Quuppa tag emulation

About this document

Scope and purpose

This application note explains how to use the quuppa_tag sample application available for the ModusToolbox™ Bluetooth® SDK to develop your own Quuppa Tag Emulation product to be used within a Quuppa Intelligent Locating System deployment for real-time location services and applications.

Intended audience

This document is intended for customers/software application developers interested in developing a Quuppa Tag Emulation product utilizing Infineon's Bluetooth® Low Energy-capable devices.

Table of contents

About this document **1**
Table of contents **2**
1 Required hardware **3**
2 Bluetooth® SDK **4**
2.1 Importing the Quuppa Pro CE package 4
2.2 Building and programming 4
2.3 Tag operation 4
References **7**
Revision history **8**
Disclaimer **9**

1 Required hardware

CYW920835M2EVB-01: CYW20835B1 evaluation board

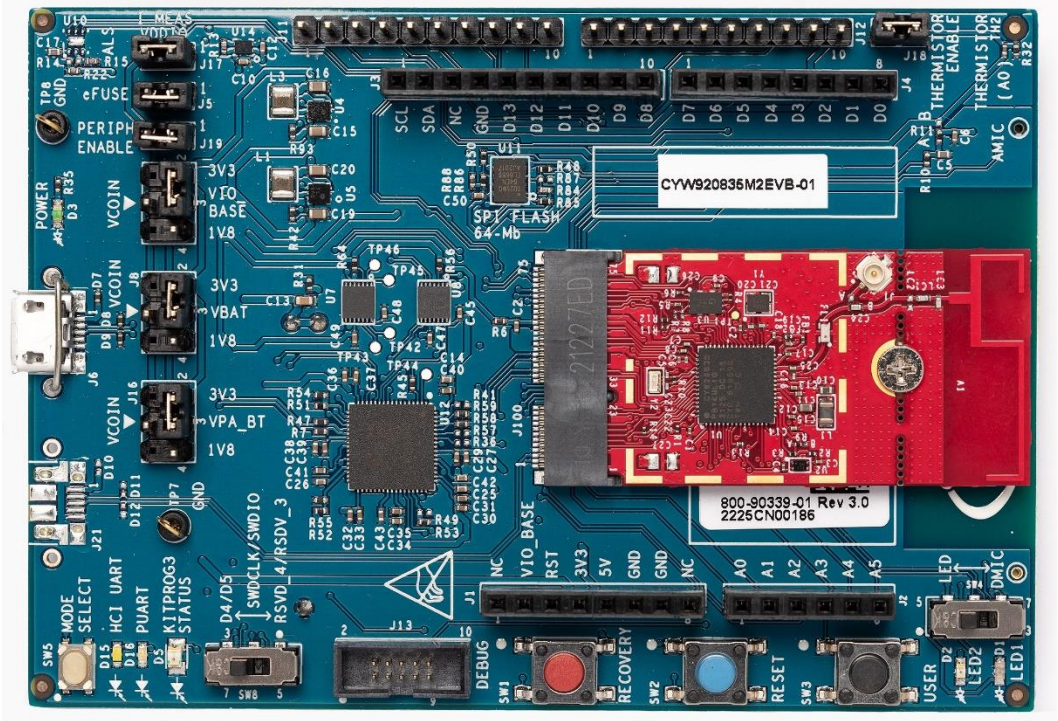


Figure 1 CYW920835M2EVB-01: CYW20835B1 evaluation board

For more details, see the CYW920835M2EVB-01: CYW20835B1 evaluation board webpage.

2 AIROC™ Bluetooth® SDK

The Quuppa Profile library and sample tag application are included in an optional AIROC™ Bluetooth® SDK Pro code example (CE) package. Contact your local sales representative to request this package.

2.1 Importing the Quuppa Pro CE package

1. Download the *mtb-example-btsdk-pro-quuppa-tag-emulation.zip* file and extract it to your workspace such as *C:\myapps\pro-quuppa-tag-emulation.*
2. From Eclipse IDE for ModusToolbox™, select **File** > **New ModusToolbox™ Application**.
3. In Project Creator, click **AIROC™ Bluetooth® BSPs**.
4. Select the CYW920835M2EVB-01 board and click **Next**.
5. Click **Browse**, and select the root folder of the extracted zip file.
   For example, *c:\myapps\pro-quuppa-tag-emulation.*
6. Click **Next** and then click **Finish**.

2.2 Building and programming

Before building and programming your project, you must configure the following items for the quuppa_tag sample app:

- Developer ID
  - By default, the sample app uses Infineon's Quuppa Developer ID value of 0x0094. If you do not have one already, contact your Quuppa representative to establish an ID for your company. Update the `QUUPPA_PROFILE_DEV_ID` variable in the application Makefile to use the new ID.
- Firmware version
  - By default, the sample app uses a `fw_version` value of 0x0001. Update the `QUUPPA_PROFILE_FW_VERSION` variable in the application Makefile.
- Hardware version
  - By default, the sample app uses a `hw_version` value of 0x0001. Update the `QUUPPA_PROFILE_HW_VERSION` variable in the application Makefile.
- Supported features
  - By default, the sample app is configured to use the following supported features bitmap. Update this in the main source file per your project requirements.
    - `QUUPPA_PROFILE_TEMPERATURE_METER_SUPPORTED | QUUPPA_PROFILE_BATT_VOLTAGE_METER_SUPPORTED | QUUPPA_PROFILE_BUTTON_1_SUPPORTED | QUUPPA_PROFILE_BATTERY_ALARM_SUPPORTED`

2.3 Tag operation

2.3.1 State machine

State machine depending on pre-configured profiles outlined in the Quuppa Tag Emulation Pro Implementation Guidelines and Notes (v.1.0) [1].

- Upon power up, the tag enters the default state.
- The tag enters the 'Triggered' state if any of the following conditions is met:
  1. The user button is pressed.
  2. Received the Set Tag State QPE API.

3. Motion detected (if supported by the platform)

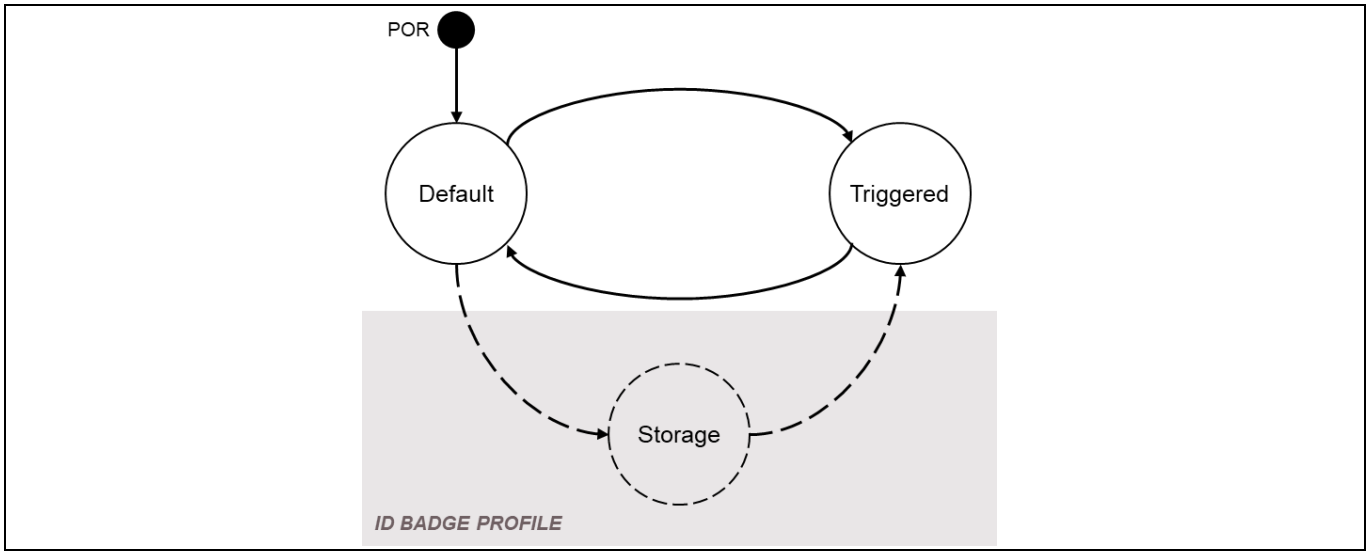


Figure 2 Tag state machine

2.3.2 Quuppa REQ/RSP

The configuration of Quuppa Tag Emulation tags is implemented using custom Quuppa Back Channel commands sent via the `sendQuuppaRequest` and `getQuuppaRequestResponse` QPE APIs. Quuppa provides the flexibility for you to define your own commands to change the configuration of your tag designs. The quuppa_tag sample app includes support for the Back Channel operation and includes implementation of four custom commands to:

1. Configure the tag including setting the Quuppa profile ID and Tx channel
2. Setting the tag's Quuppa profile state
3. Blink the tag's LED 'n' times
4. Get the current tag's configuration and state

2.3.2.1 Set Tag Configuration

Set Rag Configuration for developer-specific payload	
Octet #	**Field**
0	Payload Type = 0xFF (developer-specific)
1	Developer ID (LSB)
2	Developer ID (MSB)
3	Opcode = QUUPPA_SET_TAG_CONFIG
4	Quuppa Profile ID (quuppa_profile_t)
5	Tx Channel (0: 2402MHz(Bluetooth® LE Ch37), 1: 2403 MHz, 2: 2481 MHz)

Example Set Tag Configuration REQ to set the Quuppa Profile ID to ID BADGE and Tx Channel to 2402 MHz (Bluetooth® LE Ch 37):

http://localhost:8080/qpe/sendQuuppaRequest?tag=20835b102022&requestData=0xff34124001000000000000000000000000&humanReadable

2.3.2.2 Set Tag State

Set Tag State developer-specific payload	
Octet #	**Field**
0	Payload Type = 0xFF (developer-specific)
1	Developer ID (LSB)
2	Developer ID (MSB)
3	Opcode = QUUPPA_SET_TAG_STATE
4	State (0: Default, 1: Triggered, 2: Storage)

Example Set Tag State REQ to set the tag's Quuppa profile state to "Triggered":

http://localhost:8080/qpe/sendQuuppaRequest?tag=20835b102022&requestData=0xff34124101000000000000000000000000&humanReadable

2.3.2.3 Blink Tag LED

Blink Tag LED developer-specific payload	
Octet #	**Field**
0	Payload Type = 0xFF (developer-specific)
1	Developer ID (LSB)
2	Developer ID (MSB)
3	Opcode = QUUPPA_BLINK_TAG_LED
4	Number of blinks

Example Blink Tag LED REQ to blink the tag's LED 10 times:

http://localhost:8080/qpe/sendQuuppaRequest?tag=20835b102022&requestData=0xff3412420A000000000000000000000000&humanReadable

2.3.2.4 Get Tag Configuration

Get Tag Configuration developer-specific payload	
Octet #	**Field**
0	Payload Type = 0xFF (developer-specific)
1	Developer ID (LSB)
2	Developer ID (MSB)
3	Opcode = QUUPPA_GET_TAG_CONFIG

Example Get Tag Configuration REQ to get the current configuration of the tag:

http://localhost:8080/qpe/sendQuuppaRequest?tag=20835b102022&requestData=0xff34124300000000000000000000000000&humanReadable

References

[1] Quuppa: Quuppa Tag Emulation Pro Implementation Guidelines and Notes (v.1.0); Feb. 17, 2023

Revision history

Document revision	Date	Description of changes
**	2023-03-24	Initial release.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-03-24

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-37315 Rev. **

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie")

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.