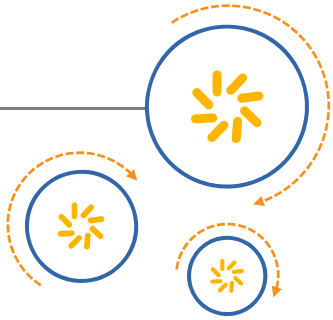
Qualcomm Technologies International, Ltd.



Confidential and Proprietary – Qualcomm Technologies International, Ltd.

(formerly known as Cambridge Silicon Radio Ltd.)

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to: DocCtrlAgent@qualcomm.com.

Restricted Distribution: Not to be distributed to anyone who is not an employee of either Qualcomm Technologies International, Ltd.or its affiliated companies without the express approval of Qualcomm Configuration Management.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies International, Ltd.

Any software provided with this notice is governed by the Qualcomm Technologies International, Ltd. Terms of Supply or the applicable license agreement at https://www.csrsupport.com/CSRTermsandConditions.

Qualcomm is a trademark of Qualcomm Incorporated, registered in the United States and other countries. All Qualcomm Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer (“export”) laws. Diversion contrary to U.S. and international law is strictly prohibited.

Qualcomm Technologies International, Ltd.
Churchill House
Cambridge Business Park
Cambridge, CB4 0WZ
United Kingdom

Push every boundary.™

CSR µEnergy™

CSRmesh 2.0 Bridge Application

Application Note

Issue 3

Document History

Revision	Date	History
1	21 SEP 15	Original publication of this document
2	27 OCT 15	Editorial updates.
3	30 OCT 15	Editorial updates.

Contacts

General information	www.csr.com
Information on this product	sales@csr.com
Customer support for this product	www.csrsupport.com
More detail on compliance and standards	product.compliance@csr.com
Help with this document	comments@csr.com

Trademarks, Patents and Licences

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

Document History 2
Contacts 2
Trademarks, Patents and Licences 2
Safety-critical Applications 2
Performance and Conformance 2
Contents 3
Tables and Figures 3
1. Introduction 5
1.1. Application Overview 6
2. Using the Application 8
2.1. Demonstration Kit 8
3. Application Structure 11
4. Code Overview 14
4.1. Application Entry Points 14
4.2. Internal State Machine for GATT connection 16
4.3. Synchronising with CSRmesh Activity 17
5. NVM Map 18
6. Customising the Application 19
6.1. Advertisement Timers 19
6.2. Connection Parameters 19
6.3. Device Name 19
6.4. Device Address 19
6.5. Non-volatile Memory 20
6.6. Application Features 20
Appendix A CSRmesh Application GATT Database 21
Document References 24
Terms and Definitions 25

Tables and Figures

Table 1.1: CSRmesh Control Profile Roles 6
Table 1.2: Application Topology 6
Table 1.3: Role and Responsibilities 6
Table 2.1: CSRmesh Components 8
Table 3.1: Source Files 11
Table 3.2: Header Files 12
Table 3.3: Database Files 13
Table 5.1: NVM Map for Application 18
Table 5.2: NVM Map for GAP Service 18
Table 6.1: Advertisement Timers 19
Table 6.2: Connection Parameters 19
Table 6.3: Application Configurations 20
Table A.1: GATT Service Characteristics 21
Table A.2: GAP Service Characteristics 21
Table A.3: CSR OTA Update Application Service Characteristics 22
Table A.4: Mesh Control Service Characteristics 23

Figure 1.1: CSRmesh Use Case 5

Figure 1.2: Primary Services....7
Figure 2.1: CSRmesh Development Board....8
Figure 2.2: CSRmesh Device Tag Sticker....9
Figure 4.1: Internal GATT State Machine....16

1. Introduction

This document describes the CSRmesh™ Bridge on-chip application built using the CSR µEnergy™ SDK.

The application has the following use case:

- CSRmesh GATT Bridge: The Bridge application implements the CSR custom Mesh Control Service. This service allows a Bluetooth® Smart enabled phone to connect using the Mesh Control GATT Service to send and receive CSRmesh messages to other devices. The bridge application does not support CSRmesh association.

The CSRmesh Bridge application is part of the CSRmesh release and shows the CSR custom-defined Mesh Control Service in GATT server role. It uses the CSRmesh library provided as part of the CSRmesh release. For more information, see the *CSRmesh API Guide* documentation.

The CSRmesh Bridge application does not support bonding and is not compatible with previous CSRmesh releases.

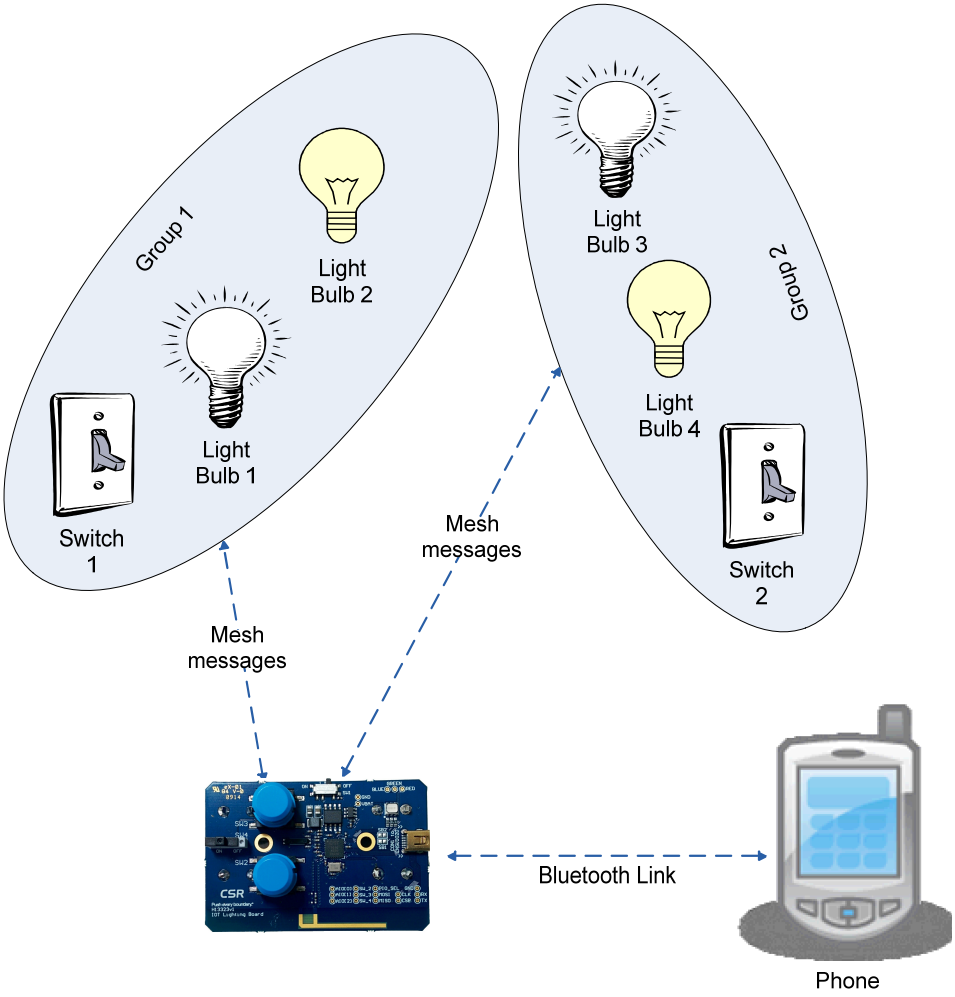


Figure 1.1: CSRmesh Use Case

1.1. Application Overview

The CSRmesh Bridge application uses the CSRmesh library API to communicate with other associated devices in the same CSRmesh network. It also supports a custom GATT profile to allow control of the CSRmesh from a Bluetooth Smart enabled device. This application shows a bridge use case and does not support association with the CSRmesh.

1.1.1. Supported Profiles

The CSRmesh Bridge application implements the following CSR custom profiles to support the use cases.

1.1.1.1. CSRmesh Control Profile

The CSRmesh Control Profile defines the behaviour when:

- A network of devices such as lights and switches need to be created.
- Controlling the device after a network is created, for example, switching the light on/off or changing the intensity or colour of a light.
- Reading the status of a device in the network, for example, on/off state, colour or intensity of a light.

Table 1.1 lists the two roles that the CSRmesh Control Profile defines.

Role	Description	Implementation
CSRmesh Bridge Device	Receives commands from host and sends them over the CSRmesh network to associated devices. Receives responses from associated devices over the CSRmesh and forwards them to the host over a Bluetooth Smart connection.	On the CSRmesh Bridge application
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 Smart connection to the CSRmesh devices.	On a Bluetooth Smart enabled phone or tablet

Table 1.1: CSRmesh Control Profile Roles

1.1.2. Application Topology

Table 1.2 and Table 1.3 lists the topology that the CSRmesh Bridge uses.

Role	Mesh Control Service	GAP Service	GATT Service	CSR OTA Update Application 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	Accepts incoming commands and requests from a client and sends responses, indications and notifications to the client.
GAP Peripheral	Accepts connections from the 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 v2.0
- GATT Service
- GAP Service
- CSR OTA Update Application Service v7

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

Figure 1.2 shows the services supported in the GATT Server Role.

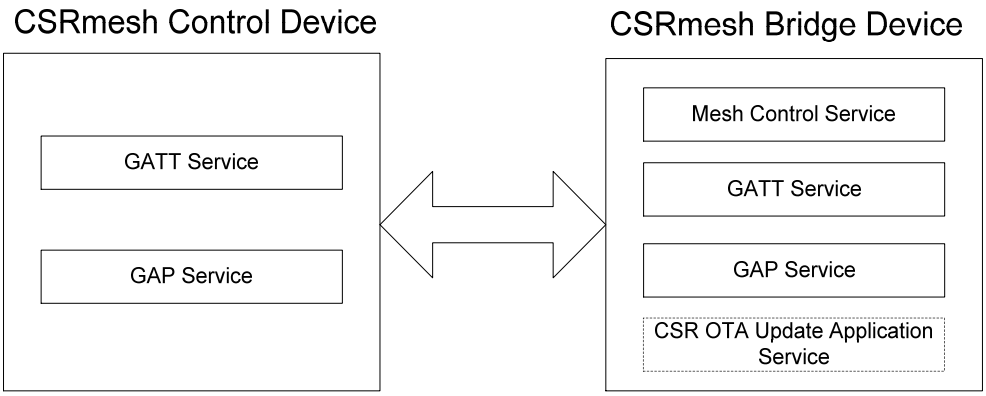


Figure 1.2: Primary Services

1.1.3.1. CSR OTA Update Application Service

The CSR OTA Update Application Service enables wireless update of the application software. A PC or mobile phone application provided by the device manufacturer enables the end-user to keep their device up-to-date with the latest features and bug fixes.

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

- Add OTA Update functionality to the on-chip application
- Add support for the CSR OTA Update Application Service and GATT Services to an application
- Configure the on-chip bootloader

The CSR OTA Update bootloader image must be present on the device and configured to contain the correct device name and optional shared authentication key.

When the device is enabled for OTA Update, the CSR µEnergy Over-the-Air Updater host application included in the SDK can update the device.

For more information about CSR OTA Update, see:

- *CSR µEnergy Over-the-Air (OTA) Update System Application Note*
- *CSR µEnergy Modifying an Application to Support OTA Update Application Note*
- *CSR µEnergy Over-the-Air (OTA) Update Application and Bootloader Services Specification*

For information about CSR OTA Update applications for iOS and Android, see www.csrsupport.com

2. Using the Application

This section describes how to use the CSRmesh Bridge application with CSRmesh Android Application to control devices.

2.1. Demonstration Kit

Table 2.1 lists the components that the application can use for demonstration.

Component	Hardware	Application
Bridge Device	CSRmesh Development PCB (DB-CSR1010-10185-1A)	CSRmesh Bridge Application v2.0
CSRmesh Android Control Device	Android Bluetooth LE Device	CSRmesh Android Application v2.0
CSRmesh iOS Control Device	iOS Bluetooth LE Device	CSRmesh iOS Application v2.0

Table 2.1: CSRmesh Components

2.1.1. CSRmesh Development Board

The CSR µEnergy SDK is used to download the CSRmesh Bridge application on the development boards. See *CSR µEnergy xIDE User Guide* for further information.

Ensure that the development board is switched on using the Power On/Off switch. Figure 2.1 shows the switch in the Off position.

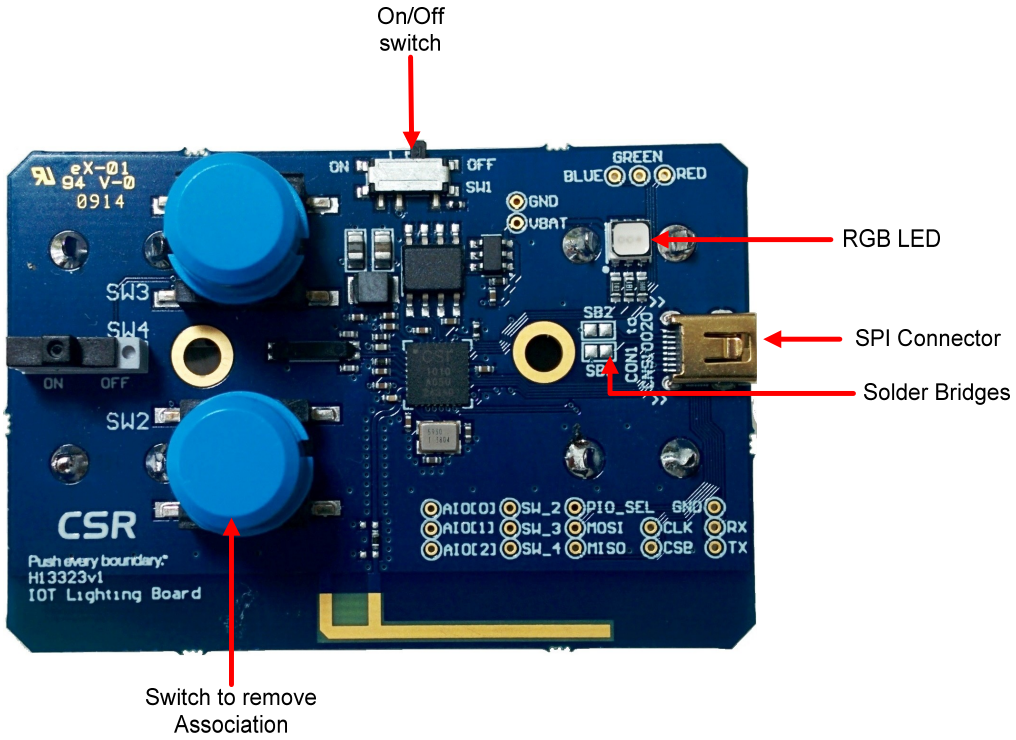


Figure 2.1: CSRmesh Development Board

Note:

When the development board is disconnected from the USB to SPI Adapter, wait at least 1 minute before powering the board on to allow dissipation of any residual charge received from the SPI connector.

Shorting the solder bridges exposes the UART through the USB to SPI Adapter. The PIOs connected to SW2 and SW3 are also mapped to UART Rx and Tx lines. Pressing any of these buttons shorts the UART lines to

ground and corrupts data on the UART. CSR recommends not pressing these buttons during UART communication.

2.1.1.1. CSRmesh Device Tag

The CSRmesh Development board is supplied with a CSRmesh Device Tag sticker shown in Figure 2.2. The sticker contains:

- BT: Device Bluetooth Address
- SN: 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 device can be programmed with the Bluetooth address and the XTAL Trim printed on the sticker by setting these values in the `bridge_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 Table 5.1 using the USB to SPI adapter.

To program the example UUID `0x0123456789ABCDEFFEDCBA9876543210` and Authorisation Code `0x0123456789ABCDEF` on the NVM:

1. Open a command prompt.
2. Program the device UUID and Authorisation Code to the device EEPROM over the SPI link:
   - If base `NVM_START_ADDRESS` is defined in the `.keyr` file, program the device UUID and Authorisation Code to the device as follows:

     ```
     <CSR_uEnergy_Tools path>\uEnergyProdTest.exe -k
     CSRmeshbridge_csr101x_A05.keyr -m1 0x02 0x3210 0x7654 0xBA98 0xFEDC
     0xCDEF 0x89AB 0x4567 0x0123 0xCDEF 0x89AB 0x4567 0x0123
     ```

     The first value following `-m1` is the NVM offset from the `NVM_START_ADDRESS`. The command takes the NVM offset as the byte address. Word offset 1 for device UUID is byte offset 2, see Table 5.1.
   - If the `.keyr` file is not included in the command, program the device UUID and Authorisation Code to the device as follows:

     ```
     <CSR_uEnergy_Tools path>\uEnergyProdTest.exe -m1 0x4102 0x3210 0x7654
     0xBA98 0xFEDC  0xCDEF 0x89AB 0x4567 0x0123 0xCDEF 0x89AB 0x4567
     0x0123
     ```

     The first value following `-m1` is the NVM address obtained by adding base address `4100` and word offset 1, see Table 5.1 for the device UUID. The command takes the NVM address as the byte address. Word offset 1 is byte offset 2, so effective address is `0x4102`.

The CSRmesh Control application reads the device Authorisation Code and the UUID from the QR-Code printed on the sticker during association. See *CSRmesh 2.0 Android Control Application Note* or the *CSRmesh 2.0 iOS Control Application Note* for details. For further information about programming the NVM, see *CSRmesh 2.0 Production Test Tool User Guide.*

2.1.2. CSRmesh Control Application

The CSRmesh Control application runs on an Android or iOS device that supports BLE. It communicates with the CSRmesh devices by connecting to one of the devices that support 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 about using the CSRmesh Control application on an Android device, see *CSRmesh 2.0 Android Control Application Note.*

For details about using the CSRmesh Control Application on an iOS device, see *CSRmesh 2.0 iOS Control Application Note*.

For details about the supported iOS and Android version, see *CSRmesh 2.0 Mobile Applications Release Note*.

3. Application Structure

This section describes the source files, head files and database files.

3.1.1. Source Files

Table 3.1 lists the source files.

File Name	Description
app_fw_event_handler.c	Defnes the handler functions for firmware events.
csr_mesh_bridge.c	Implements all the entry functions such as AppInit(), AppProcessSystemEvent(),CSRmeshAppProcessMeshEvent() and AppProcessLmEvent(). Events received from the hardware and firmware are first handled here. This file contains handling functions for all the LM and system events
csr_mesh_bridge_gatt.c	Implements routines for triggering advertisement procedures.
csr_mesh_bridge_util.c	Implements some utilities functions called by other handlers.
csr_ota_service.c	Implements the routines for handling the read/write access on the CSR custom-defined OTA Update Service.
gap_service.c	Implements routines for GAP Service. For example, handling read/write access indications on the GAP Service characteristics, reading/writing device name on NVM.
gatt_service.c	Defines routines for using GATT Service.
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

3.1.2. Header Files

Table 3.2 lists the header files.

File Name	Description
app_debug.h	Contains macro definitions for enabling debug prints.
app_fw_event_handler.h	Contains prototypes of the externally referenced functions defined in app_fw_event_handler.c.
app_gatt.h	Contains macro definitions, user-defined data type definitions and function prototypes used across the application.
appearance.h	Contains the appearance value macro of the application.
csr_mesh_bridge.h	Contains data structures and prototypes of externally referenced functions defined in the csr_mesh_bridge.c file.
csr_mesh_bridge_gatt.h	Contains timeout values and prototypes of externally referenced functions defined in the csr_mesh_bridge_gatt.c file.
csr_mesh_bridge_util.h	Contains prototypes of the externally referenced functions defined in the csr_mesh_bridge_util.c file.
csr_ota_service.h	Contains prototypes of the externally referenced functions defined in the csr_ota_service.c file.
csr_ota_uuids.h	Contains macros for UUID values for CSR OTA Update Service.
gap_conn_params.h	Contains macro definitions for fast/slow advertising, preferred connection parameters, idle connection timeout values and so on.
gap_service.h	Contains prototypes of the externally referenced functions defined in the gap_service.c file.
gap_uuids.h	Contains macros for UUID values for GAP Service.
gatt_service.h	Contains prototypes of the externally referenced functions defined in the gatt_service.c file.
gatt_service_uuids.h	Contains macros for UUID values for GATT Service.
mesh_control_service.h	Contains prototypes of the externally referenced functions defined in the mesh_control_service.c file.
mesh_control_service_uuids.h	Contains macros for UUID values for Mesh Control Service
ota_customisation.h	Contains defines for the application data variables used by the csr_ota_service.c file.
user_config.h	Contains macros for customising the application.

Table 3.2: Header Files

3.1.3. Database Files

Table 3.3 lists the database files.

File Name	Description
app_gatt_db.db	Master database files including all service specific database files. Is imported by the GATT Database Generator.
csr_ota_db.db	Contains information related to CSR OTA Update Service characteristics, their descriptors and values.
gap_service_db.db	Contains information related to GAP Service characteristics, their descriptors and values. See Table A.2 for GAP characteristics.
gatt_service_db.db	Contains information related to GATT Service characteristics, their descriptors and values.
mesh_control_service_db.db	Contains information related to CSRmesh Control Service characteristics, their descriptors and values. See Table A.4 for CSRmesh Control Service characteristics.

Table 3.3: Database Files

4. Code Overview

This section describes 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. It 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 I²C EEPROM.
- Initialises all the services.
- Reads the persistent store. Sets a random device UUID based on the Bit 3 of the `CS_USER_KEY 1` setting when the application runs for the first time.
- Initialises the CSRmesh stack:
  - Configures CSRmesh bearer parameters and initialises the CSRmesh Scheduler:

    ```
    CSRSchedSetConfigParams(params)
    CSRSchedStart()
    ```
  - Registers an application call-back function for handling core CSRmesh events: `CSRmeshRegisterAppCallback(CSRmeshAppProcessMeshEvent)`
  - Initialises the Core Stack: `CSRmeshInit()`
  - Initialises the supported model server and client modules.
- Registers the attribute database with the firmware.

4.1.2. AppProcessLmEvent()

This function is invoked whenever an LM-specific event is received by the system. The following events are handled in this function:

4.1.2.1. Database Access

- `GATT_ADD_DB_CFM`: This confirmation event marks the completion of database registration with the firmware. On receiving this event, the 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 can be a CSRmesh advertisement. The application passes the event data to the CSRmesh library to process the packet by calling the `CSRSchedHandleIncomingData()` API.
- `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 can be locally triggered or triggered by the remote connected device due to link loss.
- `LM_EV_ENCRYPTION_CHANGE`: This event indicates a change in the link encryption.
- `LM_EV_CONNECTION_UPDATE`: This event indicates that the connection parameters are updated to a new set of values and is generated when the connection parameter update procedure is initiated either by the master or by the slave. These new values are stored by the application for comparison against the preferred connection parameter, see Section 6.2.

4.1.2.4. SMP Events

- `SM_SIMPLE_PAIRING_COMPLETE_IND`: This indication event indicates that pairing completes successfully or otherwise. See Volume 3, Part H, Section 2.4 and Section 3.6 of *Bluetooth Core Specification Version 4.1*.

4.1.2.5. Connection Events

- `GATT_CONNECT_CFM`: This confirmation event indicates that the connection procedure completes. If it does not successfully complete, the application goes to the idle state and waits for a user action. See Section 4.2 for more information about application states.
- `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.

4.1.3. AppProcessSystemEvent()

This function handles the system events such as a low battery notification or a PIO change. 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. CSRmeshAppProcessMeshEvent ()

This function handles the CSRmesh core stack events received from the CSRmesh network or caused by internal state changes. The Bridge application implements only the GATT bridge function and does not enable device association with the network.

4.2. Internal State Machine for GATT connection

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

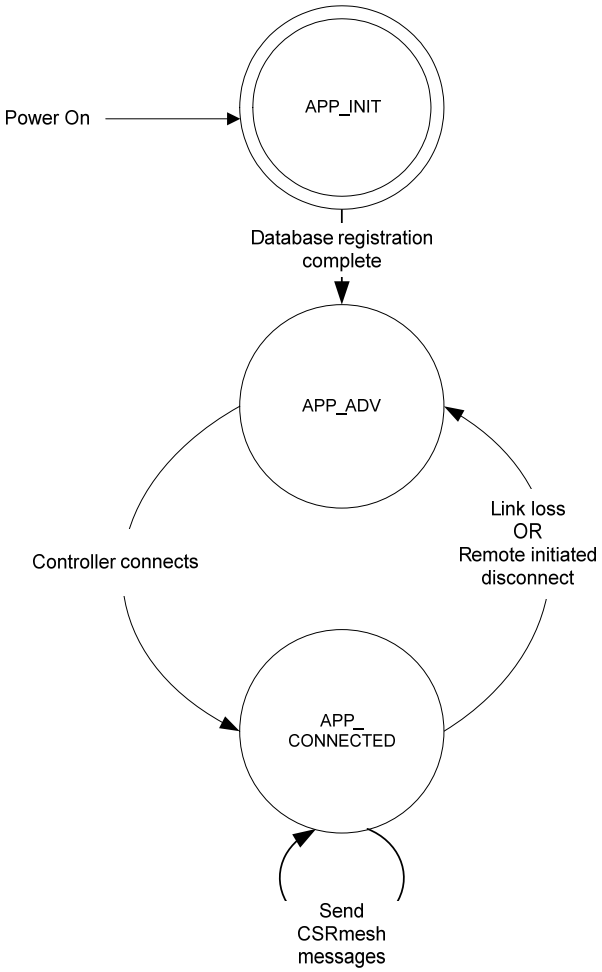


Figure 4.1: Internal GATT State Machine

4.2.1. app_state_init

When the application is powered on or the chip resets, it enters the `app_state_init` 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 the `app_state_advertising` state and transmits connectable advertising events at an interval defined by `ADVERT_INTERVAL`. When a central device connects to it, the advertisements are stopped and the application enters the `app_state_connected` state. See Section 6.1 for more information about advertisement timers.

4.2.3. app_state_connected

In the `app_state_connected` state, the CSRmesh application is connected to a CSRmesh control device using connection intervals specified in Table 6.2. It can receive commands from the 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.

4.2.4. app_state_disconnecting

The CSRmesh application never triggers a disconnection on its own. The application no longer stops the CSRmesh activity on disconnection.

- If the disconnection is triggered, the application enters the `app_state_advertising` state.

4.3. Synchronising with CSRmesh Activity

The CSRmesh Bridge application connects to the CSRmesh control application in a bridge device role. The application must 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.

4.3.1. Application Connectable Advertising

The application sends connectable advertisements at regular intervals when the device is powered and not connected. The interval at which the advertising events are sent is defined by `ADVERT_INTERVAL`.

The application calls `CSRSchedSendUserAdv()` rather than the firmware API to send connectable adverts. This function schedules one application advertising event when called.

4.3.2. Connection Events

The application must notify the CSRmesh stack of the GATT connection events to schedule the CSRmesh activity.

- The application calls `CSRSchedNotifyGattEvent(ucid, conn_interval)` for the CSRmesh library to synchronise with connection events. It is called in the following event handlers:
  - `LM_EV_CONNECTION_COMPLETE`
  - `LS_CONNECTION_PARAM_UPDATE_IND`
  - `LM_EV_DISCONNECT_COMPLETE`

5. NVM Map

Table 5.1 lists the application stores the parameters 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)
CSRmesh Stack NVM	uint16 array	32	0
CSRmesh Device UUID (This is part of 31 words of Stack NVM)	uint16 array	8	1
CSRmesh Device Authorisation Code (This is part of 31 words of Stack NVM)	uint16 array	4	9
Sanity Word	uint16	1	32

Table 5.1: NVM Map for Application

Entity Name	Type	Size of Entity (Words)	NVM Offset (Words)
GAP Device Name Length	uint16	1	33
GAP Device Name	uint8 array	20	34

Table 5.2: NVM Map for GAP Service

Note:

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

6. Customising the Application

This section describes how to customise some parameters of the CSRmesh application.

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

6.1. Advertisement Timers

The CSRmesh Bridge application sends connectable advertisements when powered on and not connected. It uses a timer to send a connectable advertising event at regular intervals. The advertising interval is defined in the file `csr_mesh_bridge_gatt.h`.

Timer Name	Timer Value
`ADVERT_INTERVAL`	1250 ms ± (0~10) ms

Table 6.1: Advertisement Timers

6.2. Connection Parameters

The CSRmesh application uses the connection parameters listed in Table 6.2 by default. The macros for these values are defined in the file `gap_conn_params.h`. These values are chosen by considering the overall current consumption of the device and optimum performance of the device in the CSRmesh network. CSR recommends not modifying this parameter. If the connection interval is set to less than 13 ms, the device cannot scan for CSRmesh messages from the associated network but only messages received from the GATT connection. 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 6.2: Connection Parameters

6.3. Device Name

The device name for the application can be changed. The default name is `CSRmesh` in the file `gap_service.c`. The maximum length of the device name is 20 octets.

6.4. Device Address

The application uses a public address by default. The `USE_STATIC_RANDOM_ADDRESS` macro in the `app_gatt.h` file enables the support for static random addresses. If enabled, the application sets a new random address during the application initialisation upon a power on reset.

6.5. 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 I²C EEPROM
- `NVM_TYPE_FLASH` for SPI Flash

Note:

The macros are enabled by selecting the NVM type using the Project Properties in xIDE. This macro is defined during compilation to let the application know for which NVM type it is built. If EEPROM is selected, `NVM_TYPE_EEPROM` is defined; if SPI Flash is selected, the macro `NVM_TYPE_FLASH` is defined. Follow the comments in the `.keyr` file.

6.6. Application Features

This section describes how to customise some parameters of the CSRmesh Bridge application. The application can be configured by uncommenting the required definition in `user_config.h`.

Configuration	Description
`USE_STATIC_RANDOM_ADDRESS`	If this feature is enabled, the application uses static random address for sending.

Table 6.3: Application Configurations

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 Minimum: 90 ms Maximum: 120 ms Slave latency: 0 Connection timeout: 6 s

Table A.2: GAP Service Characteristics

For more information about GAP service and security permissions, see the *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. 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
Version	0x0015	Read	Firmware	Security Mode 1 and Security Level 2	Service version Format: uint8

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	0x0018	Write	Application	Security Mode 1 and Security Level 1	0
Device UUID	0x001a	Read	Application	Security Mode 1 and Security Level 1	22e4-b12c-5042-11e3-9618-ce3f-5508-acd9
Device ID	0x001c	Read Write	Application	Security Mode 1 and Security Level 1	0x8001
MTL Continuation Control Point	0x001e	Write	Application	Security Mode 1 and Security Level 1	Dynamic
MTL Continuation Control Point Client Characteristic Configuration	0x001f	Read Write	Application	Security Mode 1 and Security Level 1	Current client configuration for MTL Continuation Control Point characteristic
MTL Complete Control Point	0x0021	Write Notify	Application	Security Mode 1 and Security Level 1	Dynamic
MTL Complete Control Point Client Characteristic Configuration	0x0022	Read Write	Application	Security Mode 1 and Security Level 1	Current client configuration for MTL Complete Control Point characteristic
MTL TTL	0x0024	Read Write	Application	Security Mode 1 and Security Level 1	50
MESH Appearance	0x0026	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	www.bluetooth.org/
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
CSR µEnergy xIDE User Guide	CS-212742-UG
CSRmesh 2.0 Android Controller Application Note	CS-337680-AN
CSRmesh 2.0 Gateway SB User Guide	CS-332701-UG
CSRmesh 2.0 iOS Controller Application Note	CS-337682-AN
CSRmesh 2.0 Mobile Application User Guide	CS-337051-UG
CSRmesh 2.0 Production Test Tool User Guide	CS-335123-UG
CSRmesh 2.0 Node API Guide	www.csrsupport.com
CSRmesh 2.0 Node Release Note	CS-339050-RN
CSRmesh Application 1.x to 2.0 Porting Guide	CS-335300-DC
GATT Database Generator	CS-219225-UG
Installing the CSR Driver for the Profile Demonstrator Application	CS-235358-UG
Over-the-Air Update Application and Bootloader Services Specification	CS-316220-SP
Service Characteristics And Descriptions	developer.bluetooth.org

Terms and Definitions

API	Application Programmer's Interface
BLE	Bluetooth Low Energy (now known as Bluetooth Smart)
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
PC	Personal Computer
PCB	Printed Circuit Board
PIO	Programmable Input Output
PWM	Pulse Width Modulation
Rx	Receive
SDK	Software Development Kit
SPI	Serial Peripheral Interface
Tx	Transmit
USB	Universal Serial Bus
UUID	Universally Unique Identifier