HT MICRON SEMICONDUTORES S.A.
Av. Unisinos, 1550 | 93022-750 | São Leopoldo – RS | Brasil
www.htmicron.com.br

HT MICRON semicondutores

iMCP HTNB32L-XXX Application Note MQTT Example

MQTT Example for iMCP HTNB32L-XXX System-in-Package

Classification: CONFIDENTIAL

Doc. Type: USER MANUAL

Revision: v.01

Date: 12/05/2023

Code: HTNB32L-XXX-AN0001

SUMMARY

DOCUMENT INFO

This document provides technical details about the MQTT Example available on HTNB32L-XXX SDK. It describes implementation details, the connection procedure between HTNB32L-XXX device and a MQTT broker, as well as setup and test environment.

1. GENERAL DESCRIPTION

The iMCP HTNB32L-XXX is a highly compact and low-power wireless communication MCO/SiP featuring Qualcomm QCX-212 LTE IoT Modem supporting single-mode 3GPP Release 14 Cat. NB2 IoT connectivity. Its SDK (Software Development Kit) provides OpenCPU solutions based on a FreeRTOS system, where users can embed their own IoT application, as well as AT Commands, used in a master-slave model.

The MQTT Example is an application developed to demonstrate a usage case where the HTNB32L-XXX is connected to a MQTT broker through Narrowband IoT. The whole firmware was programed based on a Demo Board designed especially for this purpose, which interacts with its related digital twin through MQTT protocol. Implementation details are discussed throughout this document.

2. ARCHITECTURE

Developed to demonstrate a use case of MQTT protocol implemented on a HTNB32L device, the MQTT Example follows a simple clint-server architecture, where clients publish and subscribe data to specific topics and a MQTT broker acts as an intermediary (or a message hub) between them. Figure 1 illustrate the application architecture designed for the MQTT Example and Table 1 describe the agents involved:

Table 1: MQTT Example agent description.

	Agent	Description
1	Demo Board	PCB, with a HTNB32L-XXX device embedded, designed especially for this application. Acts as a MQTT client, publishing and subscribing data to the MQTT broker.
2	Digital Twin	Demo Board digital twin implemented in Python, which interacts with the HTNB32L-XXX through MQTT protocol. Acts as a MQTT client, publishing and subscribing data to the MQTT broker.
3	NB-IoT Network	NB-IoT network responsible to receive and forward data from the HTNB32L-XXX device to the MQTT broker.
4	MQTT Broker	MQTT server that acts as an intermediary between the Demo Board and the Digital Twin.

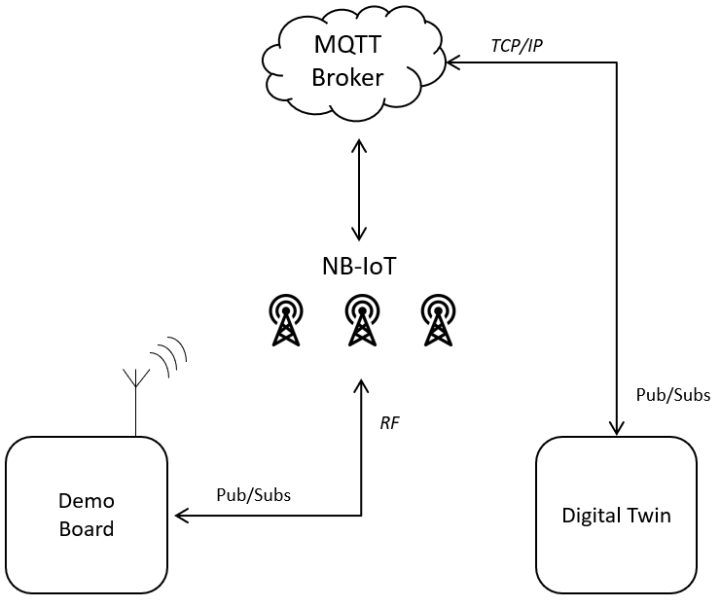

Figure 1: MQTT Example application architecture.

3. DEMO BOARD

A Printed Circuit Board, with similar characteristics to an Evaluation Board, was designed especially for this application. Pins that are not used in specific functions, such as buttons, LEDs or serial interface, are available in headers and can be used externally. Design and manufacture files can be found HERE.

4. HTNB32L-XXX FIRMWARE

This section describes the firmware developed for the MQTT Example, available on the HTNB32L-XXX SDK. Featuring OpenCPU, the whole application is implemented hardcoded, without the need for an external microcontroller. For details about the programming workspace, compiling and flashing can be found at *HTNB32L-XXX-UM001-Getting_Started,* stored at Docs directory, at the root of HTNB32L-XXX SDK.

4.1. FILE DESCRIPTION

Table 1 describes the source and headers files where the MQTT Example is implemented. All these files can be found at "*Applications/MQTT_Example*" directory.

Table 2: MQTT Example file description.

File (src/hdr)	Description
HT_BSP_Custom	Custom settings for HTNB32L. Configures the UART0, used to generated logs from the network, set the wake-up pins and retrieve network information from flash memory.
HT_Fsm	Implements the finite state machine responsible for controlling the MQTT Example application. It is the central file of this example.
HT_GPIO_Api	Configures the GPIO features used in this application.
HT_LED_Task	Configures and initializes a thread exclusively to blink the green LED, used to signalize the connection status.
HT_MQTT_Api	MQTT API that implements the basic functions used to publish and subscribe data to a MQTT broker.
HT_Peripheral_Config	File responsible to enable/disable serial peripherals, such as UART, SPI and I2C.

4.2. FINITE STATE MACHINE

MQTT Example is modulated in the finite state machine drawn in Figure 2. States are described Table 3.

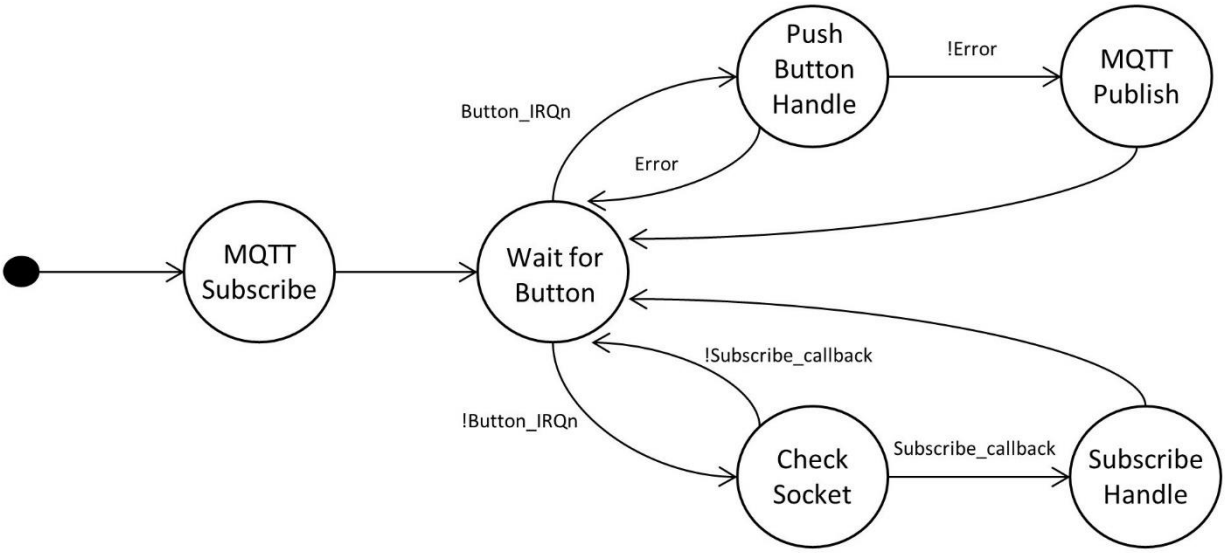

Figure 2: MQTT Example finite state machine.

Table 3: FSM description.

State	Description
MQTT Subscribe	Subscribe to the MQTT topics. These topics are responsible to transmit the commands coming from the digital twin.
Wait for Button	Keeps waiting until an user button (blue or white button) is pressed and sets the FSM to Push Button Handle states after a push button event. Also checks if a subscribe data was received.
Check Socket	Waits until the subscribe callback is called and sets the FSM to Wait For Button State after that.
Push Button Handle	Process a push button event by turning on the respective LED and publishing the respective button color to the MQTT broker.
Subscribe Handle	MQTT API that implements the basic functions used to publish and subscribe data to a MQTT broker.
MQTT Publish	File responsible to enable/disable serial peripherals, such as UART, SPI and I2C.

4.3. MQTT SETTINGS

The following settings can be customized according to the application's requirements.

- **Broker address:** broker.hivemq.com (Please check HiveMQ website for more details)
- **TCP port:** 1883
- **Client ID:** SIP_HTNB32L-XXX
- **Username:** HTNB32L-XXX
- **Password:** HTmicron

4.4. MQTT TOPICS

The objective of MQTT Example is to demonstrate the MQTT protocol functionality using the HTNB32L-XXX. To this end, the application is controlled by commands exchanged between the Demo Board and its digital twin, through MQTT topics that are published and subscribed by them. Table 4 describes all these topics:

Table 4: MQTT Example topics description.

Topic	Description
htnb32l_bluebutton_sw	Topic where the **digital twin** is supposed to publish a push button event occurred in its blue button.
htnb32l_whitebutton_sw	Topic where the **digital twin** is supposed to publish a push button event occurred in its white button.
htnb32l_set_state	Although it refers as a "set" function, this topic is used to retrieve the LED status of the digital twin. Therefore, it is a "set" topic from the point of view of the digital twin.
htnb32l_get_state	Although it refers as a "get" function, this topic is used to transmit a command to the digital twin, asking for its status (LED status). Therefore, it is a "get" topic from the point of view of the digital twin.
htnb32l_bluebutton_fw	Topic where the **demo board** publishes a push button event occurred in its blue button.
htnb32l_whitebutton_fw	Topic where the **demo board** publishes a push button event occurred in its white button.

4.5. PERIPHERALS

Mainly, the MQTT Example was implemented using two different peripherals:

Table 5: MQTT Example peripheral description.

Peripheral	Description
UART0	Used to transmitting logs informing the network status.
UART1	Used for printf.
GPIO19	Blue button.
GPIO7	White button.
GPIO3	Blue LED.
GPIO4	White LED.

5. DIGITAL TWIN

A digital twin of the Demo Board was implemented in Python, in order to explore the data exchange through MQTT protocol. The idea is to reproduce every event that happens on both sides, by publishing and subscribing their status. The source code is available at *Software_Apps* directory.

5.1. MQTT SETTINGS

Although the following settings can be customized according to the application's requirements, it must be the same as the ones configured on Subsection 4.3.

- **Broker address:** broker.hivemq.com
- **TCP port:** 1883
- **Client ID:** HTNB32L-XXX_MQTT_Client-<random_value>
- **Username:** HTNB32L-XXX_MQTT_Client
- **Password:** HTmicron

5.2. CODE DESCRIPTION

All methods implemented for the *Backend* class are listed and described at Table 6:

Table 6: Methods description.

Method	Description
blue_button_clicked	Blue button clicked event handle.
white_button_clicked	White button clicked event handle.
blue_led_on	Turn on the blue LED of the digital twin.
blue_led_off	Turn off the blue LED of the digital twin.
white_led_on	Turn on the white LED of the digital twin.
white_led_off	Turn off the white LED of the digital twin.
blink_white_led	Blink white LED for five times.
connect_mqtt	Establish a connection to the MQTT broker.
Publish	Publishes a message to a MQTT protocol.
Subscribe	Subscribes a MQTT protocol.

6. RUNNING EXAMPLE

1. Connect your HTNB32L device to the NB-IoT network, by running the AT Commands firmware available on the SDK. Details about how to do it can be found at *HTNB32L-XXX-UM001-Getting_Started.*
2. Run the HTNB32L-XXX-MQTT-Demo-SW software, available on the Software_Apps directory, by double clicking the "*backend.py*" file.

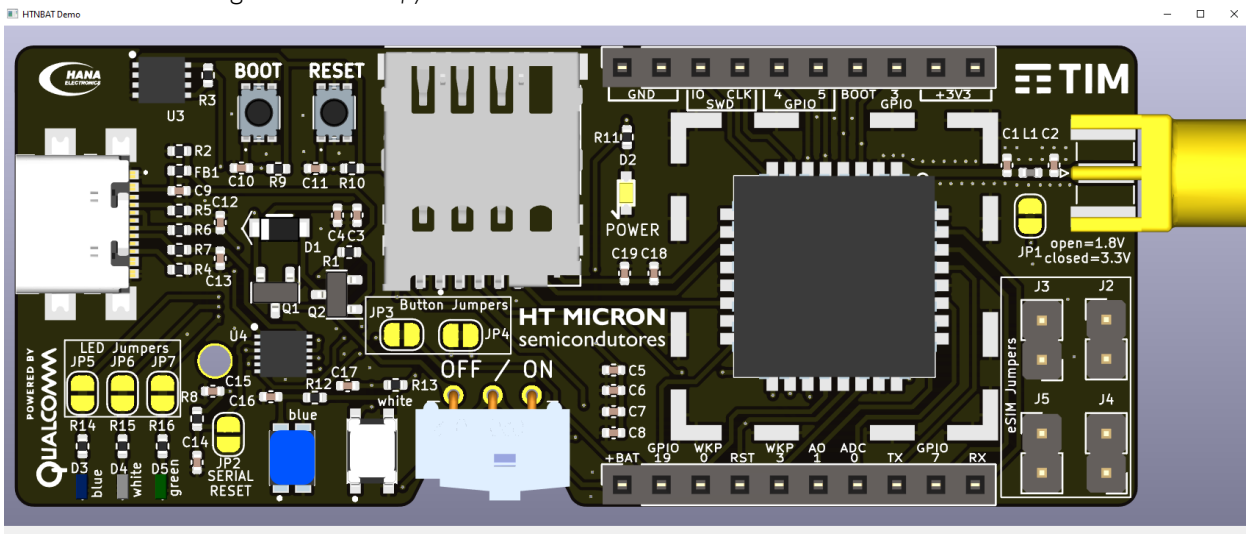

Figure 3: HTNB32L-XXX-MQTT-Demo-SW software.

NOTE

This software is implemented in Python. Any package issue can be easily fixed by installing its dependencies.

3. Wait until the green LED of the digital twin starts blinking. The green LED is used to inform the connection status between the MQTT Demo software and the MQTT broker. It will blink if the Python application was successfully connected to the host.
4. Compile and flash the MQTT_Example firmware on your HTNB32L-XXX device. Please refer to Sections **Error! Reference source not found.** and **Error! Reference source not found.** for more details about the co mpiling and flashing procedures.
5. Connect your HTNB32L-XXX device to the local NB-IoT network by following the steps detailed at Subsection **Error! Reference source not found.**.
6. Wait until the green LED embedded on the physical MQTT Demo board starts blinking. Similar as its digital-twin, this LED is responsible to signalize if the connection was successfully established between the HTNB32L-XXX, the NB-IoT network, the MQTT broker and also the MQTT Demo software.
7. The idea of the digital twin is to replicate everything that happens in the MQTT Demo board. For example:
    a. If the blue button of the MQTT Demo board is pressed, the blue LED of both PCBs will turn on or turn off, depending on its previous state.
    b. If the white button of the digital twin board is pressed, the white LED of both PCBs will turn on or turn off, depending on its previous state.

ABBREVIATIONS

Table 7: Abbreviations

Acronym	Description
GPIO	General Purpose Input Output
PCB	Printed-Circuit Board
SDK	Software Development Kit
VSCODE	Visual Studio Code
MQTT	Message Queuing Telemetry Transport
LIB	Library
TCP	Transmission Control Protocol
APP	Application
DEMO	Demonstration
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
I2C	Inter-integrated Circuit.
ID	Identifier

LIST OF FIGURES

LIST OF TABLES

REVISION HISTORY

Version	Date	Changes	Authors
00	12/05/2023	- Initial draft	HBG

CONTACT

HT MICRON SEMICONDUTORES S.A.
Av. Unisinos, 1550 | 93022-750 | São Leopoldo | RS | Brasil
www.htmicron.com.br

DOCUMENT INFORMATION

Document Title:	iMCP HTNB32L-XXX Application Note MQTT Example
Document Subtitle:	MQTT Example for iMCP HTNB32L-XXX System-in-Package
Classification:	CONFIDENTIAL
Doc. Type:	USER MANUAL
Revision:	v.01
Date:	12/05/2023
Code:	HTNB32L-XXX-AN0001

DISCLAIMER