Firmware Update Guide for Wireless Protocol Modules MiP Series

Description

This document provides step-by-step instructions to safely perform a firmware update on the MiP series module. It covers two different methods: updating via LoRa MiP GUI and flashing the firmware through an external microcontroller (Host MCU) using the module's communication channels (UART, SPI and I^2C).

Following this guide ensures optimal performance, unlocks new features, and maintains system stability.

Index

1. Updating Firmware via LoRa MiP GUI

1.1. Introduction

The purpose of this section is to outline the automation capabilities of the LoRa MiP GUI. Please note that all the manual steps and procedures detailed in Chapter 2 are executed automatically by the software.

1.2. Hardware Setup

Based on the specific product model, refer to the relevant document provided in the table below.

DevKit	AN_HWS001
Dongle	UG_USB001

1.3. Software Setup

The software LoRa MiP GUI can be downloaded directly from the official Mipot website, link below.

https://mipot.com/en/

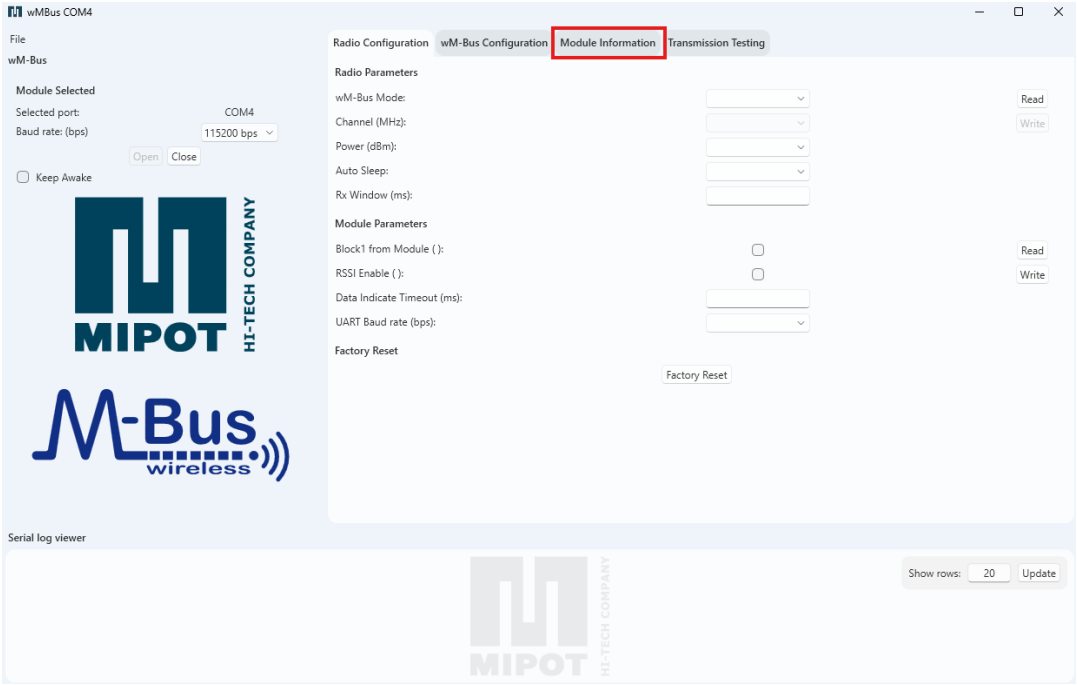
After logging in, in the documents section of your product, you can find two versions (32 and 64 bit) of the software.

1.4. Procedure

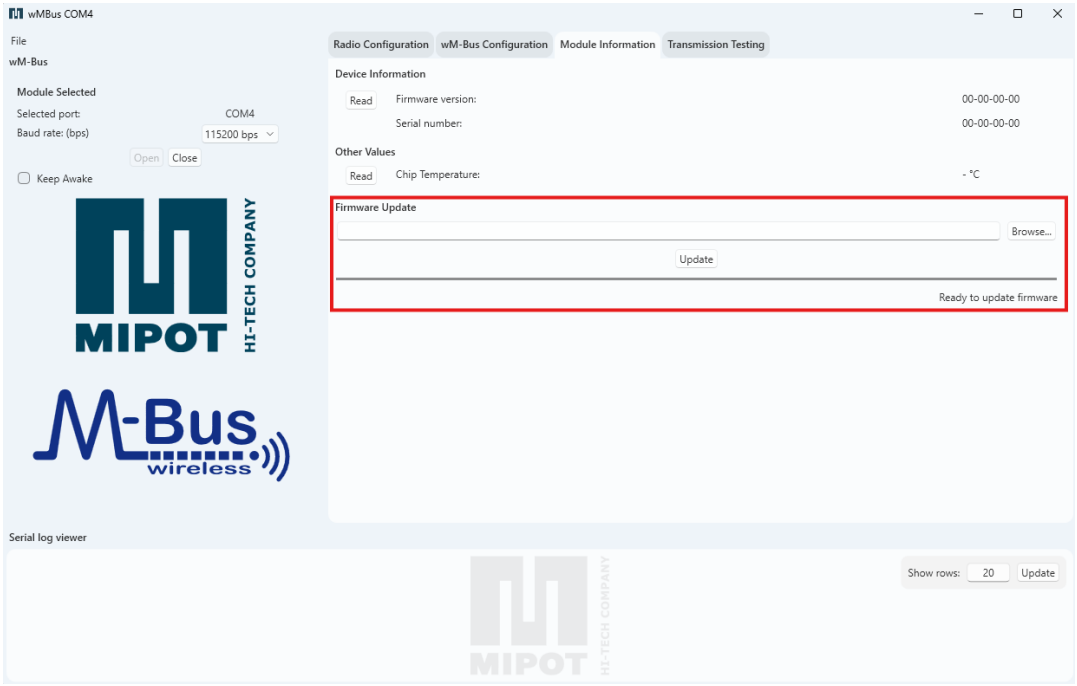
Once all connections are established and the module is powered on, launch the LoRa MiP GUI. Upon startup, the application automatically scans available serial ports for connected devices. Multiple devices can be connected to the same PC simultaneously.

Once the scan is complete, the serial port connected to the module is highlighted in green. Double-click the corresponding COM port to connect and use the module.

In the window that opens, navigate to the tab labelled “Module Information”.



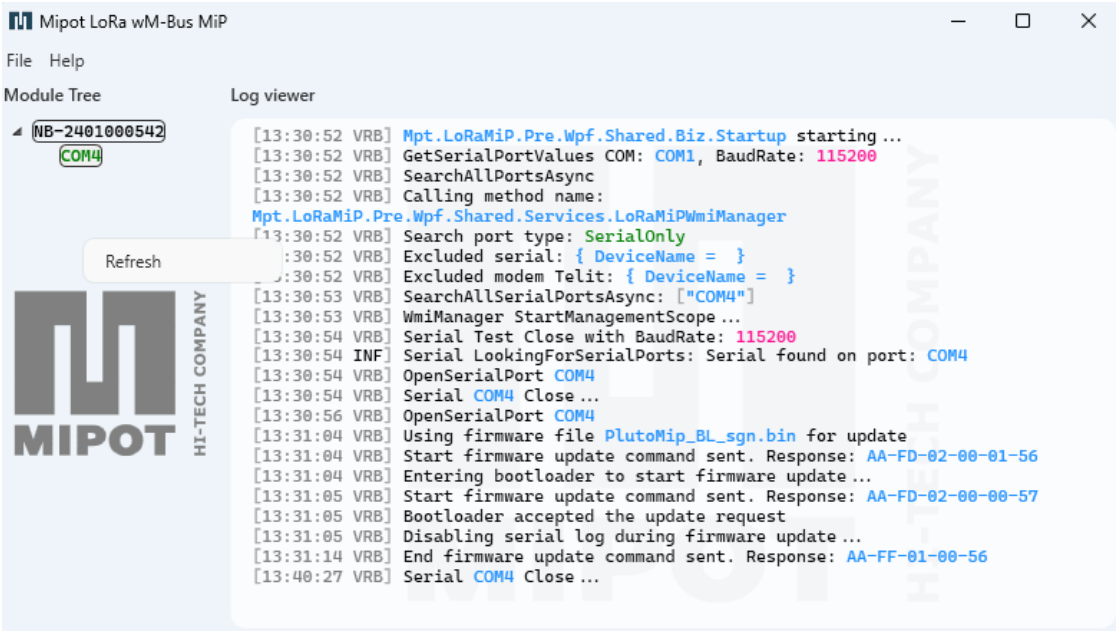
The firmware update section is highlighted in the image below



Click the “Browse” button to select the firmware file, then click “Update” to start the upgrade procedure.

Once the update is successful, a confirmation message will appear.

Next, close the current window you were working on. In the remaining window, right-click and select “Refresh” to allow the GUI to recognize the new firmware version. The module is now successfully updated and ready to use.



2. Updating Firmware via Serial Interfaces

This section provides a step-by-step guide on how to update the module's firmware using a standard serial interface (UART, SPI, I^2C). To perform this update, an external microcontroller is required to implement and execute the specific steps detailed in this guide. Alternatively, you can use a PC to communicate with the module to be updated via a USB-UART/SPI/I^2C converter.

The firmware update is valid for both the single-core and dual-core versions. In the latter case, the update must be managed by the M4 core, and the messages reported below must be written to the IPCC buffer.

1. Initializing and Opening the Serial Port

The first step requires the external microcontroller to configure and open its serial communication peripheral. The host MCU must set the correct communication parameters - such as baud rate, data bits, parity and stop bits – to match the module's specifications. Once configured, the serial port must be opened to enable data transmission and reception.

2. Driving the NWAKE Pin Low

To ensure uninterrupted communication, the external microcontroller must drive the module's NWAKE pin to a logic low level. This pin must be held low for the entire duration of the firmware update procedure. Keeping the NWAKE signal low prevents the module from entering any low-power or sleep states, ensuring it remains fully awake and responsive throughout the process.

3. Sending the Initialization Command and Awaiting Response

The external microcontroller must transmit the specific initialization command sequence through the serial interface to start the update procedure:

0xAA 0x7D 0x00 0xD8 (1)

After transmission, the host MCU must wait for a response frame formatted as:

0xAA 0xFD 0x02 status loc cks (2)

The host MCU must then parse the loc byte to determine the module's current state:

- loc = 0: the module is already in the bootloader. You can proceed directly to the next step.
- loc = 1: the module is currently running the application firmware. Upon receiving the command, the module will automatically reboot into bootloader mode. In this case, the host MCU must re-transmit the command (1). The module will then reply with a new frame where loc = 0, allowing the procedure to move forward.

4. Fragmenting the Firmware Binary File

The external microcontroller must divide the new firmware binary file into fixed-size blocks of exactly 256 bytes each. If the final block of the binary is smaller than 256 bytes, the host MCU must pad the remaining space with 0xFF bytes until the full 256-bytes size is reached. This ensures that every data payload transmitted to the module has a uniform length.

5. Structuring and Transmitting the Firmware Blocks

The external microcontroller must transmit each prepared 256-byte block to the module sequentially. For every chunk, the host MCU construct a specific message frame that includes a progressive block index. This index increments with each subsequent packet, allowing the module to track the data sequence and ensure no blocks are missed during the transfer.

0xAA	0x7E	0x11	BLK_IDX_0	BLK_IDX_1	BLK_IDX_C0	BLK_IDX_C1	7 …. 18	BLOCK	CKS

- BLK_IDX_0/BLK_IDX_1: block index written in little endian (2 bytes)
- BLK_IDX_C0/BLK_IDX_C1: bitwise NOT of the 2-byte block index written in little endian
- 7……18: initially zeroed bytes
- BLOCK: firmware block to send
- CKS: checksum of the entire message

Before calculating the CKS, the checksum of the message section from BLK_IDX0 to the end of the BLOCK must be computed. This checksum must be calculated as an XOR of the bytes, and the resulting value must be inserted at position 18. Once the value is updated, you can proceed to calculate the CKS value.

Once all fields have been updated, you can proceed to transmit the message.

6. Receiving the Module Status Response

For every message transmitted, the module replies with a status message featuring the following structure. The external microcontroller must capture this response to verify the outcome of the specific block transfer before proceeding.

0xAA 0xFE 0x01 status cks (3)

- Status = 0: all good
- Status ≠ 0: error

7. Transmitting the Procedure Termination Message

Once Steps 5 and 6 have been repeated for every firmware block, the external microcontroller must transmit the final termination message to conclude the update procedure. This message signals to the module that the entire binary file has been transferred and triggers the final validation phase.

Send:	0xAA 0x7F 0x00 0xD7	(4)
Reply:	0xAA 0xFF 0x01 status cks	(5)

- Status = 0: all good
- Status ≠ 0: error

Afterward, it restarts automatically. If the update is successful, the new firmware will boot; otherwise, it will get stuck in the bootloader, and the update procedure can be repeated.

3. Revision History

Revision	Date	Description
0.0	23.09.2026	First version