Morse Micro
reaching farther™

MM8108-EKH05

USER GUIDE

Table of Contents

1 Introduction

The MM8108-EKH05 evaluation kit is a fully integrated Wi-Fi HaLow development platform, designed for a wide range of IoT applications—from smart home devices to industrial automation systems. It features the MM8108-MF15457 module, incorporating Morse Micro's MM8108 Wi-Fi HaLow low-power SoC, alongside the STM32U585 low-power microcontroller (MCU) and the BlueNRG-M2 Bluetooth® SoC, the board provides robust wireless connectivity, low power consumption, and an extensive range of programmable interfaces and sensors. This versatile platform is ideal for software engineers developing energy-efficient IoT solutions.

This user guide provides a comprehensive overview of the MM8108-EKH05 evaluation kit, offering step-by-step instructions for setup, configuration, and programming. It covers all aspects of the board's capabilities, including the integration of Wi-Fi HaLow and Bluetooth® connectivity, utilization of the STM32U585 MCU, and leveraging the onboard sensors and interfaces. Detailed examples and practical use cases are provided to help developers quickly prototype and validate IoT applications, ensuring they can harness the full potential of this evaluation kit. The guide also includes troubleshooting tips, optimization techniques for low-power operation, and insights into best practices for IoT system design. Whether the user is new to IoT development or an experienced engineer, this guide is structured to help efficiently create innovative, connected solutions.

Note: At this time, a Wi-Fi HaLow access point is required in addition to the MM8108-EKH05. Refer to Section 4 for setup details.

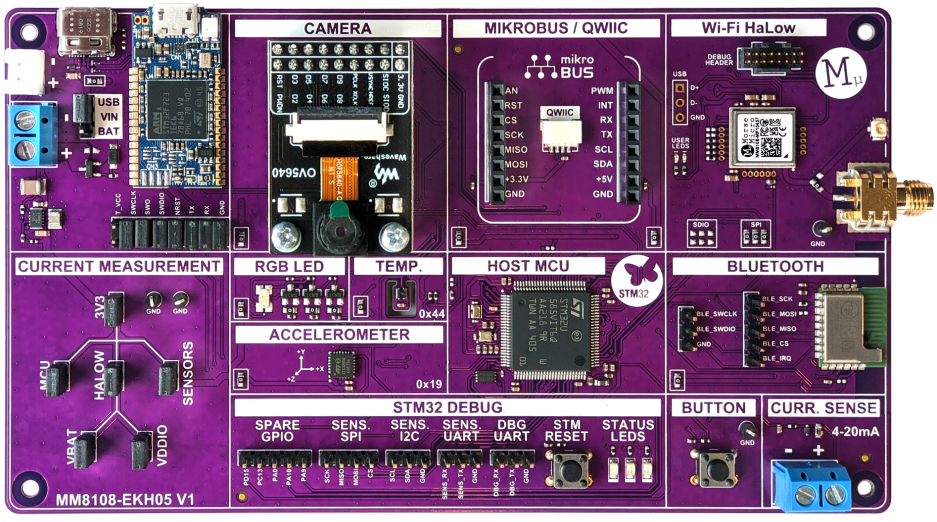


Figure 1. MM8108-EKH05 evaluation kit.

2 Features

This section highlights the key features of the MM8108-EKH05 Wi-Fi HaLow evaluation kit, showcasing the advanced capabilities that make it a versatile platform for IoT development.

Key features:

- Wi-Fi HaLow support for long-range, low-power wireless connectivity.
- STM32U585 microcontroller with energy-efficient architecture.
- Integrated programmable interface for flexible development.
- Integrated sensors for enhanced IoT applications.
- Multiple power input sources, including battery connectors.

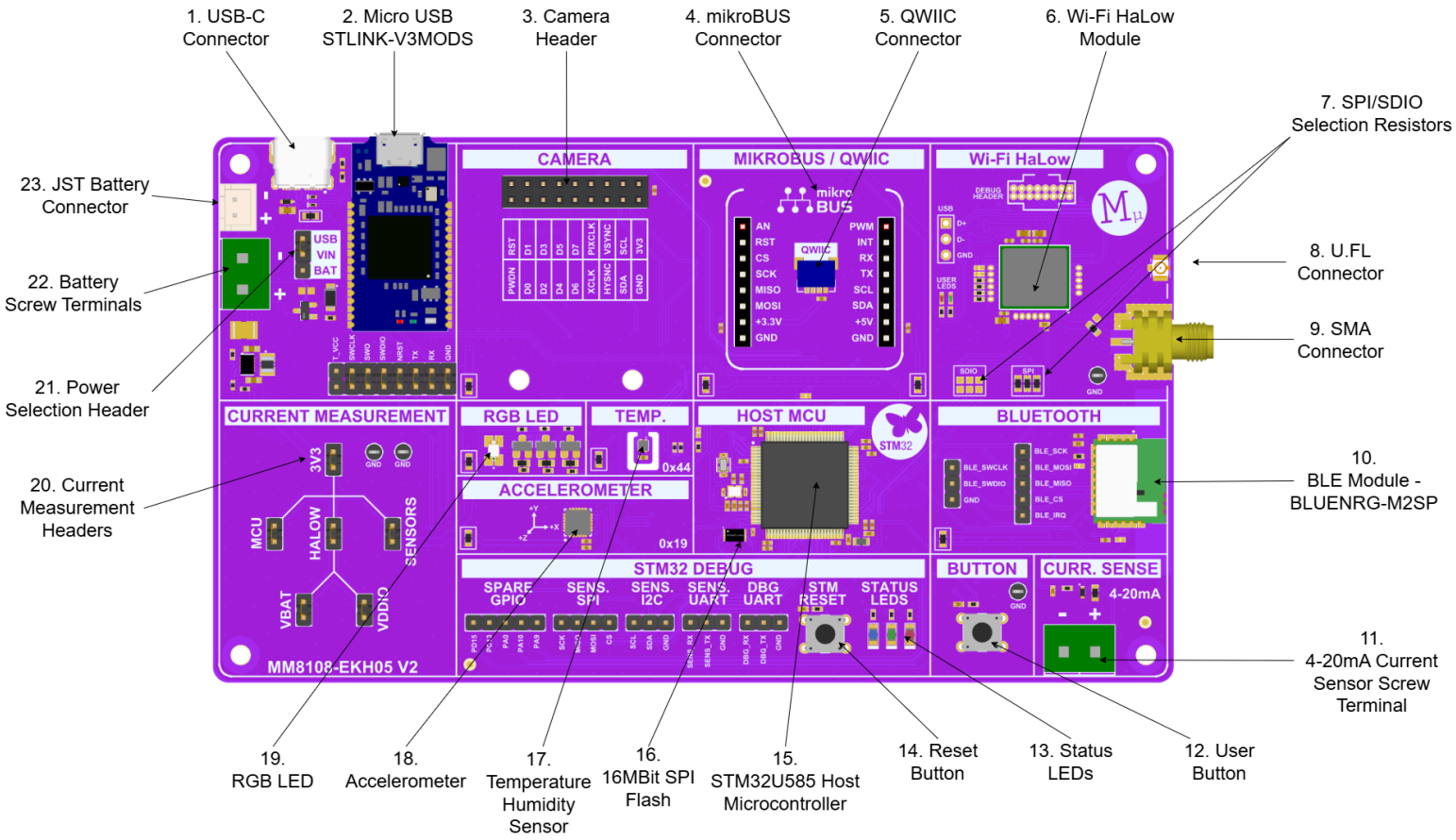


Figure 2. Features of the MM8108-EKH05.

#	Item	Description
1.	**USB-C Connector**	5V power supply. USB data lines connecting to STM32U585 host microcontroller.
2.	**Micro USB STLINK-V3MODS**	5V power supply. Programming and debugging access to STM32U585 host microcontroller. Access to STM32U585 host microcontroller UART. Note: The header pins below the STLINK-V3MODS module allow for disconnection from the host microcontroller for low-power testing.
3.	**Camera Header**	Camera connector to suit a DCMI (Digital Camera Interface) camera module. Connects to DCMI interface on STM32U585 host microcontroller. Note: A Camera module is provided in the MM8108-EKH05 kit.
4.	**mikroBUS Connector**	Standard connector for mikroBUS modules. Interfaces connect to STM32U585 host microcontroller.
5.,	**QWIIC Connector**	Standard connector for QWIIC modules. Interface connects to STM32U585 host microcontroller.
6.	**HaLow Module**	Morse Micro MM8108-MF15457 module with the MM8108 Morse Micro chipset. Connects to SPI interface (by default) on STM32U585 host microcontroller.
7.	**SPI/SDIO Selection Resistors**	Option to populate 0-ohm resistors for either SPI (default) or SDIO operation to STM32U585 host microcontroller. Note: Changing this setting also requires software modifications.

8.	**U.FL Connector**	Antenna output for U.FL connections. Note: SMA Connector is set by default. 0-ohm resistor change required to use U.FL connector. Instructions available in a chapter below.
9.	**SMA Connector**	Antenna output for SMA connections. Note: SMA is the default antenna output.
10.	**BLUENRG-M2SP Module**	ST BLUENRG-M2SP BLE module for provisioning devices. Connects to SPI interface on STM32U585 host microcontroller. Note: Access to SPI data lines and SWD data lines through header pins beside the BLUENRG module.
11.	**4-20mA Current Sensor Screw Terminal**	Connects to an ADC pin on the STM32U585 host microcontroller for 4-20mA operation.
12.	**User Button**	Programmable button for user applications. Connects to GPIO on STM32U585 host microcontroller.
13.	**Status LEDs**	Individual red, green, and blue LEDs for user applications. Connects to GPIO on STM32U585 host microcontroller.
14.	**Reset Button**	Resets the STM32U585 host microcontroller.
15.	**STM32U585 host microcontroller**	Ultra-low-power MCU with FPU ARM Cortex-M33 MCU with TrustZone, 160 MHz with 2 MBytes of Flash memory.
16.	**16MBit SPI Flash**	Winbond W25Q16JV SPI flash. Available as extra memory for user applications on STM32U585 host microcontroller

17.	**Temperature Humidity Sensor**	Sensirion SHT40 Ultra-Low-Power, 16-bit relative humidity and temperature sensor. Connects to I2C Interface on STM32U585 host microcontroller
18.	**Accelerometer**	ST IIS328DQ Ultra low-power 3-axis accelerometer with digital output. Connects to I2C Interface on STM32U585 host microcontroller.
19.	**RGB LED**	RGB LED connected to PWM interface on STM32U585 host microcontroller.
20.	**Current Measurement Headers**	Headers for power measurement. Follows a tree structure allowing measurement of either the entire system or individual blocks.
21.	**Power Selection Header**	Header to select system power source. Options: 1. USB power from USB-C or STLINK 2. Battery JST Connector or Battery screw terminals
22.	**Battery Screw Terminals**	Option for connecting a battery to the system through screw terminals.
23.	**JST Battery Connector**	Option for connecting a battery to a system through a standard JST connector.

3 Development Environment

3.1 System Environment

All major operating systems are supported:

- Windows 10, Windows 11
- Linux 64-bit
- macOS (macOS 14 and below.)

The development options are described in the following section.

3.2 Development Options

Morse Micro provides three development options for the MM8108-EKH05. The CMSIS Pack with STM32CubeIDE fully supports all hardware and includes pre-loaded example applications for immediate, out-of-box use. The other development options are available but have limited support for the hardware peripherals of the MM8108-EKH05. The rest of this document assumes the CMSIS pack will be used and describes how to use it with the MM8108-EKH05 hardware.

3.2.1 CMSIS Pack

A CMSIS (Cortex Microcontroller Software Interface Standard) pack is a standardized software package format defined by ARM to facilitate the development of software for ARM Cortex-based microcontrollers. It bundles together all necessary components, such as device drivers, middleware, libraries, and configuration files, into a single package that can be easily distributed and integrated into development environments. The Morse Micro CMSIS pack includes everything needed to integrate Wi-Fi HaLow applications using this standardized package.

Morse Micro CMSIS pack releases can be found via the following link. The .pack file should be downloaded, although the source code is also available in each release.

https://github.com/MorseMicro/mm-iot-cmsis/releases

The instructions for using this package with STM32CubeIDE as the supported IDE are included in section 5 of this document.

3.2.2 MM IoT SDK

The Morse Micro IoT SDK provides a framework for developing embedded applications that include connectivity provided by Morse Micro MM8108 chip. This SDK package contains the software components to build the application firmware for the STM32 processor. These components include the RTOS (FreeRTOS), network stack (LwIP or FreeRTOS + TCP), Morse Micro host software (morselib), and the platform-specific board support package (BSP).

The MM IoT SDK release packages can be found via:
https://github.com/MorseMicro/mm-iot-sdk/releases

NOTE: This development option does NOT support existing peripherals on the MM8108-EKH05. The rest of this user guide is tailored for the CMSIS pack development option.

3.2.3 PlatformIO + MM IoT SDK

PlatformIO is an open source multi-platform build system with integration available for various IDEs. It has powerful built-in features such as debugging, unit testing, and static code analysis and it supports a multitude of hardware platforms. PlatformIO also has a vast amount of libraries and example code available for various sensors and actuators used in IoT applications. The MM IoT SDK provides an integration for PlatformIO that supports multiple development boards from ST Microelectronics combined with Morse Micro extension board for developing Wi-Fi HaLow enabled applications.

The user guide for the PlatformIO can be found from the following link on the Morse Micro support portal:
https://www.morsemicro.com/resources/user_guides/Platform-IO-MM-IoT-SDK-User-Guide.pdf.

NOTE: This development option does NOT support existing peripherals on the MM8108-EKH05. The rest of this user guide is tailored for the CMSIS pack development option.

4 Getting Started

The most straightforward setup involves connecting a host PC to both the MM8108-EKH05 as a station (STA), and a separate Wi-Fi HaLow access point (AP). The recommended AP is the HaLowLink 1, which will be referenced throughout this document.

⚠️Note: At this time, the MM8108-EKH05 **cannot** operate as an AP.

The MM8108-EKH05 connects through the STLINK Micro USB connector, and the AP connects through an Ethernet cable to the LAN port. The Host PC must set the Ethernet interface to a DHCP client - see AP Device Settings . The HaLow connection can either be Over The Air (OTA) through antennas, or cabled. For cabled, it is recommended to use 40 dB of external attenuation as shown below. The device can be damaged if connected directly without attenuation.

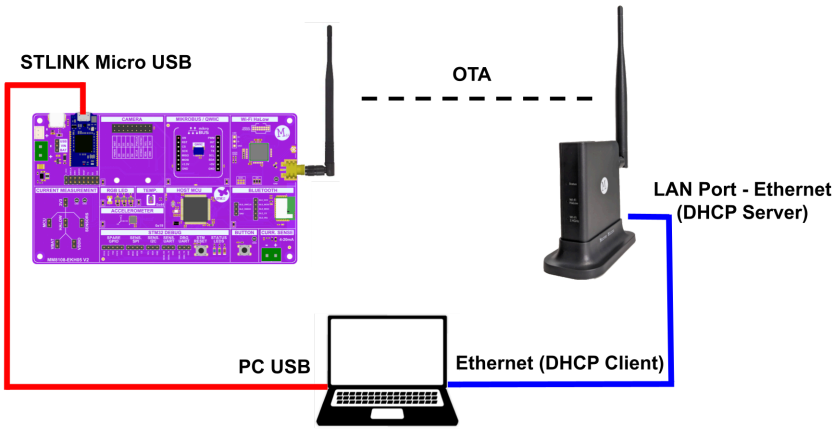


Figure 3. Hardware setup with antennas.

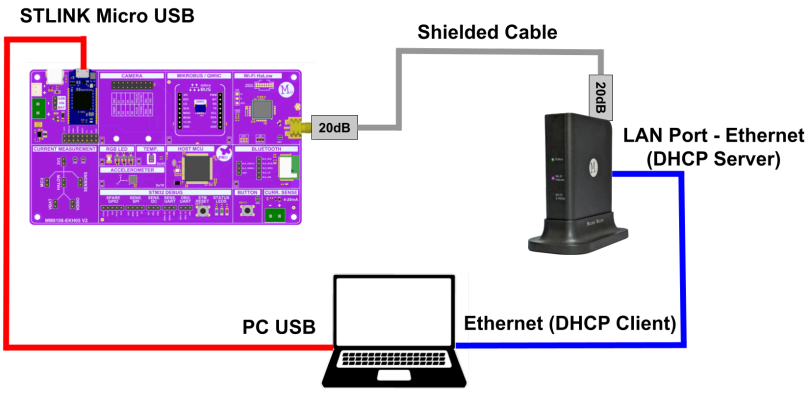


Figure 4. Hardware setup with shielded cable.

4.1 Default Jumper Configuration

By default, the following headers are populated:

1. USB power selection header
2. All STLINK headers (except for T_VCC)
3. Current measurement headers

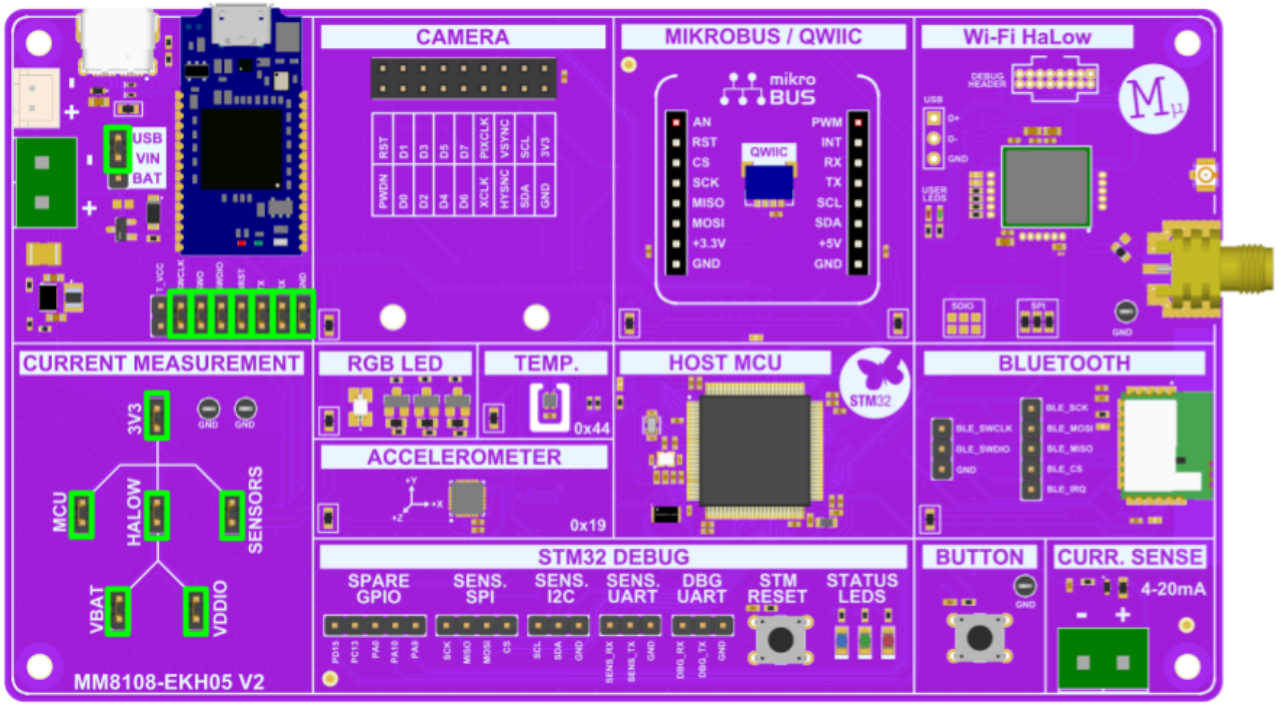


Figure 5. Default jumper placements on MM8108-EKH05.

4.2 AP Device Settings

This section describes how to set up the HaLowLink 1 access point (AP). These steps assume the device is being configured for the first time. For more information, the User Guide for the HaLowLink 1 can be found on the Customer Portal.

Ensure the PC used is set up as a DHCP client for its ethernet configuration. Navigate to **192.168.12.1** in a web browser. Enter the username and password on the HaLowLink 1 label. Click **Log in**.

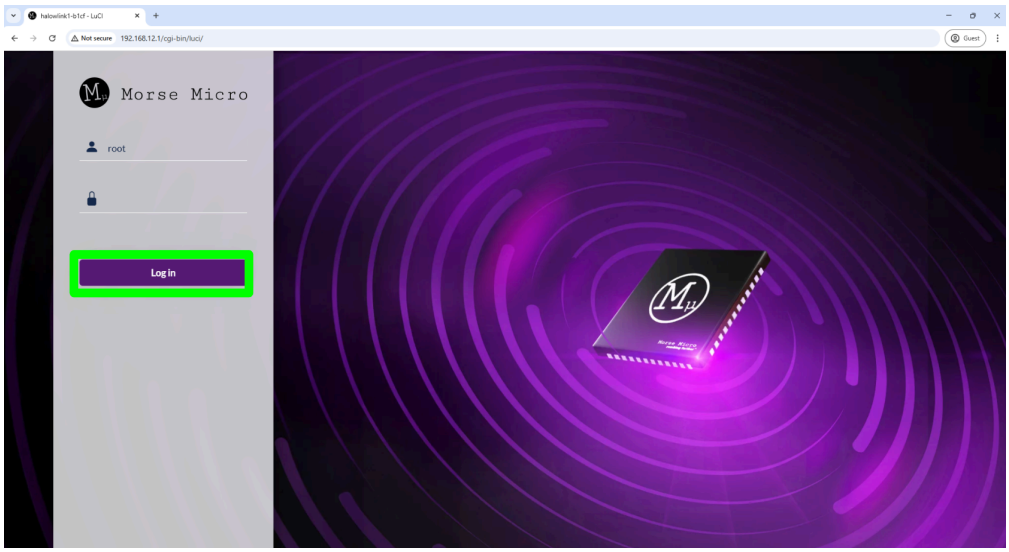


Figure 6. OpenWRT login screen.

Select **Quick Config**.

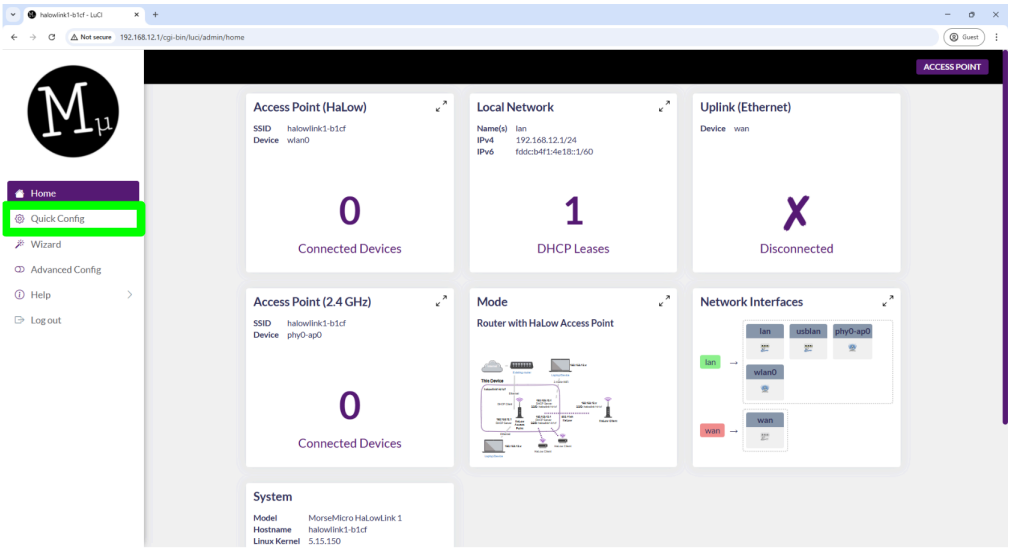


Figure 7. Highlighting the Quick Config menu.

Under the Wireless → Morse Micro HaLow Wi-Fi 802.11ah (radio1) section is where the **SSID**, **Passphrase**, **Bandwidth, Channel** and **Country Code** can be set.

All MM8108-EKH05 example applications expect the following SSID and Passphrase. Set the following to ensure the MM8108-EKH05 associates with the AP.

SSID: MorseMicro
Passphrase: 12345678

The default **Country Code** loaded on the HaLowLink 1 and the MM8108-EKH05 is **US**. If operating these devices in a different country, these will need to be changed. On the HaLowLink1, this can be changed from the same page. On the MM8108-EKH05, please see section 5.4.

After configuring, click **Save & Apply**. These parameters can be changed later if needed.

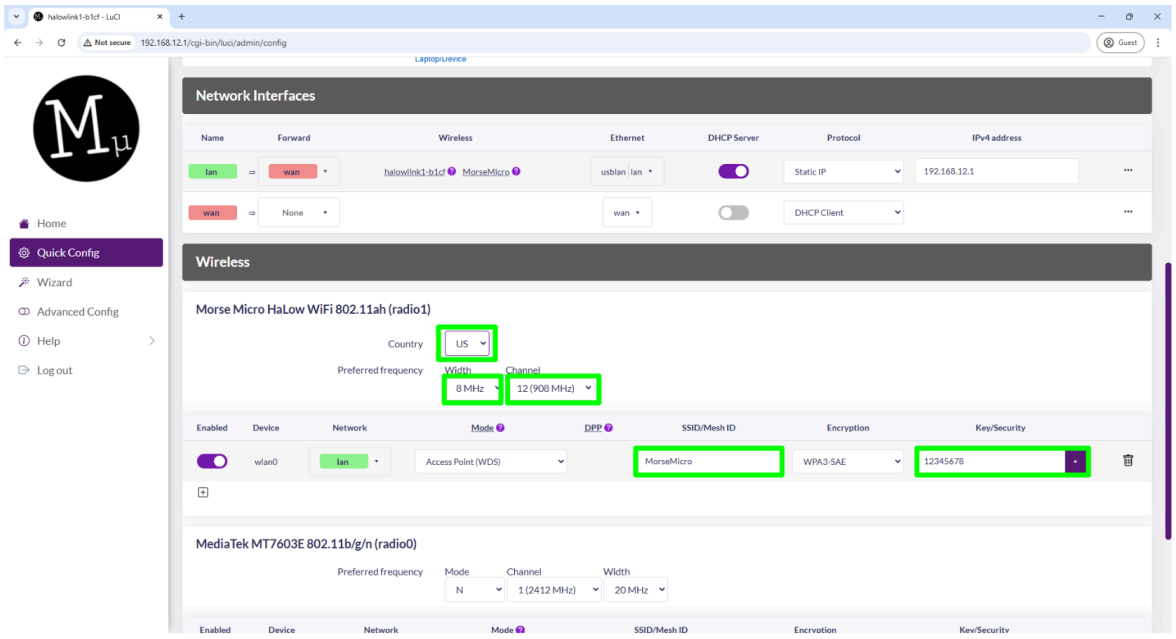


Figure 8. Quick config menu showing where to change country code, SSID, passphrase, bandwidth, and channel.

4.2.1 Changing Channel, Bandwidth, DTIM Period

After initial configuration, the channel, bandwidth, and DTIM period can be changed by navigating to the **Advanced Config → Network → Wireless** page, and then clicking **Edit** on the HaLow radio.

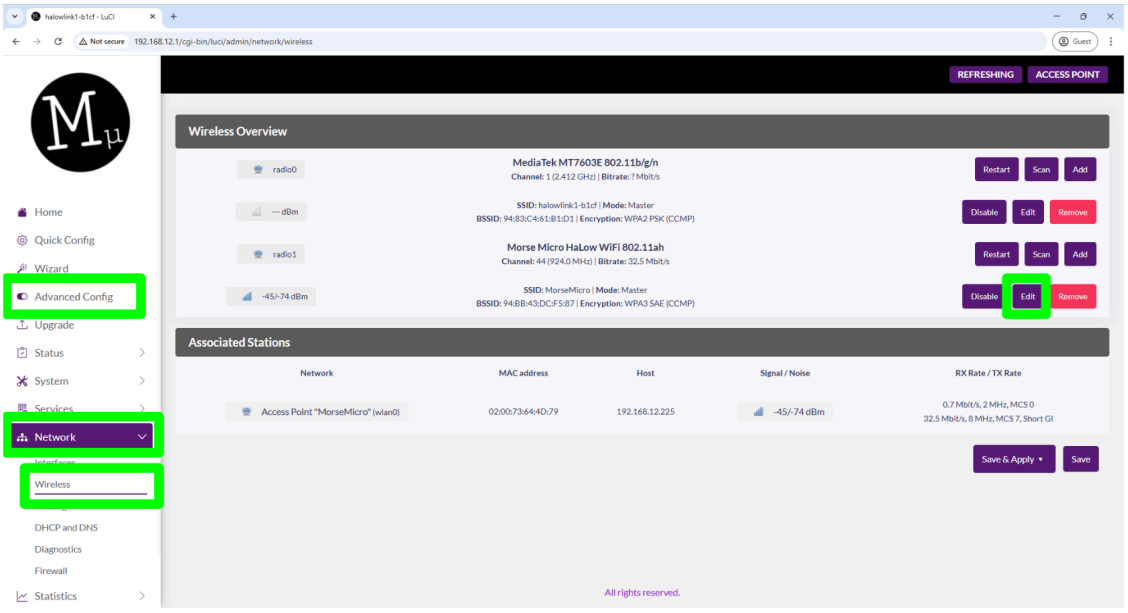


Figure 9. Wireless network configuration page on OpenWRT.

The **Bandwidth** and **Channel** can then be changed from the **Device Configuration** menu. The **DTIM Interval** can then be changed by selecting **Advanced Settings** from the **Interface Configuration** menu.

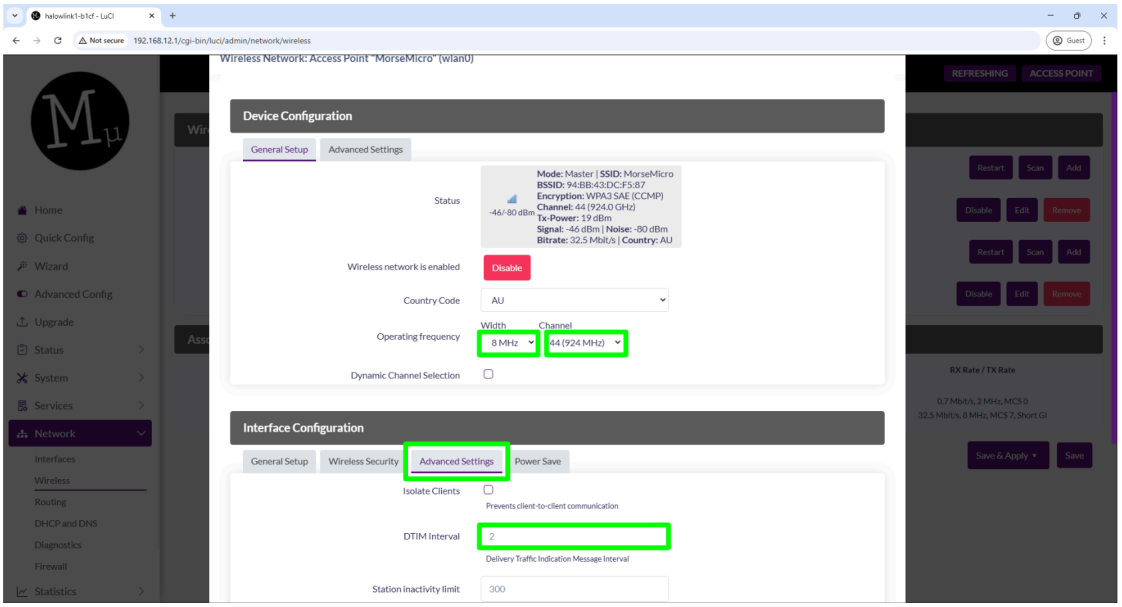


Figure 10. Advanced setup showing where to set DTIM interval, bandwidth, and channel.

4.3 Software Examples

The following table will describe the list of supported software examples that are included with the CMSIS pack for the MM8108-EKH05. A limited version of the "EKH05 Demo" example is loaded on the device out of the box. Instructions for flashing the full functionality version can be found in section 4.4 Loading Full EKH05 Demo.

Software Example	Description
EKH05 Demo	The EKH05 Demo showcases the operation of all on-board peripherals of the MM8108-EKH05. A HTTP server runs on the device and displays images from the camera, temperature, humidity and accelerometer values. The BLE module is also activated and acts as a sensor. **Note:** A limited version of this example is loaded onto the device out of the box.
Ping	Simple ping demonstration. Also used to stay connected to an AP while idling to measure DTIM sleep currents.
WNM Sleep	Example utilizing WNM sleep to conserve power in between periodic transmissions.
iPerf	Example showing how to perform throughput measurements using iPerf.
AWS IoT	AWS IoT example to demonstrate connecting to AWS Shadow service and demonstrate a simple light bulb example.
Scan	Example application to demonstrate scanning for Wi-Fi HaLow networks.
CLI	Example application communicating with the host through a UART connection. Can control Wi-Fi parameters, scan, connect, initiate ping, iPerf, sleep mechanisms and more.

4.4 Loading Full EKH05 Demo Firmware

The MM8108-EKH05 out-of-box demo application does not include RF functionality for regulatory adherence. For quick programming, pre-built binary files of the full EKH05 Demo application for each region (US, JP, EU, etc) are available via Morse Micro github or Morse Micro website download dashboard. Take caution to download the right binary file for your region.

Once downloaded, the firmware binary can be flashed easily to the device through the STM32CubeProgrammer software. This can be downloaded from the following link - https://www.st.com/en/development-tools/stm32cubeprog.html.

With STM32CubeProgrammer software running, connect the provided USB-micro cable to the EKH05 and your PC. In STM32CubeProgrammer, click the **Connect** button to attach to the device.

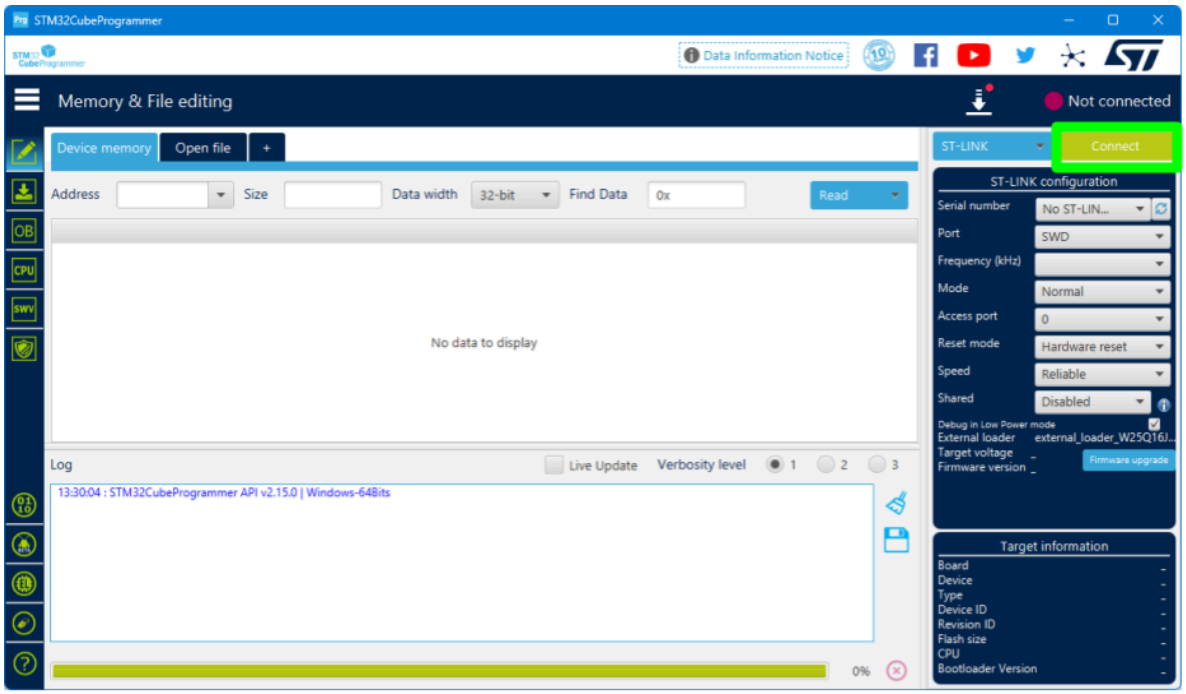


Figure 11. Initial screen of the STM32CubeProgrammer.

Once connected, click on the **Erasing & Programming** button on the left hand panel.

From this screen, browse to the downloaded EKH05_Demo elf file on your PC, turn on "Verify programming" and "Run after programming" and click "Start programming".

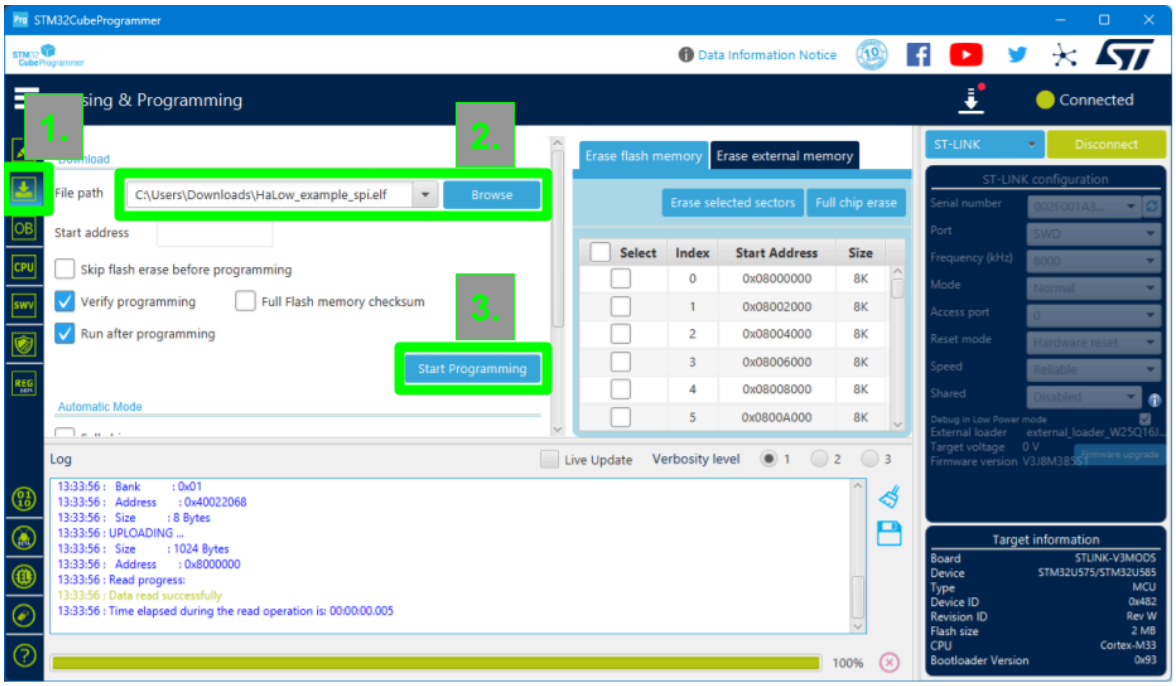


Figure 12. STM32CubeProgrammer - How to select and flash firmware binary.

A pop-up window will indicate if the operation was successful, and the log will display the following.

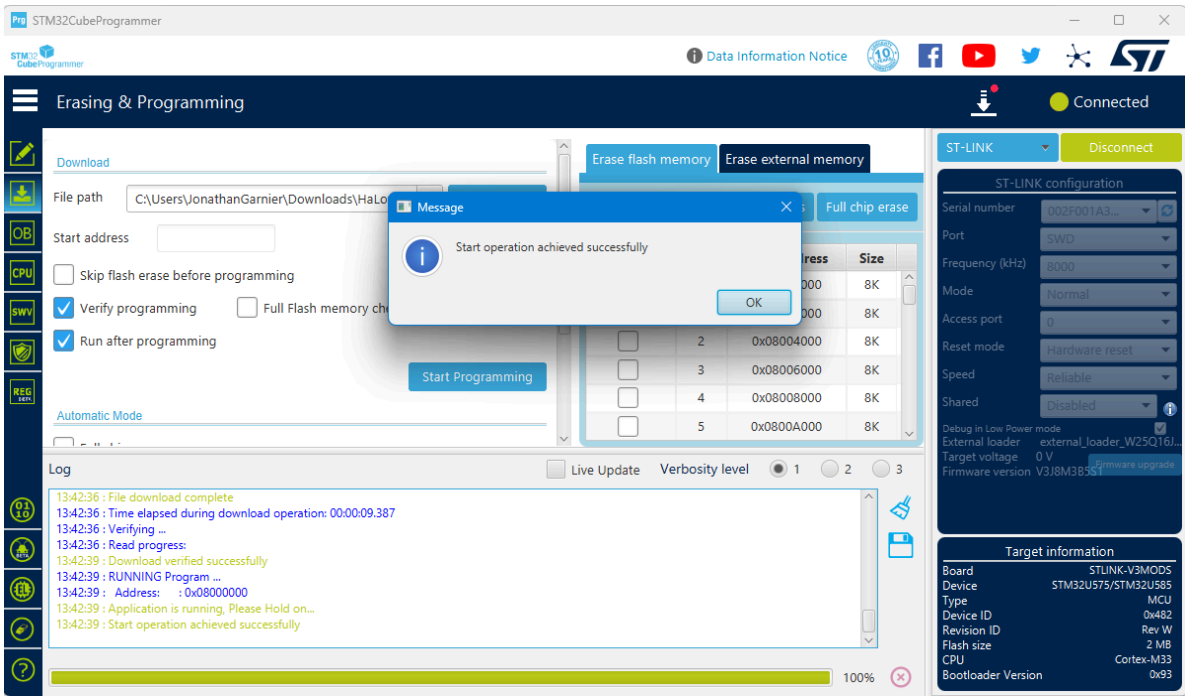


Figure 13. Successful flashing of firmware using STM32CubeProgrammer.

4.5 Viewing MM8108-EKH05 Demo HTTP Server

To view the HTTP server running on the MM8108-EKH05 device from a PC, the IP address of the MM8108-EKH05 needs to be found. To do this, navigate back to the **Home** section, and click on the **Connected Devices** link.

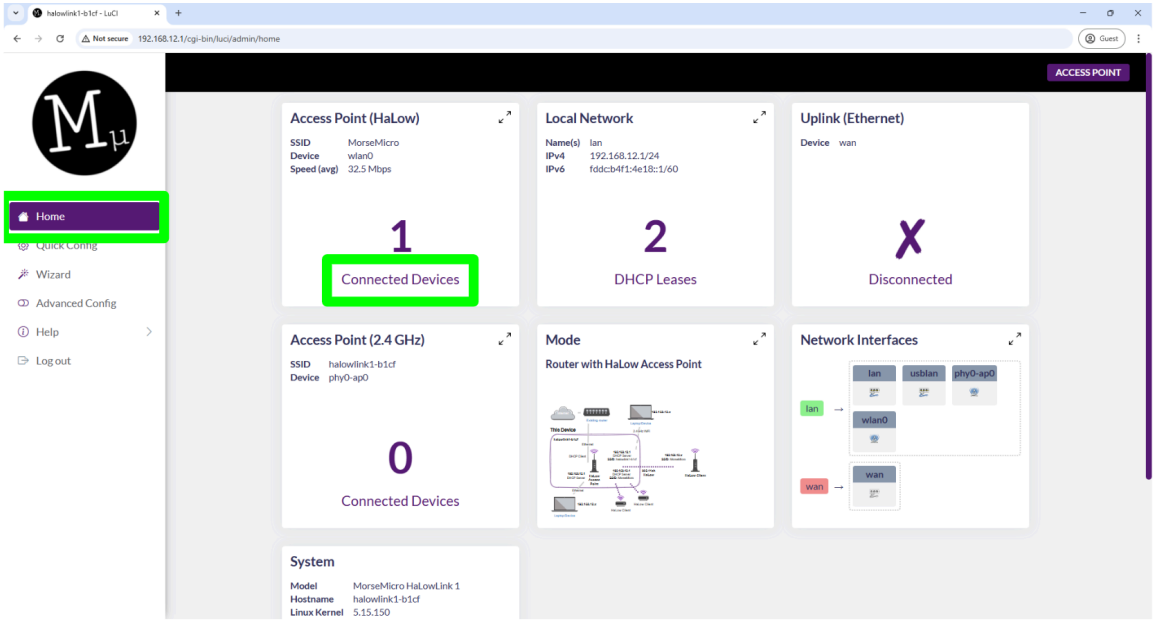


Figure 14. Home OpenWRT menu showing connected devices.

This will bring up a list of DHCP leases the AP has distributed to connected stations. In this case, the MM8108-EKH05 connected to the AP has an IP address of 192.168.12.225.

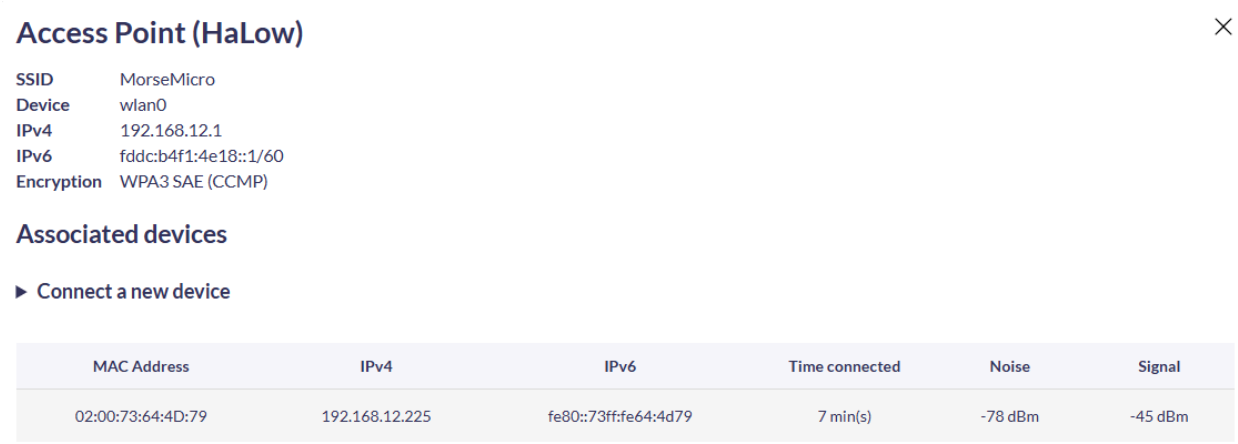


Figure 15. Connected devices and their connection details.

Navigate to the IP address of the MM8108-EKH05 (192.168.12.225 in this case) in a web browser to view the contents of the HTTP server.

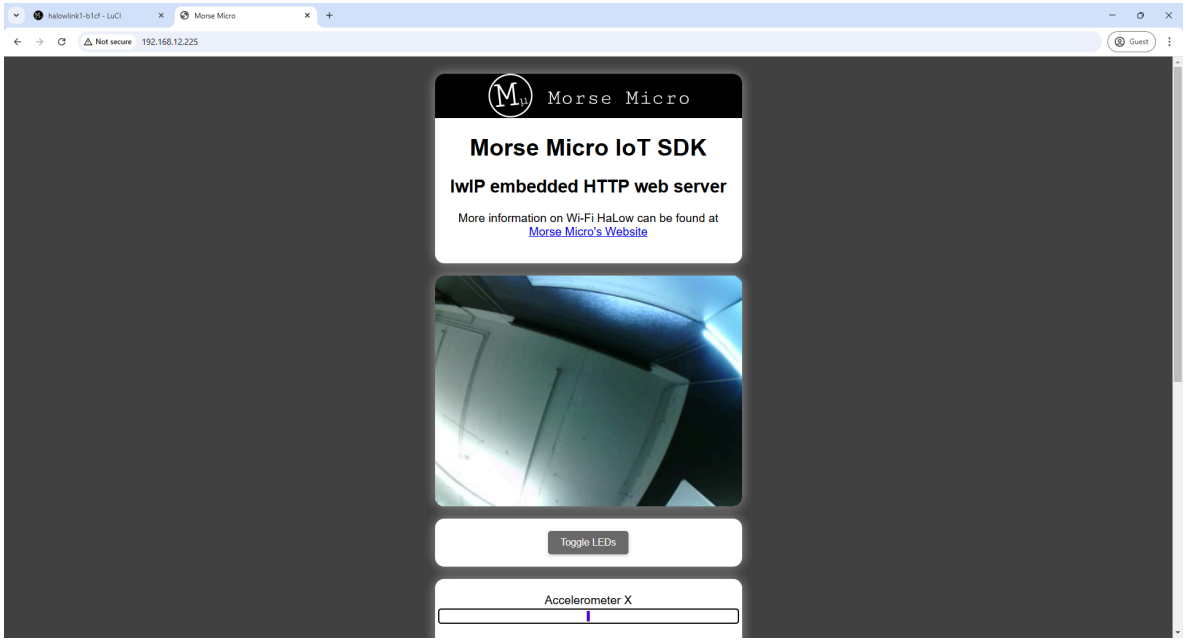


Figure 16. HTTP server running on MM8108-EKH05 demo application.

5 Software Development

This section will describe how to install the CMSIS package, how to build and run Morse Micro example applications within the STM32CubeIDE and how to update the firmware on the BLUENRG-M2SP module.

STM32CubeIDE can be downloaded from the ST website - https://www.st.com/en/development-tools/stm32cubeide.html.

The CMSIS package can be downloaded from the Morse Micro GitHub page - https://github.com/MorseMicro/mm-iot-cmsis/releases.

5.1 Installing CMSIS Package

After Starting STM32CubeIDE, click on **Manage Embedded Software Packages** from the **Help** menu:

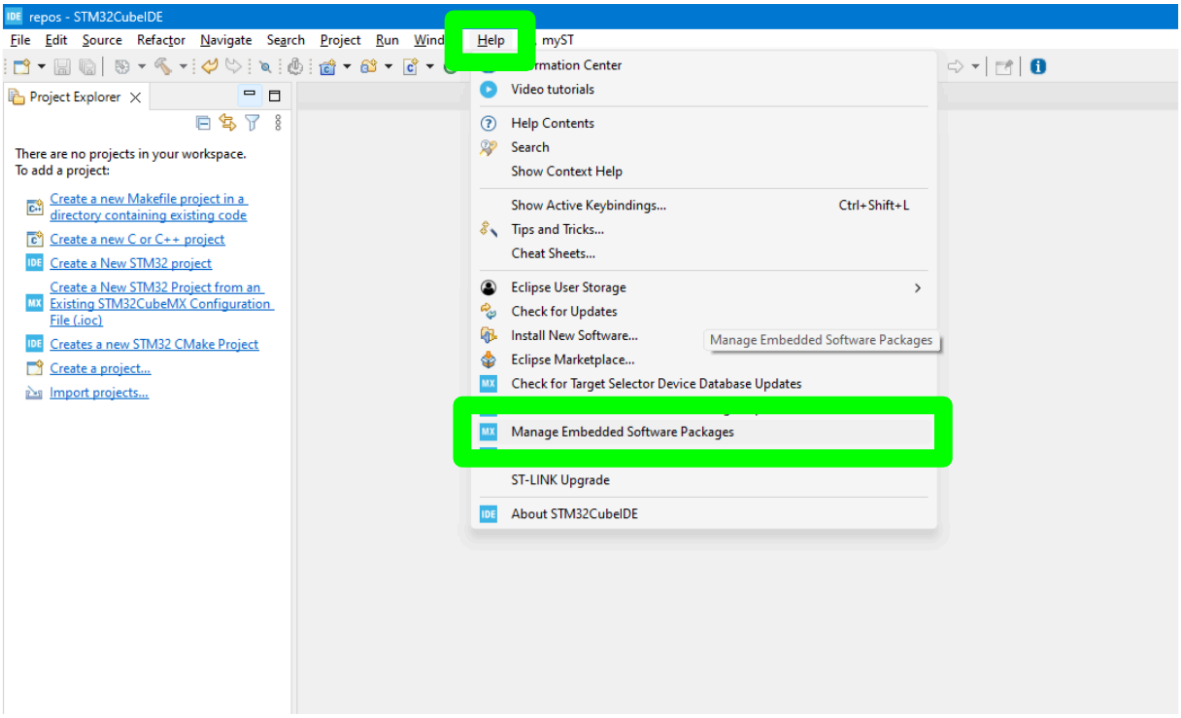


Figure 17. STM32CubeIDE help menu dropdown.

Click on **From Local ...**:

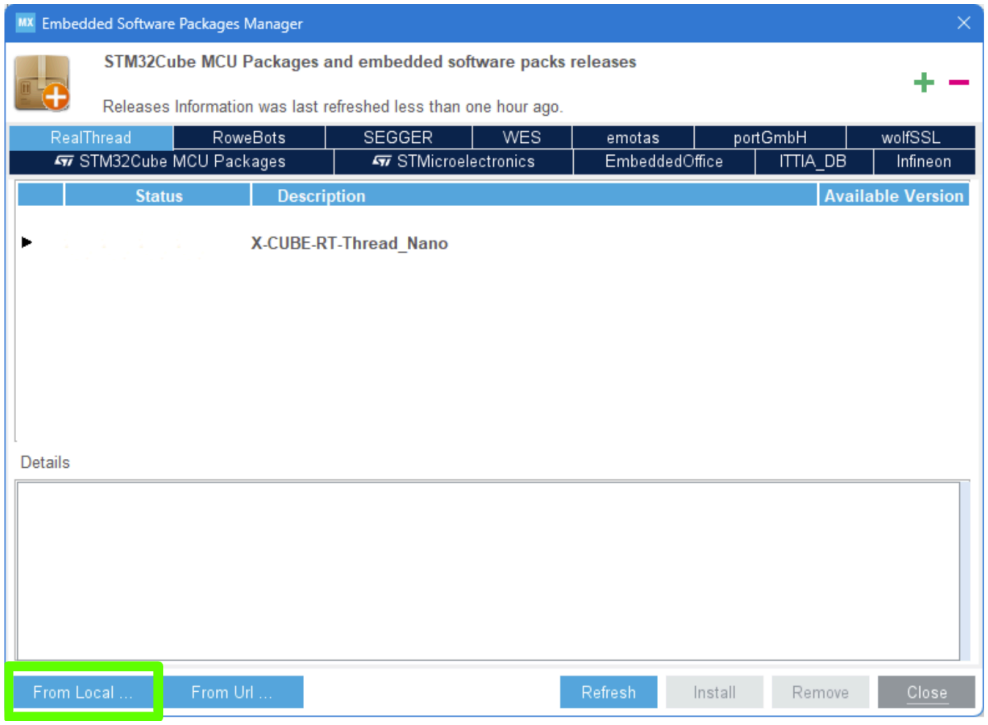


Figure 18. Embedded software packages manager menu.

Navigate to where the Morse Micro CMSIS package is downloaded, and click **Open**:

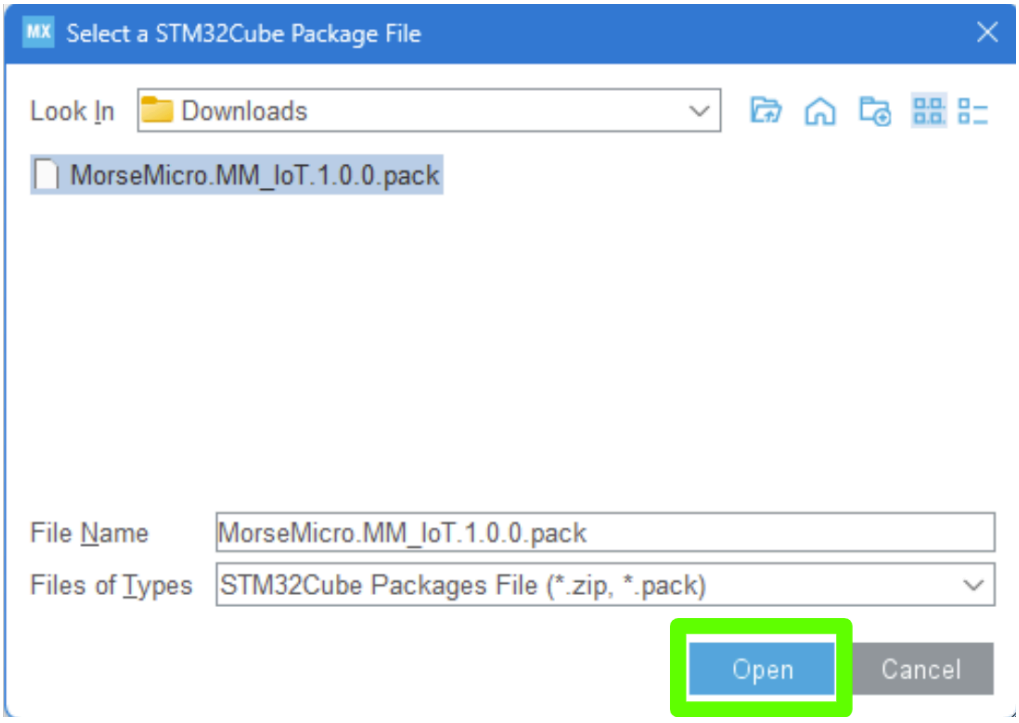


Figure 19. Opening CMSIS pack from downloaded location.

Accept the license agreement, and click **Finish**:

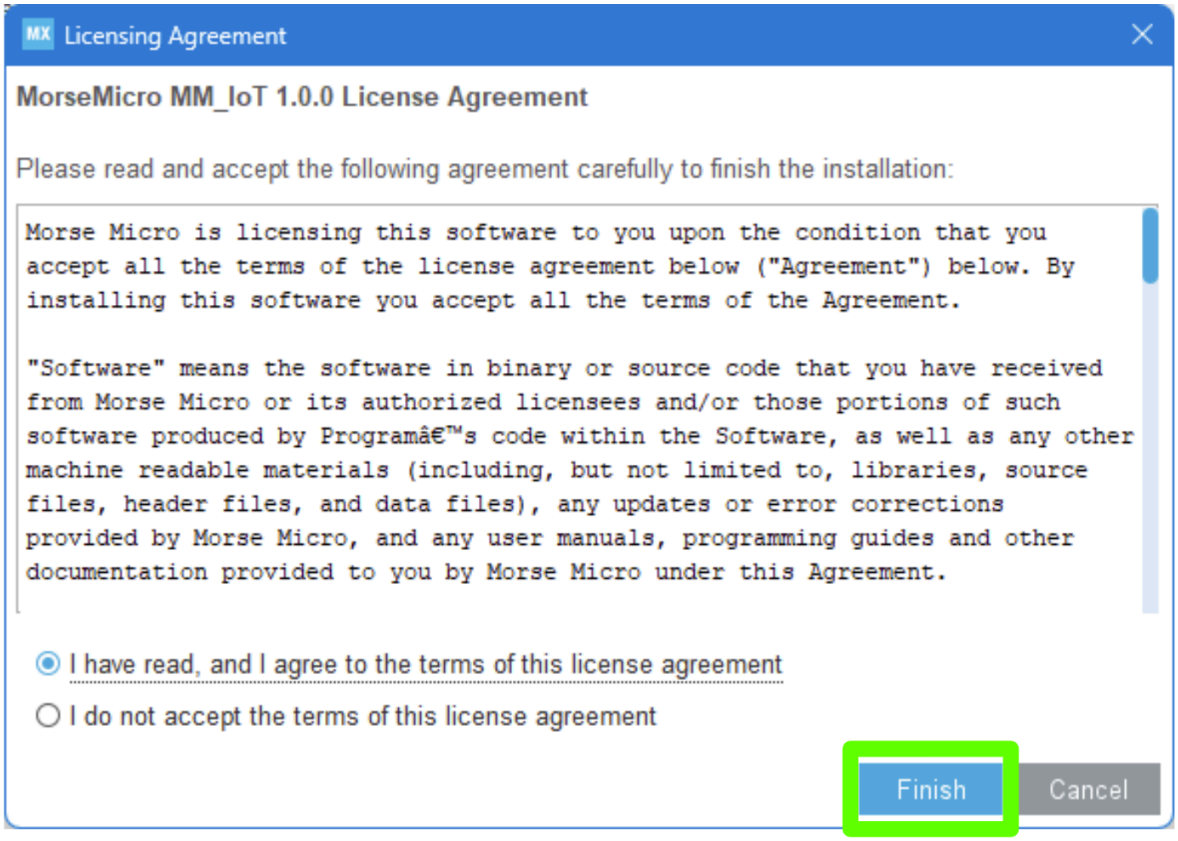


Figure 20. License agreement menu for installing CMSIS pack.

The package is now installed.

5.2 Build and Run Example Applications

After installing the Morse Micro CMSIS package, an example application can be run. As an example, this will run the “HaLow Example” application.

First, create and log into a myST account. This is needed to install other packages included within the HaLow example.

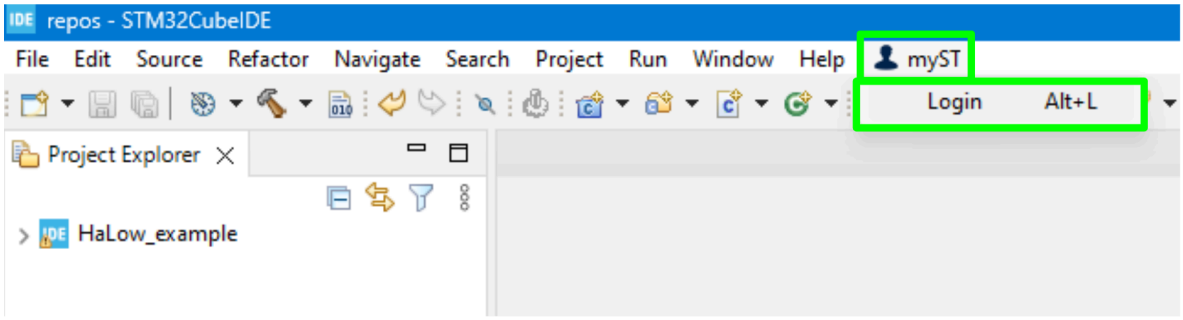


Figure 21. Where to login to ST account in STM32CubeIDE.

From **File**, click **Open Projects from File System**:

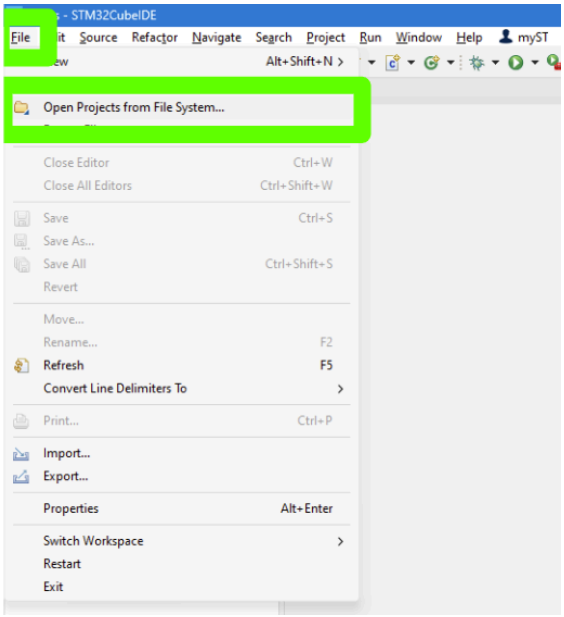


Figure 22. Where to open project in STM32CubeIDE.

Click on **Directory...**:

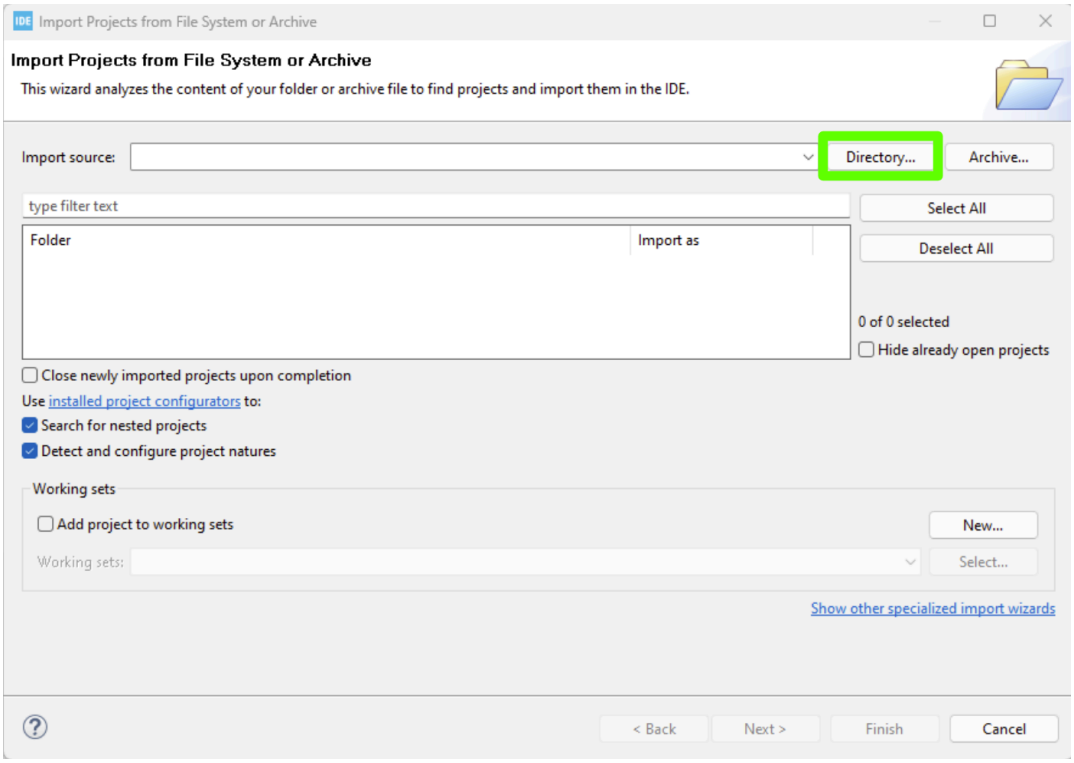


Figure 23. Highlighting directory button to search for project.

Navigate to the default STM32 package location, and then find the HaLow_example application. Either HaLow_example_spi or HaLow_example_sdio can be chosen to use SPI or SDIO respectively. SPI will be chosen for the rest of this document.

On Windows, the default path is:

```
C:\Users\USER_NAME\STM32Cube\Repository\Packs\MorseMicro\MM_IoT\1.6.0\HaLow_example_spi
```

On Linux, the default path is:

```
/home/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.6.0/HaLow_example_spi
```

On Mac, the default path is:

```
/Users/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.6.0/HaLow_example_spi
```

Note that the version number may change depending on which version of the CMSIS pack is downloaded.

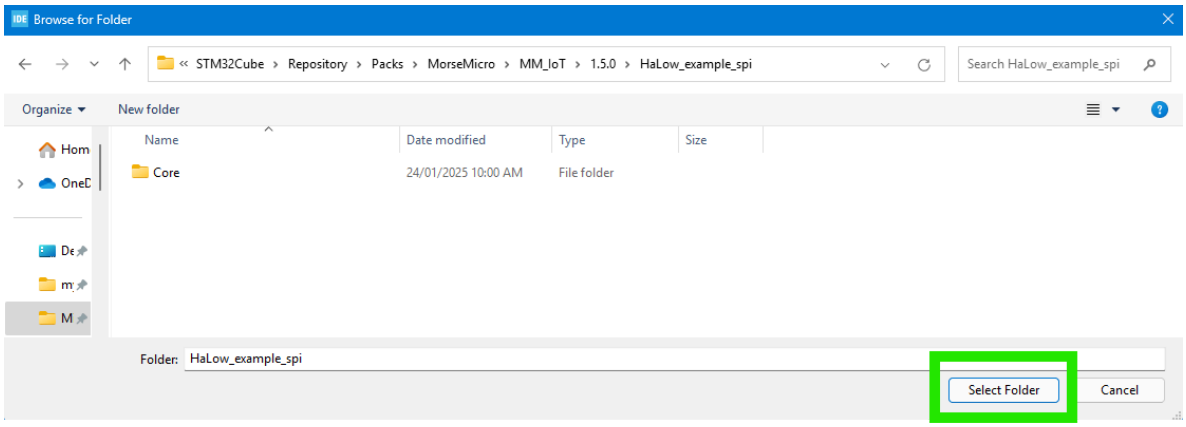


Figure 24. Example location of where to find the example application on a windows machine.

Click **Finish**:

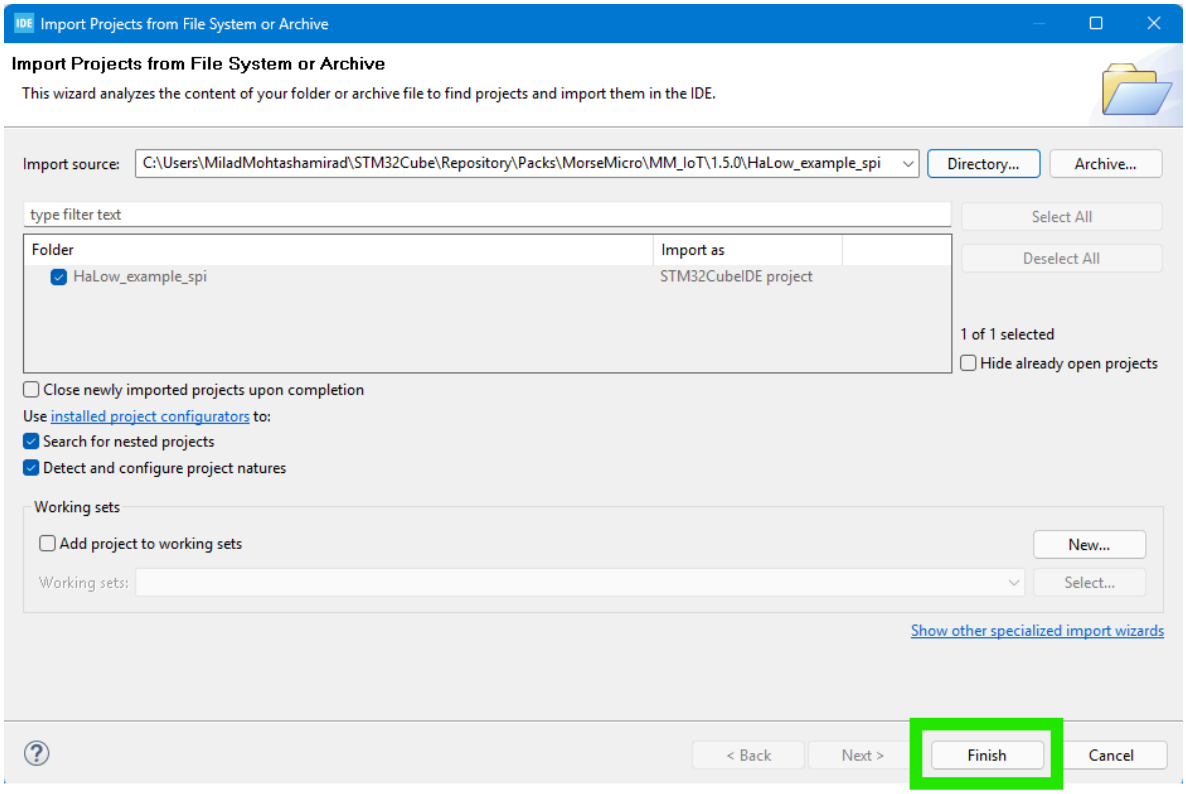


Figure 25. Finish menu importing the example application.

Expand the Halow_example, and double click on the .ioc file:

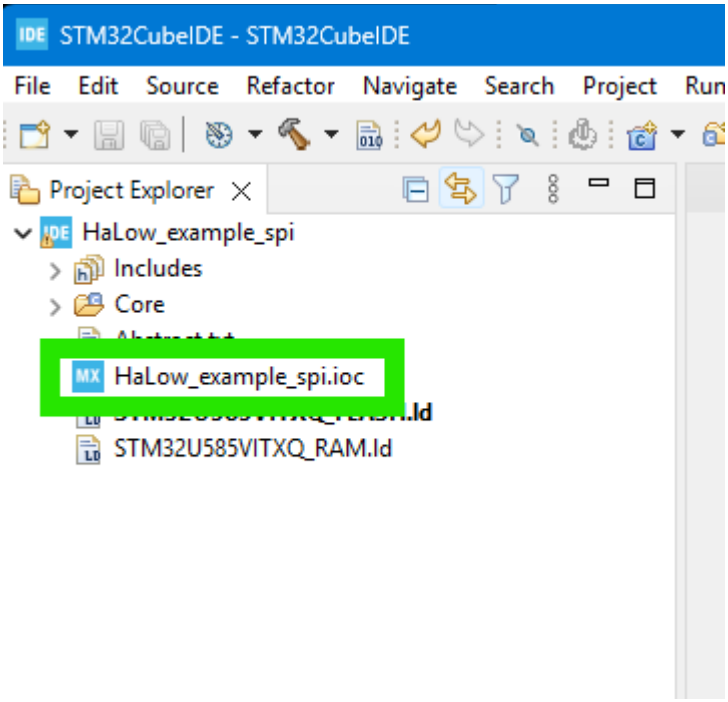


Figure 26. Highlighting .ioc file in project explorer.

Click **Continue**.

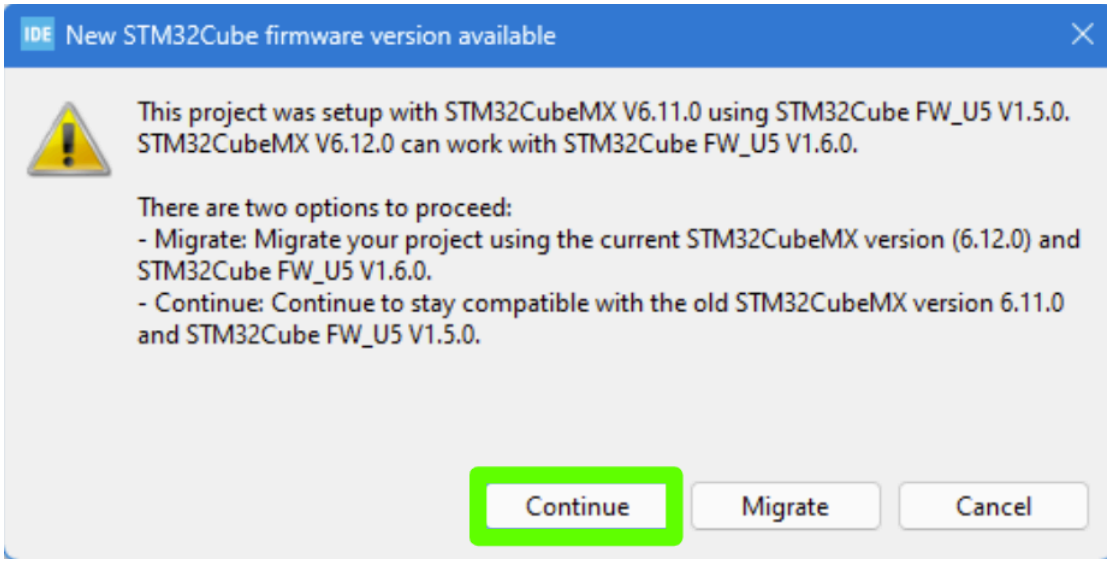


Figure 27. Pop-up menu after opening .ioc file.

There may be additional software packs that need to be installed including STMicroelectonics.X-CUBE-FREERTOS.1.2.0, and STMicroelectronics.X-CUBE-BLE2.3.3.0. Download these items in the subsequent pop-up windows.

Click **Select Components** from the **Software Packs** menu.

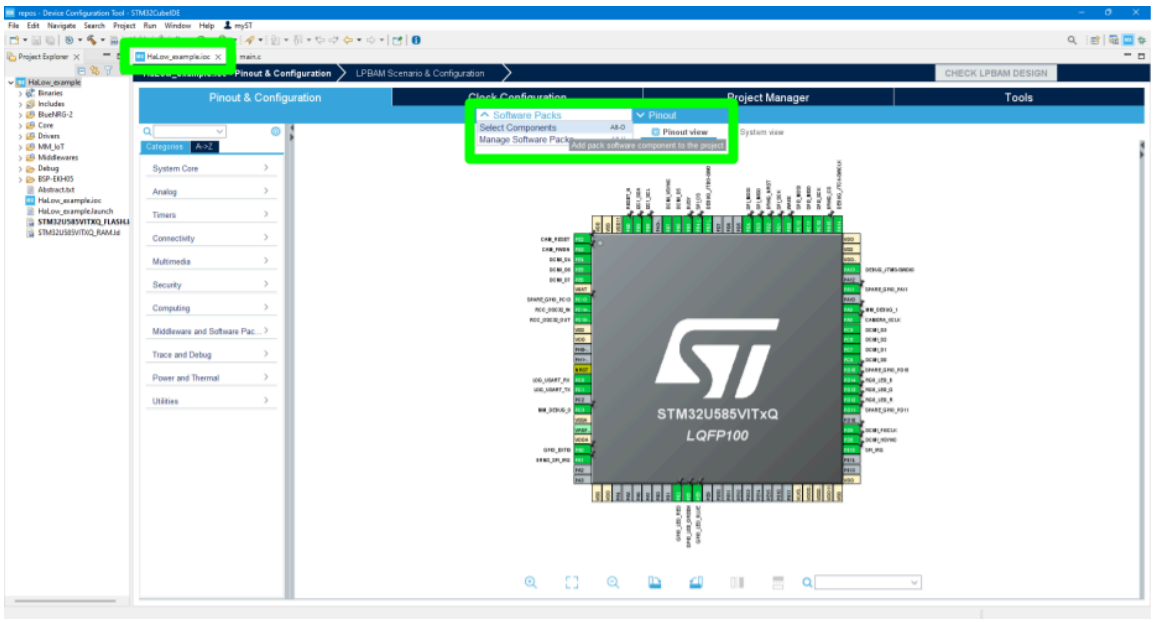


Figure 28. Select components menu in the .ioc file.

Expand **MorseMicro MM_IoT**, and then **MM_IoT**. From the **morselib** drop down box, select **MM8108**. Click **Ok**.

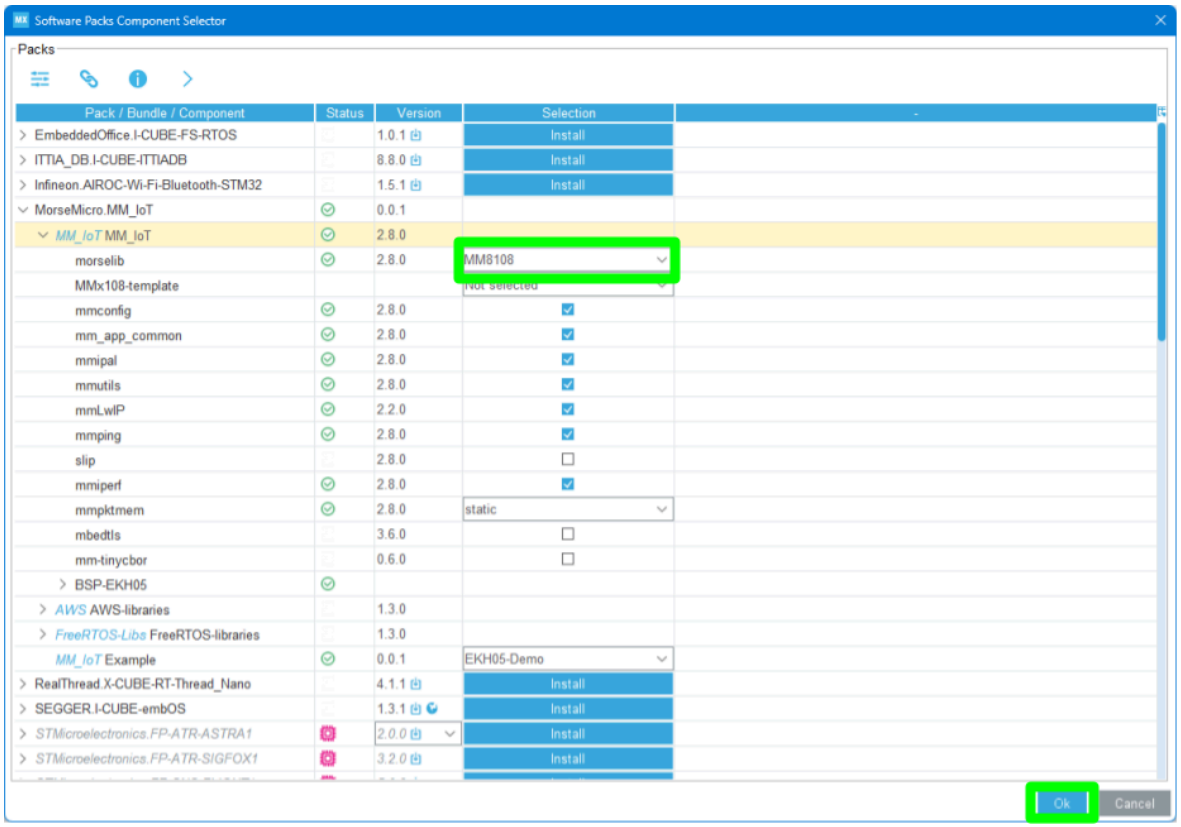


Figure 29. Select chip type in the components selector menu.

Open the **Core→Inc→main.h** file from the project explorer. Change the #define for **MM_CHIP** from **MM6108** to **MM8108**, and save the file.

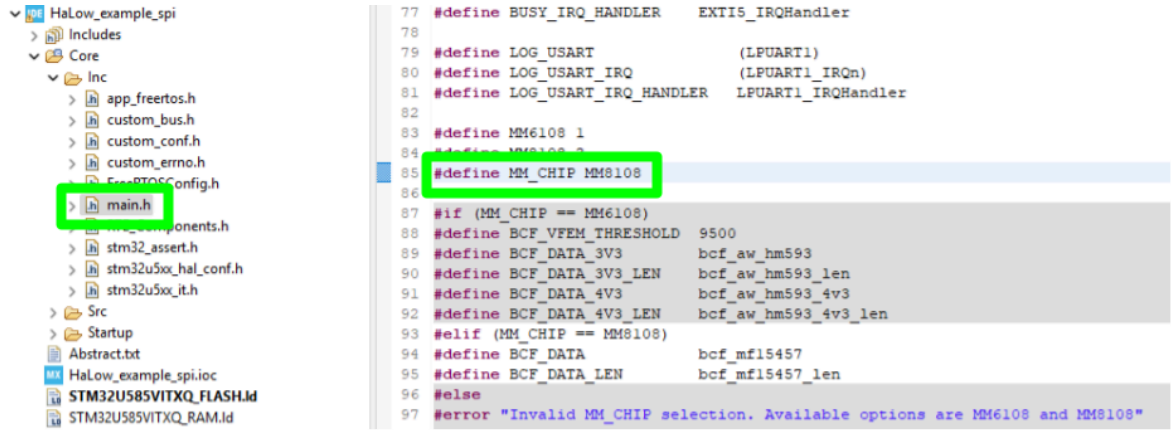


Figure 30. Select chip type in the main.h file.

Click **Project**, **Generate Code**. This may also start a download of the STM32U5 firmware if this is not downloaded already.

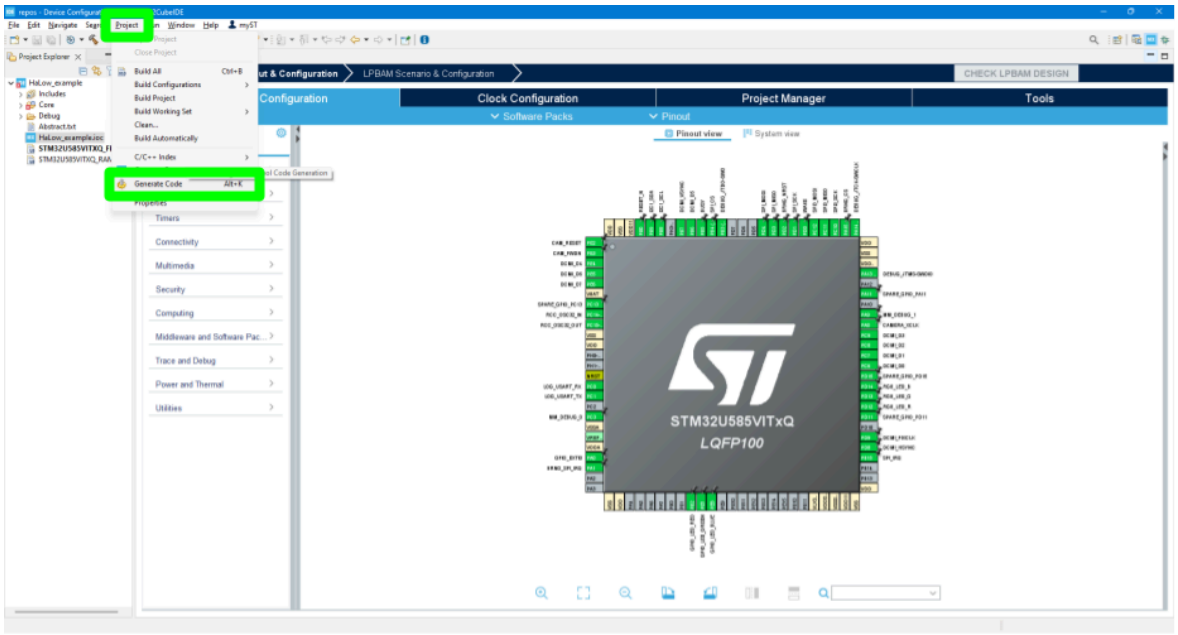


Figure 31. Highlighting where to generate code from after opening the .ioc file.

The example can then be built by clicking the hammer icon in the toolbar. If successful, there should be no errors in the Console window:

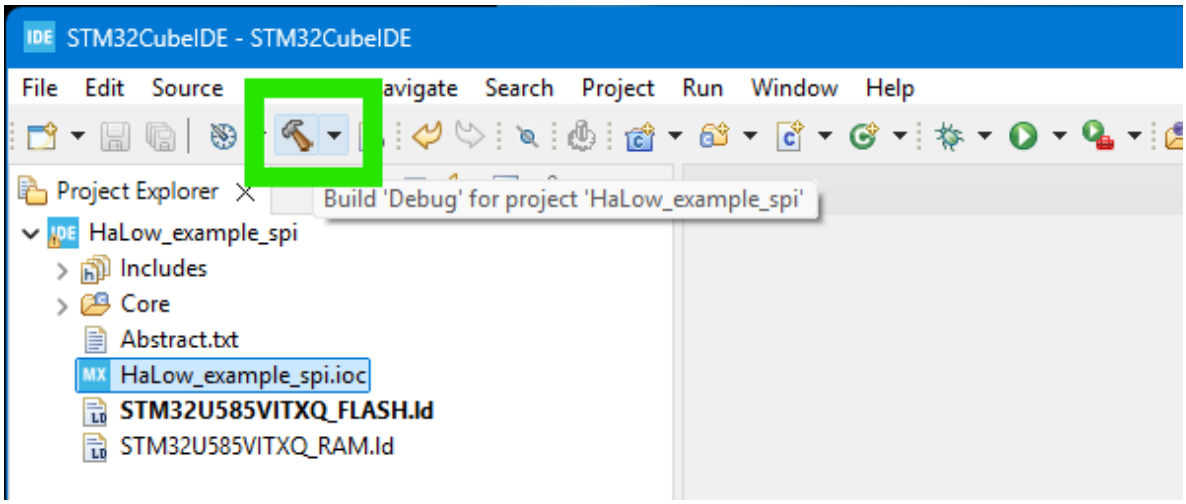


Figure 32. Icon in STM32CubeIDE for building application.

The program can now be run by clicking on the green Run icon:

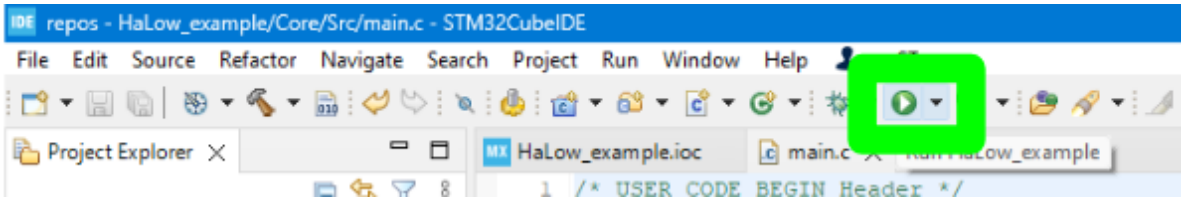


Figure 33. Icon in STM32CubeIDE for running application.

This will then bring up a window for the Run Configuration. All settings can be left as default. Click **Ok**:

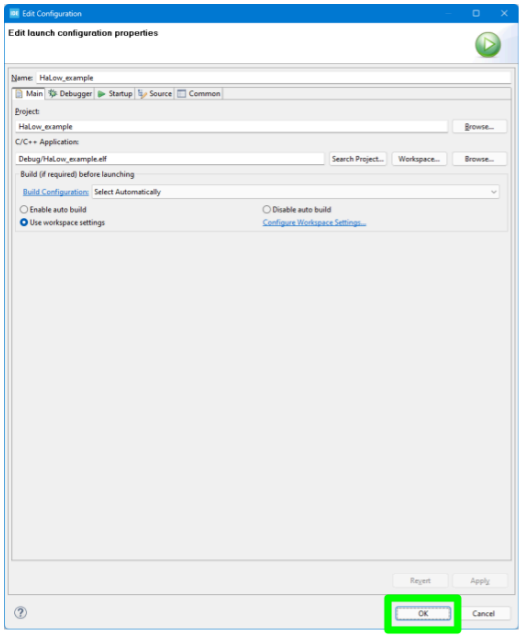


Figure 34. Run configuration menu.

The example program will now build again if not built already, and then be flashed to the device.

5.2.1 UART Output

The UART output can be monitored by opening a serial monitor with the following settings:

Item	Setting
Baud Rate	115200
Hardware Flow Control	None
Data Bits	8
Stop Bits	1
Parity	None

5.2.1.1 Windows

The STLINK-V3MODS will show up as a Virtual COM port on windows machines.

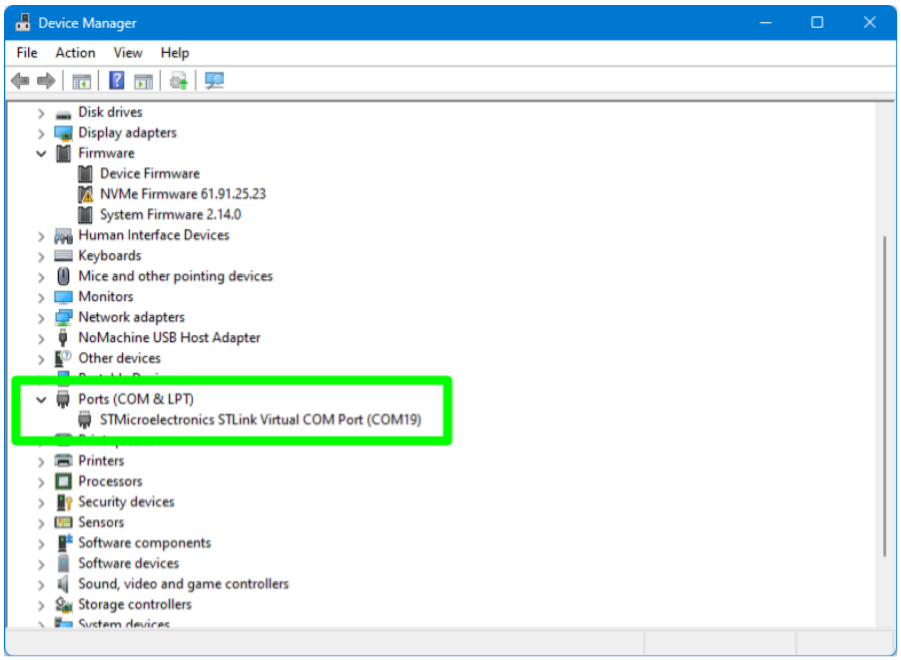


Figure 35. Device manager on windows machine showing where the STLINK-V3 COM port is located.

This COM port can then be opened with tools such as PuTTY.

```
putty -serial COM10 -sercfg 115200
```

5.2.1.2 Linux

On Linux machines, the device can be detected with the following command:

```
ls /dev/serial/by-id/usb-STMicroelectronics_STLINK-V3_*
```

And a serial terminal can be opened with tools such as minicom:

```
minicom -D /dev/serial/by-id/usb-STMicroelectronics_STLINK-V3_*
-b 115200
```

5.2.1.3 Mac OS

On Mac machines, the device can be detected with the following command:

```
ls /dev/tty.*
```

And similarly, a tool such as minicom can be used to open the serial terminal:

```
minicom -D /dev/tty<YOUR-DEVICE-HERE> -b 115200
```

5.3 Changing Example Application

To change the example application, open the HaLow_example.ioc file and click **Select Components** from the **Software Packs** menu.

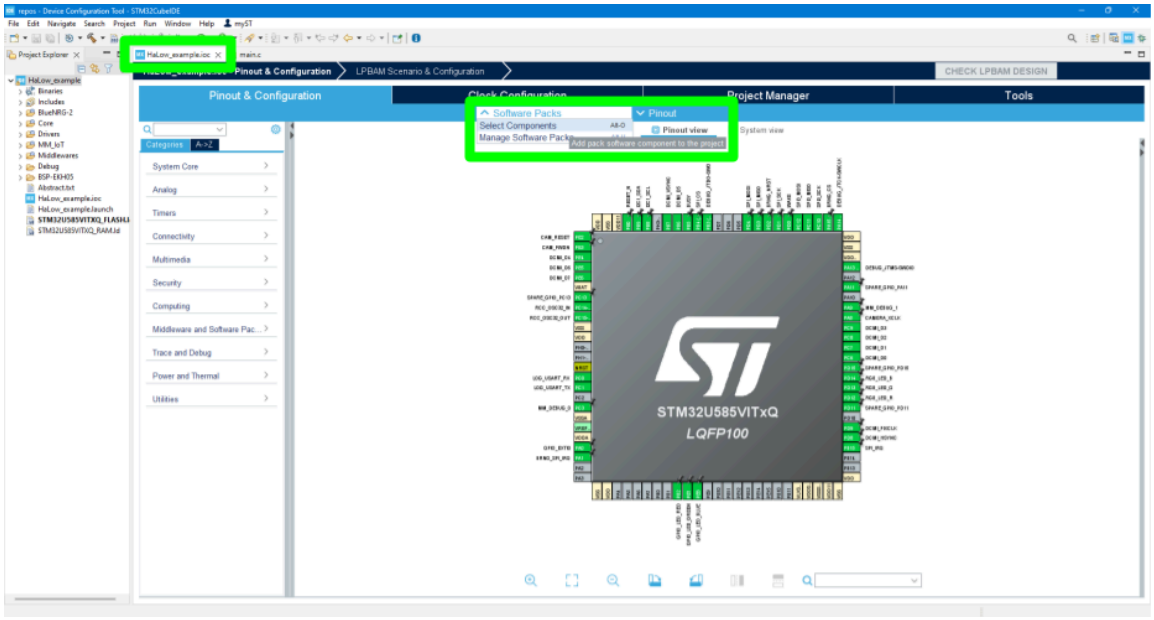


Figure 36. Select components menu in the .ioc file.

Click on the MorseMicro.MM_IoT arrow, and then select the desired application from the drop down list. Please note that the HTTP server is only available on the EKH05-Demo application. Changing to a different application will mean the user will be unable to connect to the IP address of the MM8108-EKH05 in a web browser.

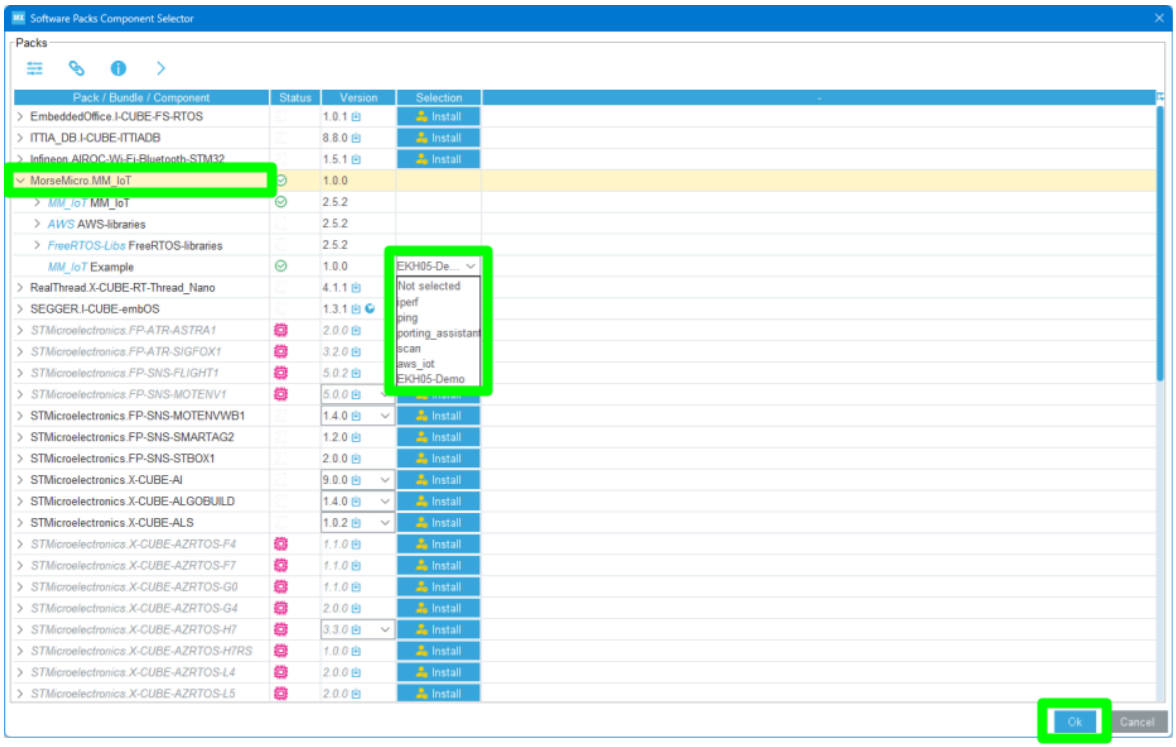


Figure 37. List of example applications within the CMSIS pack.

Click **Generate Code** from the **Project** menu again. The **Run** button can then be clicked again to build and run the new example.

5.3.1 rf-test Example Considerations

In case of selecting the rf-test application, a global define for DISABLE_UART_LOG is needed. To do so, open the project properties from **Project->Properties:**

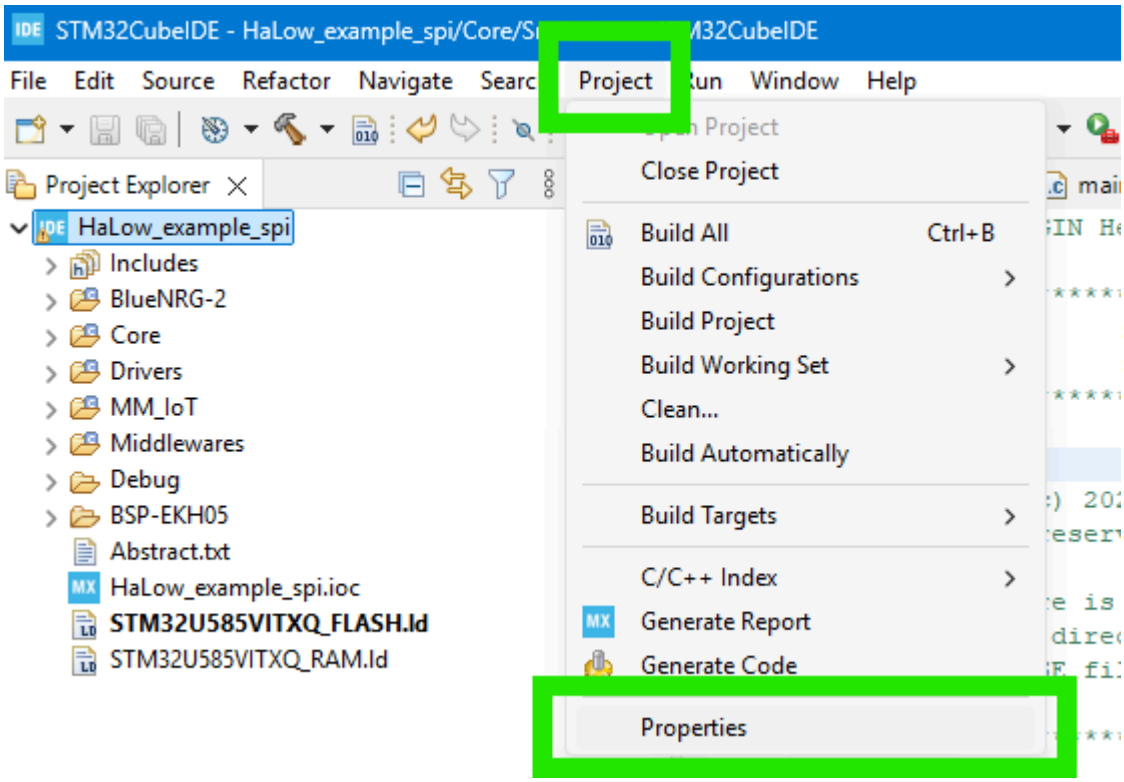


Figure 38. Project properties menu in STM32CubeIDE.

From **C/C++ Build -> Settings** select **Tools Settings** tab and from **MCU GCC Compiler -> Preprocessor** click on **"Add..."** button. In the text box add "DISABLE_UART_LOG=1" and click OK:

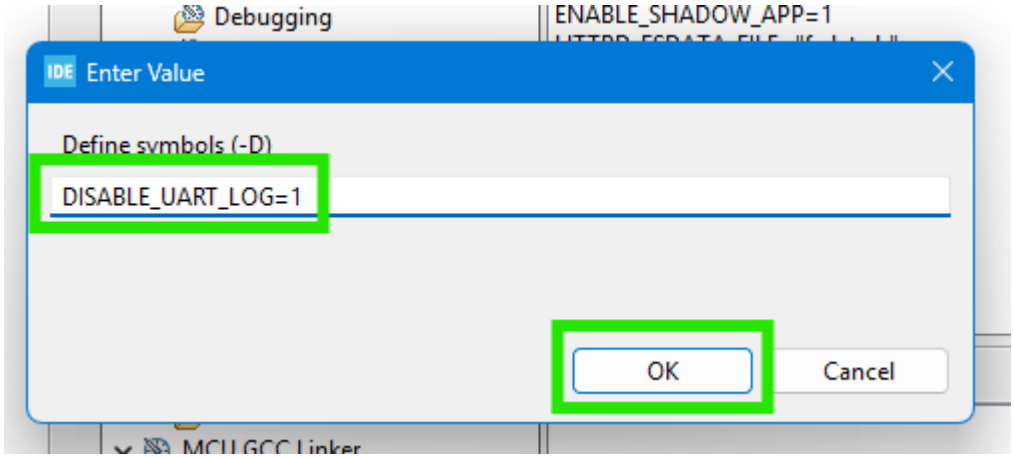


Figure 39. Adding the symbol necessary to build the rf-test application.

Click on **Apply and Close** and rebuild the project.

5.4 Changing Country Code

The country code parameter can be changed by navigating to the **Core → Src → configs.c** file. Search for "wlan.country_code". Replace the parameter and save the file. The changes will take effect upon the next build of the firmware.

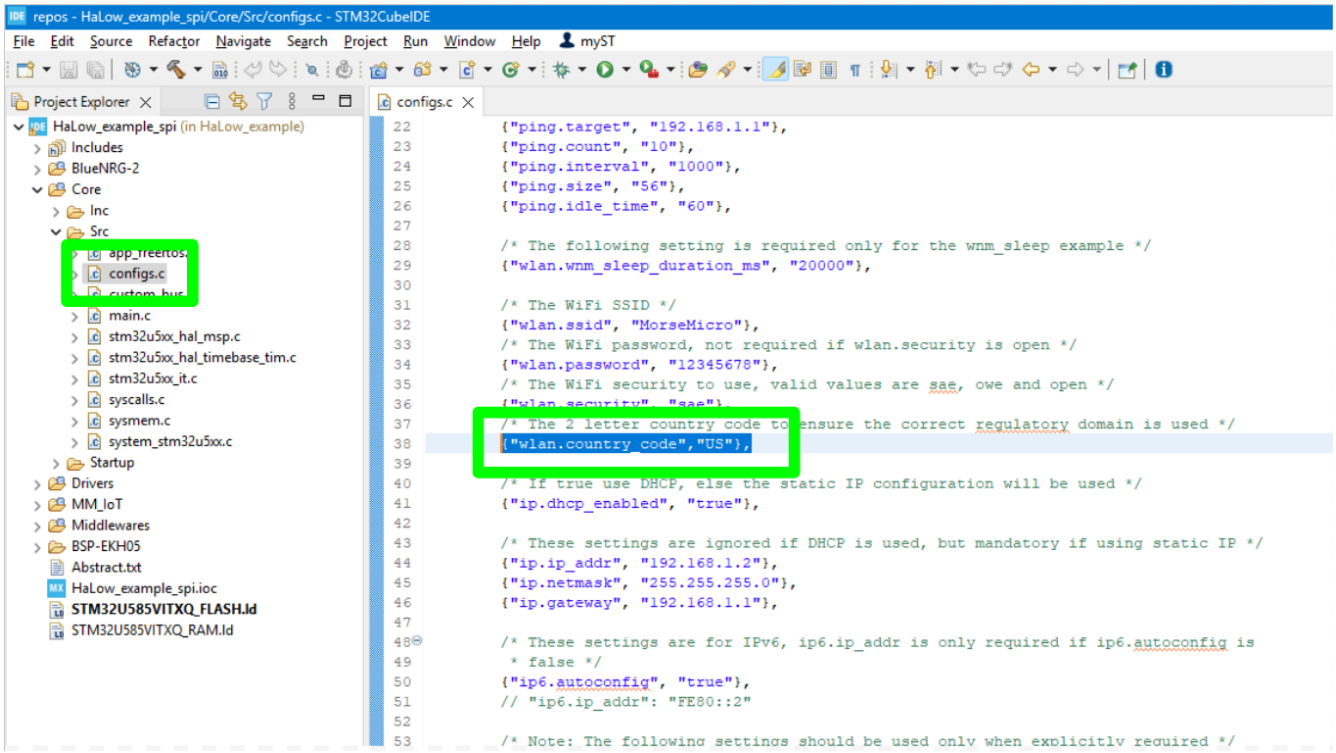


Figure 40. Source code location to change the country code.

5.5 Changing Other Example Configurations

Other parameters like SSID, passphrase, IP addresses, ping count etc can also be changed by navigating to the **Core → Src → configs.c** file. Replace the parameter and save the file. The changes will take effect upon the next build of the firmware.

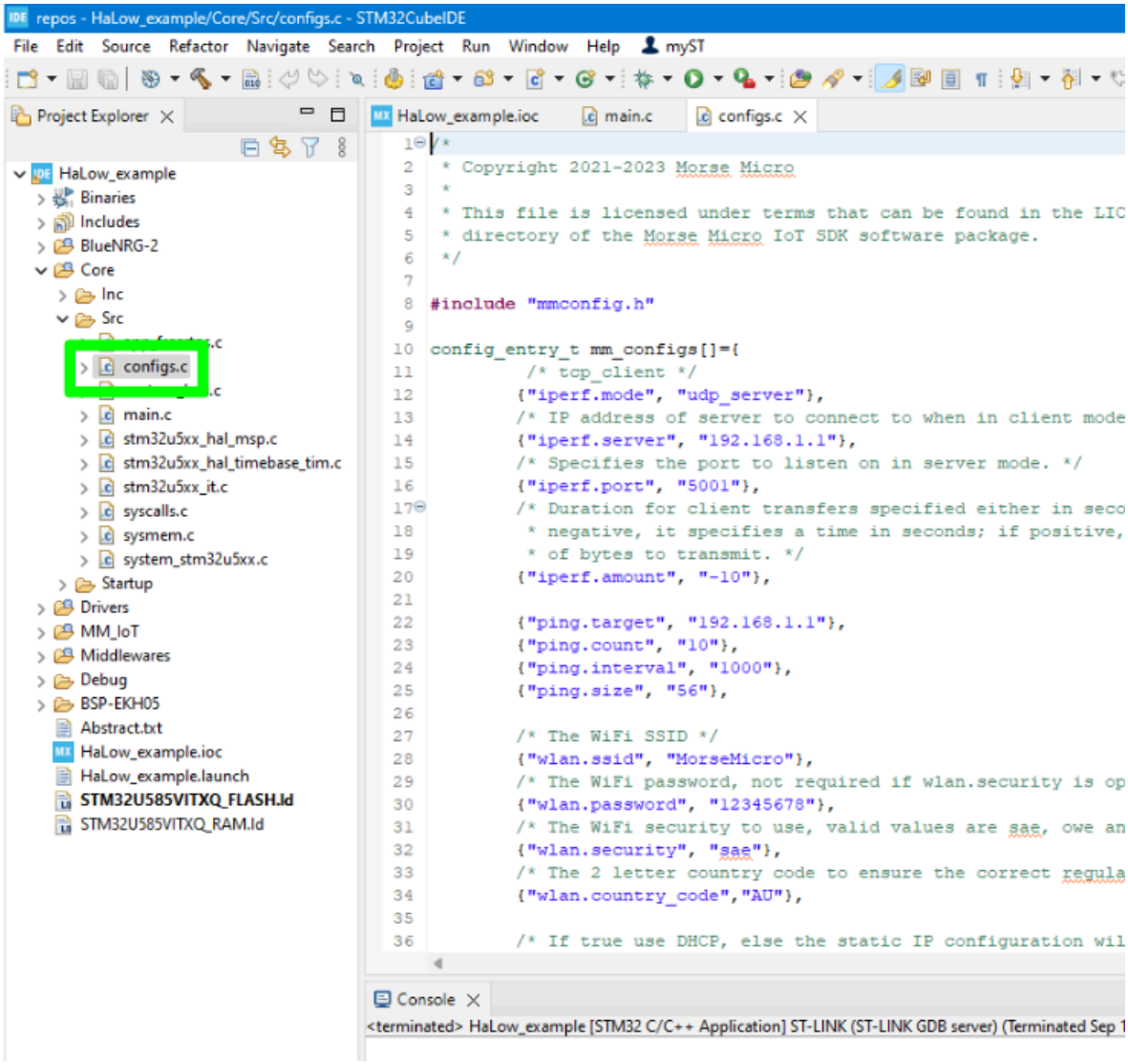


Figure 41. Showing all configurations available.

5.6 Changing Between SPI and SDIO

In order to change between using SPI and SDIO as the communication protocol between the STM32U585 and the HaLow module, a different project is needed. Either the HaLow_example_spi project or HaLow_example_sdio project can be selected, as shown in section 5.2. Note that hardware changes are also required to function properly. These changes are described in section 6.4.

5.7 Changing Network Stacks

This section will document how to change between using LwIP and FreeRTOS+TCP as the network stack. For almost all situations, LwIP should be used. FreeRTOS+TCP should only be used when validating the best low-power performance. Not all examples are compatible with FreeRTOS+TCP.

Open the HaLow_example.ioc file and click **Select Components** from the **Software Packs** menu.

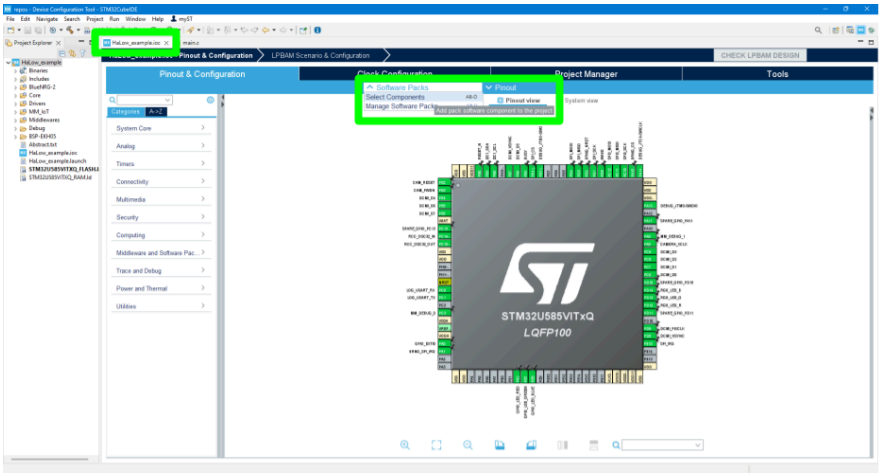


Figure 42. Select components menu in the .ioc file.

Select the **MM_IoT** and **FreeRTOS-Libs** drop down menus. Select the desired network stack and ensure the other is deselected. There should be all green ticks if there are no issues.

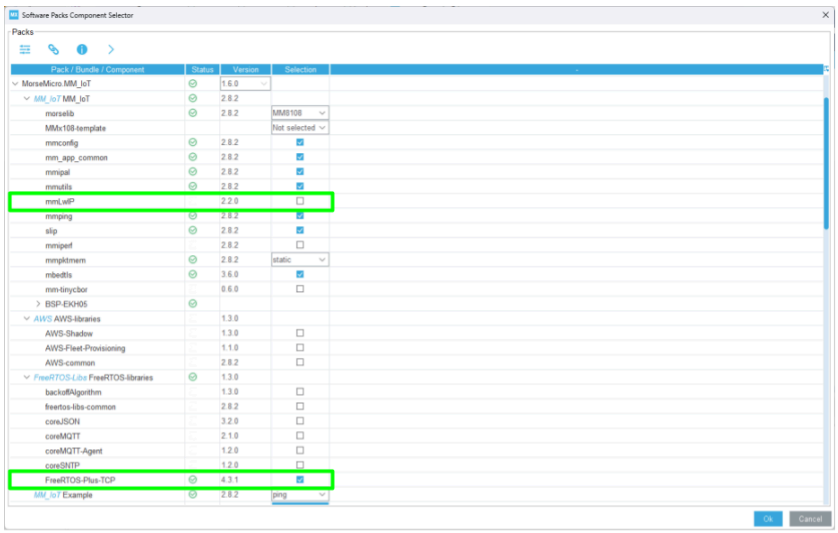


Figure 43. Unchecking the mmLwIP box, checking the FreeRTOS-Plus-TCP box.

Next, click the **Middleware and Software Packs** menu from the .ioc file. Select **MM_IoT** and ensure the **FreeRTOS-Libs FreeeRTOS-libraries** checkbox is ticked.

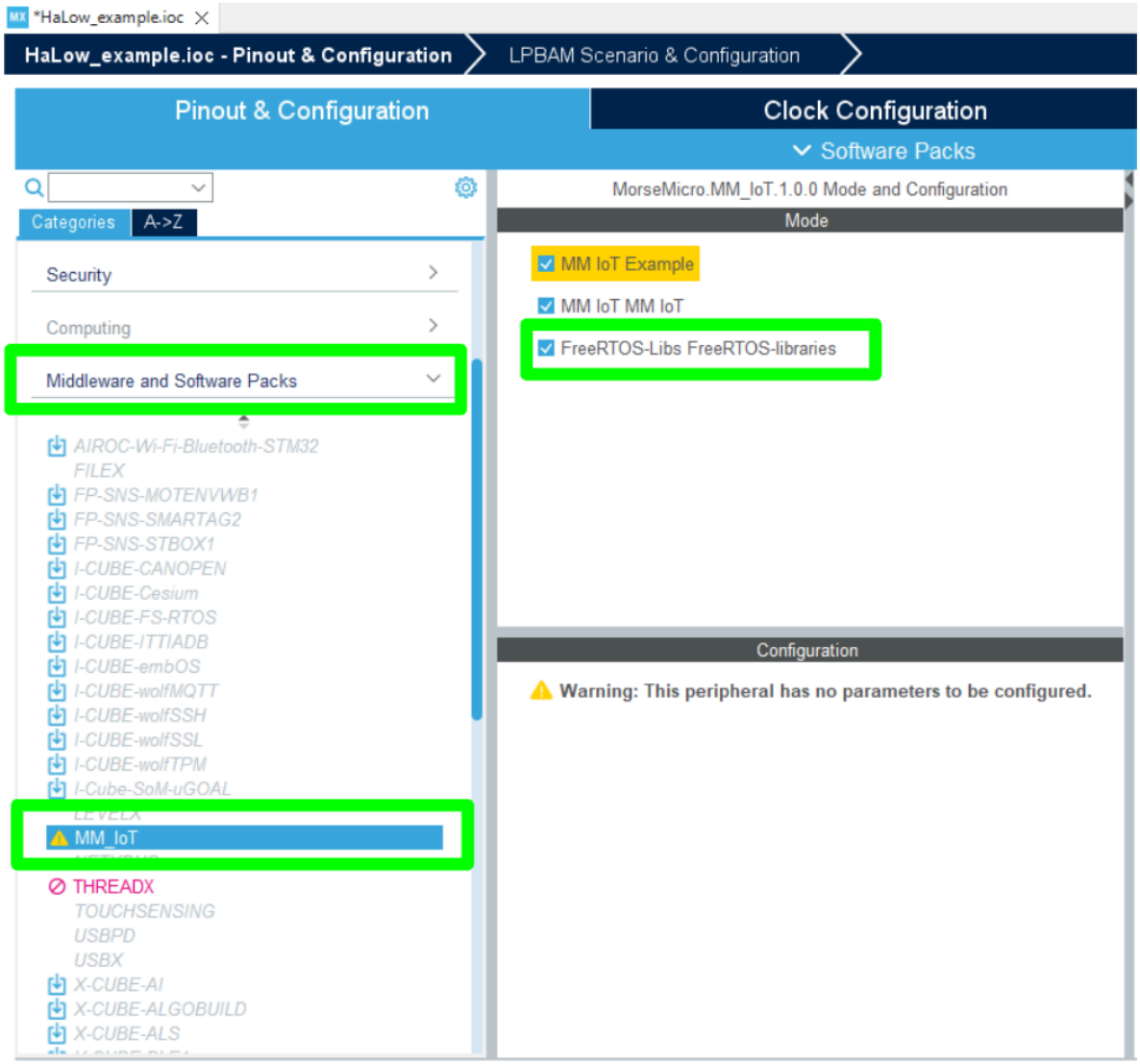


Figure 44. Middleware and software packs menu to add FreeRTOS libraries.

The code will need to be regenerated, and the firmware rebuilt and uploaded to the device for the changes to take effect.

5.8 Updating BLUENRG-M2SP Module Firmware

The most straightforward way to flash the firmware on the BLUENRG-M2SP module is to use the on board STLINK V3. The SWCLK and SWDIO jumpers can be removed, and wires connecting to the BLE SWD headers can be attached as shown below.

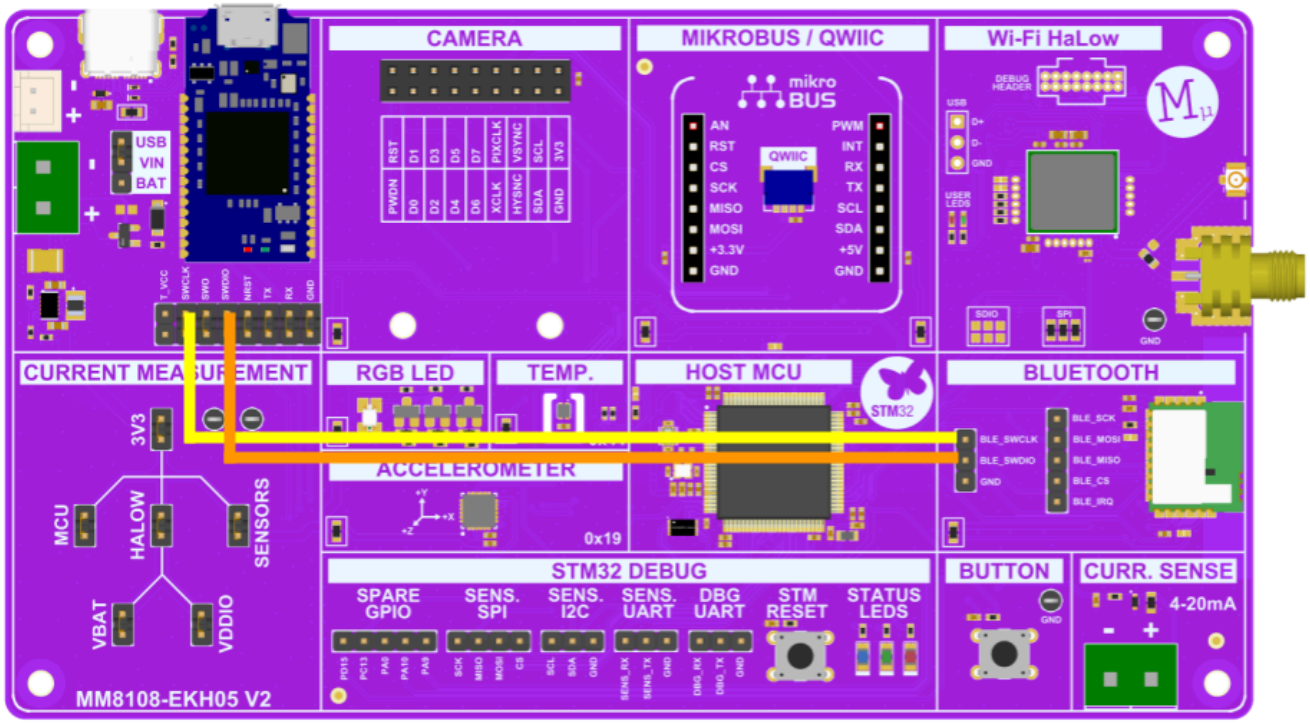


Figure 45. Connecting the STLINK V3 SWCLK and SWDIO pins to the BLUENRG-M2SP module.

Ensure the wires are attached to the top pins of the header as these are connected to the STLINK V3.

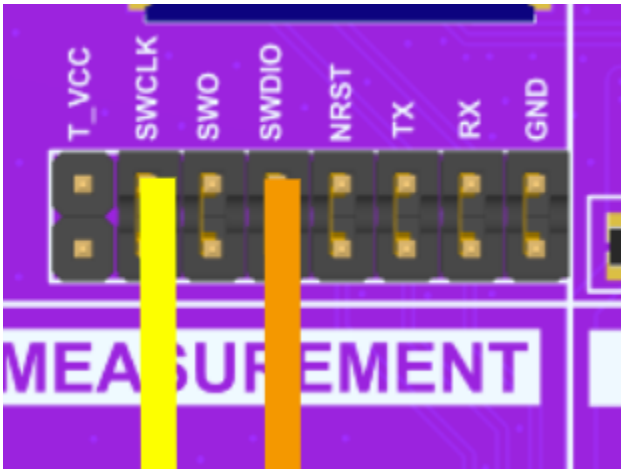


Figure 46. Close up view of connections, wires must be placed on top pins of the header.

Before doing this, the BLE RESET line must be held high by the STM32U585. Please ensure that pin PD2 is configured as an output, and is set to high. The BLUENRG-M2SP module will be held in reset if this is not done, and will not be able to be programmed.

The next step is to download the RF-Flasher Utility from ST. This can be found here - https://www.st.com/en/embedded-software/stsw-bnrgflasher.html

Once installed, open the program and do the following:

1. Click on the **SWD** tab.
2. Select your connected STLINK device
3. Click **Select Image File**. Select the firmware image file to load.
4. Click **Flash**.

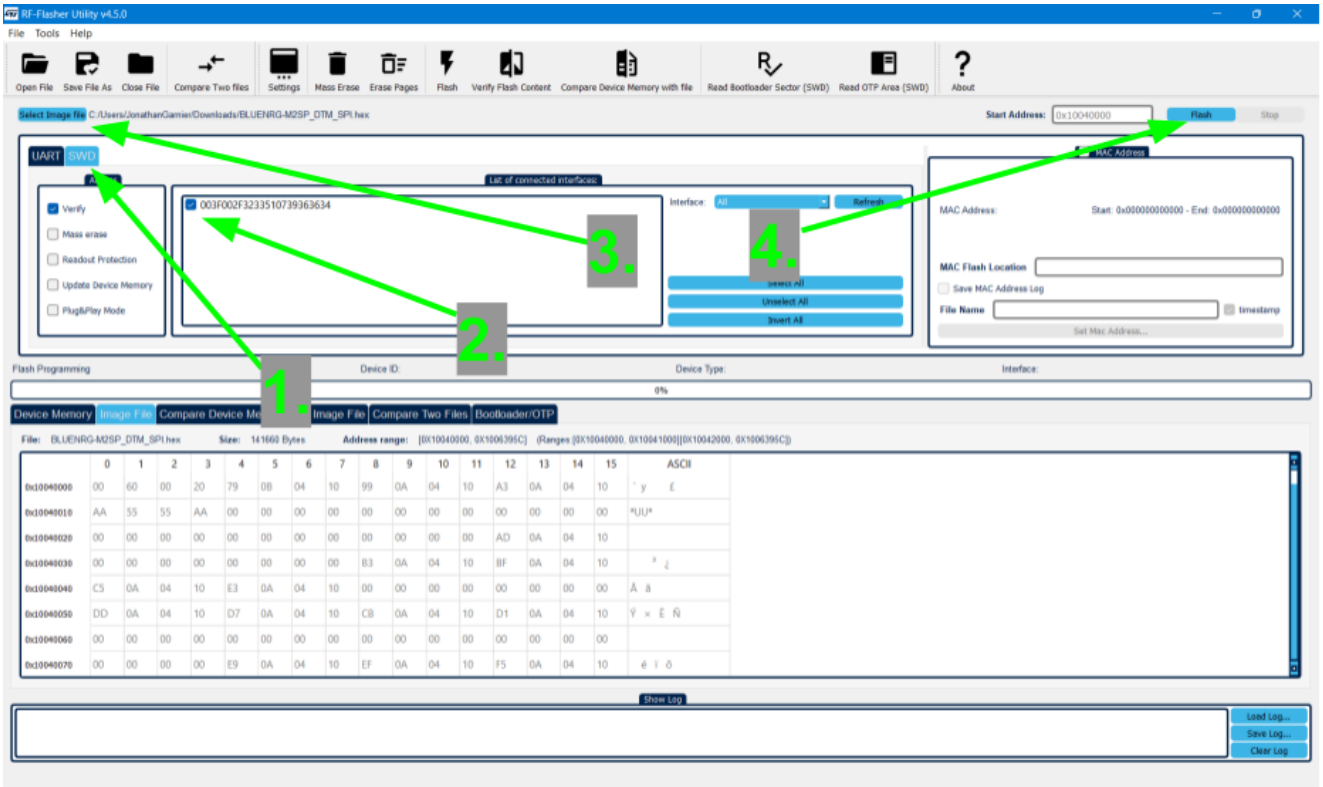


Figure 47. Steps in the RF-Flasher utility to program new firmware to the BLUENRG-M2SP module.

To verify this operation was successful, click the **Verify Flash Content** button. A popup window will appear notifying if the memory content matches the file or not.

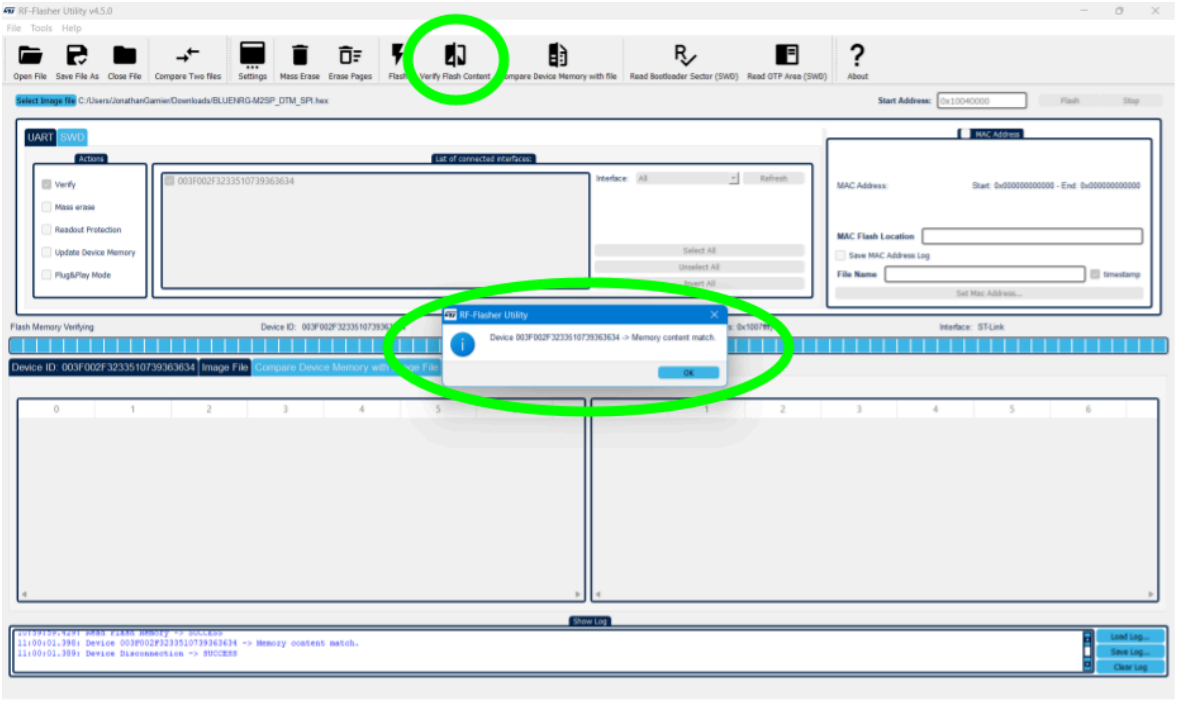


Figure 48. Verify firmware flash was successful.

6 Hardware Layout and Configuration

6.1 Power Selection

The voltage supply input to the buck-boost regulators can come from multiple sources. If the jumper is placed on the USB and VIN header, the supply can come from either the USB-C or STLINK Micro USB connectors. If the jumper is placed on the BAT and VIN header, the supply will come from the battery connections.

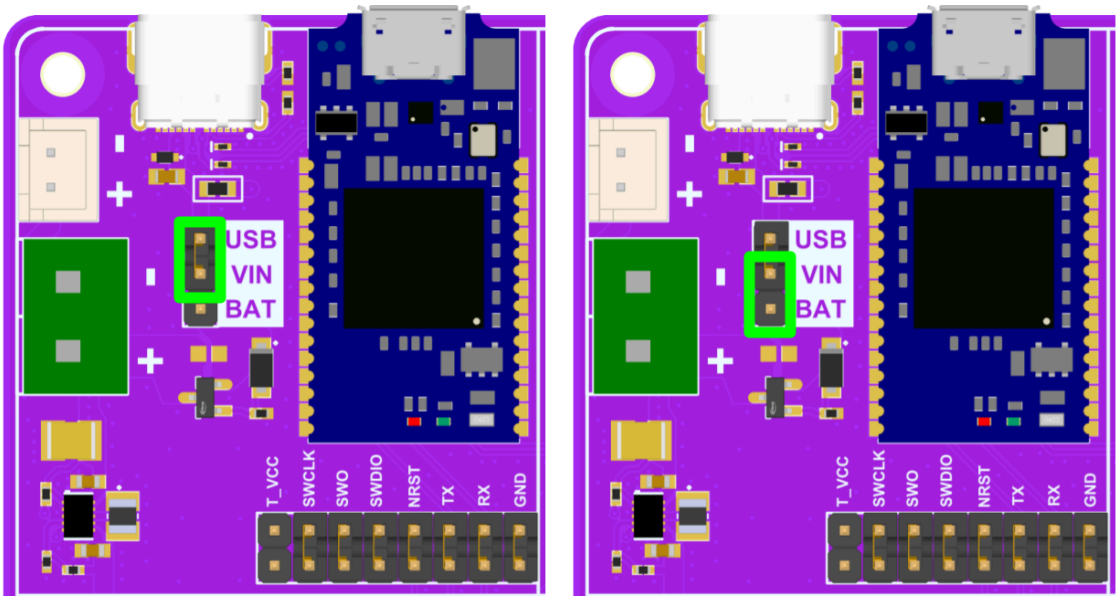


Figure 50. Changing power input between 5V USB on left (from USB-C or STLINK-V3) and battery inputs on right (JST connector or battery screw terminal).

6.2 Using an External Debugger/Programmer

The STLINK V3 MODS is a modular in-circuit debugger and programmer for STM32 and STM8 microcontrollers. It programs the device through a Single Wire Debug (SWD) interface, and connects to a UART interface on the STM32U585 microcontroller in the one cable.

Developers can opt to use their own external debug tool if desired. The jumpers below the module can be removed, and the bottom side of the headers (circled in green) can be attached to with an external debug tool to connect to the relevant pins on the STM32U585 microcontroller. The T_VCC header (circled in red) connects to 3.3V. The target voltage pin of an external debug tool can be attached here.

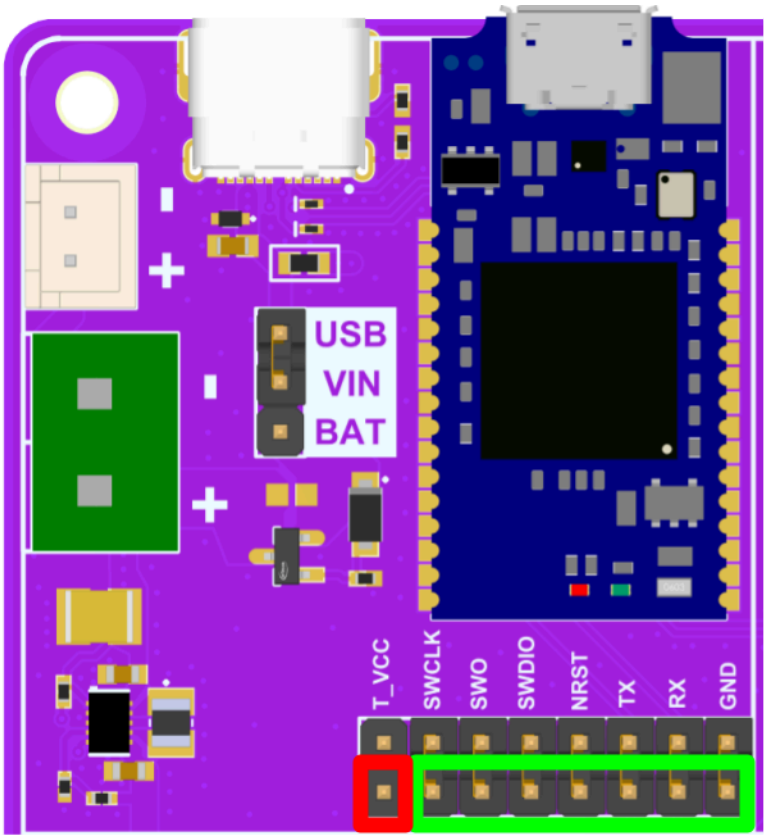


Figure 51. How to attach an external programmer.

6.3 Switching Between SDIO and SPI

The MM8108-EKH05 gives users the option to interface the STM32U585 Microcontroller with the HaLow module either through SDIO or SPI. SPI is the default configuration. If using SDIO, 47 kOhm pull-up resistors must be populated as pointed out by the arrow below. For SPI, these pull up resistors are not needed for the protocol and will cause **extra leakage current** on the VDDIO rail if not removed. The pull up resistor on the SD_D2 line **must be populated**.

A different pin configuration in the software is also needed when changing between the two. Section 5.6 details how to configure this.

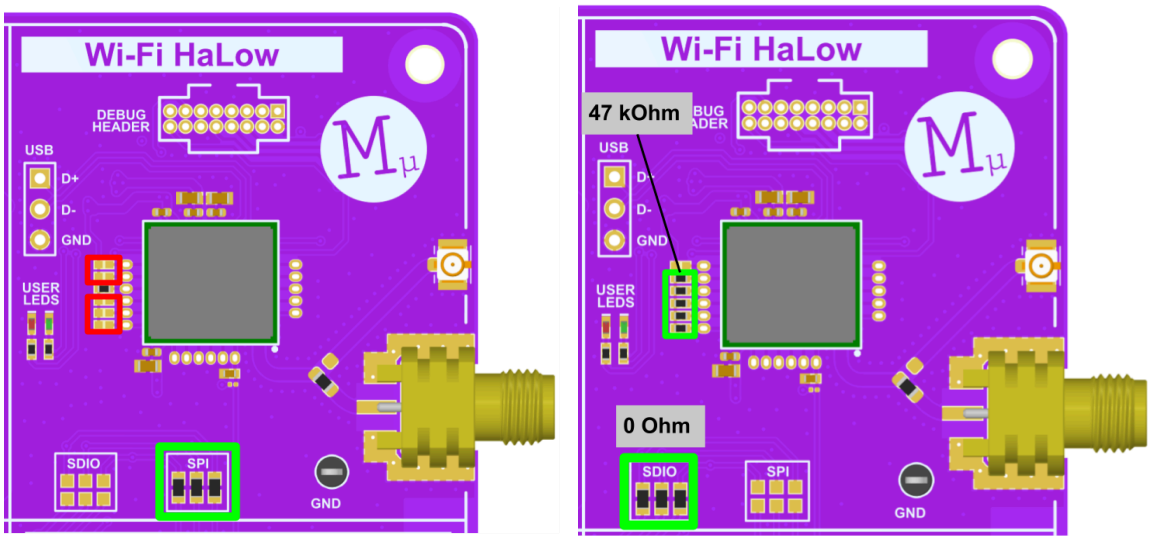


Figure 53. SPI configuration (left) and SDIO configuration (right).

6.4 Switching Between SMA and U.FL Connector

The user has the option to use either a SMA or U.FL connector for the HaLow antenna. The two can be switched between by changing the orientation of a 0 ohm resistor.

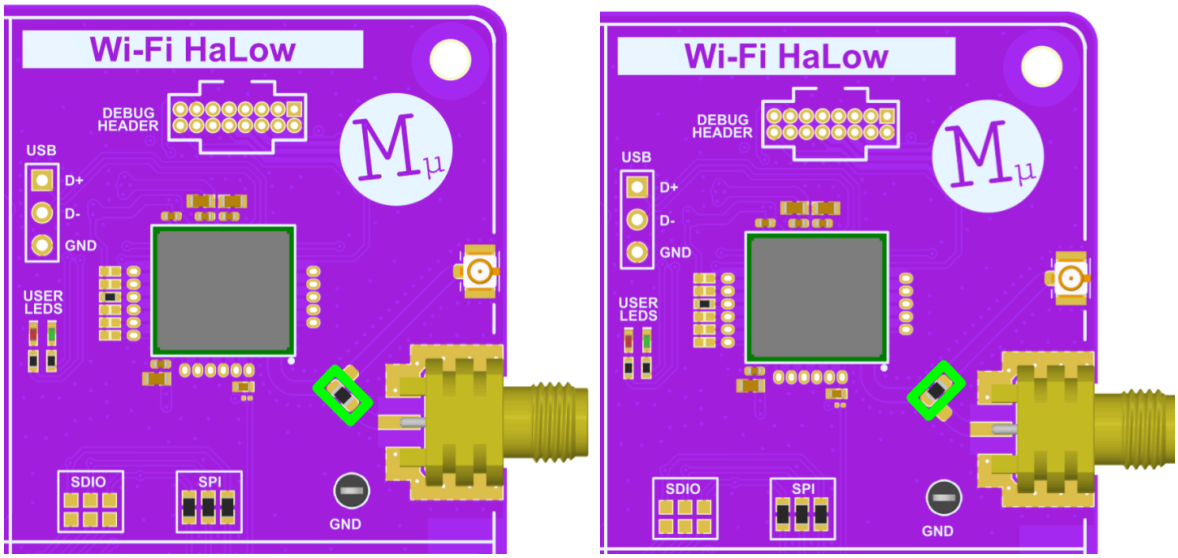


Figure 55. SMA connection (left) and U.FL connection (right).

6.5 Disconnecting Sensors

The sensor blocks on the MM8108-EKH05 feature a 0 ohm resistor that supplies power from the SENSOR header in the power tree to each individual sensor. These sensors include the temperature and humidity sensor, accelerometer, RGB LED, camera header, mikroBUS connector (3.3V on the left, 5V on the right), QWIIC connector, and the BLE module. By removing these 0 ohm resistors, the power can be cut off to the corresponding sensor individually. All these resistors are populated by default.

Please note to achieve optimal power consumption, it may be necessary to upload new firmware that ensures any I/Os connected to the STM32U585—which are now unpowered—are not actively driven.

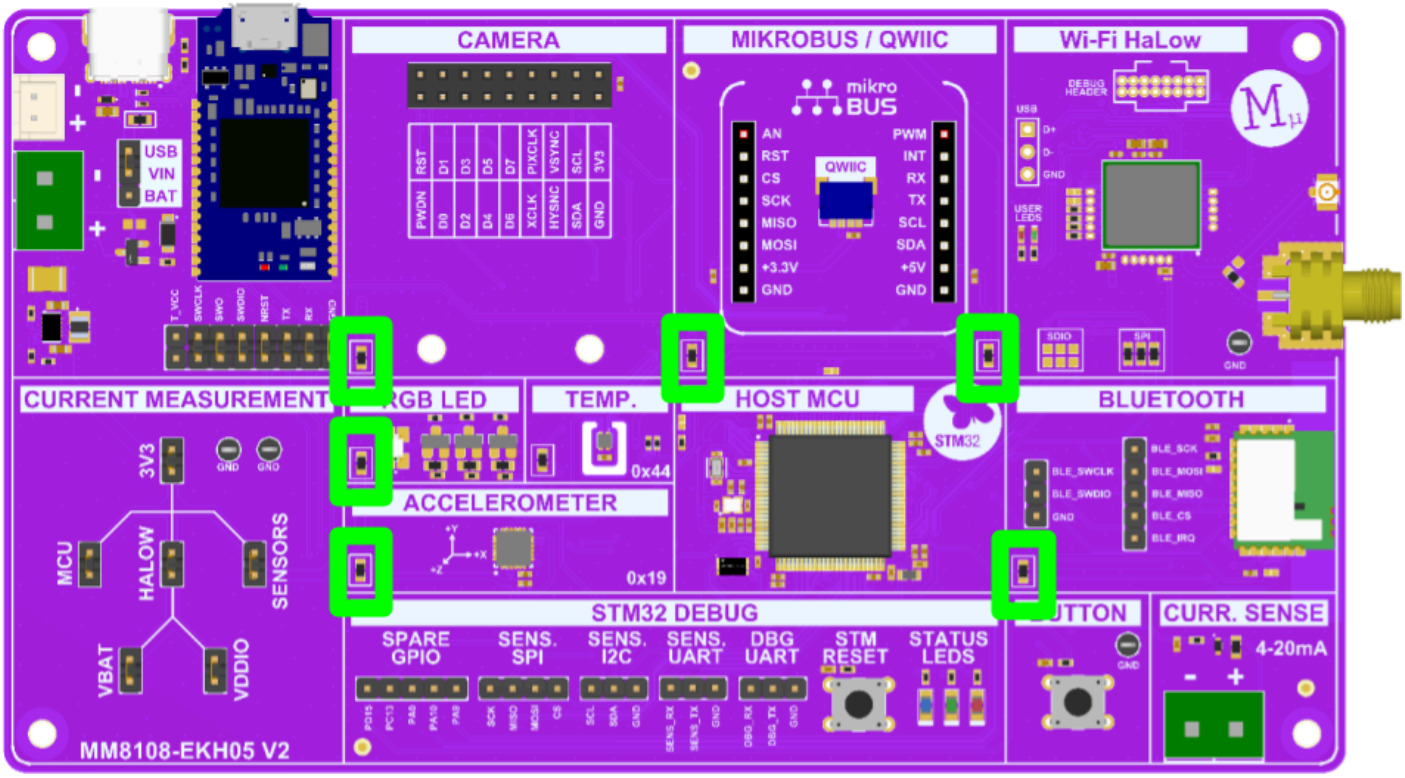


Figure 56. 0 ohm links which connect power to each sensor.

7 Power Consumption Measurements

7.1 Power Consumption Measurement Points

7.1.1 General Structure

The MM8108-EKH05 is designed to make key power consumption measurements straightforward. Its power distribution follows a tree structure, with each sub-power section branching from the main supply. For instance, to measure the total current draw on the 3.3V rail, an ammeter can be connected across the 3V3 header. If the user wishes to measure only the power consumption of the MCU for example, the ammeter can instead be placed across the MCU header.

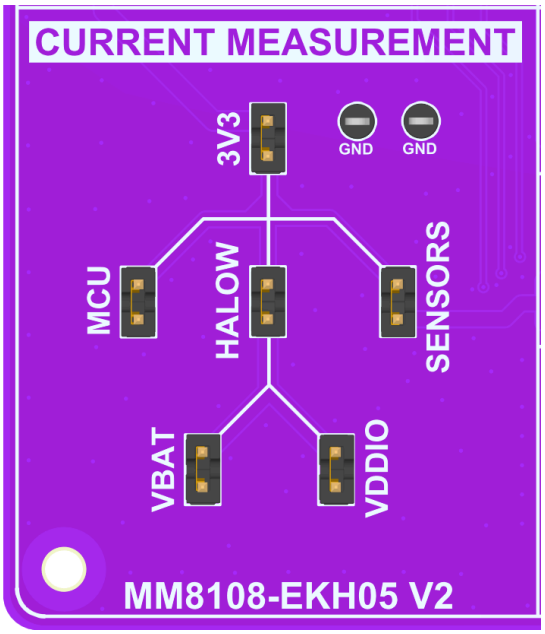


Figure 57. Current measurement tree on MM8108-EKH05.

7.1.2 Whole System Power Consumption

To measure the whole system power consumption, including efficiency losses of the 3.3V on-board buck-boost regulator, connect ammeter as per Figure 55.

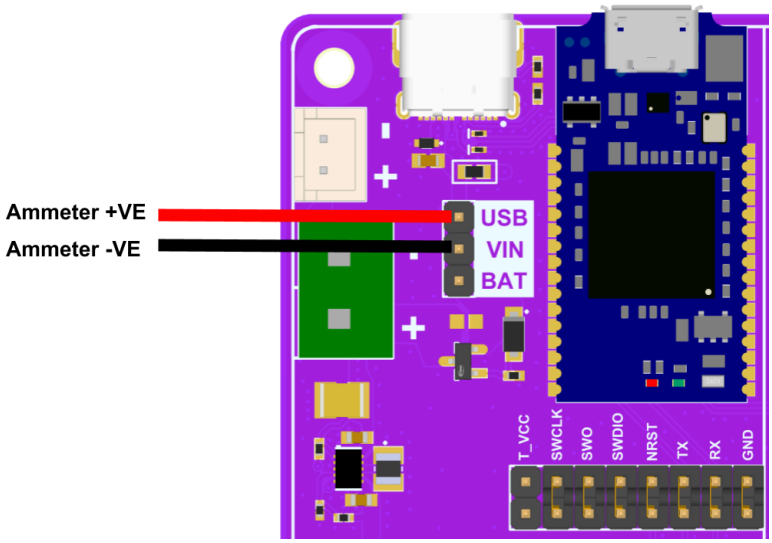


Figure 58. Ammeter positions to measure the entire board including the on-board regulators.

7.2 Power Consumption Measurement Procedure

A high-precision power analyzer, such as the Joulescope JS220, is recommended for low power measurements. This device can measure both voltage and current, allowing for a precise calculation of overall power. The suggested configuration is shown below.

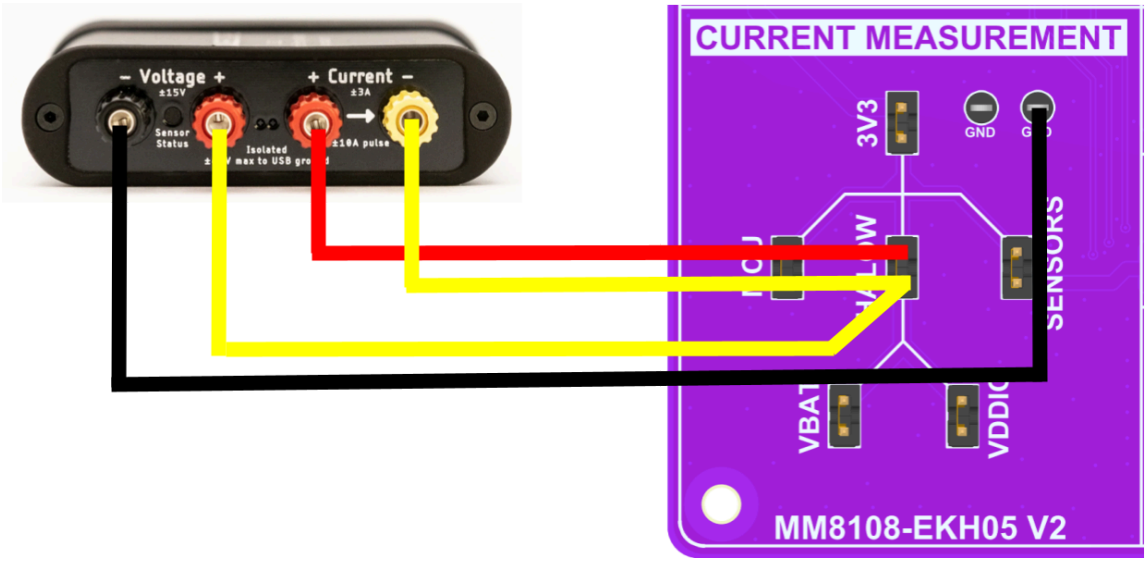


Figure 59. Example joulescope connections measuring HALOW header on MM8108-EKH05.

This configuration can be used for the 3V3, MCU, HALOW, SENSORS, VBAT and VDDIO headers, with the positive ammeter position at the top pin of the header, and the negative at the bottom.

7.3 Optional Debug GPIO Connections

Certain example applications utilise PA9 and PA10 of the STM32U585 as debug GPIOs. These debug GPIOs will toggle when the device is transitioning between different states and can be useful for measuring different parts of the example applications as they execute. These GPIOs are broken out to the SPARE GPIO header on the MM8108-EKH05 and can be connected to the Joulescope as follows.

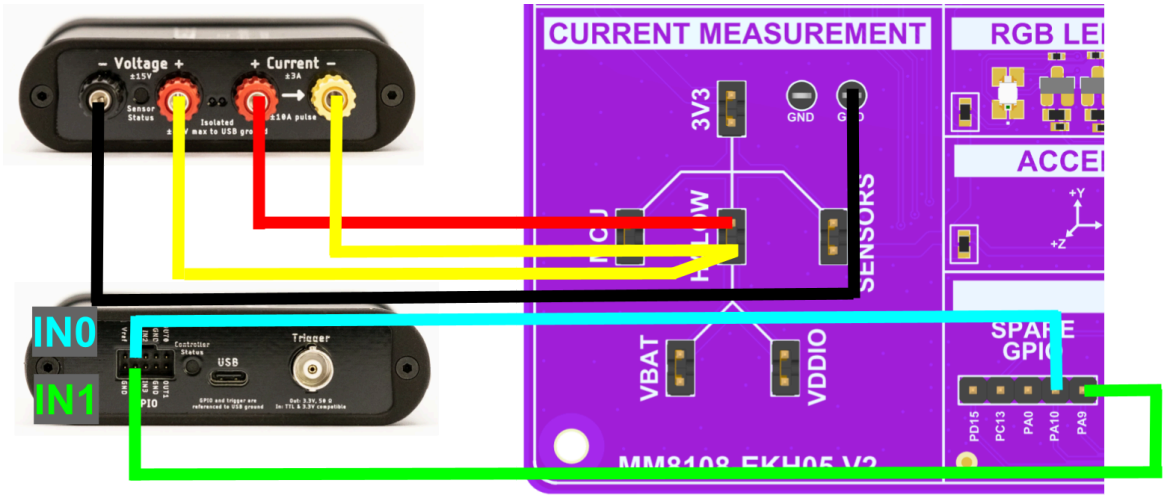


Figure 60. Showing where to connect debug GPIO's from MM8108-EKH05 to a joulescope.

On the Joulescope UI, the toggling debug GPIOs can be seen at the bottom and are annotated over to show the different states in a ping example application.

Figure 58. Power consumption profile of the ping app with the states from the debug GPIO's.

Figure 61.

Definitions for each debug state can be found in the example folder under each application of interest. For example, the definitions for the ping application can be found in the following locations.

Windows:

```
C:\Users\USER_NAME\STM32Cube\Repository\Packs\MorseMicro\MM_IoT\1.6.0\Example\ping\ping.c
```

Linux:

```
/home/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.6.0/Example/ping/ping.c
```

Mac:

```
/Users/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.6.0/Example/ping/ping.c
```

A search for the **enum** of **debug_state** will show the definitions. The function **set_debug_state** is used to set these throughout the application.

7.4 Measuring DTIM Power

To measure DTIM power consumption, use the ping application. Make sure to use the correct country code, BCF, SSID, and passphrase to connect the device to the AP. The DTIM period and bandwidth/channel can also be changed by following the steps in section 4.2.1.

The IP address that the MM8108-EKH05 will ping will also need to be set. See section 5.5 for how to change example parameters. The **ping.target** parameter needs to be set to 192.168.12.1 to match the default IP address of the HaLowLink 1.

For CMSIS Pack 1.5.0, the **ping.post_ping_delay_ms** parameter should also be changed from 60 to 600000. This will set the device to idle while connected for 10 minutes after pinging to take DTIM measurements.

For the lowest power consumption, it is recommended to use FreeRTOS Plus TCP as the network stack, please see section 5.7 to change this.

The ping application will perform the following actions:

- Boot the MM8108 chip.
- Connect to the AP.
- Ping the AP 10 times.
- Remain connected in power save mode while idling for 600 seconds.
- Disconnect from the AP.

Before starting, ensure that the ping application is loaded onto the device and that the device is properly connected to the AP. This can be confirmed by monitoring the UART output. Note that the UART baud rate is set to 115200.

```
Morse Ping Demo (Built Sep 16 2024 13:44:00)

---------------------------------
  BCF API version:          6.7.0
  BCF build version:        fbed8cb 244ffd771b
  BCF board description:    BAILEY_V12
  Morselib version:         998b9a497_NFP
  Morse firmware version:   1.13.0
  Morse chip ID:            0x0306
---------------------------------
Initialize IPv4 with static IP: 192.168.1.2...
Initialize IPv6 using Autoconfig...
Morse FreeRTOS+ TCP interface initialised. MAC address 94:bb:43:dc:f9:52
Attempting to connect to MorseMicro with passphrase 12345678
This may take some time (~30 seconds)
WLAN STA connecting
WLAN STA connected
DNS server 0: 192.168.1.1
Link is up. Time: 7068 ms, IP: 192.168.1.2, Netmask: 255.255.255.0, Gateway: 192.168.1.1, D1


Ping 192.168.1.1 56(64) bytes of data.
(192.168.1.1) packets transmitted/received = 2/2, round-trip min/avg/max = 12/12/12 ms
(192.168.1.1) packets transmitted/received = 3/3, round-trip min/avg/max = 11/11/12 ms
(192.168.1.1) packets transmitted/received = 4/4, round-trip min/avg/max = 11/11/12 ms
(192.168.1.1) packets transmitted/received = 5/5, round-trip min/avg/max = 11/11/12 ms
(192.168.1.1) packets transmitted/received = 6/6, round-trip min/avg/max = 11/11/13 ms
(192.168.1.1) packets transmitted/received = 7/7, round-trip min/avg/max = 11/11/13 ms
(192.168.1.1) packets transmitted/received = 8/8, round-trip min/avg/max = 11/11/13 ms
(192.168.1.1) packets transmitted/received = 9/9, round-trip min/avg/max = 11/11/13 ms
(192.168.1.1) packets transmitted/received = 10/10, round-trip min/avg/max = 11/11/13 ms

--- 192.168.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0.000% packet loss
round-trip min/avg/max = 11/11/13 ms
WLAN STA disabled
Link is down. Time: 27868 ms
```

Figure 62. Example UART output from ping application.

The below screenshot shows the power consumption profile as the program executes.

Figure 63. Example current consumption profile from ping application.

The green highlighted section below illustrates the measurement of a DTIM period, zoomed in from the DTIM Wakeups section displayed above. In this example, the measurement is for a DTIM3 with an 8 MHz channel and a 102.4 ms beacon interval. Adjusting these parameters will impact the measured power consumption.

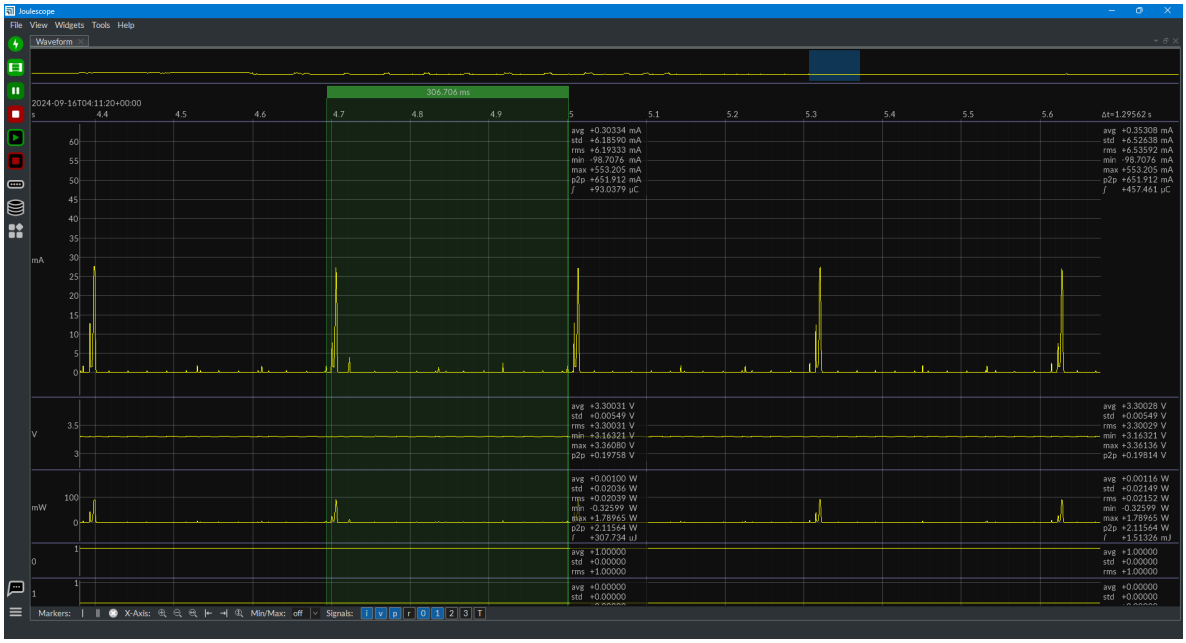


Figure 64. Zoomed in view of a DTIM3 measurement.

7.5 Measuring WNM Sleep Power

WNM Sleep provides a more sophisticated sleep mechanism, where the sleep schedule is explicitly negotiated between the station and the AP. This allows for longer sleep intervals and more efficient data retrieval when the client wakes up.

To measure WNM sleep power consumption, use the wnm_sleep application. Ensure that the correct country code, BCF, SSID, and passphrases are configured so the device can connect to the AP.

Similar to the ping example mentioned in section 7.4, the IP address that the MM8108-EKH05 will ping in the WNM sleep example will also need to be set. See section 5.5 for how to change example parameters. The ping.target parameter needs to be set to 192.168.12.1 to match the default IP address of the HaLowLink 1.

The wnm_sleep application will perform the following actions:

- Boot the MM8108 chip.
- Connect to the AP.
- Ping the AP 10 times.
- Enter WNM sleep for 20 seconds.
- Exit WNM sleep.
- Ping the AP another 10 times.
- Enter WNM sleep with chip in shutdown mode for 20 seconds.
- Exit WNM sleep.
- Disconnect from the AP.

As with DTIM measurements, start by verifying that the wnm_sleep example is correctly loaded and that the device can successfully associate with the AP via the MM8108-EKH05's UART. Below are screenshots of the complete UART output from the program, along with an annotated power trace.

```
Morse WNM Sleep Demo (Built Sep 16 2024 14:15:05)

---------------------------------
  BCF API version:         6.7.0
  BCF build version:       fbed8cb 244ffd771b
  BCF board description:   BAILEY_V12
  Morselib version:        998b9a497_NFP
  Morse firmware version:  1.13.0
  Morse chip ID:           0x0306
---------------------------------
Initialize IPv4 with static IP: 192.168.1.2...
Initialize IPv6 using Autoconfig...
Morse FreeRTOS+ TCP interface initialised. MAC address 94:bb:43:dc:f9:52
Attempting to connect to MorseMicro with passphrase 12345678
This may take some time (~30 seconds)
WLAN STA connecting
WLAN STA connected
DNS server 0: 192.168.1.1
Link is up. Time: 8176 ms, IP: 192.168.1.2, Netmask: 255.255.255.0, Gateway: 192.168.1.1, D1


Ping 192.168.1.1 56(64) bytes of data.
(192.168.1.1) packets transmitted/received = 2/2, round-trip min/avg/max = 12/12/12 ms
(192.168.1.1) packets transmitted/received = 3/3, round-trip min/avg/max = 12/12/12 ms
(192.168.1.1) packets transmitted/received = 4/4, round-trip min/avg/max = 11/11/12 ms
(192.168.1.1) packets transmitted/received = 5/5, round-trip min/avg/max = 11/11/12 ms
(192.168.1.1) packets transmitted/received = 6/6, round-trip min/avg/max = 11/12/14 ms
(192.168.1.1) packets transmitted/received = 7/7, round-trip min/avg/max = 11/12/14 ms
(192.168.1.1) packets transmitted/received = 8/8, round-trip min/avg/max = 11/12/14 ms
(192.168.1.1) packets transmitted/received = 9/9, round-trip min/avg/max = 11/11/14 ms
(192.168.1.1) packets transmitted/received = 10/10, round-trip min/avg/max = 11/11/14 ms

--- 192.168.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0.000% packet loss
round-trip min/avg/max = 11/11/14 ms

Entering WNM sleep with chip power down disabled
Expected sleep time 20000ms.

Enter WNM sleep took 16 ms.

Exit WNM sleep took 17 ms.
```

Figure 65. Example UART output from wnm_sleep application.

```
Ping 192.168.1.1 56(64) bytes of data.
(192.168.1.1) packets transmitted/received = 2/2, round-trip min/avg/max = 3/7/11 ms
(192.168.1.1) packets transmitted/received = 3/3, round-trip min/avg/max = 3/9/13 ms
(192.168.1.1) packets transmitted/received = 4/4, round-trip min/avg/max = 3/9/13 ms
(192.168.1.1) packets transmitted/received = 5/5, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 6/6, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 7/7, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 8/8, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 9/9, round-trip min/avg/max = 3/14/35 ms
(192.168.1.1) packets transmitted/received = 10/10, round-trip min/avg/max = 3/14/35 ms

--- 192.168.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0.000% packet loss
round-trip min/avg/max = 3/14/35 ms

Entering WNM sleep with chip power down enabled.
Expected sleep time 20000ms.

Enter WNM sleep took 13 ms.

Exit WNM sleep took 623 ms.
WLAN STA disabled
Link is down. Time: 69345 ms
```

Figure 66. Example UART output from wnm_sleep application continued.

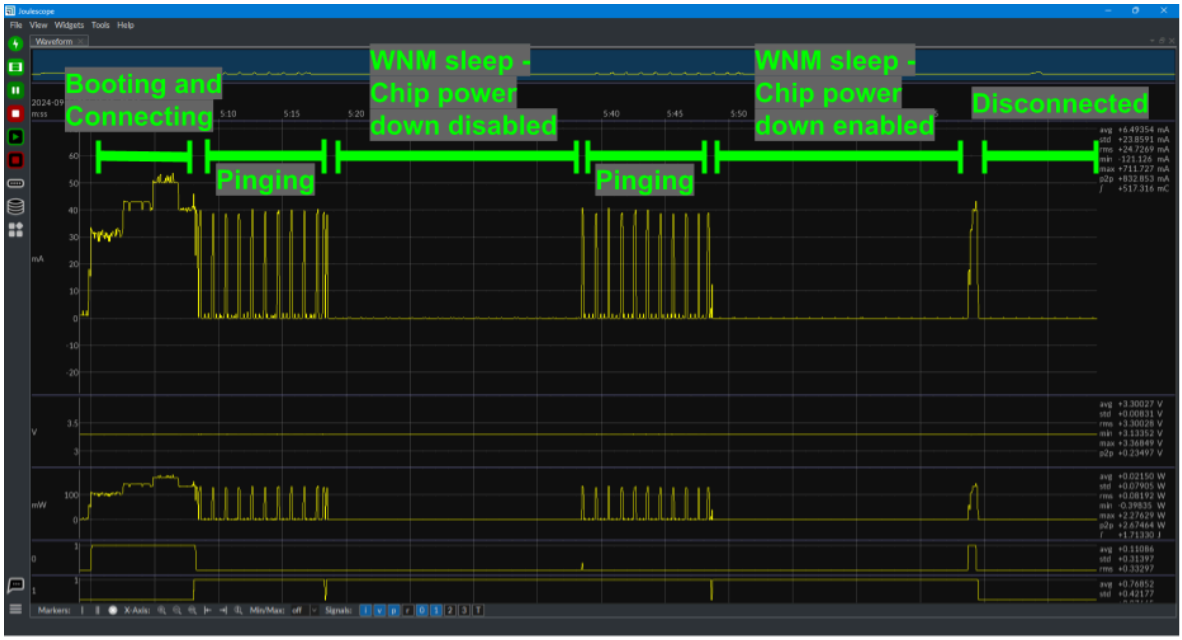


Figure 67. Example current consumption profile from wnm_sleep application.

7.6 Current Spikes When Waking Up

Users may observe a momentary current spike when MM8108 based devices wake up from sleep modes. Typically, these spikes will be less than 200mA. The spikes are expected, and are caused from the operation of the internal buck converter. The buck converter generates ~1.4 V during active operation to supply the internal chip power rails. In power save mode, the chip enters a sleep mode where the Vbuck voltage is dropped to ~0.7 V. This voltage is used to keep the internal memory in a retention state; while the rest of the chip's power supplies are disabled to save power.

When the chip wakes up, the buck capacitor is charged from ~0.7 V to ~1.4 V. This rapid charging, occurring over tens of microseconds, generates the observed current spike.

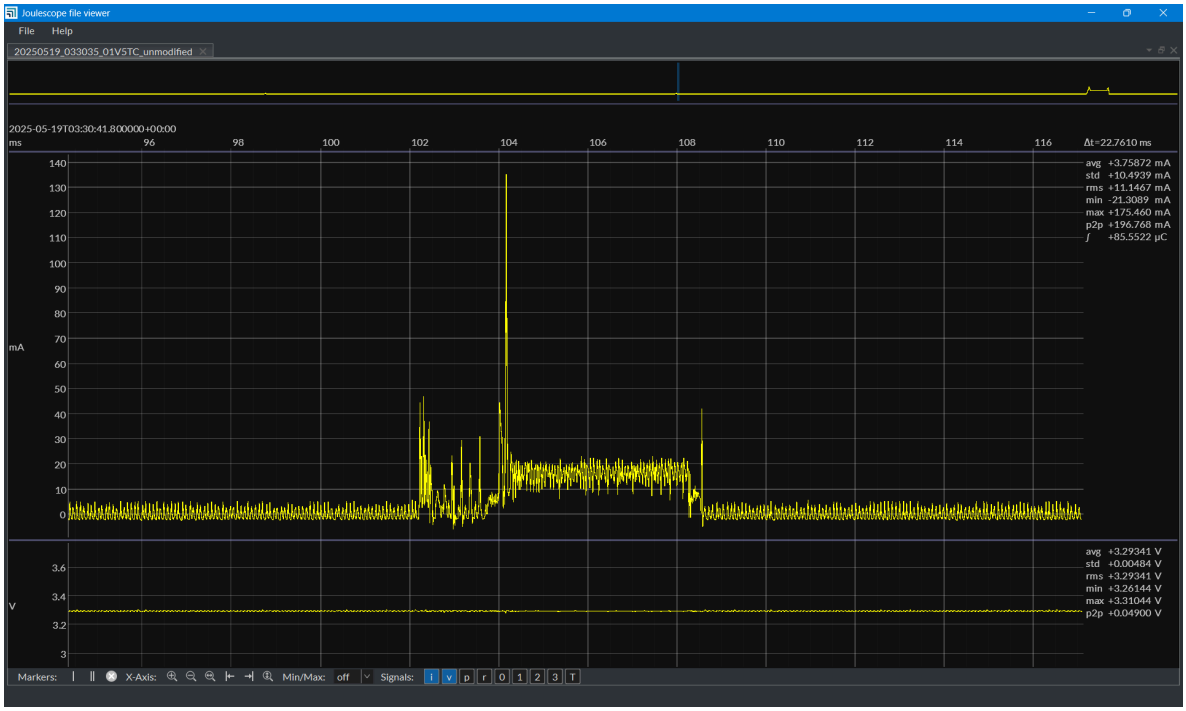


Figure 68. MM8108 DTIM wakeup zoomed out.

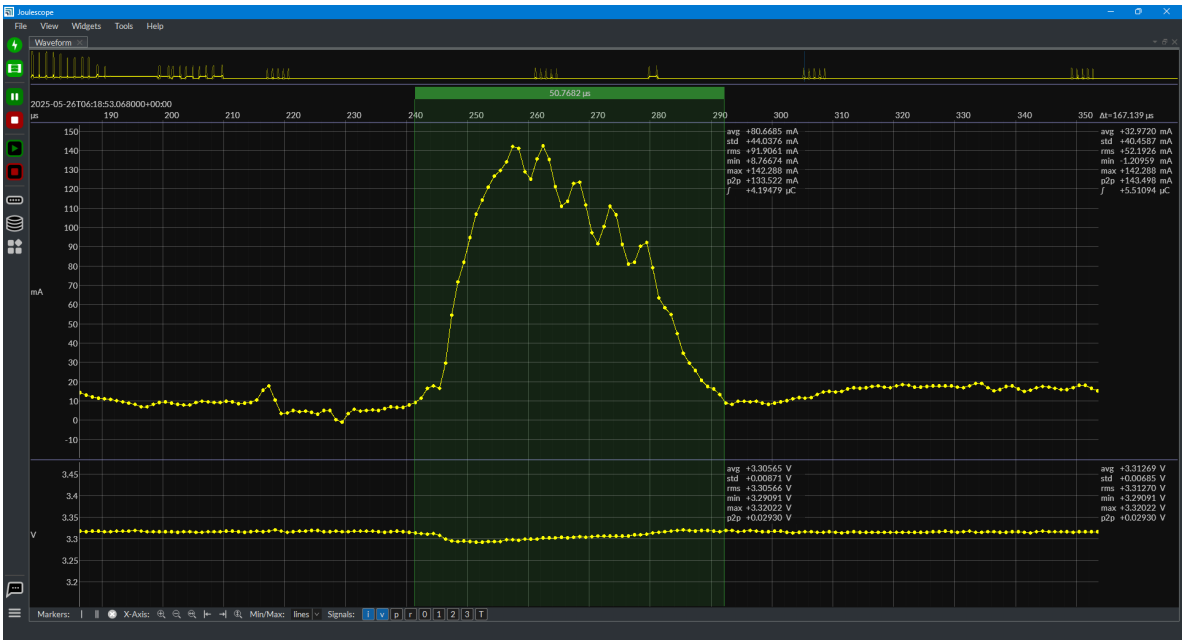


Figure 69. MM8108 wakeup current spike zoomed in.

Although the peak of the current is large, the duration is short, so the energy is small (in this instance less than 4 µC) and the instantaneous current spike is sourced primarily from decoupling capacitors on the VBAT power rail. Batteries connected to the power supply are decoupled further by regulators and additional bulk capacitors upstream of the MM8108. All power consumption figures published in Morse Micro datasheets already include these transient spikes, ensuring there are no additional power costs beyond the specified values.

7.7 Negative Current Consumption

Users may observe brief negative current pulses when measuring the power consumption of the HaLow module, particularly during snooze periods. The screenshot below shows a zoomed-in view of the snooze current as measured on the HaLow header when running the ping application.

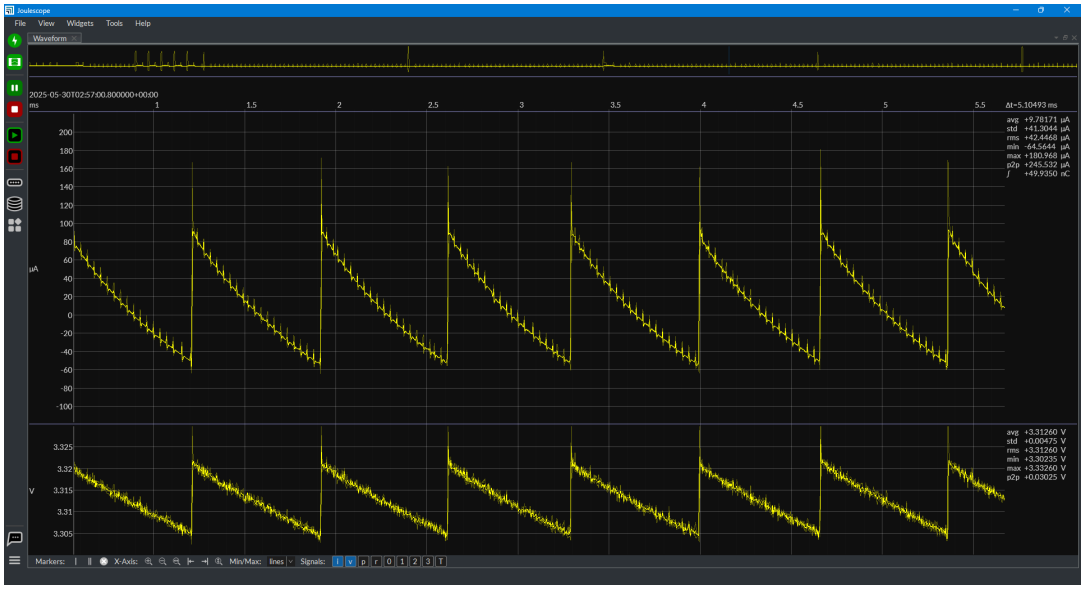


Figure 70. Zoomed in view of a snooze measurement showing negative current spikes.

This behavior is caused by the on-board buck-boost regulator, the TPS63802. In the MM8108-EKH05 design, the MODE pin is tied low, which places the regulator in power-save mode (PFM) to optimize efficiency at light load:

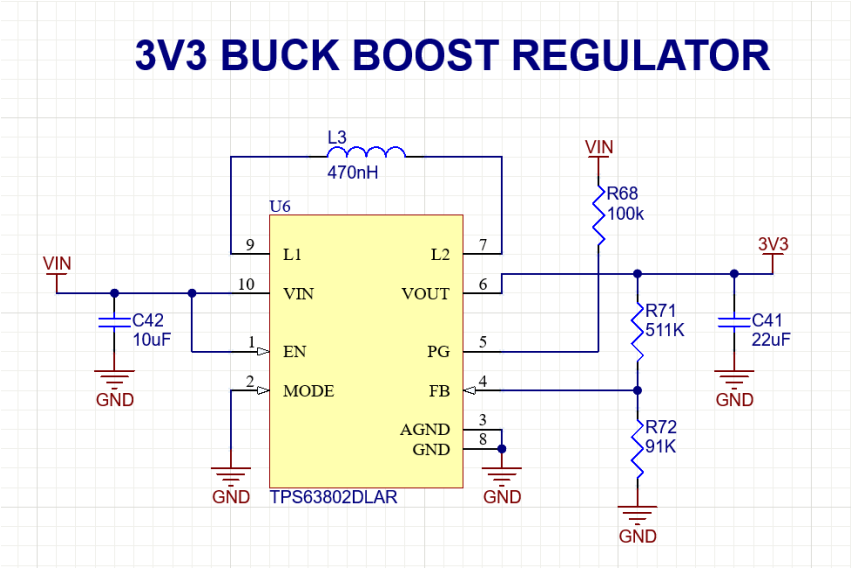
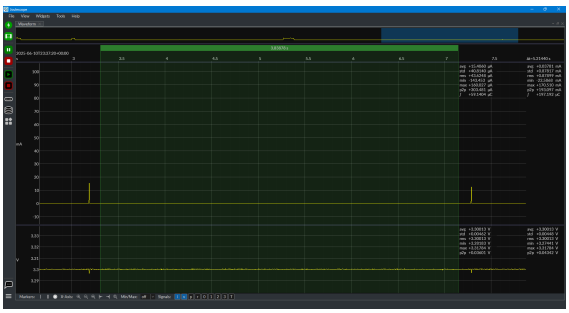
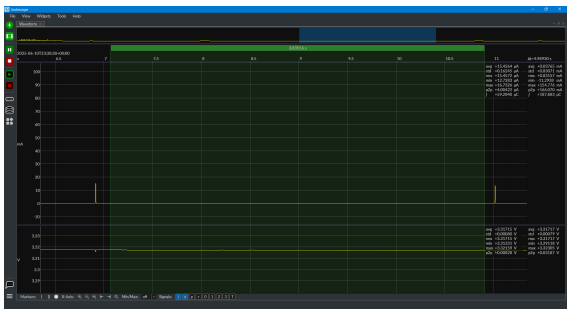


Figure 71. Schematic of TPS63802 in the MM8108-EKH05.

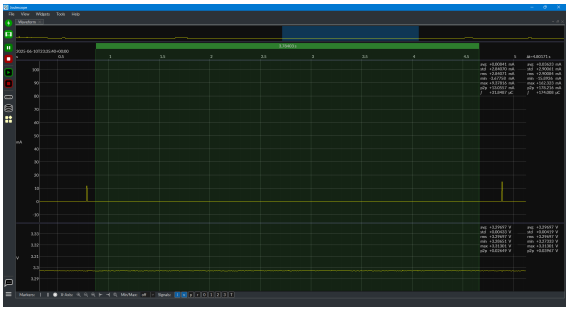
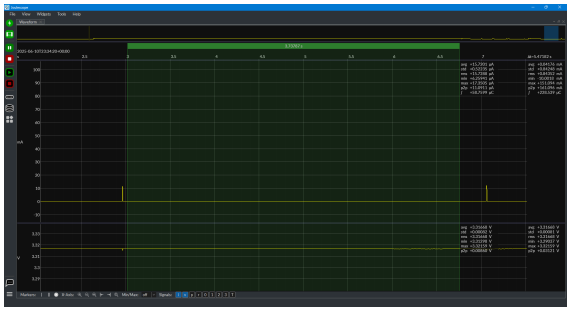
In PFM mode, the regulator delivers energy in short bursts and then enters an idle or "coast" phase. During these bursts, energy is stored in the output capacitor and inductor. If the system load is very low or momentarily inactive — such as during HaLow snooze — that energy may exceed what the load can absorb. This can lead to a temporary overshoot in output voltage and current to flow back into the regulator, through its internal switches or body diodes, or along parasitic paths.

This reverse current is captured by the Joulescope and appears as negative current, even though the HaLow module itself is not actively drawing current. These transients are a normal artifact of operating in PFM mode with a low or pulsed load and do not indicate an error or inefficiency. However, they may cause measured average power to appear lower or distorted unless properly interpreted.

The simplest way to eliminate the reverse current effect entirely is to bypass the on-board regulator and instead power the 3V3 rail using a lab bench power supply. However, under light load conditions, the difference in measured HaLow current between the on-board regulator and an external supply is negligible. For example, when running the ping application — which places the STM32U585 host microcontroller into a low-power sleep state — the difference in measured current is typically less than 0.1 µA. In this case, the backflow effect is present but extremely minor.

Ping Application - On-board regulator	**Ping Application - Lab bench power supply**
	
Snooze current: 15 uA	Snooze current: 15 uA

In contrast, when using the CLI application, the STM32U585 host microcontroller remains active by default, drawing approximately 6 mA continuously. Under this heavier load, the reverse current pulses become more pronounced, with deeper negative current spikes observed on the Joulescope. This is due to larger energy bursts during power-save switching, which exacerbate transient effects.

CLI Application - On-board regulator	CLI Application - Lab bench power supply
 Snooze current: 8 uA	 Snooze current: 16 uA

8 Troubleshooting

8.1 Device Is Not Programming

If the STLINK V3-MODS is unable to connect to the STM32U585, please check the following steps.

1. Ensure the STLINK jumpers are populated. Without these, the programmer will be disconnected from the microcontroller.

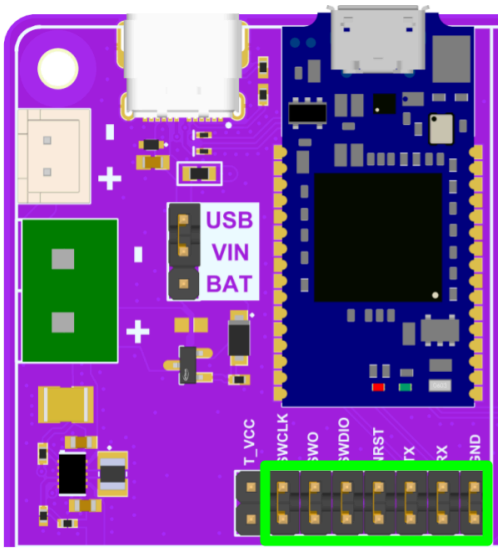


Figure 76. Troubleshooting step checking if the highlighted jumpers are connected.

2. Ensure the 3V3 and MCU jumpers are populated. The microcontroller will not be powered without these.

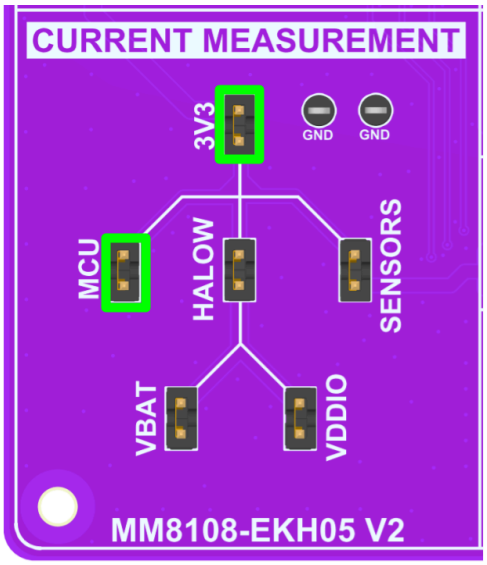


Figure 74. Troubleshooting step checking if the 3V3 and MCU jumpers are connected.

3. Ensure the power select jumper is on USB. If using batteries, check VIN voltage to ensure the battery is functioning normally.

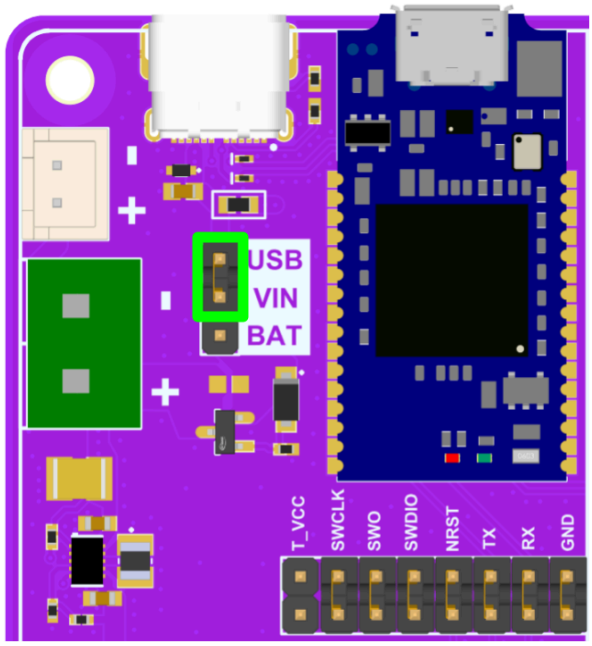


Figure 78. Troubleshooting step checking the input power supply.

4. Ensure the device is not in a low power state. If the microcontroller is in certain deep sleep states, the programmer will not be able to attach. To recover in this situation, hold the STM RESET button as the device is attempting to connect, and release. This will ensure the microcontroller can be attached to.

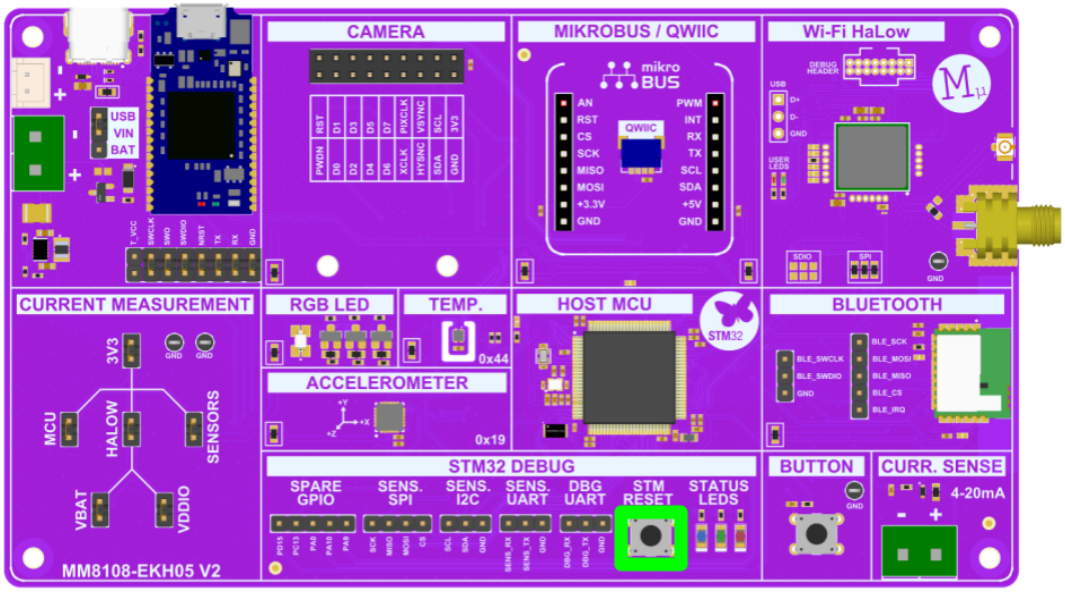


Figure 79. Reset button on the MM8108-EKH05. Resets the host MCU.

8.2 Poor HaLow Performance

If the HaLow performance is poor, the following items should be checked.

1. Ensure the AP configuration is expected. I.e. the correct channel and bandwidth are configured.
2. Ensure there are no other HaLow devices on the same channel causing in-band interference.
3. Ensure antenna connection is on securely.
4. If cabled, ensure there is sufficient attenuation (~40 dB) between the devices. If the power is too high, it will saturate the LNA.

8.3 High Power Consumption

There are many reasons that high power consumption may be measured. Please check the following:

1. If the T_VCC jumper is populated, this will add ~300uA to the 3V3 power consumption.
2. For lowest power consumption, ensure the network stack is set to FreeRTOS plus TCP instead of LwIP. See section 5.7 for details.
3. To replicate Morse Micro DTIM power consumption numbers, the devices should be placed inside a shielded box away from interference. If the device fails to receive the beacon from the AP, it will spend a longer time in active RX mode and consume more power resulting in misleading DTIM measurements.
4. The temperature plays a big factor in power consumption. All testing should be done at room temperature of ~77 degrees Fahrenheit (25 degrees Celsius) to match Morse Micro DTIM power consumption numbers.
5. Ensure the MM8108-EKH05 is associated with the AP. Double check SSID and passphrase match.
6. It is recommended to use as short leads as possible from the ammeter to the MM8108-EKH05. Longer leads will add equivalent series resistance (ESR) and inductance which will impact the measurements.
7. Traffic from the network. If the AP is set to a mode where traffic gets bridged from the Ethernet interface to the HaLow interface, the MM8108-EKH05 will get sent any multicast traffic, and wake up to receive this. A good way to see this, is to set the security of both the AP and STA to an open network, and connect a sniffer to the setup to see what traffic is being sent while observing the power consumption. A setup script to launch the sniffer can be found on the Morse Micro customer portal after logging in - https://www.morsemicro.com/downloads-dashboard/#/tools. An easy way to remove all multicast traffic is to either set the AP to a mode where the interfaces are not bridged, or just simply disconnect the ethernet cable from the AP.

9 Known Limitations

9.1 Reduced Receive Sensitivity when Programmer connected

When connected via the micro-USB port, harmonics from the 25MHz crystal on the STLINK-V3MODS (U7), though small, are above the sensitivity limit of the receiver.

To maximise transmit range, either:

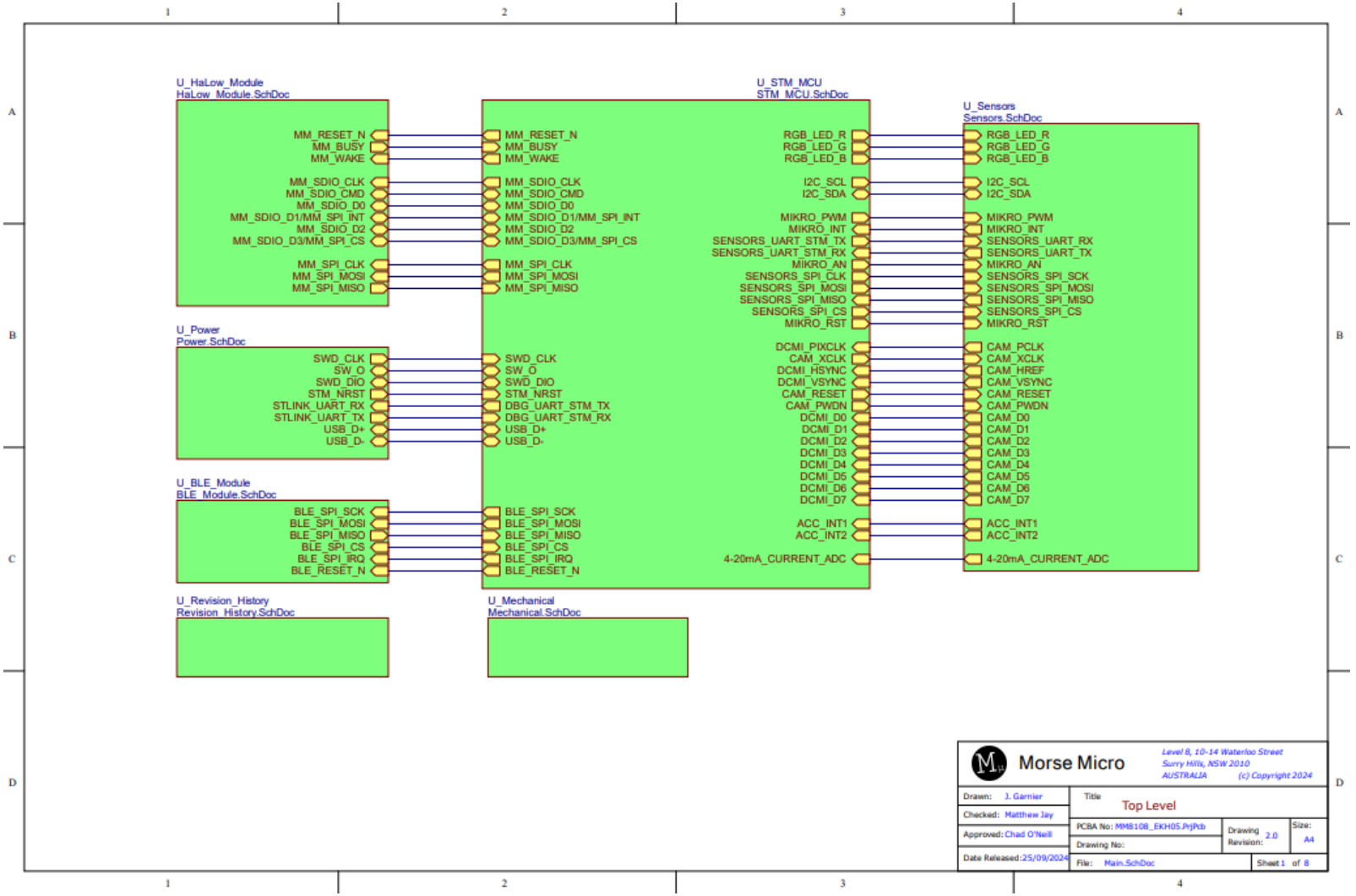
A. Do not connect the USB-Micro port. USB-C or battery are suitable alternatives
B. Operate on a channel below 923MHz or above 927MHz

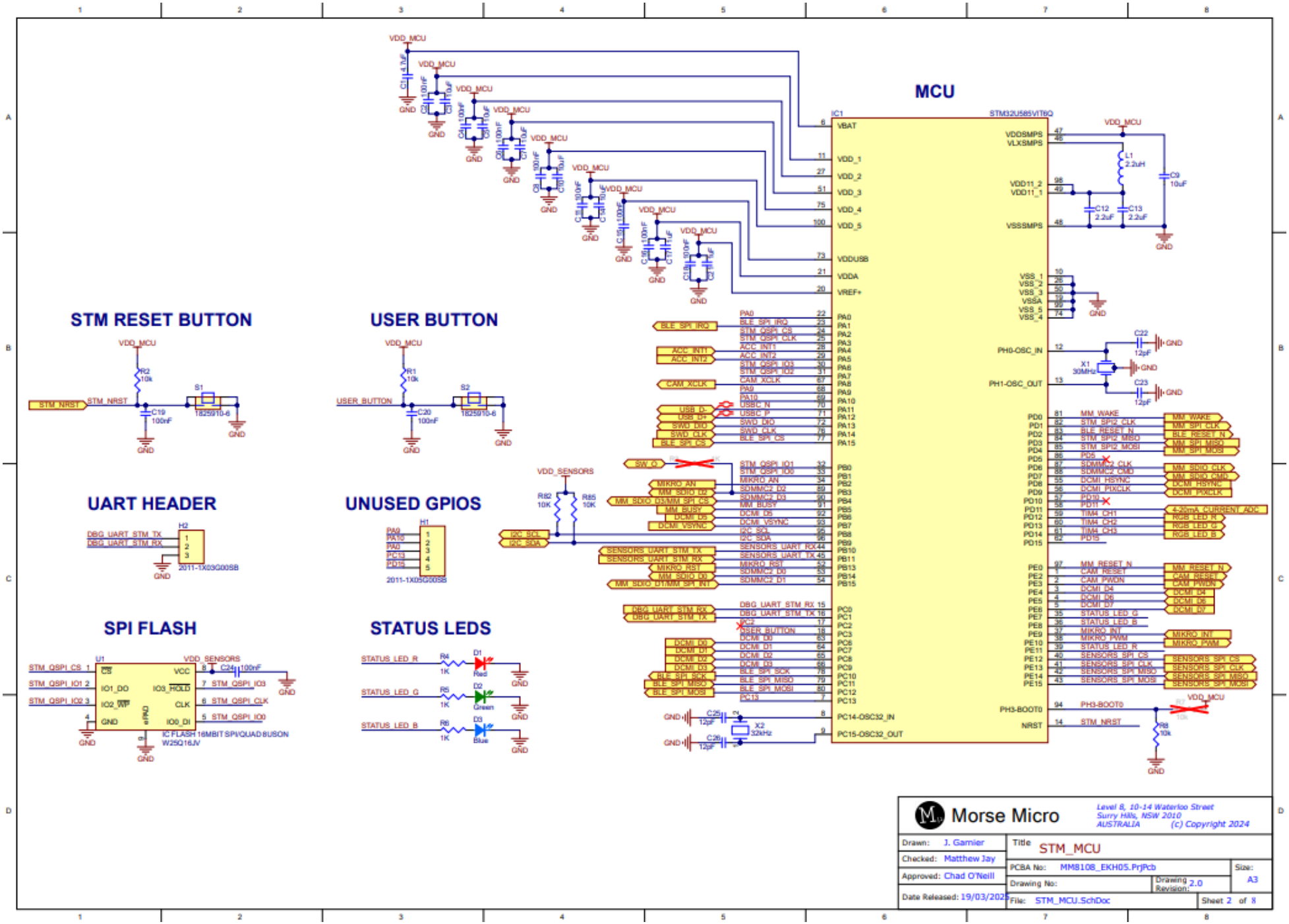
This receiver desensitisation effect does not originate from the Morse Micro HaLow module. It is a natural consequence of a 25MHz square wave clock having a harmonic at 925 MHz. Any external noise source with similar spectral content could produce a comparable effect.

10 Revision History

Release Number	Release Date	Release Notes
01	17/04/2025	• Initial release.
02	31/10/2025	• Updated section 7.7 explaining negative current consumption.
03	31/10/2025	• Removed references of the full EKH05 Demo application being loaded on by default. • Added CLI example application. • Added explanation of reduced sensitivity when using programmer.

Appendix A: Schematics




MCU
STM RESET BUTTON
USER BUTTON
UART HEADER
UNUSED GPIOS
SPI FLASH
STATUS LEDS
STM32U585VIT6Q
IC FLASH 16MBIT SPI/QUAD 8USON W25Q16JV
2011-1X03G00SB
2011-1X05G00SB
Morse Micro
Level 8, 10-14 Waterloo Street
Surry Hills, NSW 2010
AUSTRALIA
(c) Copyright 2024
Drawn: J. Garnier
Checked: Matthew Jay
Approved: Chad O'Neill
Date Released: 19/03/2025
Title STM_MCU
PCBA No: MM8108_EKH05.PrjPcb
Drawing No:
Drawing Revision: 2.0
Size: A3
File: STM_MCU.SchDoc
Sheet 2 of 8

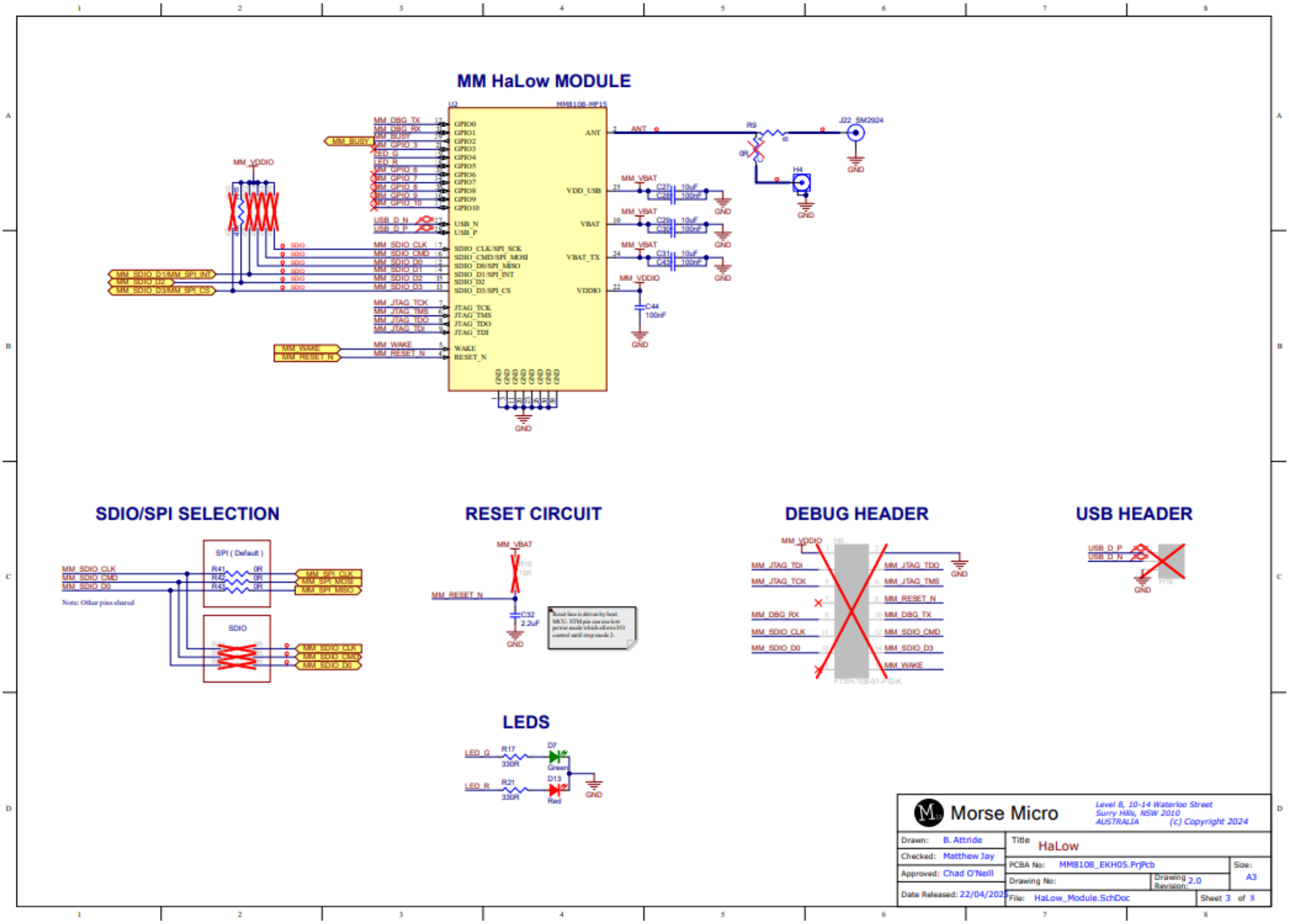
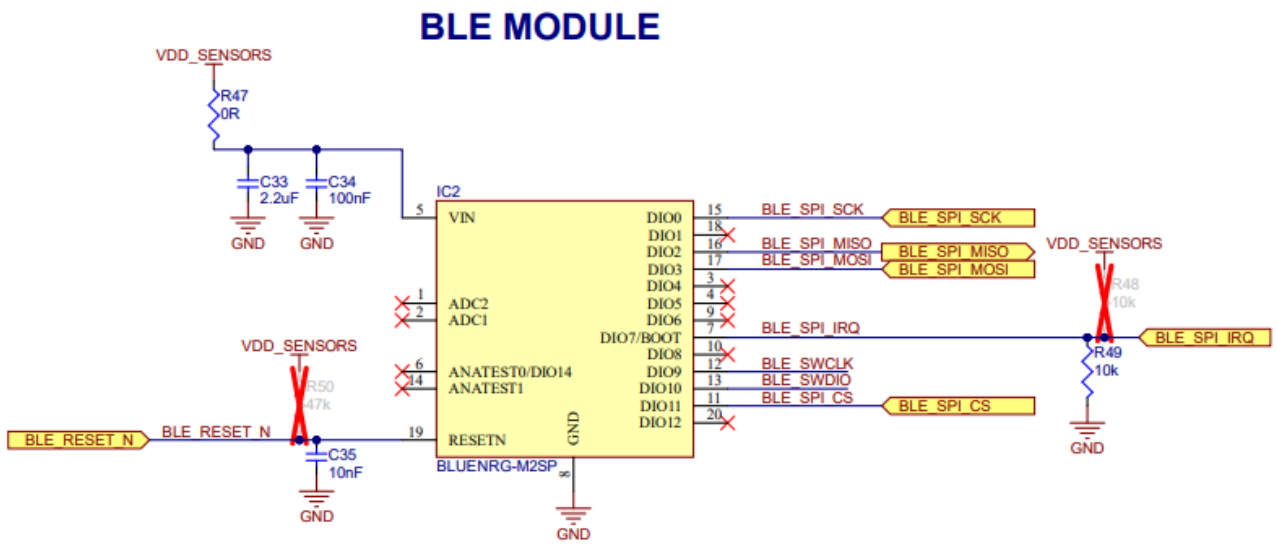

MM HaLow MODULE
U2
MM8108-MF15
GPIO0
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
GPIO7
GPIO8
GPIO9
GPIO10
USB_N
USB_P
SDIO_CLK/SPI_SCK
SDIO_CMD/SPI_MOSI
SDIO_D0/SPI_MISO
SDIO_D1/SPI_INT
SDIO_D2
SDIO_D3/SPI_CS
JTAG_TCK
JTAG_TMS
JTAG_TDO
JTAG_TDI
WAKE
RESET_N
ANT
VDD_USB
VBAT
VBAT_TX
VDDIO
MM_VDDIO
MM_VBAT
MM_BUSY
MM_DBG_TX
MM_DBG_RX
MM_GPIO_3
LED_G
LED_R
MM_GPIO_6
MM_GPIO_7
MM_GPIO_8
MM_GPIO_9
MM_GPIO_10
USB_D_N
USB_D_P
MM_SDIO_CLK
MM_SDIO_CMD
MM_SDIO_D0
MM_SDIO_D1
MM_SDIO_D2
MM_SDIO_D3
MM_JTAG_TCK
MM_JTAG_TMS
MM_JTAG_TDO
MM_JTAG_TDI
MM_WAKE
MM_RESET_N
MM_SDIO_D1/MM_SPI_INT
MM_SDIO_D2
MM_SDIO_D3/MM_SPI_CS
SDIO
R9
0R
J22_SM2924
H4
C27
C28
C29
C30
C31
C43
10uF
100nF
C44
100nF
GND
SDIO/SPI SELECTION
SPI (Default)
R41
R42
R43
0R
MM_SPI_CLK
MM_SPI_MOSI
MM_SPI_MISO
Note: Other pins shared
SDIO
MM_SDIO_CLK
MM_SDIO_CMD
MM_SDIO_D0
RESET CIRCUIT
MM_VBAT
MM_RESET_N
C32
2.2uF
DEBUG HEADER
H5
MM_VDDIO
MM_JTAG_TDI
MM_JTAG_TCK
MM_DBG_RX
MM_SDIO_CLK
MM_SDIO_D0
MM_JTAG_TDO
MM_JTAG_TMS
MM_RESET_N
MM_DBG_TX
MM_SDIO_CMD
MM_SDIO_D3
MM_WAKE
USB HEADER
USB_D_P
USB_D_N
LEDS
LED_G
LED_R
R17
R21
330R
D7
D13
Green
Red
Morse Micro
Level 8, 10-14 Waterloo Street
Surry Hills, NSW 2010
AUSTRALIA
(c) Copyright 2024
Drawn: B. Attride
Checked: Matthew Jay
Approved: Chad O'Neill
Date Released: 22/04/2025
Title HaLow
PCBA No: MM8108_EKH05.PrjPcb
Size: A3
Drawing No:
Drawing Revision: 2.0
File: HaLow_Module.SchDoc
Sheet 3 of 8

Table 8. **Pin connection**

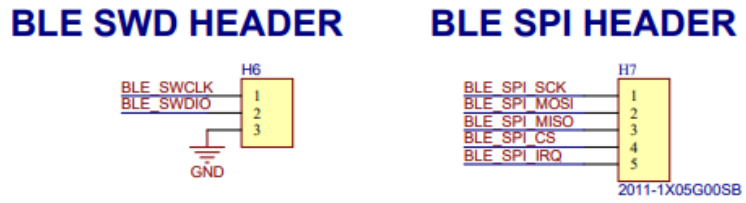
Pin #	Name	Function			
		Mode "000"	Mode "001"	Mode "100"	Mode "010"
1	ADC2	ADC input 2			
2	ADC1	ADC input 1			
3	DIO4	GPIO4	UART_RXD	I2C2_CLK	PWM0
4	DIO5	GPIO5	UART_TXD	I2C2_DAT	PWM1
5	VIN	Power Supply			
6	ANATEST0/DIO14	GPIO14	I2C1_CLK	SPI_CLK	ADC_DAT
7	DIO7/BOOT [1]	GPIO7	UART_CTS	I2C2_DAT	PDM_CLK
8	GND	Ground			
9	DIO6	GPIO6	UART_RTS	I2C2_CLK	PDM_DAT
10	DIO8	GPIO8	UART_TXD	SPI_CLK	PDM_DAT
11	DIO11	GPIO11	UART-RXD	SPI_CS1	
12	DIO9	GPIO9	SWCLK	SPI_IN [2]	
13	DIO10	GPIO10	SWDIO	SPI_out [3]	
14	ANATEST1	Anatest1			
15	DIO0	GPIO0	UART_CTS	SPI_CLK	
16	DIO2	GPIO2	PWM0	SPI_OUT	PDM_CLK
17	DIO3	GPIO3	PWM1	SPI_IN	ADC_CLK
18	DIO1	GPIO1	UART_RTS	SPI_CS1	PDM_DAT
19	RESETN	Reset			
20	DIO12	GPIO12 [4]		I2C1_CLK	
21 [5]	GND	Ground or leave unconnected			
22 [5]	N.C.	Leave unconnected			
23 [5]	N.C.	Leave unconnected			

1. *The pin DIO7/BOOT is monitored by bootloader after power-up or hardware reset and it should be low to prevent unwanted bootloader activation.*
2. *The function SPI_IN indicates that the pin is always an input when configured for SPI. Thus in case of SPI master role, it acts as MISO pin. In case of SPI slave role, this pin act as MOSI*
3. *The function SPI_OUT indicates that the pin is always an output when configured for SPI. Thus in case of SPI master role, it acts as MOSI pin. In case of SPI slave role, this pin act as MISO.*
4. *DIO12 can only be general purpose input pins (not output), or I^2C1 clock pin.*
5. *BLUENRG-M2SA only.*



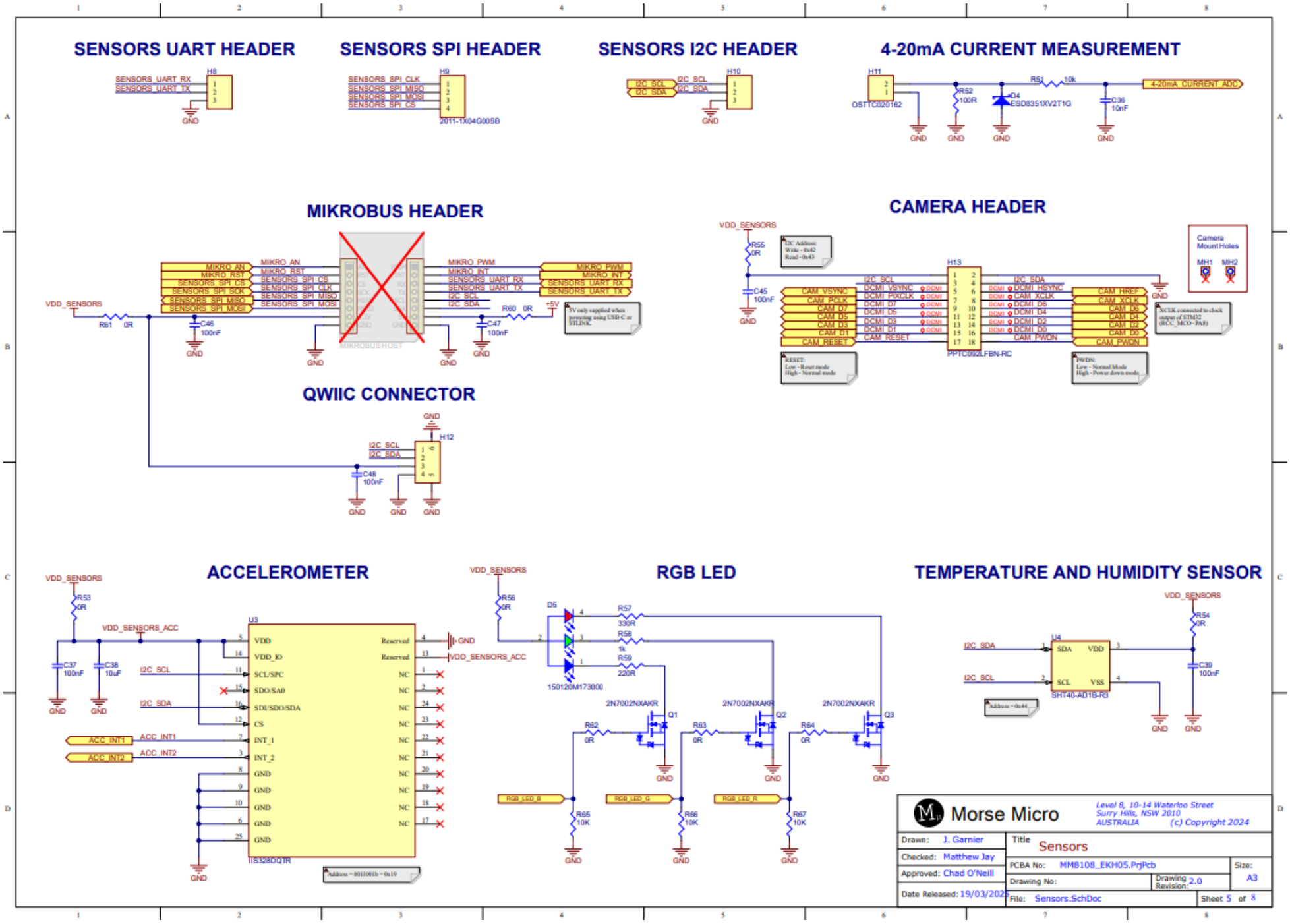
Note:

- *All unused pins should be left floating; do not ground*
- *All GND pins must be well grounded*
- *The area around the module should be free of any ground planes, power planes, trace routings, or metal for 6 mm from the module antenna position, in all directions.*
- *Traces should not be routed underneath the module*

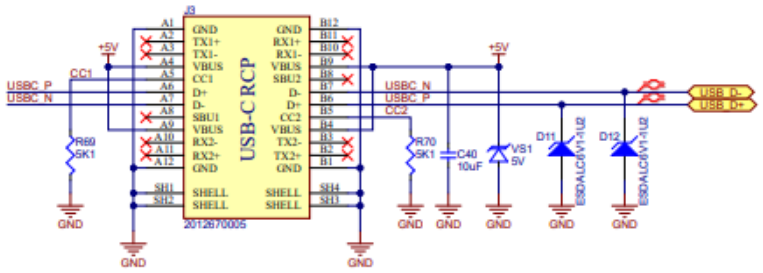


Morse Micro
Level 8, 10-14 Waterloo Street
Surry Hills, NSW 2010
AUSTRALIA
(c) Copyright 2024

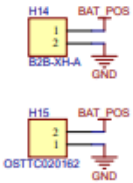
Drawn: J. Garnier	Title: BLE		
Checked: Matthew Jay	PCBA No: MM8108_EKH05.PrjPcb	Drawing Revision: 2.0	Size: A4
Approved: Chad O'Neill	Drawing No:		
Date Released: 19/03/2025	File: BLE_Module.SchDoc	Sheet 4 of 8	


SENSORS UART HEADER
SENSORS_UART_RX
SENSORS_UART_TX
H8
GND
SENSORS SPI HEADER
SENSORS_SPI_CLK
SENSORS_SPI_MISO
SENSORS_SPI_MOSI
SENSORS_SPI_CS
H9
2011-1X04G00SB
SENSORS I2C HEADER
I2C_SCL
I2C_SDA
H10
4-20mA CURRENT MEASUREMENT
H11
OSTTC020162
R52
100R
D4
ESD8351XV2T1G
R51
10k
C36
10nF
4-20mA_CURRENT_ADC
MIKROBUS HEADER
MIKRO_AN
MIKRO_RST
SENSORS_SPI_CS
SENSORS_SPI_CLK
SENSORS_SPI_MISO
SENSORS_SPI_MOSI
MIKRO_PWM
MIKRO_INT
SENSORS_UART_RX
SENSORS_UART_TX
I2C_SCL
I2C_SDA
MIKROBUSHOST
VDD_SENSORS
R61 0R
C46
100nF
R60 0R
+5V
C47
100nF
CAMERA HEADER
R55
0R
C45
100nF
H13
PPTC092LFBN-RC
CAM_VSYNC
CAM_PCLK
CAM_D7
CAM_D6
CAM_D5
CAM_D4
CAM_D3
CAM_D2
CAM_D1
CAM_D0
CAM_RESET
CAM_HREF
CAM_XCLK
CAM_PWDN
DCMI_VSYNC
DCMI_PIXCLK
DCMI_D7
DCMI_D6
DCMI_D5
DCMI_D4
DCMI_D3
DCMI_D2
DCMI_D1
DCMI_D0
DCMI_HSYNC
Camera Mount Holes
MH1
MH2
QWIIC CONNECTOR
H12
C48
100nF
ACCELEROMETER
R53
0R
VDD_SENSORS_ACC
C37
100nF
C38
10uF
U3
IIS328DQTR
VDD
VDD_IO
SCL/SPC
SDO/SA0
SDI/SDO/SDA
CS
INT_1
INT_2
Reserved
NC
ACC_INT1
ACC_INT2
RGB LED
R56
0R
D5
150120M173000
R57
330R
R58
1k
R59
220R
2N7002NXAKR
Q1
Q2
Q3
R62
R63
R64
0R
R65
R66
R67
10K
RGB_LED_B
RGB_LED_G
RGB_LED_R
TEMPERATURE AND HUMIDITY SENSOR
R54
0R
C39
100nF
U4
SHT40-AD1B-R3
SDA
SCL
VDD
VSS
Morse Micro
Level 8, 10-14 Waterloo Street
Surry Hills, NSW 2010
AUSTRALIA
(c) Copyright 2024
Drawn: J. Garnier
Checked: Matthew Jay
Approved: Chad O'Neill
Date Released: 19/03/2025
Title Sensors
PCBA No: MM8108_EKH05.PrjPcb
Drawing No:
Drawing Revision: 2.0
Size: A3
File: Sensors.SchDoc
Sheet 5 of 8

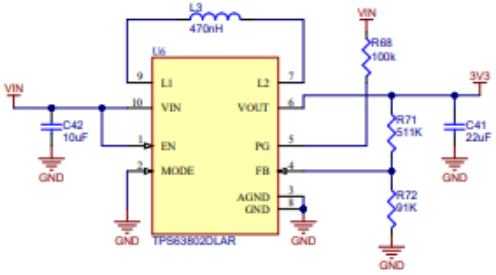
USB TYPE-C



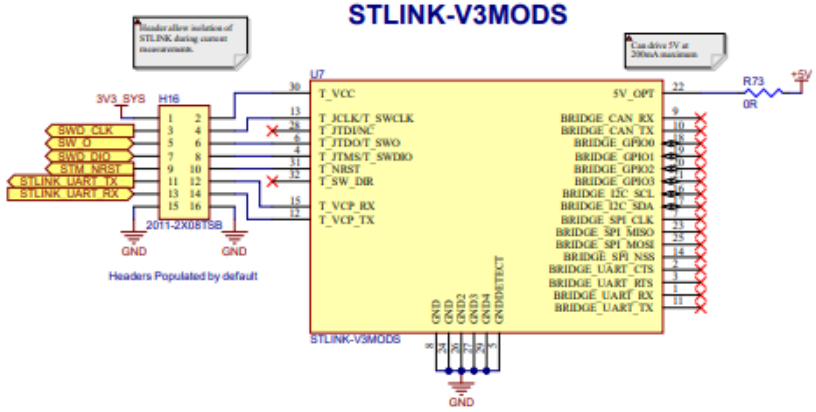
BATTERY CONNECTORS



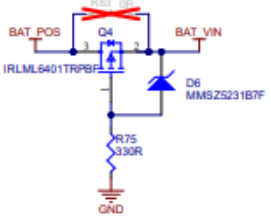
3V3 BUCK BOOST REGULATOR



STLINK-V3MODS

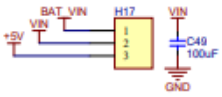


BATTERY REVERSE POLARITY PROTECTION

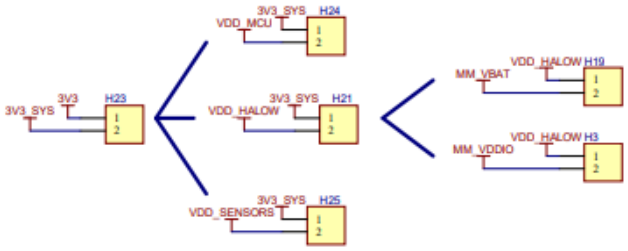


POWER INPUT SELECT

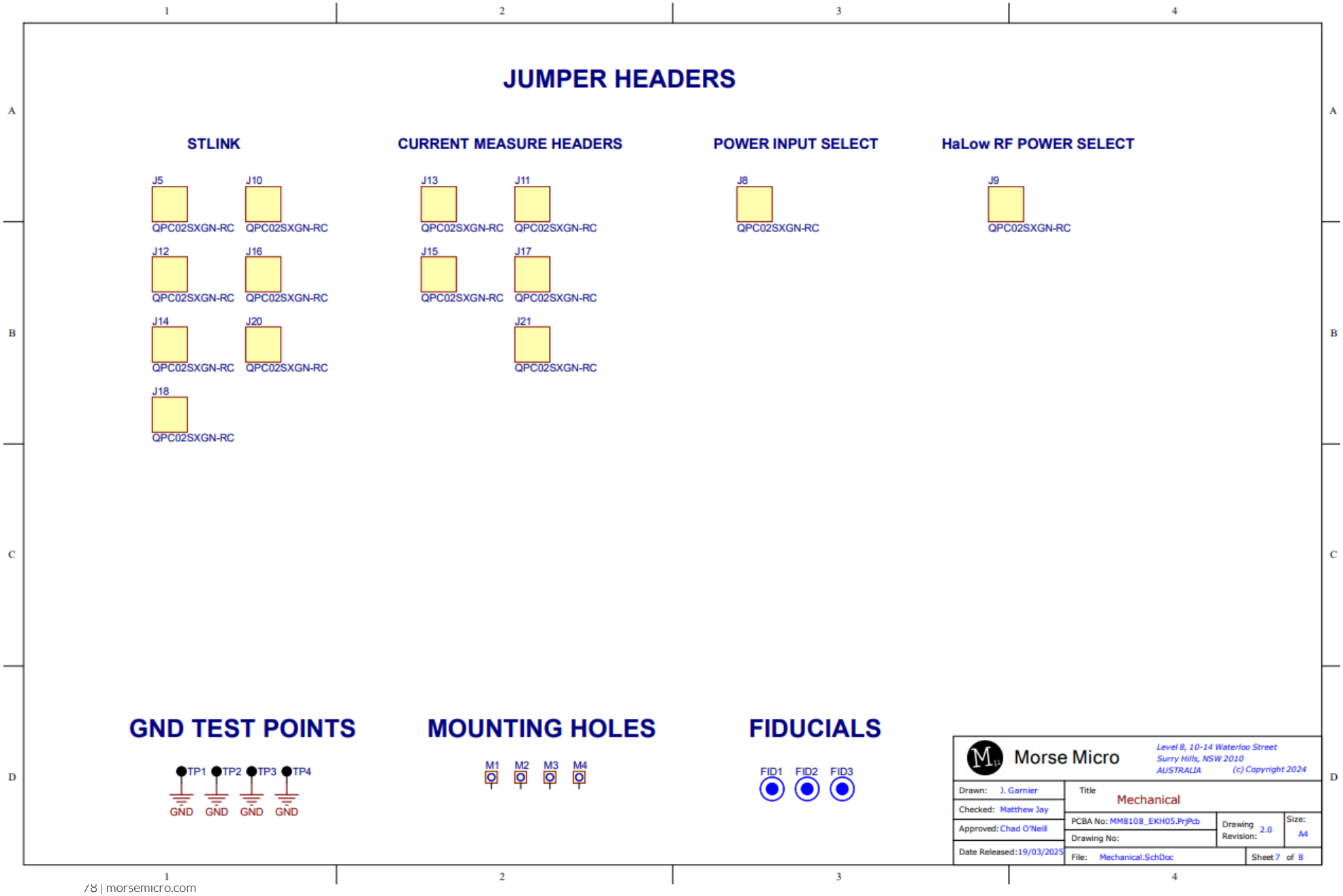
Options: Battery or USB



CURRENT MEASUREMENTS



Morse Micro	Level 8, 10-14 Waterloo Street Surry Hills, NSW 2010 AUSTRALIA	(c) Copyright 2024	
Drawn: J. Garnier	Title: Power		
Checked: Matthew Jay	PCBA No: MM8108_EKH05.PrjPcb		Size: A3
Approved: Chad O'Neill	Drawing No:	Drawing Revision: 2.0	
Date Released: 19/03/2025	File: Power.SchDoc		Sheet 6 of 8


JUMPER HEADERS
STLINK
J5
QPC02SXGN-RC
J10
QPC02SXGN-RC
J12
QPC02SXGN-RC
J16
QPC02SXGN-RC
J14
QPC02SXGN-RC
J20
QPC02SXGN-RC
J18
QPC02SXGN-RC
CURRENT MEASURE HEADERS
J13
QPC02SXGN-RC
J11
QPC02SXGN-RC
J15
QPC02SXGN-RC
J17
QPC02SXGN-RC
J21
QPC02SXGN-RC
POWER INPUT SELECT
J8
QPC02SXGN-RC
HaLow RF POWER SELECT
J9
QPC02SXGN-RC
GND TEST POINTS
TP1
TP2
TP3
TP4
GND
GND
GND
GND
MOUNTING HOLES
M1
M2
M3
M4
FIDUCIALS
FID1
FID2
FID3
Morse Micro
Level 8, 10-14 Waterloo Street
Surry Hills, NSW 2010
AUSTRALIA
(c) Copyright 2024
Drawn: J. Garnier
Checked: Matthew Jay
Approved: Chad O'Neill
Date Released: 19/03/2025
Title
Mechanical
PCBA No: MM8108_EKH05.PrjPcb
Drawing No:
Drawing Revision: 2.0
Size: A4
File: Mechanical.SchDoc
Sheet 7 of 8

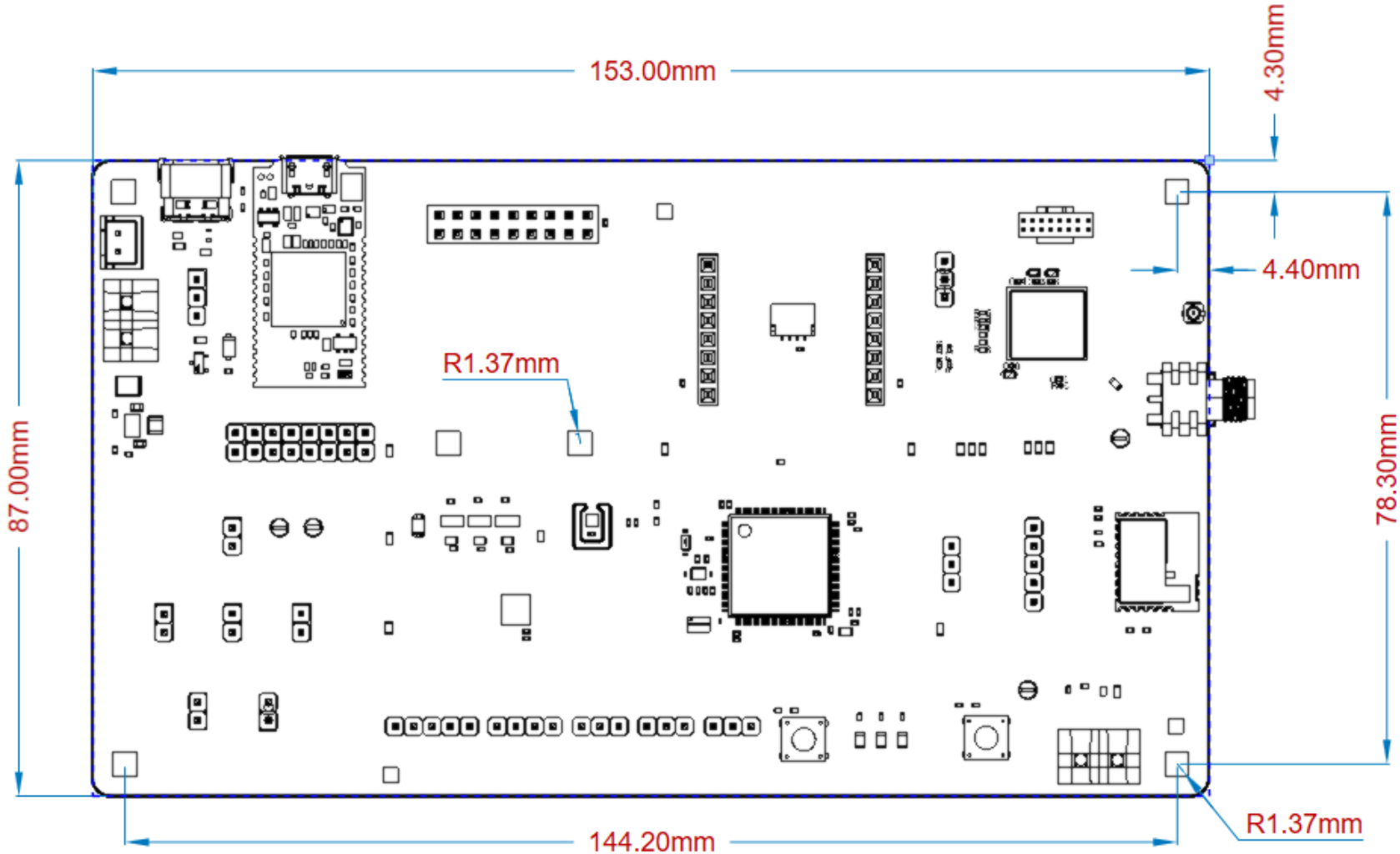
Revision History	
1.0	- Initial release.

Morse Micro

Level 8, 10-14 Waterloo Street
Surry Hills, NSW 2010
AUSTRALIA

Drawn: J. Garnier	Title: Revision History		
Checked: Matthew Jay			
Approved: Chad O'Neill	PCBA No: MM8108_EKH05.PrjPcb	Drawing Revision: 2.0	Size: A4
Date Released: 25/09/2024	Drawing No:		
	File: Revision_History.SchDoc	Sheet 8 of 8	

Appendix B: Mechanical Drawing



Morse Micro
reaching farther™

Morse Micro Pty. Ltd. Corporate Headquarters
Level 8, 10-14 Waterloo Street, Surry Hills, NSW 2010, AUSTRALIA

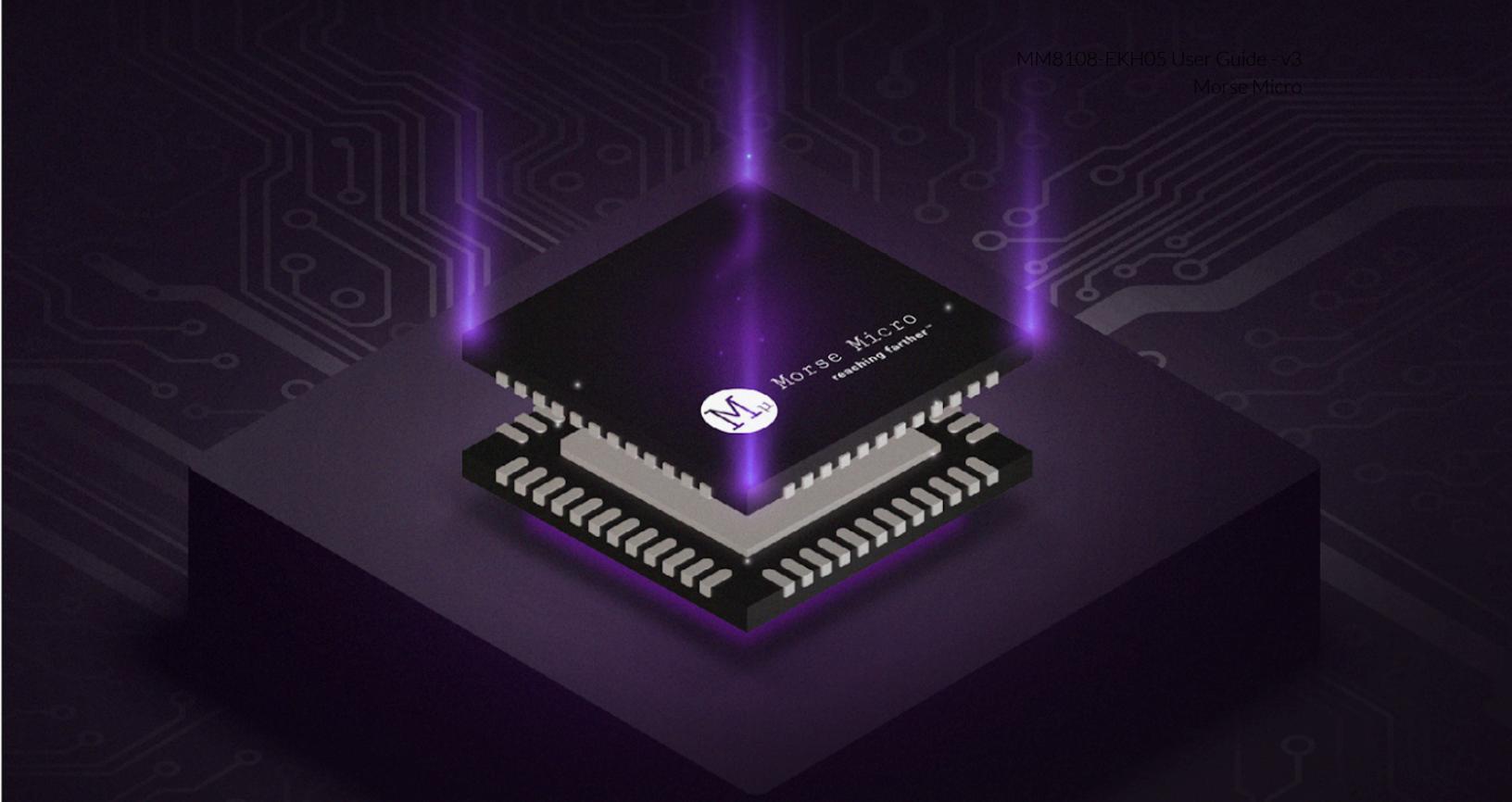
Morse Micro Inc. – USA
40 Waterworks Way
Irvine, CA 92618, UNITED STATES

Morse Micro – China
Floor 7, Room 08-210, Wework office 292 Yan'an Rd, Shangcheng District, Hangzhou 310003

USA: +1.949.501.7080
Australia: +61.02.905.45922
China: +86.571.229.30277
Email: sales@morsemicro.com
www.morsemicro.com

About Morse Micro

Morse Micro is producing IEEE 802.11ah / Wi-Fi HaLow solutions for Internet of Things (IOT) - based on a newly certified Wi-Fi standard called HaLow. Morse Micro is a VC-backed Startup headquartered in Sydney, Australia. Learn more at www.morsemicro.com



Morse Micro
reaching farther™