels every 1 second until the minimum level is reached.

SW3 Button Press Behaviour

- Pressing the button once increases the brightness by 1 level.
- Keeping the button pressed continuously will result in the following behaviour
  - The destination light brightness will increase by 1 level immediately.
  - The destination light brightness will increase by 5 levels every 1 second until the max level is reached.

2.1.1.2. CSRmesh Device Tag

The IOT Lighting board is supplied with a CSRmesh Device Tag sticker as shown in the Figure 2.2. The sticker contains

- BT: Device Bluetooth Address
- Serial Number
- XTAL Trim
- UUID :CSRmesh Device UUID
- AC: CSRmesh Authorisation Code
- QR - Code : Encodes UUID and AC

Figure 2.2: CSRmesh Device Tag Sticker

The user can program the device with BT Address and the XTAL Trim printed on the sticker by setting these values in the `switch_csr101x_A05.keyr` file.

The Device UUID and the Authorisation Code printed on the sticker can be programmed on the NVM at the offsets defined in

Bearer Model Data	structure	3	32

Table 5.1 using the USB to SPI Adapter.

The example UUID = `0x9EC07735AC234DD2B28600025B041725` and Authorisation code = `0xC151DAB6678B787B` can be programmed on the NVM as described in the following steps

1. Open a command prompt
2. Program the Device UUID to the device EEPROM over the SPI link as follows

   ```
   \CSR_uEnergy_SDK-2.4.0\tools\bin\e2cmd -trans "SPITRANS=USB SPIPORT=2"
   writeblock F824 1725 5B04 0002 B286 4DD2 AC23 7735 9EC0
   ```

   First value after `writeblock` is the NVM address, obtained by adding base address `F800` and word offset 18 (See Table 5.1) for Device UUID. The command takes NVM address as byte address. Word offset 18 will be byte offset 36 decimal = 0x24. So effective address will be 0xF824

   Replace the `trans` option with values appropriate to the SPI connection in use.
3. Program the Device Authorisation code

   ```
   \CSR_uEnergy_SDK-2.4.0\tools\bin\e2cmd -trans "SPITRANS=USB SPIPORT=2"
   writeblock F834 787B 678B DAB6 C151
   ```

   First value after `writeblock` is the NVM address, obtained by adding base address `F800` and word offset `26` (See Table 5.1) for Authorisation code. The command takes NVM address as byte address. Word offset 26 will be byte offset 52 decimal or 0x34. So effective byte address will be 0xF834

   Replace the `trans` option with values appropriate to the SPI connection in use.

The CSRmesh Control application reads the Authorisation Code and the UUID from the QR-Code printed on the sticker during association. Refer to CSRmesh 1.0 Control Application Note for details.

2.1.1.3. Button De-bouncing

The application uses the following procedure to handle button de-bouncing.

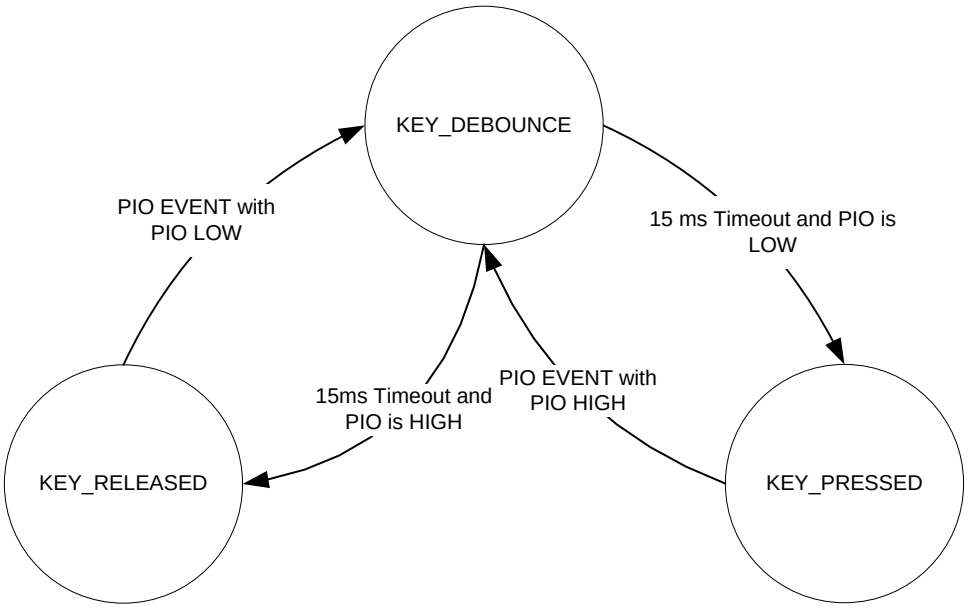

Figure 2.3: Handling of button De-bounce

Note:

The CSRmesh Switch application blinks the blue LED until it is associated to the CSRmesh Network. When the application gets an association request it will blink yellow till the time the association is completed.

2.1.2. CSRmesh Control Application

The CSRmesh Control Application runs on a phone or a Tablet which has the Android version 4.3 or higher. It communicates with the CSRmesh devices by connecting to one of the devices which supports CSR custom defined CSRmesh Control Profile. This application is required for

- Setting up a network by associating devices
- Configuring and grouping the network devices

Note:

For details on using the CSRmesh Control Application on an Android device refer to *CSRmesh 1.0 Control Application Note*.

3. Application Structure

3.1. Source Files

The table below lists the source files and their purpose.

File Name	Purpose
battery_hw.c	Implements the routines to read the battery level.
csr_mesh_switch.c	Implements all the entry functions e.g. AppInit(), AppProcessSystemEvent(), AppProcessCsrMeshEvent() and AppProcessLmEvent(). Events received from the hardware, CSRmesh network and firmware are first handled here. This file contains handling functions for all the LM, CSRmesh and system events
csr_mesh_switch_gatt.c	Implements routines for triggering advertisement procedures.
csr_ota_service.c	Implements the routines for handling the read/write access on the CSR custom defined Over-the-Air update service.
gap_service.c	Implements routines for GAP service e.g. handling read/write access indications on the GAP service characteristics, reading/writing device name on NVM etc.
gatt_service.c	Defines routines for using GATT service.
iot_hw.c	Implements the hardware interface to configure and control the peripherals on the CSRmesh Development Board
mesh_control_service.c	Implements routines for handling read/write access on Mesh Control characteristics and for sending notifications of mesh device responses.
nvm_access.c	Implements the NVM read/write routines.

Table 3.1: Source Files

3.2. Header Files

The table below lists the header files and their purpose.

File Name	Purpose
appearance.h	Contains the appearance value macro of the application.
app_debug.h	Contains macro definitions for enabling debug prints.
app_gatt.h	Contains macro definitions, user defined data type definitions and function prototypes which are being used across the application.
battery_hw.h	Contains prototypes of the externally referred functions defined in battery_hw.c.
csr_mesh_switch.h	Contains data structures and prototypes of externally referred functions defined in the csr_mesh_switch.c

	file
csr_mesh_switch_gatt.h	Contains timeout values and prototypes of externally referred functions defined in the csr_mesh_switch_gatt.c file.
csr_ota_service.h	Contains prototypes of the externally referred functions defined in the csr_ota_service.c file.
csr_ota_uuids.h	Contains macros for UUID values for CSR OTA Update service
customisation.h	Contains defines for the application data variables used by the csr_ota_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 the gap_service.c file.
gap_uuids.h	Contains macros for UUID values for GAP service.
gatt_service_uuids.h	Contains macros for UUID values for GATT service.
iot_hw.h	Contains the function declarations t to control the hardware interfaces.
mesh_control_service.h	Contains prototypes of the externally referred functions defined in the mesh_control_service.c file.
mesh_control_service_uuids.h	Contains macros for UUID values for Mesh Control service.
nvm_access.h	Contains prototypes of externally referred NVM read/write functions defined in the nvm_access.c file.
ota_customisation.h	Customisation definitions for the integration of CSR OTAU functionality with the application.
user_config.h	Contains macros for customising the application.

Table 3.2: Header Files

3.3. Database Files

The SDK uses database files to generate attribute database for the application. For more information on how to write database files, see *GATT Database Generator User Guide*.

The table below 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.
gatt_service_db.db	Contains information related to GATT service characteristics, their descriptors and values. See Table A.1 for GATT service characteristics.
gap_service_db.db	Contains information related to GAP service

	characteristics, their descriptors and values. See Table A.2 for GAP characteristics.
`csr_ota_db.db`	Contains information related to CSR Over-the-Air Update Application service characteristics, their descriptors and values. See Table A.3 for CSR OTA service characteristics.
`mesh_control_service.db`	Contains information related to CSRmesh Control service characteristics, their descriptors and values. See Table A.3 for CSRmesh Control service characteristics.

Table 3.3: Database Files

4. Code Overview

The following sections describe significant functions of the application.

4.1. Application Entry Points

4.1.1. AppInit ()

This function is invoked when the application is powered on or the chip resets and performs the following initialisation functions:

- Initialises the application timers, hardware and application data structures.
- Configures GATT entity for server role.
- Configures the NVM manager to use I^2C EEPROM.
- Initialises all the services.
- Reads the persistent store.
- If the Bit 0 of the CS_USER_KEY 2 is set, it generates a random device UUID when the application is executed for the first time after it is programmed. See

2	CSRmesh Configuration Bitmask	0000 to 0007	Bit-0 : CSRmesh Random Device UUID Enable 1 - Enables generation of Random Device UUID. 0 - Reads the UUID from NVM. If this key is set to 1 the application generates a random UUID and stores it in the NVM when it is run for the first time. This can be used to avoid having same UUID on multiple devices without explicitly programming a UUID on each device. Bit-1 : CSRmesh Relay Enable 1 - Enables relay of CSRmesh messages. 0 - Disables relay of CSRmesh messages. Bit-2 : CSRmesh Bridge Enable 0 - Disables connectable advertisements. 1 - Enables connectable advertisements. Disabling Bridge leaves the device non connectable on any service. The connectable adverts can be enabled by sending a `CSR_MESH_BEARER_SET_STATE` message with the LE GATT server bearer bit set Bits 3 to 15 are ignored.

- Table 6.6 for details.

- Registers the ATT database with the firmware.
- Initialises the CSRmesh Library and all the models that are supported.

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 application starts advertising.
- `GATT_ACCESS_IND`: This indication event is received when the CSRmesh Control device tries to access an ATT characteristic managed by the application.

4.1.2.2. LS Events

- `LS_CONNECTION_PARAM_UPDATE_CFM`: This confirmation event is received in response to the connection parameter update request by the application. The connection parameter update request from the application triggers the L2CAP connection parameter update signalling procedure. See Volume 3, Part A, Section 4.20 of *Bluetooth Core Specification Version 4.1*.
- `LS_CONNECTION_PARAM_UPDATE_IND`: This indication event is received when the remote central device updates the connection parameters. On receiving this event, the application validates the new connection parameters against the preferred connection parameters and triggers a connection parameter update request if the new connection parameters do not comply with the preferred connection parameters.
- `LS_RADIO_EVENT_IND`: This radio event indication is received when the chip firmware receives an acknowledgement for the Tx data sent by the application. On receiving this event, the application aligns the timer wakeup (which sends data periodically to the collector) with the latent connection interval.

4.1.2.3. LM Events

- `LM_EV_ADVERTISING_REPORT`: This event is received when an advertisement packet is received. This could be a CSRmesh advertisement. The application passes the event data to the CSRmesh library to process the packet by calling the `CsrMeshProcessMessage()` API.

4.1.2.4. SMP Events

- `SM_KEYS_IND`: This indication event is received on completion of the bonding procedure. It contains keys and security information used on a connection that has completed the short term key generation. The application stores the received diversifier (DIV) and Identity Resolving Key (IRK) (if the collector device uses resolvable random address) to NVM. See Volume 3, Part H, Section 2.4 and Section 3.6 of the *Bluetooth Core Specification Version 4.1*.
- `SM_SIMPLE_PAIRING_COMPLETE_IND`: This indication event indicates that the pairing has completed successfully or otherwise. See Volume 3, Part H, Section 2.4 and Section 3.6 of *Bluetooth Core Specification Version 4.1*. In the case of a successful completion of the pairing procedure, the application is bonded with the control device and bonding information is stored in the NVM. The bonded device address will be added to the white list, if it is not a resolvable random address.
- `SM_DIV_APPROVE_IND`: This indication event is received when the remote connected device re-encrypts the link or triggers encryption at the time of reconnection. The firmware sends the diversifier in this event and waits for the application to approve or disapprove the encryption. The application shall disapprove the encryption if the bond has been removed by the user. The firmware compares this diversifier with the one it had received in `SM_KEYS_IND` at the time of the first encryption. If similar, the application approves the encryption, otherwise it disapproves it.

4.1.2.5. 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 for more information on application states. If the application is bonded to a device with

resolvable random address and connection is established, the application tries to resolve the connected device address using the IRK stored in NVM. If the application fails to resolve the address, it disconnects the link and restarts advertising.

- GATT_CANCEL_CONNECT_CFM: This confirmation event confirms the cancellation of connection procedure. When the application stops advertisements to change advertising parameters or to save power, this signal confirms the successful stopping of advertisements by the CSRmesh application.
- LM_EV_CONNECTION_COMPLETE: This event is received when the connection with the master is considered to be complete and includes the new connection parameters.
- LM_EV_DISCONNECT_COMPLETE: This event is received on link disconnection. Disconnection could be due to link loss, locally triggered or triggered by the remote connected device.
- LM_EV_ENCRYPTION_CHANGE: This event indicates a change in the link encryption.
- LM_EV_CONNECTION_UPDATE: This event indicates that the connection parameters have been updated to a new set of values and is generated when the connection parameter update procedure is either initiated by the master or the slave. These new values are stored by the application for comparison against the preferred connection parameter, see section 6.3.

4.1.3. AppProcessSystemEvent ()

This function handles the system events such as a low battery notification or a PIO change. The CSRmesh applications currently handle the following system events:

- 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.

4.1.4. AppProcessCsrMeshEvent ()

This function handles the CSRmesh events received from the CSRmesh network, such as setting the light state, status or any other responses for commands sent over CSRmesh.

4.1.4.1. Raw Messages

- CSR_MESH_RAW_MESSAGE: This event is received whenever the device receives a mesh message. CSRmesh application notifies the message to the control device on the Mesh Control Service if connected and notifications are enabled.

The following events are handled by the CSRmesh Switch Application.

4.1.4.2. Light State Messages

- CSR_MESH_LIGHT_STATE: This event is received as a response to a command sent to update or query the state of a light device.

4.1.4.3. Power State Messages

- CSR_MESH_POWER_STATE: This event is received as a response to a command sent to update or query the power state of a device.

4.1.4.4. Network Association Messages

- CSR_MESH_ASSOCIATION_REQUEST: This event is received when a CSRmesh Control application sends an association request to a CSRmesh device which is ready for association. The application starts blinking in yellow to indicate association in progress and stops sending Device UUID advertisements.
- CSR_MESH_KEY_DISTRIBUTION: This event is received when the CSRmesh Control device provides the network key to be used for all future messages to communicate on the network. The application switches the state to associate and turns off the LED to indicate association completion and stores the association status on the NVM.

4.1.4.5. Device Configuration Messages

- CSR_MESH_CONFIG_DEVICE_IDENTIFIER: This message is received when a configuring device assigns a new device identifier once the device has received the network key.

- CSR_MESH_CONFIG_RESET_DEVICE: This message is received when a configuring device wants to remove all the CSRmesh network information from device. The application resets the assigned group IDs and starts sending Device UUID advertisements.

4.1.4.6. Battery Model Messages

- CSR_MESH_BATTERY_GET_STATE: This message is received when the control device queries the battery status.

4.1.4.7. Group Model Messages

- CSR_MESH_GROUP_SET_MODEL_GROUPID: This message is received when the control device sets a new group ID to a supported model. The CSRmesh Switch Application uses this message to set the destination group IDs which it will control.

4.1.4.8. Bearer Model Messages

- CSR_MESH_BEARER_SET_STATE: The application enables or disables the relay as set in the message when this message is received
- CSR_MESH_BEARER_GET_STATE: The application returns the current bear state data when this message is received.

4.2. Internal State Machine for GATT connection

This section describes the different state transition in the application during connection.

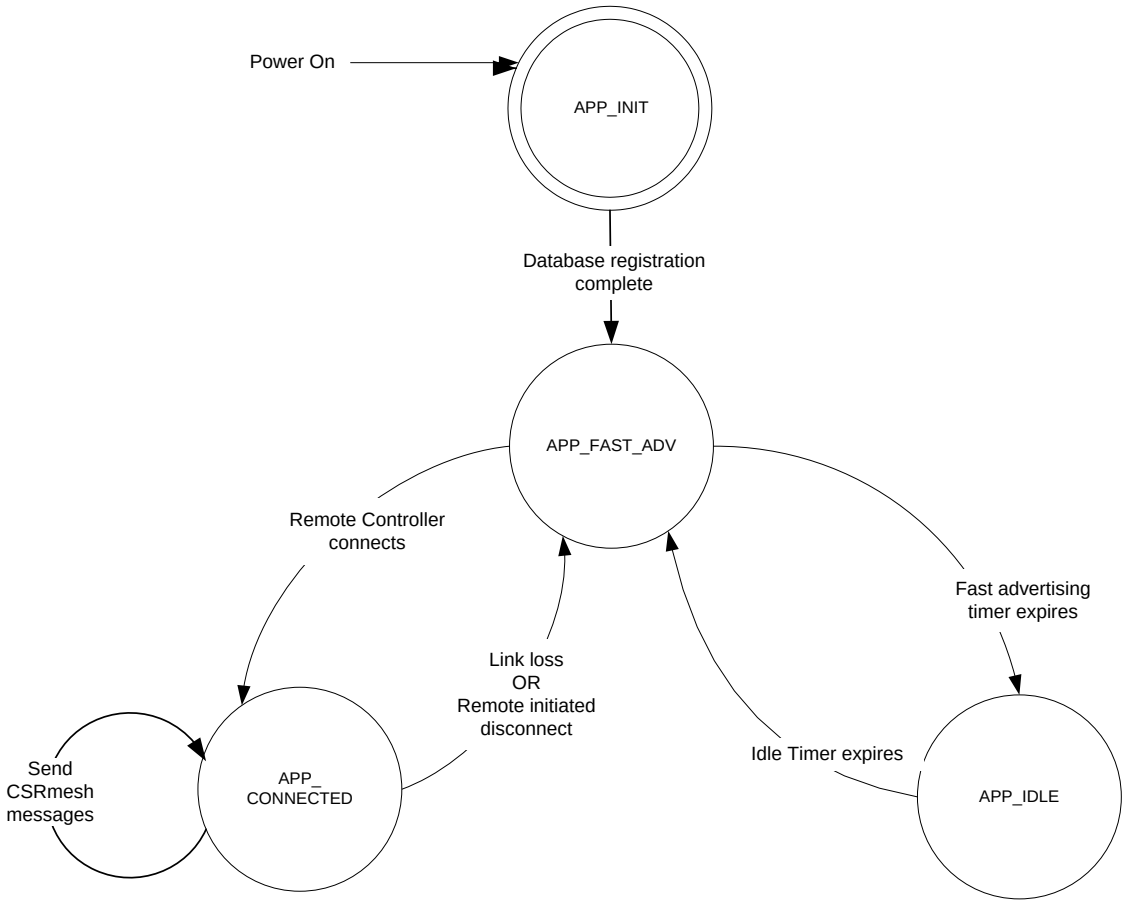

Figure 4.1: Internal GATT State Machine

4.2.1. app_state_init

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

4.2.2. app_state_advertising

The application starts in this state and transmits one shot advertisements every one second. If a remote collector connects to it, it stops advertisements and enters the `app_state_connected`. If the fast advertising timer expires before connection is made, the `app_state_idle` is entered. See section 6.2 for more information on advertisement timers.

While in the above state:

- If the application is bonded to some remote device, it will add the bonded device's address to its white list. This means that it will accept connections only from this bonded device. While triggering advertisements, it starts a Bonded Device Advertisement Timer. If the bonded remote device connects to it within this interval, it stops advertising and enters the `app_state_connected`.
- If the Bonded Device Advertisement Timer expires before the remote bonded device connects to it, it stops advertising, disables the white list and again starts fast advertising for a certain interval period. If

the remote collector connects to it, it stops advertisements and enters the `app_state_connected`. See the *Bluetooth Core specification Version 4.1* for more information on white list.

4.2.3. app_state_idle

The CSRmesh application enters this state on expiry of `app_state_advertising`. It remains in this state until idle timer expiry and then moves back to `app_state_advertising`.

4.2.4. app_state_connected

In this state, the CSRmesh application is connected to a CSRmesh Control device using connection intervals specified in Table 6.4. It can receive commands from Control device or send responses received over CSRmesh to control device.

- If link loss occurs, the application switches to the `app_state_advertising` state.
- In the case of a Remote triggered disconnection, it again starts advertising and enters the `app_state_advertising` state.

4.2.5. app_state_disconnecting

The CSRmesh application never triggers a disconnection on its own.

- If the disconnection was triggered by then, the application will enter state `app_state_advertising`.

4.3. CSRmesh Association

The application needs to be associated with a CSRmesh network to communicate with other devices in the network. The application sends CSRmesh Device UUID advertisements every 5 seconds for the CSRmesh Control device to find the device and send an association request. For details on device association refer to *CSRmesh 1.0 Control Application Note*.

The following diagram describes the application association state machine.

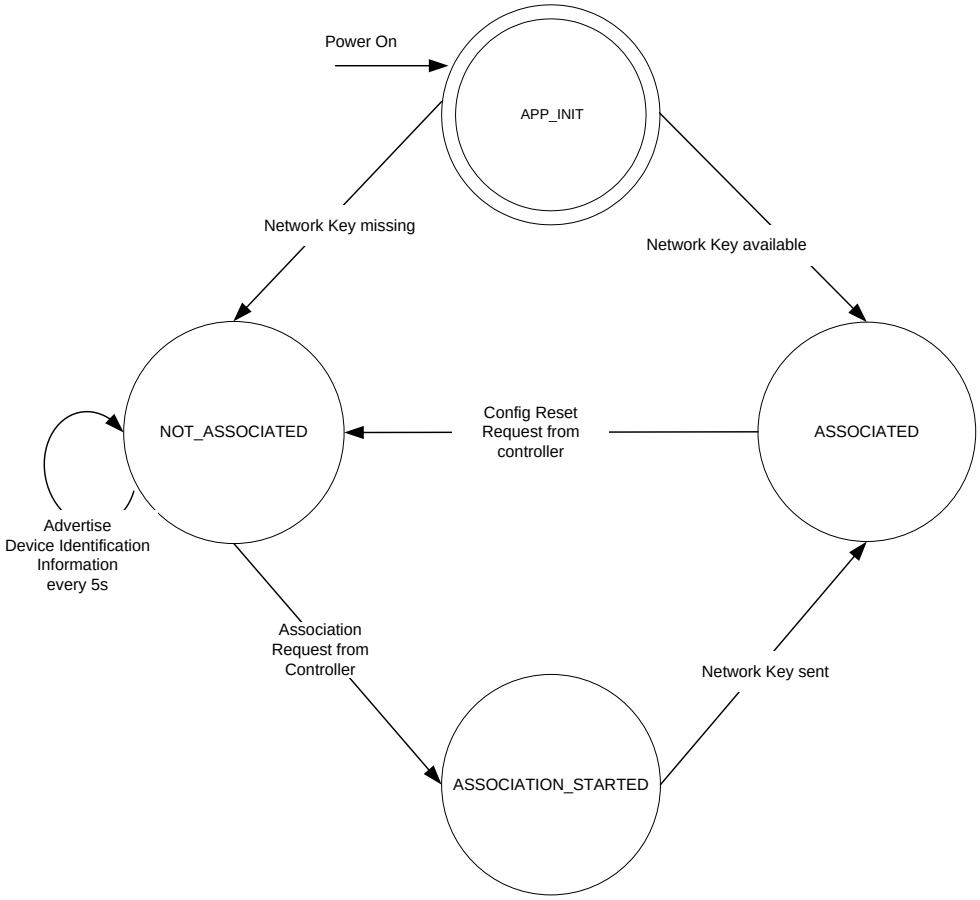

Figure 4.2: CSRmesh association State Machine

4.3.1. app_state_not_associated

When the application is first flashed on the device it will be in this state. In this state the application will be ready to associate with a CSRmesh network and sends the CSRmesh Device UUID advertisements every 5 seconds.

4.3.2. app_state_association_started

The application enters this state when it receives an association request from the control device.

4.3.3. app_state_associated

The application enters this state when association is complete. The application saves the association state on the NVM and it continues to be associated even after power cycle. The application moves to app_state_not_associated when it receives a `CSR_MESH_CONFIG_RESET_DEVICE`.

4.4. CSRmesh Models Supported

The switch application supports the following CSRmesh models

CSRmesh Model	Application action
Light Model	Sets the light level as per button press.
Power Model	Sends power set commands to the assigned target device when the switch SW4 is toggled. Handles the POWER_STATE message received from the target device.
Bearer Model	Handles the set state and get state messages. It enables or disables relay of mesh messages as given in the set state message. Responds with the bearer type, and message relay status parameters to the get state message
Battery Model	Responds to the battery status messages.
Group Model	This application handles the Set Model Group ID message. It uses the assigned Group ID as the target device ID to send the Light Set Level and Power Set State commands.
Switch Model	This application enables the switch model. This allows the control application to identify the device as a switch and assign group IDs.

Table 4.1: CSRmesh models supported

4.5. Synchronising with CSRmesh activity

The CSRmesh Switch application can connect to the CSRmesh control application in a bridge device role. The application has to synchronise with the CSRmesh library to avoid collision of the advertisements and connection events with the CSRmesh activity. The application calls the CSRmesh APIs to synchronise the connection radio events and the connectable advertisements with the CSRmesh library.

- The application sends connectable advertisements upon expiry of the idle timer, as long as it is not connected. Before sending the connectable advertisements the application stops the CSRmesh activity by calling the `CsrMeshStop(FALSE)`.
- It restarts the CSRmesh after `FAST_CONNECTION_ADVERT_TIMEOUT_VALUE` by calling the `CsrMeshStart()`. See section 6.2 for the timer values.
- When connected, it calls the `CsrMeshHandleDataInConnection(ucid, conn_interval)` for the CSRmesh library to synchronise with connection events. This is called in the
  - `LM_EV_CONNECTION_COMPLETE` event handler
  - `LS_CONNECTION_PARAM_UPDATE_IND` event handler and
  - `LM_EV_DISCONNECT_COMPLETE` event handler
- It calls the `CsrMeshHandleRadioEvent()` in the `LS_RADIO_EVENT_IND` event handler.

5. NVM Map

The application stores the parameters listed below in the NVM to prevent loss in the event of a power off or a chip panic.

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
App NVM Version	uint16	1	0
Sanity Word	uint16	1	1
Bonded Flag	boolean	1	2
Bonded Device Address	structure	5	3
Diversifier	uint16	1	8
IRK	uint16 array	8	9
CSRmesh Association State	boolean	1	17
CSRmesh Device UUID	uint16 array	8	18
CSRmesh Device Authorisation Code	uint16 array	4	26
Switch Level	uint16	1	30
Destination Device Address	uint16	1	31
Bearer Model Data	structure	3	32

Table 5.1: NVM Map for Application

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
CSRmesh NVM	uint16 array	14	35

Table 5.2: NVM Map for CSRmesh Library

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
GAP Device Name Length	uint16	1	49
GAP Device Name	uint8 array	20	50

Table 5.3: NVM Map for GAP Service

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
MTL_CP Characteristic client configuration descriptor	uint16	1	70

Table 5.4: NVM Map for CSRmesh Control Service

Note 1:

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.

5.1. Application NVM Version

The CSRmesh Light application reads the persistent parameters at fixed Offsets on the NVM as given in the tables 5.1 to 5.4. In the case of an application update where the NVM offsets got modified, the contents of the NVM will no longer be valid.

Before reading the parameters from the NVM, the application reads NVM version at Offset 0. If the version number stored on the NVM does not match with the defined `APP_NVM_VERSION`, the application erases the NVM and stores the defined `APP_NVM_VERSION` at offset 0.

This can be used to ensure that the contents of the NVM are valid for the NVM offsets defined in the application. In case the NVM offsets of the parameters change upon an application update, this value can be incremented to invalidate the contents of the NVM.

6. Customising the Application

The developer can customize the application by modifying the following parameter values.

This section provides details on how to customize some parameters of the application.

6.1. Advertising Parameters

The CSRmesh applications use 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 and optimum performance of the device in the CSRmesh. See *Bluetooth Core Specification Version 4.1* for the advertising parameters range. It is recommended not to change these parameters.

Parameter Name	Fast Advertisements
Minimum Advertising Interval	90 ms
Maximum Advertising Interval	90 ms

Table 6.1: Advertising Parameters

6.2. Advertisement Timers

The CSRmesh application enters the appropriate state on expiry of the advertisement timers. See section 4.2.2 for more information. The macro for this timer value is defined in the file `csr_mesh_switch_gatt.h`

Timer Name	Timer Value
`FAST_CONNECTION_ADVERT_TIMEOUT_VALUE`	30 ms

Table 6.2: Advertisement Timer

Note:

There will be no CSRmesh activity when this timer is running. It is recommended to keep this value small to avoid losing CSRmesh messages.

6.3. Idle Timer

The CSRmesh application enters the `app_state_advertising` state on expiry of the idle timer. See section **Error! Reference source not found.** for more information on the usage of this timer. The value for this timer is defined in `csr_mesh_switch_gatt.c`

Timer Name	Timer Value
`IDLE_TIMEOUT`	1 s

Table 6.3: Idle timer

6.4. Connection Parameters

The CSRmesh switch application uses the connection parameters listed in Table 6.4 by default. The macros for these values are defined in the file `gap_conn_params.h`. These values have been chosen by considering the overall current consumption of the device and optimum performance of the device in the CSRmesh network. It is

recommended not to modify these parameters. See *Bluetooth Core Specification Version 4.1* for the connection parameter range.

Parameter Name	Parameter Value
Minimum Connection Interval	90 ms
Maximum Connection Interval	120 ms
Slave Latency	0 intervals
Supervision Timeout	6000 ms

Table 6.4: Connection Parameters

6.5. Device Name

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

6.6. Device Address

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

6.7. Non-Volatile Memory

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

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

Note:

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

6.8. Application Features

This section provides details on how to customize some parameters on the CSRmesh Switch application. The application can be configured by uncommenting the required define in the `user_config.h`

Configuration	Description
`ENABLE_GATT_OTA_SERVICE`	Enables the CSR OTA service. With this feature enabled the user can request for firmware update over the air using the CSR OTA Update tool.
`DEBUG_ENABLE`	Enable application debug logging on UART. The debug messages can be viewed on a HyperTerminal or any other terminal application by connecting the device to the PC and opening the corresponding COM Port with 2400-8-N-1 configuration.

Configuration	Description
	Note: If enabled, the brightness control function of the buttons SW2 and SW3 will be disabled.
ENABLE_BATTERY_MODEL	Enables the CSRmesh Battery Model support

Table 6.5: Application configurations

6.9. Configuring the CSRmesh Parameters

The CS User keys can be defined in the .keyr file to override the default CSRmesh parameters. The CSRmesh Library sets the parameters based on the CS User Key values. The following table provides the recommended values for these values for optimal performance of the devices over the CSRmesh network.

CS User Key Index	Parameter	Recommended Value	Description
0	CSRmesh Transmit Interval (ms)	003C	The CSRmesh library uses this interval for transmitting messages. This is the recommended value as this allows the device with optimal time for Receiving and transmitting messages.
1	CSRmesh Transmit Time (ms)	01E0	The CSRmesh library uses this value for transmitting data over the network for this period of time. This is the recommended value for transmitting a message with high reliability.
2	CSRmesh Configuration Bitmask	0000 to 0007	Bit-0 : CSRmesh Random Device UUID Enable 1 - Enables generation of Random Device UUID. 0 - Reads the UUID from NVM. If this key is set to 1 the application generates a random UUID and stores it in the NVM when it is run for the first time. This can be used to avoid having same UUID on multiple devices without explicitly programming a UUID on each device. Bit-1 : CSRmesh Relay Enable 1 - Enables relay of CSRmesh messages. 0 - Disables relay of CSRmesh messages. Bit-2 : CSRmesh Bridge Enable 0 - Disables connectable advertisements. 1 - Enables connectable advertisements. Disabling Bridge leaves the device non connectable on any service. The connectable adverts can be enabled by sending a CSR_MESH_BEARER_SET_STATE message with the LE GATT server bearer bit set Bits 3 to 15 are ignored.

Table 6.6: Configuring CSRmesh Parameters

Appendix A CSRmesh Application GATT Database

A.1 GATT Service Characteristics

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Service Changed	0x0003	Indicate	Application	Security Mode 1 and Security Level 1	Service Changed Handle value
Service Changed Client Characteristic Configuration descriptor	0x0004	Read, Write	Application	Security Mode 1 and Security Level 1	Current client configuration for “Service Changed " characteristic

Table A.1: GATT Service Characteristics

A.2 GAP Service Characteristics

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Device Name	0x0007	Read, Write	Application	Security Mode 1 and Security Level 1	Device name Default value : "CSRmesh"
Appearance	0x0009	Read	Firmware	Security Mode 1 and Security Level 1	Unknown: 0x0000
Peripheral preferred connection parameters	0x000b	Read	Firmware	Security Mode 1 and Security Level 1	Connection interval - Min 90 ms - Max 120 ms Slave latency - 0 Connection timeout - 6 s

Table A.2: GAP Service Characteristics

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

A.3 CSR OTA Update Application Service Characteristics

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Current Application	0x000e	Read, Write	Application	Security Mode 1 and Security Level 2	Current live application 0x0 - OTA Update Bootloader 0x1 - Identifies application 1 0x2 - Identifies application 2
Read CS Block	0x0010	Write	Application	Security Mode 1 and Security Level 2	Format - uint16[2] Index 0 - An offset in 16-bit words into the CS defined in the SDK Documentation.

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
					Index 1 - The size of the CS block expected, in octets.
Data Transfer	0x0012	Read, Notify	Application	Security Mode 1 and Security Level 2	This characteristic is ATT_MTU-3 (20)-bytes long. The format of the 20-bytes is defined by the message context.
Data Transfer Client Characteristic Configuration	0x0013	Read, Write	Application	Security Mode 1 and Security Level 2	Current client configuration for “Data Transfer" characteristic

Table A.3: CSR OTA Update Application Service Characteristics

A.4 Mesh Control Service Characteristics

Characteristic Name	Database Handle	Access Permissions	Managed By	Security Permissions	Value
Network Key	0x0016	Write	Application	Security Mode 1 and Security Level 1	0
Device UUID	0x0018	Read	Application	Security Mode 1 and Security Level 1	22e4-b12c-5042-11e3-9618-ce3f-5508-acd9
Device ID	0x001a	Read, Write	Application	Security Mode 1 and Security Level 1	0x8001
MTL Continuation Control Point	0x001c	Write	Application	Security Mode 1 and Security Level 1	Dynamic
MTL Continuation Control Point Client Characteristic Configuration	0x001d	Read, Write	Application	Security Mode 1 and Security Level 1	Current client configuration for “MTL Continuation Control Point " characteristic
MTL Complete Control Point	0x001f	Write, Notify	Application	Security Mode 1 and Security Level 1	Dynamic
MTL Complete Control Point Client Characteristic Configuration	0x0020	Read, Write	Application	Security Mode 1 and Security Level 1	Current client configuration for “MTL Complete Control Point " characteristic
MTL TTL	0x0022	Read, Write	Application	Security Mode 1 and Security Level 1	50
MESH Appearance	0x0024	Read, Write	Application	Security Mode 1 and Security Level 1	0

Table A.4: Mesh Control Service Characteristics

Document References

Document	Reference
Bluetooth Core Specification Version 4.1	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
CSRmesh 1.0 Control Application Note	CS-318312-AN
Service Characteristics And Descriptions	http://developer.bluetooth.org/gatt/characteristics/Pages/default.aspx
GATT Database Generator	CS-219225-UG
CSR µEnergy xIDE User Guide	CS-212742-UG
Installing the CSR Driver for the Profile Demonstrator Application	CS-235358-UG
CSR µEnergy Modifying an Application to Support OTA Update Application Note	CS-304564-AN
CSR µEnergy Over-the-Air (OTA) Update System Application Note	CS-316019-AN
Over-the-Air Update Application and Bootloader Services Specification	CS-316220–SP
CSRmesh API Guide	https://www.csrsupport.com/CSRmesh
CSRmesh Release Notes	https://www.csrsupport.com/CSRmesh
CSRmesh Android Application	https://www.csrsupport.com/CSRmesh

Terms and Definitions

API	Application Programmer's Interface
BLE	Bluetooth Low Energy (now known as Bluetooth Smart)
BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CS	Configuration Store
CSR	Cambridge Silicon Radio
CSRmesh™	A CSR protocol that enables peer-to-peer-like networking of Bluetooth Smart devices
EEPROM	Electrically Erasable Programmable Read Only Memory
GAP	Generic Access Profile
GATT	Generic Attribute Profile
I^2C	Inter-Integrated Circuit
IOT	Internet of Things
IRK	Identity Resolving Key
LED	Light Emitting Diode
LM	Link Manager
MTL	Message Transport Layer
NVM	Non Volatile Memory
OTA	Over-the-Air
PIO	Programmable Input Output
PWM	Pulse Width Modulation
Tx	Transmit
UUID	Universally Unique Identifier