KG200Z QuecOpen(SDK) Quick Start Guide

Wi-Fi&Bluetooth Module Series

Version: 1.0.0

Date: 2023-11-24

Status: Preliminary

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2023-11-24	Quentin QIU	Creation of the document
1.0.0	2020-11-24	Quentin QIU	Preliminary

Contents

Table Index

Figure Index

1 Introduction

Quectel KG200Z module supports QuecOpen® solution. QuecOpen® is an open-source embedded development platform based on RTOS system. It is intended to simplify the design and development of IoT applications.

Quectel KG200Z is a high-performance, highly integrated LoRaWAN module that supports the LoRaWAN standard protocol. This document mainly outlines the usage of the Quectel KG200Z QuecOpen® SDK, including directory structure, environment preparation, SDK compilation, firmware downloading, common problem handling, etc., aiming to help users quickly understand the development and compilation of LoRaWAN projects.

2 Environment Preparation

2.1. Hardware Environment

Table 1: Hardware Environment

Hardware Name	Amount
Quectel KG200Z Module	1
KG200Z-TE-B EVB	1
USB Type-C Cable	1
LoRa Antenna	1
ST-LINK/V2	1
JTAG Cable	1
USB Mini-B Cable	1

2.2. Hardware Installation Figures

The top front view of the KG200Z-TE-B EVB loaded with the KG200Z module is as follows.

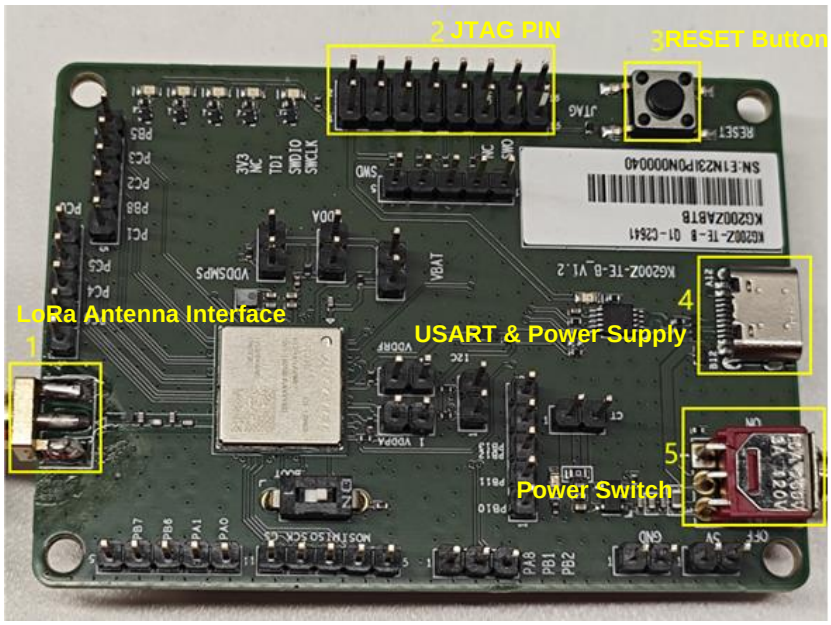


Figure 1: Top View of KG200Z-TE-B EVB

The hardware connection view of the KG200Z-TE-B EVB with the KG200Z module is as follows.

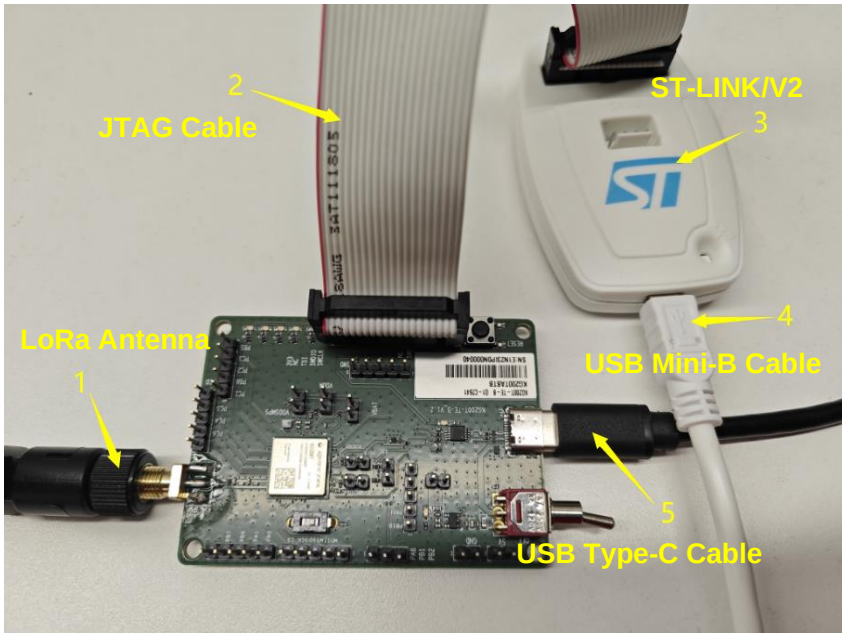


Figure 2: Hardware Connection View

The JTAG installation view of KG200Z-TE-B EVB with the KG200Z module is as follows. During installation, you need to align the installation indicating arrow at place 1 with the JTAG PIN at place 2.

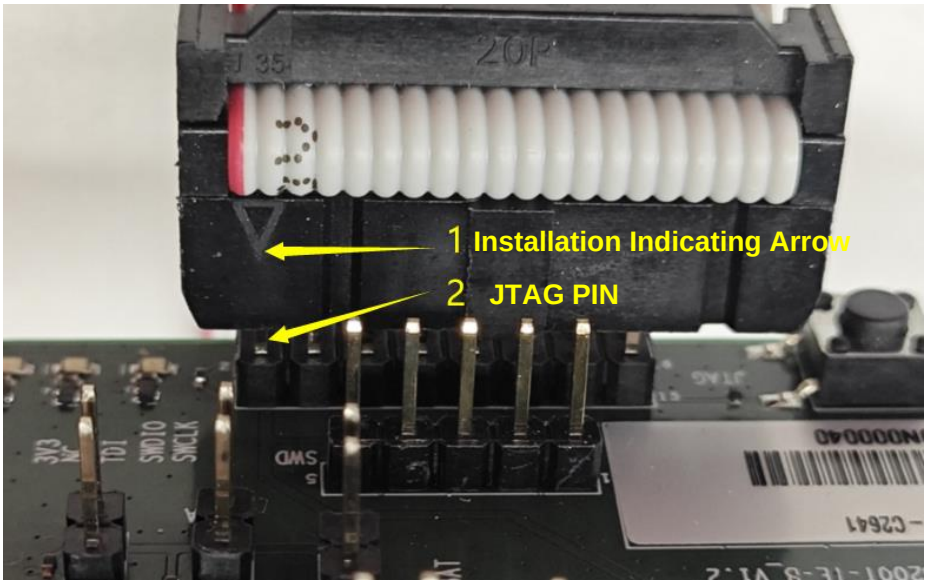


Figure 3: JTAG Installation View

The bottom view of KG200Z-TE-B EVB loaded with the KG200Z module is as follows.

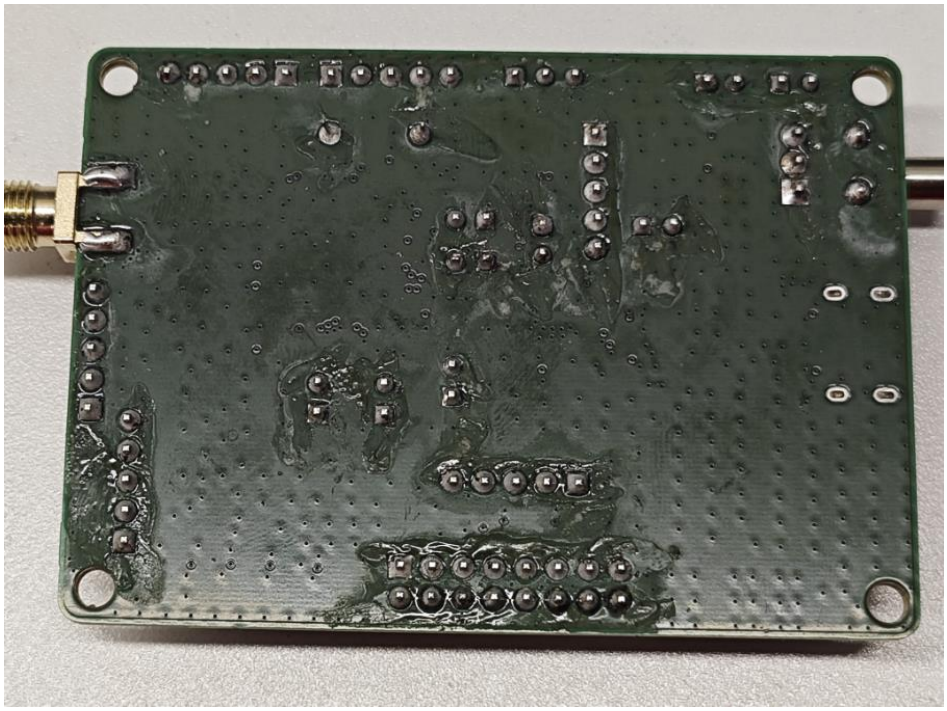


Figure 4: Bottom View of KG200Z-TE-B EVB

2.3. Software Environment Preparation

2.3.1. Obtaining SDK Package

1. Contact Quectel Technical Support for a GitLab account.

2. Visit https://git-master.quectel.com/wifi.bt/KG200Z to obtain the KG200Z QuecOpen SDK package.

The KG200Z QuecOpen SDK contains the following files:

Table 2: SDK Directory Structure

Directory Name	Description
Core	Header files and source files
Drivers	BSP drivers files
LoRaWAN	APP files and lora files
Middlewares	LoRaWAN and SubGHz_Phy middle layer files
STM32CubeIDE	STM32CubeIDE project files
Utilities	Project utility files

2.3.2. Obtaining Compilation Tool

The SDK of the module must be compiled with the STM32CubeIDE tool, which must be installed on a 32-bit or 64-bit Windows OS, and the Windows version is Windows 7 or above.

Visit https://www.st.com/en/development-tools/stm32cubeide.html#get-software to obtain the STM32CubeIDE installation package and install it on the Windows OS.

2.3.3. Obtaining Firmware Download Tool

The module can use the STM32CubeProgrammer tool for firmware downloading, which must be installed on a 32-bit or 64-bit Windows OS, and the Windows version is Windows 7 or above.

Visit https://www.st.com/en/development-tools/stm32cubeprog.html#get-software to obtain the STM32CubeProgrammer installation package and install it on the Windows OS.

3 SDK Compilation

3.1. Importing Project

Open the STM32CubeIDE tool and click "**File**" -> "**Import**" -> "**Existing Projects into Workspace**". Select the extract directory of the SDK package and import it into that directory.

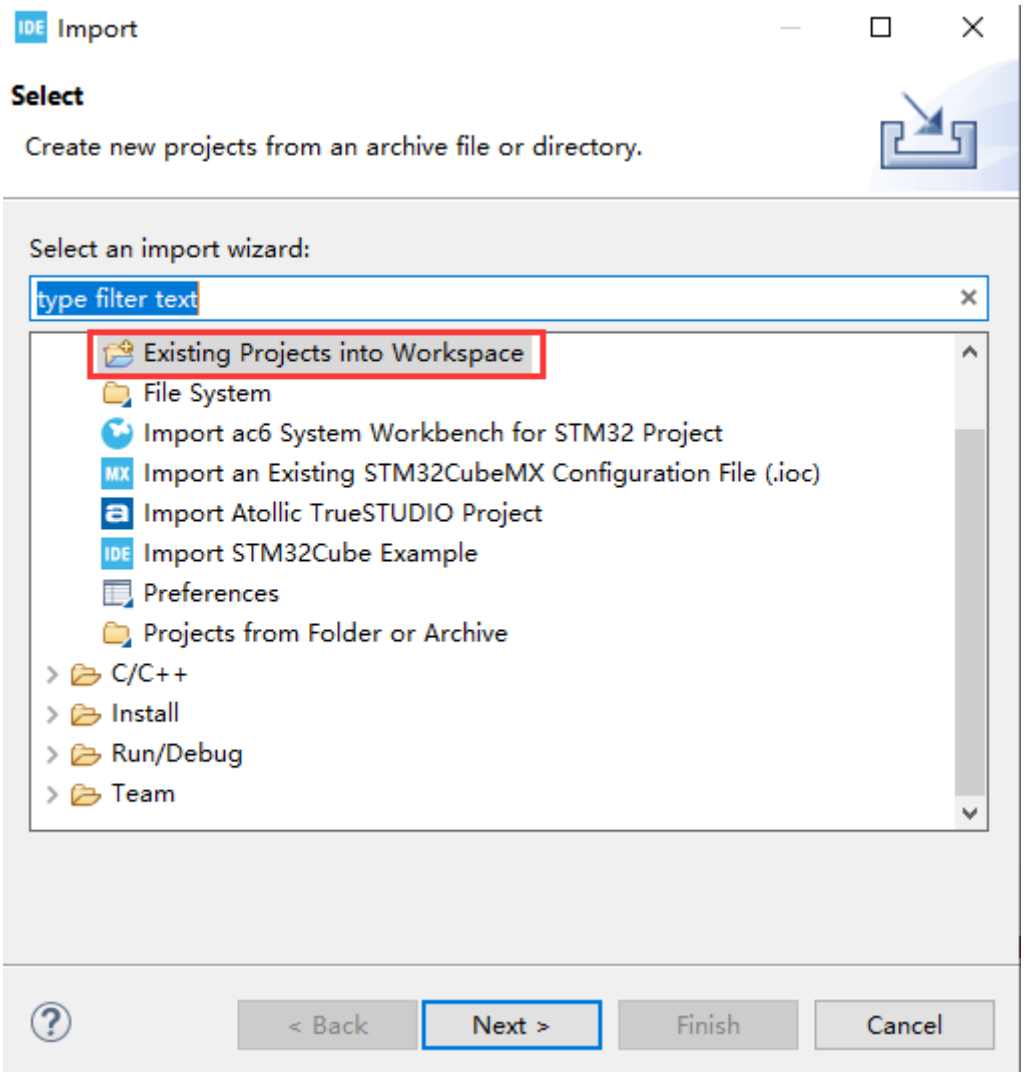


Figure 5: STM32CubeIDE Tool Interface

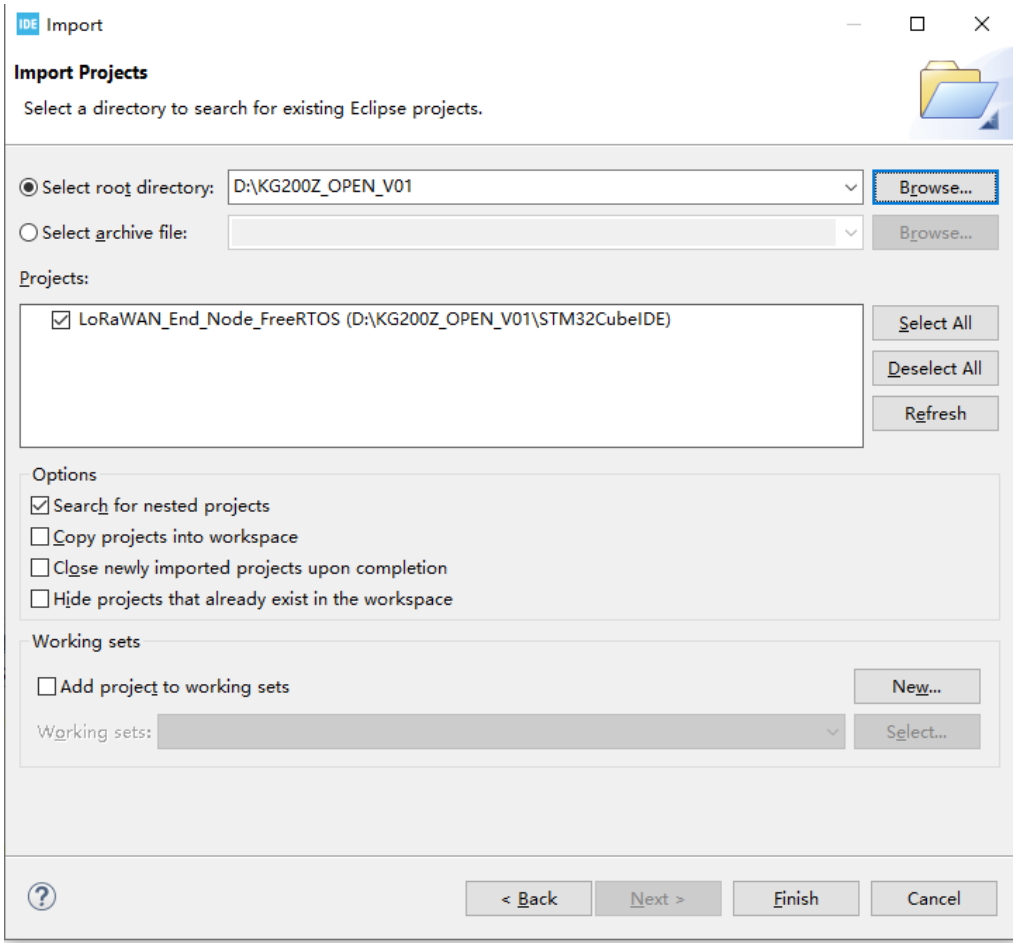


Figure 6: Importing Project

3.2. Compiling Project

In the STM32CubeIDE tool interface, click the button shown in the figure or use the keyboard shortcut "**Ctrl+B**" to compile the project.

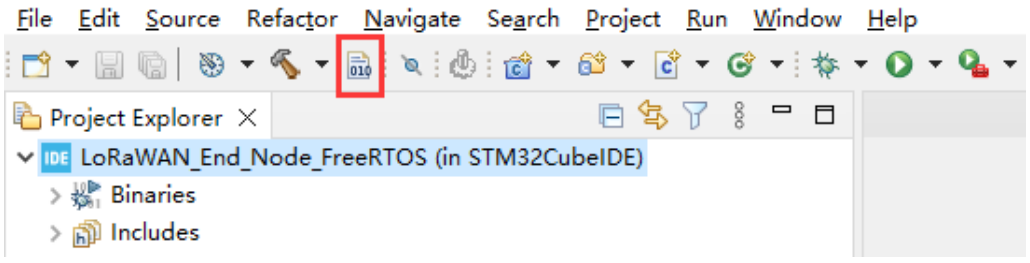


Figure 7: Compile Button

After the compilation is complete, the project is compiled successfully if the following information is displayed.

```
arm-none-eabi-size   LoRaWAN_End_Node_FreeRTOS.elf
arm-none-eabi-objdump -h -S  LoRaWAN_End_Node_FreeRTOS.elf  > "LoRaWAN_End_Node_FreeRTOS.list"
arm-none-eabi-objcopy  -O ihex  LoRaWAN_End_Node_FreeRTOS.elf  "LoRaWAN_End_Node_FreeRTOS.hex"
arm-none-eabi-objcopy  -O binary  LoRaWAN_End_Node_FreeRTOS.elf  "LoRaWAN_End_Node_FreeRTOS.bin"
   text    data     bss     dec     hex filename
  86848     296   22928  110072   1adf8 LoRaWAN_End_Node_FreeRTOS.elf
Finished building: default.size.stdout

Finished building: LoRaWAN_End_Node_FreeRTOS.hex
Finished building: LoRaWAN_End_Node_FreeRTOS.bin


Finished building: LoRaWAN_End_Node_FreeRTOS.list


10:28:02 Build Finished. 0 errors, 0 warnings. (took 2s.834ms)
```

Figure 8: Compile Console Print

The target files *LoRaWAN_End_Node_FreeRTOS.hex* and *LoRaWAN_End_Node_FreeRTOS.bin* will be generated in the *KG200Z_OPEN_V01\STM32CubeIDE\Debug* directory after the compilation is successful.

4 Downloading Firmware

There are two methods for downloading firmware: using the STM32CubeIDE tool or the STM32CubeProgrammer tool. You can choose between these methods as described in ***Chapter 4.1*** and ***4.2*** based on your requirements.

4.1. Firmware Download with STM32CubeIDE

The STM32CubeIDE tool uses source code to download firmware. After the compilation is complete, clicking the button shown in the tool interface to download firmware.

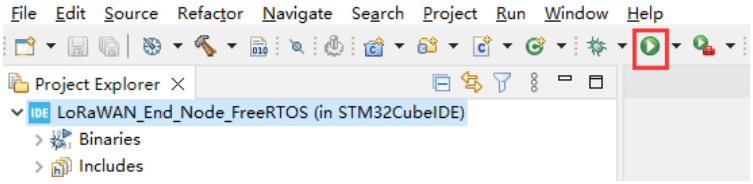


Figure 9: Download button

After the download is complete, if the following information is displayed, the download is successful.

```
Memory Programming ...
Opening and parsing file: ST-LINK_GDB_server_a73328.srec
  File          : ST-LINK_GDB_server_a73328.srec
  Size          : 85.11 KB
  Address       : 0x08000000


Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 42]
Erasing memory corresponding to segment 1:
Erasing internal memory sector 124
Download in Progress:


File download complete
Time elapsed during download operation: 00:00:02.855


Verifying ...



Download verified successfully


Shutting down...
Exit.
```

Figure 10: Download Console Print

4.2. Firmware Download with STM32CubeProgrammer

The STM32CubeProgrammer tool uses either bin or hex file to download firmware. Open the STM32CubeProgrammer tool and click "**Open file**" to import the firmware file.

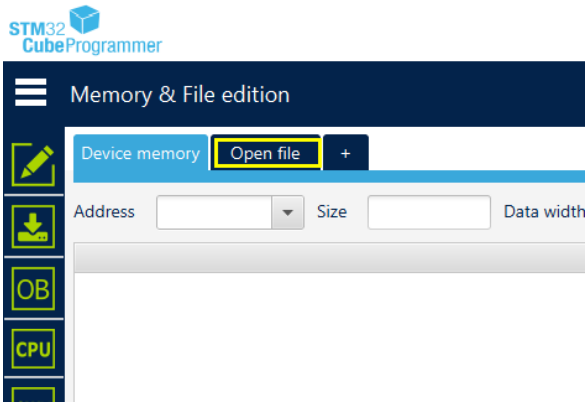


Figure 11: import hex file

Refresh and select the module's corresponding "Serial number", then click the "**Connect**" button to connect the module to STM32CubeProgrammer tool.

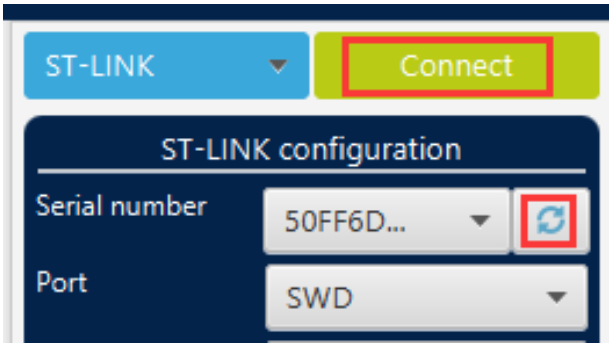


Figure 12: "Connect" Button

If the connection is successful, the status in the upper right corner will become “Connected”. Then click the “**Download**” button to start downloading the firmware.

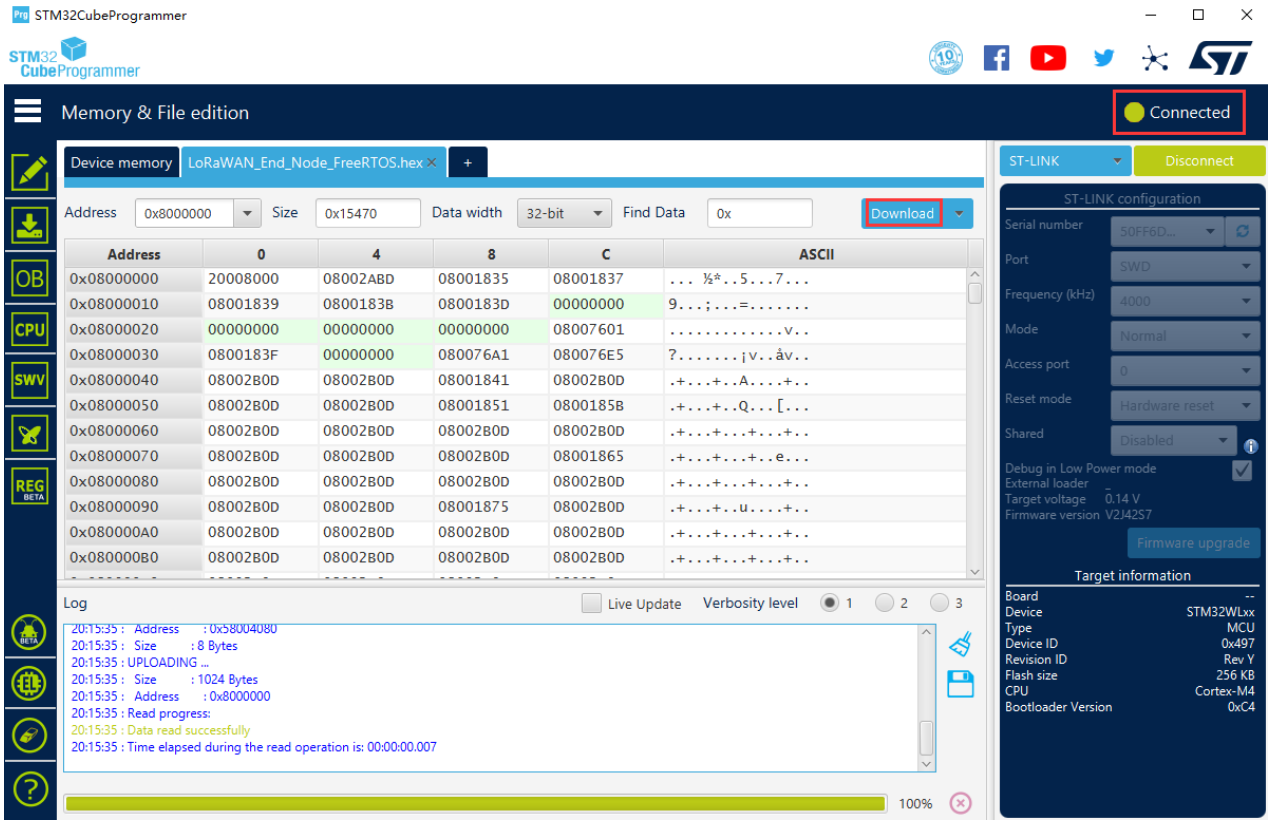


Figure 13: Downloading Firmware

After the download is successful, a message “File download complete” will be displayed.

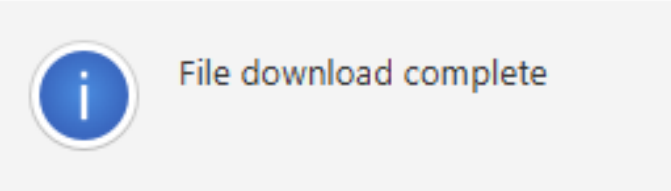


Figure 14: Download Complete

5 Common problem

5.1. Firmware Files Not Generated After Compilation

If bin and hex firmware files are not generated in the STM32CubeIDE tool interface after compilation.

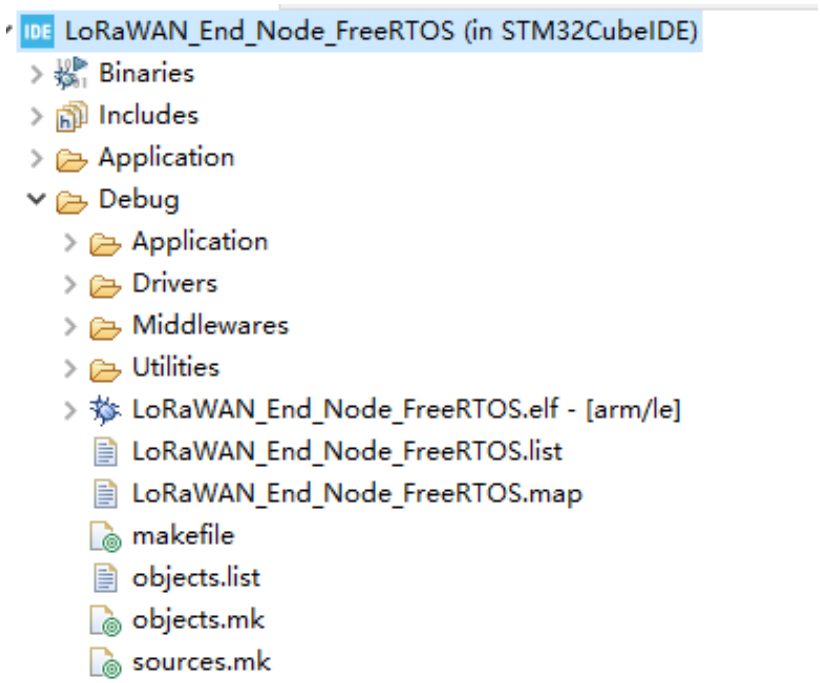


Figure 15: bin and hex Files Not Generated After Compilation

You can try the following steps to generate bin and hex firmware files again:

Step 1: Right-click on the project name in the STM32CubeIDE tool interface, select “Properties” or use the keyboard shortcut “**Alt+Enter**” to open the software settings page.

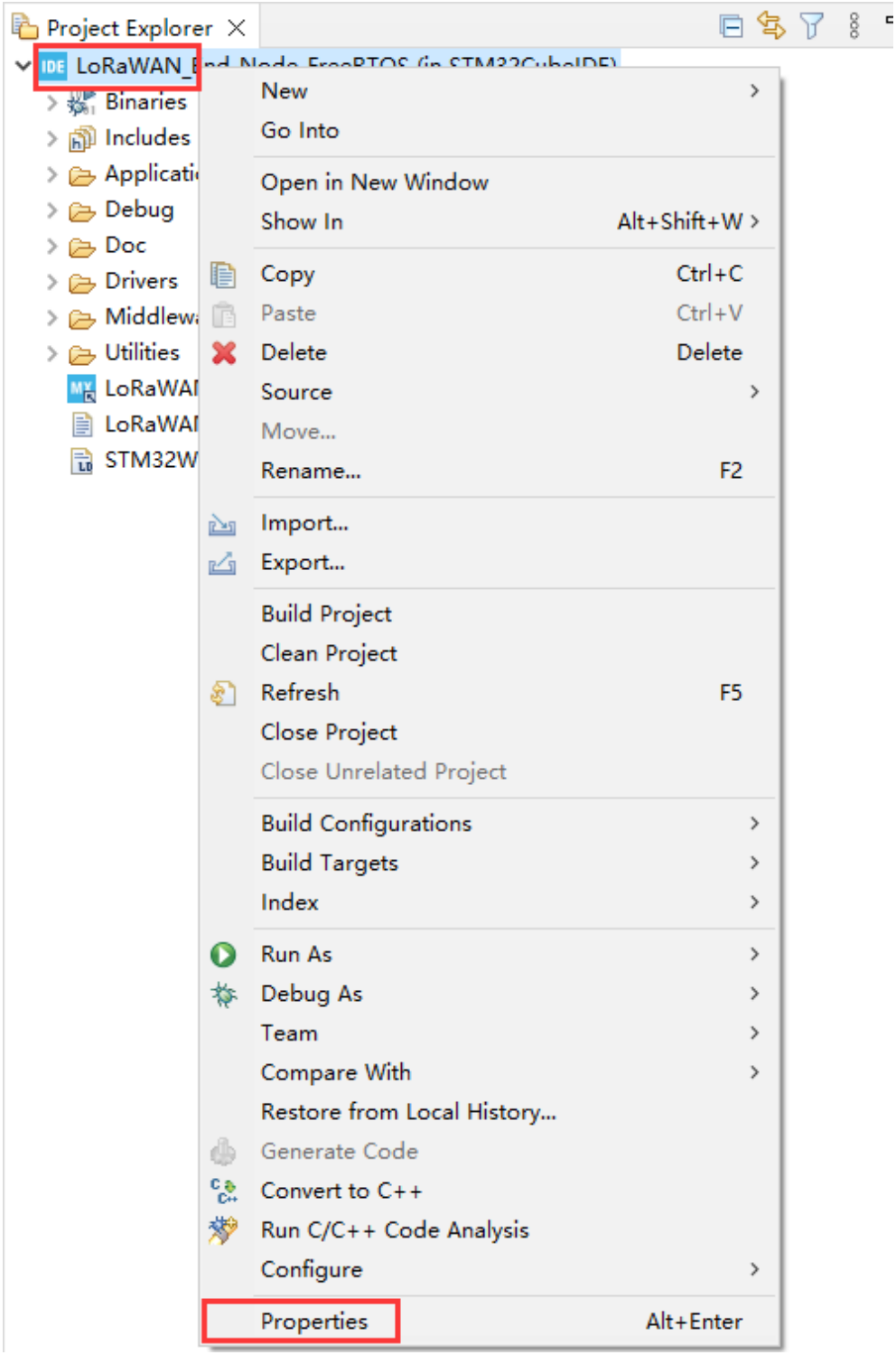


Figure 16: Opening Software Settings Page

Step 2: Click “C/C++ Build” -> “Settings” -> “Tool Settings” -> “MCU Post build outputs” and check the options “Convert to binary file (-O binary)” and “Convert to Intel Hex file (-O ihex)”.

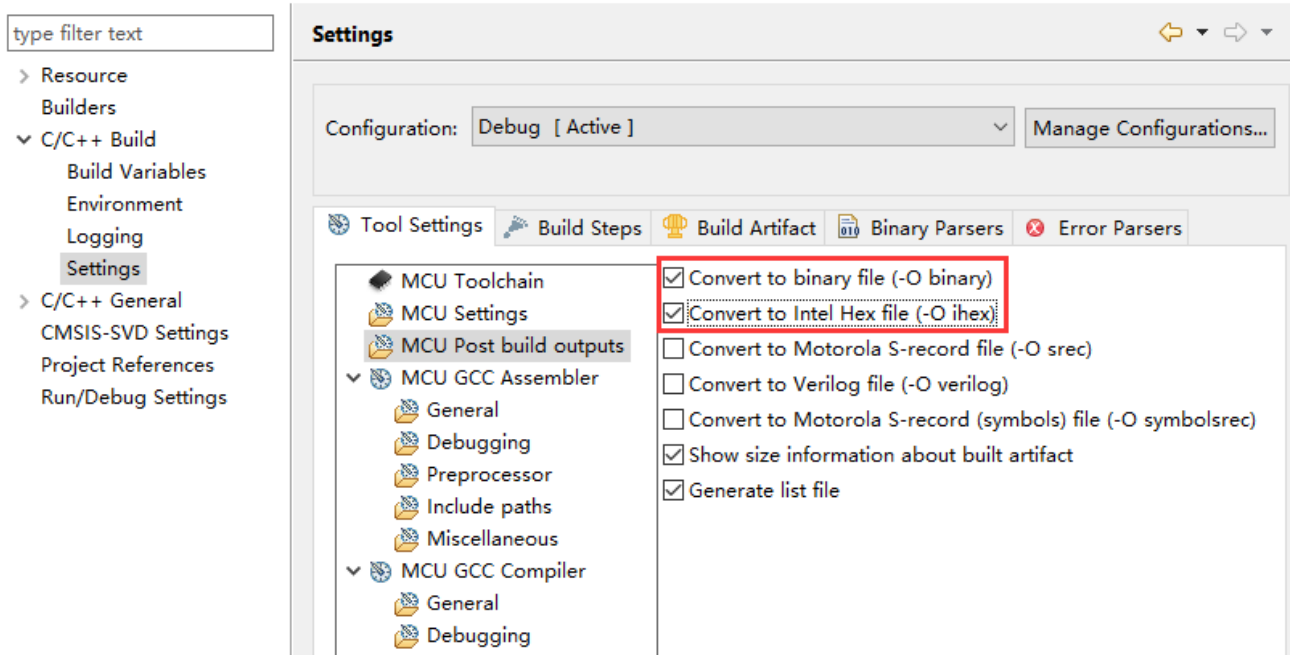


Figure 17: “MCU Post build outputs” Options

Step 3: Recompile the project by referring ***Chapter 3.2*** and then the bin and hex firmware files will be generated.

5.2. Connection Failure

If there is a connection failure when connectiong the module to the STM32CubeProgrammer tool as described in ***Chapter 4.2***.

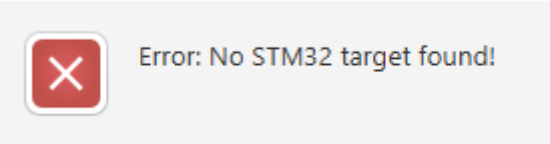


Figure 18: Not found STM32

Check whether “Reset mode” option is set to “Software reset”. If so, change the “Reset mode” option to “Hardware reset” and click the “Connect” button again to reconnect.

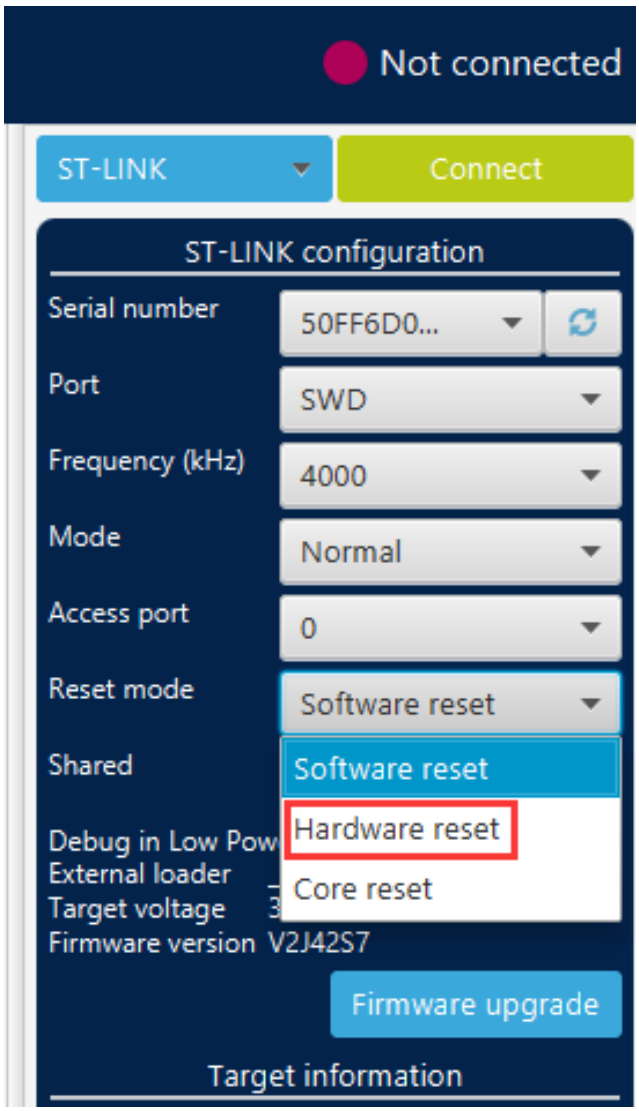


Figure 19: Change “Reset mode” Option

6 Configuration Modifications

This chapter mainly introduces various configurable items. You can modify these configuration items based on your actual situations and requirements. The default values of the configuration items are for reference only. If you choose to modify the configuration, you need to complete these modifications before compiling the SDK. The modified configurations will take effect only after recompiling and redownloading.

6.1. Debug Information Output

The debugging information of the KG200Z module is by default outputted through USART, with the USART log outputted through the Type-C interface. By default, the USART baud rate is set to 115200, with 1 stop bit and a byte size of 8 bits.

These settings can be customized by modifying the *KG200Z_OPEN_V01\Core\Src\usart.c* file.

```
hlpuart1.Instance = LPUART1;
hlpuart1.Init.BaudRate = 115200;
hlpuart1.Init.WordLength = UART_WORDLENGTH_8B;
hlpuart1.Init.StopBits = UART_STOPBITS_1;
hlpuart1.Init.Parity = UART_PARITY_NONE;
hlpuart1.Init.Mode = UART_MODE_TX_RX;
hlpuart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
hlpuart1.Init.OverSampling = UART_OVERSAMPLING_16;
hlpuart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
hlpuart1.Init.ClockPrescaler = UART_PRESCALER_DIV1;
hlpuart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
```

6.2. LoRaWAN Version

The LoRaWAN version can be chosen to use the expected feature or to be aligned with the required version defined by the Network Server. This version is defined in *KG200Z_OPEN_V01\LoRaWAN\Target\lorawan_conf.h* with the code below.

```
#define LORAMAC_SPECIFICATION_VERSION                0x01000300
```

The available values are 0x01000300 and 0x01000400, corresponding to V1.0.3 and V1.0.4, respectively. If the network server uses the LoRaWAN version V1.0.3, then the LoRaWAN version attribute must be set to 0x01000300 for data interaction with the network server.

Figure 20: Network Server Version

Users can check the LoRaWAN version currently used by the module through the serial port log.

When the attribute value is 0x01000300, the serial port log output is as follows.

```
L2_SPEC_VERSION:       V1.0.3
RP_SPEC_VERSION:       V1-1.0.3
```

When the attribute value is 0x01000400, the serial port log output is as follows.

```
L2_SPEC_VERSION:       V1.0.4
RP_SPEC_VERSION:       V2-1.0.1
```

6.3. LoRa Region/Frequency Selection

The region and its corresponding band selection are defined in *KG200Z_OPEN_V01\LoRaWAN\Target\lorawan_conf.h* file with the code below.

```
#define REGION_AS923
#define REGION_AU915
#define REGION_CN470
#define REGION_CN779
#define REGION_EU433
#define REGION_EU868
#define REGION_KR920
#define REGION_IN865
#define REGION_US915
#define REGION_RU864
```

Several regions can be enabled on the same project before compiling the SDK (by default, the EU and US regions are enabled). The default active region must be defined in the *KG200Z_OPEN_V01\LoRaWAN\App\lora_app.h* (default active region is EU).

If the working region of the target gateway is EU868, the REGION_EU868 must be enabled, and the ACTIVE_REGION must be LORAMAC_REGION_EU868.

```
#define ACTIVE_REGION                    LORAMAC_REGION_EU868
```

Figure 21: Gateway Band

Normally, setting the LoRa band and active region is enough, and there is no need to configure frequency parameters. However, the AS923 is quite different, there are a variety of frequency range configurations to be configured. If the user set LoRa region to AS923, the following configurations should be configured to determine the specific effective frequency range.

```
#define REGION_AS923_DEFAULT_CHANNEL_PLAN    CHANNEL_PLAN_GROUP_AS923_1
```

The available frequency range configurations of AS923:

- CHANNEL_PLAN_GROUP_AS923_1 (Default configuration. Freq offset: 0.0 MHz / Freq range: 915–928 MHz)
- CHANNEL_PLAN_GROUP_AS923_2 (Freq offset: -1.80 MHz / Freq range: 915–928 MHz)
- CHANNEL_PLAN_GROUP_AS923_3 (Freq offset: -6.60 MHz / Freq range: 915–928 MHz)
- CHANNEL_PLAN_GROUP_AS923_4 (Freq offset: -5.90 MHz / Freq range: 917–920 MHz)
- CHANNEL_PLAN_GROUP_AS923_1_JP (Freq offset: 0.0 MHz / Freq range: 920.6–923.4 MHz)

6.4. Activation Modes and Keys

There are two ways to activate a device on the network, either by OTAA or by ABP.

The global variable “ActivationType” in the *KG200Z_OPEN_V01\LoRaWAN\App\lora_app.c* and *KG200Z_OPEN_V01\LoRaWAN\App\lora_app.h* must be adjusted to activate the device with the selected mode (default mode is OTAA).

In *KG200Z_OPEN_V01\LoRaWAN\App\lora_app.c*:

```
static ActivationType_t ActivationType = LORAWAN_DEFAULT_ACTIVATION_TYPE;
```

In *KG200Z_OPEN_V01\LoRaWAN\App\lora_app.h*:

```
#define LORAWAN_DEFAULT_ACTIVATION_TYPE            ACTIVATION_TYPE_OTAA
```

Where ActivationType_t enum is defined in the *KG200Z_OPEN_V01\Middlewares\Third_Party\LoRaWAN\Mac\LoRaMacInterfaces.h* as follows:

```
typedef enum eActivationType {
ACTIVATION_TYPE_NONE = 0,
ACTIVATION_TYPE_ABP = 1,
ACTIVATION_TYPE_OTAA = 2,
}ActivationType_t;
```

6.4.1. OTAA Mode

If you activate the device in OTAA mode, you only need to set the values of LORAWAN_NWK_KEY and LORAWAN_JOIN_EUI.

LORAWAN_NWK_KEY is defined in *KG200Z_OPEN_V01\LoRaWAN\App\se-identity.h*. It's the network root key used for deriving session keys when joining a network in OTAA mode. The length of the LORAWAN_NWK_KEY value must be 32 characters.

```
#define LORAWAN_NWK_KEY         2B,7E,15,16,28,AE,D2,A6,AB,F7,15,88,09,CF,4F,3C
```

LORAWAN_JOIN_EUI, defined in the same file, is the IEEE EUI of the application or network server (used only in OTAA). The length of the LORAWAN_JOIN_EUI value must be 16 characters.

```
#define LORAWAN_JOIN_EUI                                    01,01,01,01,01,01,01,01
```

These attributes may have different names in the network server and gateway. For example, in the RAK gateway, LORAWAN_NWK_KEY is known as Application Key, and LORAWAN_JOIN_EUI is known as Application EUI.

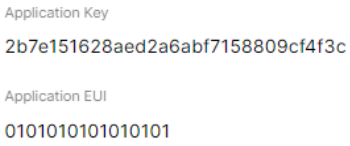


Figure 22: RAK Gateway

6.4.2. ABP Mode

The following three attributes LORAWAN_DEVICE_ADDRESS, LORAWAN_APP_S_KEY and LORAWAN_NWK_S_KEY are set only in ABP mode and generated randomly in OTAA mode. These three attributes are also defined in the *KG200Z_OPEN_V01\LoRaWAN\App\se-identity.h*.

LORAWAN_DEVICE_ADDRESS represents the terminal device address on the network (only used in ABP, generated by Network Server in OTAA). When set it to 00000000, it indicates using the MAC address of MCU device.

```
#define LORAWAN_DEVICE_ADDRESS                          00,00,00,00
```

LORAWAN_APP_S_KEY is the application session key used with the application server to encrypt/decrypt payloads (only used in ABP, generated by root key in OTAA).

```
#define LORAWAN_APP_S_KEY        2B,7E,15,16,28,AE,D2,A6,AB,F7,15,88,09,CF,4F,3C
```

ORAWAN_NWK_S_KEY is the network session key used with the network server to encrypt/decrypt payloads (only used in ABP, generated by root key in OTAA).

```
#define LORAWAN_NWK_S_KEY        2B,7E,15,16,28,AE,D2,A6,AB,F7,15,88,09,CF,4F,3C
```

When registering the KG200Z module on the network server, it's necessary to know the device's DevEUI. LORAWAN_DEVICE_EUI, defined in the same file, is the IEEE EUI of the terminal device. When set it to 0000000000000000, DevEUI is automatically set with a value provided by MCU platform.

```
#define LORAWAN_DEVICE_EUI                              00,00,00,00,00,00,00,00
```

When the device is powered on, the basic information of the device is output through USART, including the DevEUI of the device.

```
MW_LORAWAN_VERSION:   V2.5.0
MW_RADIO_VERSION:      V1.3.0
L2_SPEC_VERSION:       V1.0.3
RP_SPEC_VERSION:       V1-1.0.3
###### AppKey:         2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### NwkKey:         2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### AppSKey:        2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### NwkSKey:        2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### DevEUI:         00:80:E1:15:05:99:BA:88
###### AppEUI:         01:01:01:01:01:01:01:01
###### DevAddr:        05:99:B0:7E
```

6.5. Log level

The log printing is enabled in *KG200Z_OPEN_V01\Core\Inc\sys_conf.h* with the code below:

```
#define APP_LOG_ENABLED                 1
```

The log level is selected in *KG200Z_OPEN_V01\Core\Inc\sys_conf.h* with the code below:

```
#define VERBOSE_LEVEL                   VLEVEL_M
```

The following log levels are available:

- VLEVEL_OFF: all log levels are disabled
- VLEVEL_L: functional log is enabled
- VLEVEL_M: debug log is enabled (default level)
- VLEVEL_H: all log levels are enabled

7 Log Printing

7.1. Joining Network

7.1.1. OTAA Mode

When activating the device to join the network in OTAA mode, the device will automatically attempt to join the network upon booting. If the joining is successful, the serial port will print the following log information.

```
MW_LORAWAN_VERSION:    V2.5.0
MW_RADIO_VERSION:       V1.3.0
L2_SPEC_VERSION:        V1.0.3
RP_SPEC_VERSION:        V1-1.0.3
###### AppKey:          2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### NwkKey:          2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### AppSKey:         2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### NwkSKey:         2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### DevEUI:          00:80:E1:15:05:99:BA:88
###### AppEUI:          01:01:01:01:01:01:01:01
###### DevAddr:         05:99:BA:88
0s032:TX on freq 868300000 Hz at DR 0
1s516:MAC txDone
6s548:RX_1 on freq 868300000 Hz at DR 0
8s359:MAC rxDone


###### = JOINED = OTAA ====================
###### MCRootKey:      7D:F7:6B:0C:1A:B8:99:B3:3E:42:F0:47:B9:1B:54:6F
###### MCKEKey:        8C:B8:66:5E:0C:0E:0B:64:5B:2E:D9:E4:8A:19:27:7C
###### AppSKey:        A0:90:BB:3F:C5:C2:B6:42:9D:AB:F6:06:AA:EB:C6:38
###### NwkSKey:        53:D3:BD:4A:E4:A4:46:D1:DA:10:98:14:D0:63:53:C2
###### DBIntKey:       7A:C4:7C:65:FE:25:9B:B6:54:BD:26:35:19:F8:9C:8E
###### DevEUI:          00:80:E1:15:05:99:BA:88
###### AppEUI:          01:01:01:01:01:01:01:01
###### DevAddr:         02:0B:86:5F
10s034:VDDA: 254
10s034:temp: 24
10s039:TX on freq 867100000 Hz at DR 0
```

```
10s040:SEND REQUEST
11s687:MAC txDone
12s719:RX_1 on freq 867100000 Hz at DR 0
14s037:MAC rxDone
```

If the joining fails in OTAA mode, the serial port will print the following log information.

```
MW_LORAWAN_VERSION:    V2.5.0
MW_RADIO_VERSION:       V1.3.0
L2_SPEC_VERSION:        V1.0.3
RP_SPEC_VERSION:        V1-1.0.3
###### AppKey:          2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### NwkKey:          2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### AppSKey:         2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### NwkSKey:         2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### DevEUI:          00:80:E1:15:05:99:BA:88
###### AppEUI:          01:01:01:01:01:01:01:01
###### DevAddr:         05:99:BA:88
0s033:TX on freq 868100000 Hz at DR 0
1s517:MAC txDone
6s549:RX_1 on freq 868100000 Hz at DR 0
6s747:IRQ_RX_TX_TIMEOUT
6s747:MAC rxTimeOut
7s549:RX_2 on freq 869525000 Hz at DR 0
7s747:IRQ_RX_TX_TIMEOUT
7s747:MAC rxTimeOut



###### = JOIN FAILED
10s038:TX on freq 868300000 Hz at DR 0
11s522:MAC txDone
16s554:RX_1 on freq 868300000 Hz at DR 0
16s751:IRQ_RX_TX_TIMEOUT
16s751:MAC rxTimeOut
17s554:RX_2 on freq 869525000 Hz at DR 0
17s751:IRQ_RX_TX_TIMEOUT
17s751:MAC rxTimeOut


###### = JOIN FAILED
```

If the joining succeeds, the Join packet can be viewed in the gateway control page.

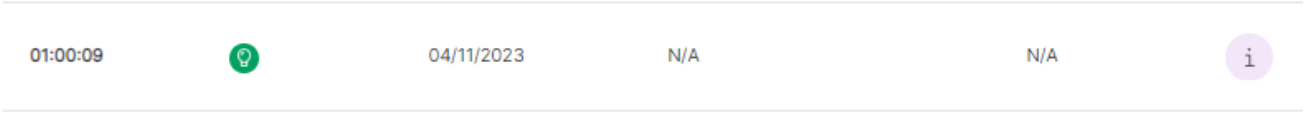


Figure 23: Join Packet

The packet types are as follows:

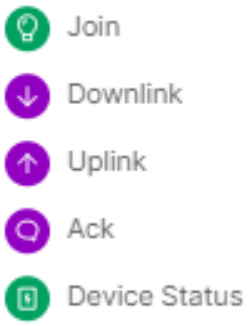


Figure 24: Packet Type

7.1.2. ABP Mode

In ABP mode, after the module is powered on, it is connected to the network by default and does not need to send Join packet to the gateway. The log print information in ABP mode is as follows:

```
APPLICATION_VERSION: V1.3.0
MW_LORAWAN_VERSION:   V2.5.0
MW_RADIO_VERSION:      V1.3.0
L2_SPEC_VERSION:       V1.0.3
RP_SPEC_VERSION:       V1-1.0.3
###### AppKey:         2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### NwkKey:         2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### AppSKey:        2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### NwkSKey:        2B:7E:15:16:28:AE:D2:A6:AB:F7:15:88:09:CF:4F:3C
###### DevEUI:         00:80:E1:15:05:99:BA:88
###### AppEUI:         01:01:01:01:01:01:01:01
###### DevAddr:        05:99:BA:88


###### = JOINED = ABP ======================
```

```
10s386:VDDA: 254
10s386:temp: 24
10s392:TX on freq 868500000 Hz at DR 0
10s393:SEND REQUEST
12s041:MAC txDone
13s074:RX_1 on freq 868500000 Hz at DR 0
14s391:MAC rxDone


###### ========== MCPS-Confirm ==========
```

There is no Join packet in ABP mode can be viewed in the gateway control page. And the module starts the packet transmission directly after startup.

02:13:40	04/11/2023	0027101801f4fe3e090d0503ab0000	2	i
02:13:31	04/11/2023	N/A	N/A	i
02:13:29	04/11/2023	0027101801f4fe3e090d0503ab0000	2	i
02:13:22	04/11/2023	8888	2	i
02:13:22	04/11/2023	0027101801f4fe3e090d0503ab0000	2	i
02:13:09	04/11/2023	0027101701f4fe3e090d0503ab0000	2	i
02:13:00	04/11/2023	0027101801f4fe3e090d0503ab0000	2	i
02:12:52	04/11/2023	0027101801f4fe3e090d0503ab0000	2	i

Figure 25: ABP Packets

7.2. Transmitting Data

When the module successfully transmits the data, the log print information is as follows:

```
40s059:TX on freq 867700000 Hz at DR 5
40s060:SEND REQUEST
40s132:MAC txDone
```

If the transmission succeeds, the uplink packet can be viewed in the gateway control page.

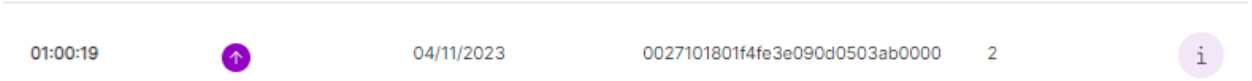


Figure 26: Uplink Packet

7.3. Receiving Data

When the module fails to receive the data, the log print information is as follows:

```
41s114:RX_1 on freq 867700000 Hz at DR 5
41s159:IRQ_RX_TX_TIMEOUT
41s160:MAC rxTimeOut
```

When the data is received successfully, the log print information is as follows:

```
###### ========== MCPS-Confirm ==========
50s061:VDDA: 254
50s061:temp: 24
50s066:TX on freq 867900000 Hz at DR 5
50s067:SEND REQUEST
50s134:MAC txDone
51s116:RX_1 on freq 867900000 Hz at DR 5
51s192:MAC rxDone
```

If the receiving succeeds, the downlink packet can be viewed in the gateway control page.

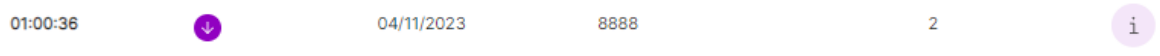
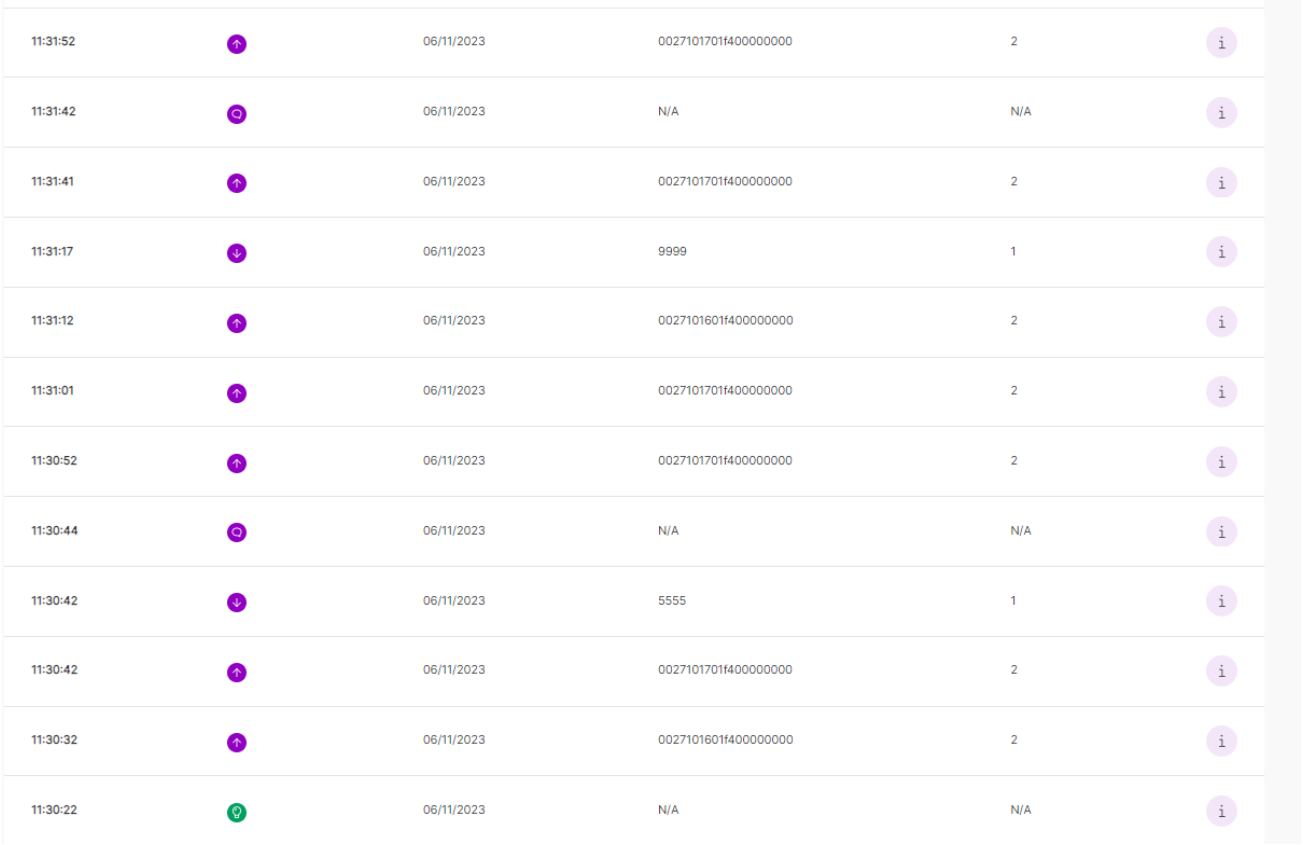


Figure 27: Downlink Packet

7.4. Duty Cycle

The uplink packet time interval can be viewed in the gateway control page has certain rules.



11:31:52	↑	06/11/2023	0027101701f400000000	2	i
11:31:42	↻	06/11/2023	N/A	N/A	i
11:31:41	↑	06/11/2023	0027101701f400000000	2	i
11:31:17	↓	06/11/2023	9999	1	i
11:31:12	↑	06/11/2023	0027101601f400000000	2	i
11:31:01	↑	06/11/2023	0027101701f400000000	2	i
11:30:52	↑	06/11/2023	0027101701f400000000	2	i
11:30:44	↻	06/11/2023	N/A	N/A	i
11:30:42	↓	06/11/2023	5555	1	i
11:30:42	↑	06/11/2023	0027101701f400000000	2	i
11:30:32	↑	06/11/2023	0027101601f400000000	2	i
11:30:22	↻	06/11/2023	N/A	N/A	i

Figure 28: Uplink Packet Time Interval

The uplink packet time interval is controlled by the duty cycle. You can define the duty cycle value used by the software in the *KG200Z_OPEN_V01\LoRaWAN\App\lora_app.h* file (Unit: ms; default value: 10000)

```
#define APP_TX_DUTYCYCLE                            10000
```

8 Appendix Terms and Abbreviations

Table 3: Terms and Abbreviations

Abbreviation	Description
APP	Application
ABP	Activation By Personalization
BSP	Board Support Package
EVB	Evaluation Board
IRQ	Interrupt Request
LoRa	Long Range Radio Technology
LoRaWAN	Lora Wide-Area Network
MAC	Media Access Control
MCPS	MAC Common Part Sublayer
MCU	Microcontroller Unit
OS	Operating System
OTAA	Over-The-Air Activation
Rx	Reception
Tx	Transmission
USART	Universal Synchronous/Asynchronous Receiver/Transmitter