



Morse Micro
reaching farther™

MM6108-EKH05

USER GUIDE

Copyright © 2025 Morse Micro

Table of Contents

1 Introduction	4
2 Features	5
3 Development Environment	9
3.1 System Environment	9
3.2 Development Options	9
3.2.1 CMSIS Pack	9
3.2.2 MM IoT SDK	10
3.2.3 PlatformIO + MM IoT SDK	10
4 Getting Started	11
4.1 Default Jumper Configuration	12
4.2 AP Device Settings	13
4.2.1 Changing Channel, Bandwidth, DTIM Period	15
4.3 Software Examples	16
4.4 Viewing MM6108-EKH05 Demo HTTP Server	17
5 Software Development	19
5.1 Installing CMSIS Package	19
5.2 Build and Run Example Applications	22
5.2.1 UART Output	28
5.2.1.1 Windows	28
5.2.1.2 Linux	29
5.2.1.3 Mac OS	29
5.3 Changing Example Application	30
5.3.1 rf-test Example Considerations	32
5.4 Changing Country Code	33
5.5 Changing Other Example Configurations	34
5.6 Changing Between SPI and SDIO	35
5.7 Changing Network Stacks	36
5.8 Updating BLUENRG-M2SP Module Firmware	38
6 Hardware Layout and Configuration	41
6.1 Power Selection	41
6.2 Using an External Debugger/Programmer	42
6.3 Changing VFEM Voltage	43
6.4 Switching Between SDIO and SPI	44
6.5 Switching Between SMA and U.FL Connector	45
6.6 Disconnecting Sensors	46

7 Power Consumption Measurements	47
7.1 Power Consumption Measurement Points	47
7.1.1 General Structure	47
7.1.2 Power Measurement Considerations	48
7.1.2.1 HaLow and VFEM	48
7.1.2.2 Whole System Power Consumption	48
7.2 Power Consumption Measurement Procedure	50
7.3 Optional Debug GPIO Connections	51
7.4 Measuring DTIM Power	53
7.5 Measuring WNM Sleep Power	56
7.6 Current Spikes When Waking Up	59
7.7 Memory Retention Current Spikes	60
7.8 Negative Current Consumption	61
8 Troubleshooting	64
8.1 Device Is Not Programming	64
8.2 Poor HaLow Performance	66
8.3 High Power Consumption	67
9 Errata	68
10 Revision History	69
Appendix A: Schematics	70
Appendix B: Mechanical Drawing	78

1 Introduction

The MM6108-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 an Azurewave AW-HM593 module, incorporating Morse Micro's MM6108 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 MM6108-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 HaLow access point is required in addition to the MM6108-EKH05. Refer to [Section 4](#) for setup details.

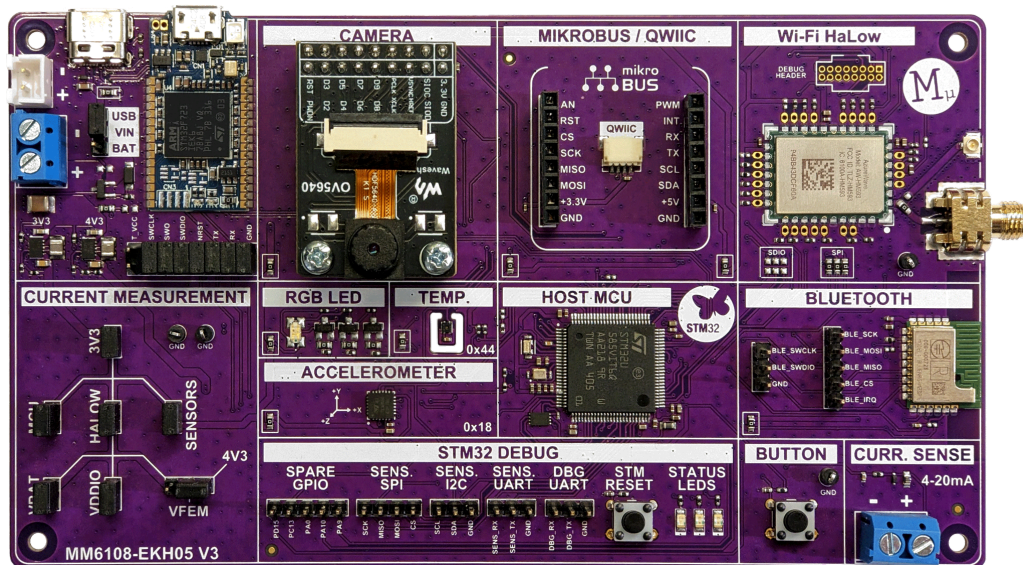


Figure 1. MM6108-EKH05 evaluation kit.

2 Features

This section highlights the key features of the MM6108-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.

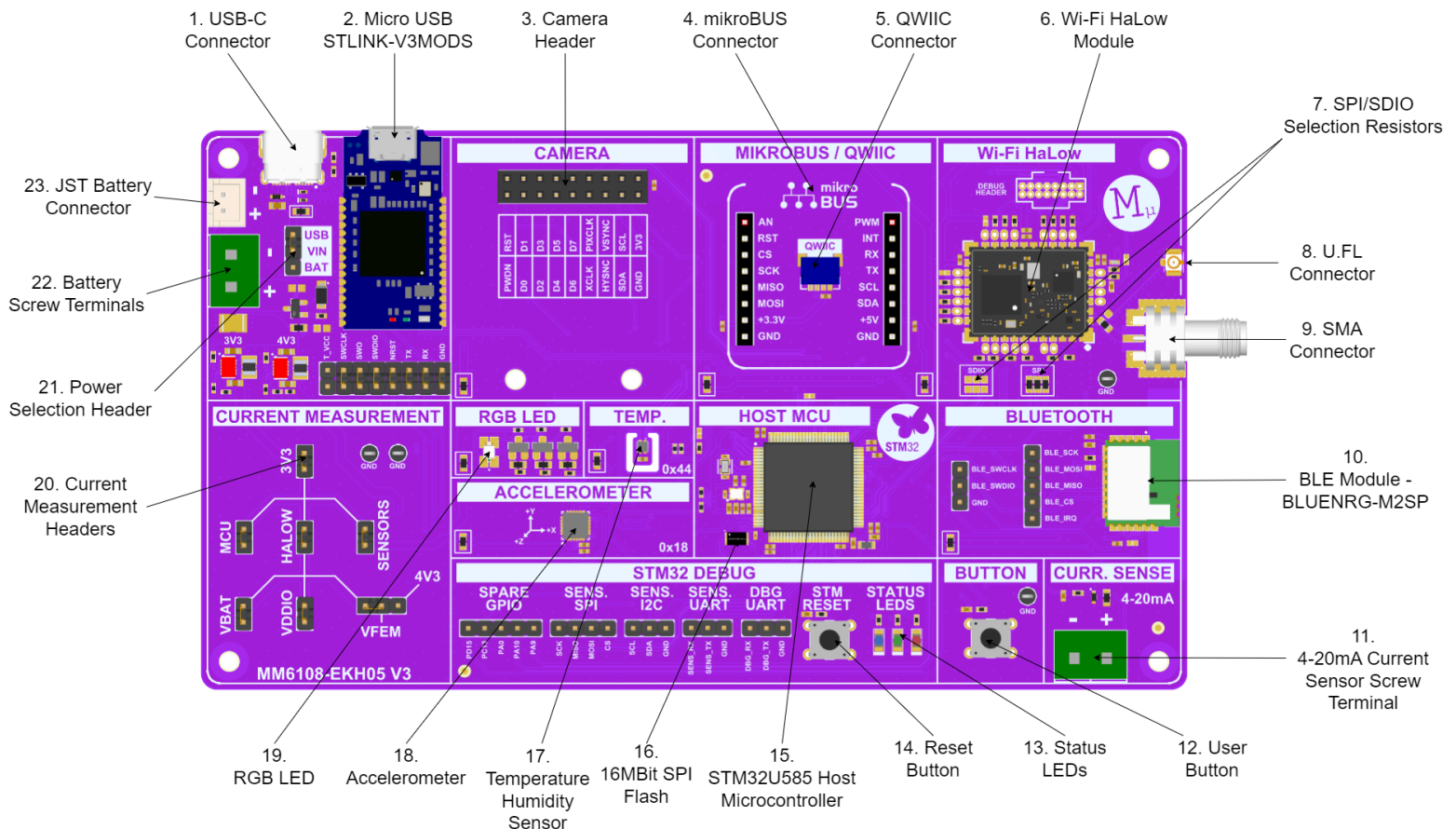


Figure 2. Features of the MM6108-EKH05.

#	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 MM6108-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	Azurewave AW-HM593 module with the MM6108 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 MM6108-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 MM6108-EKH05. The rest of this document assumes the CMSIS pack will be used and describes how to use it with the MM6108-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 MM6108 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 MM6108-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 MM6108-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 MM6108-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 MM6108-EKH05 **cannot** operate as an AP.

The MM6108-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.

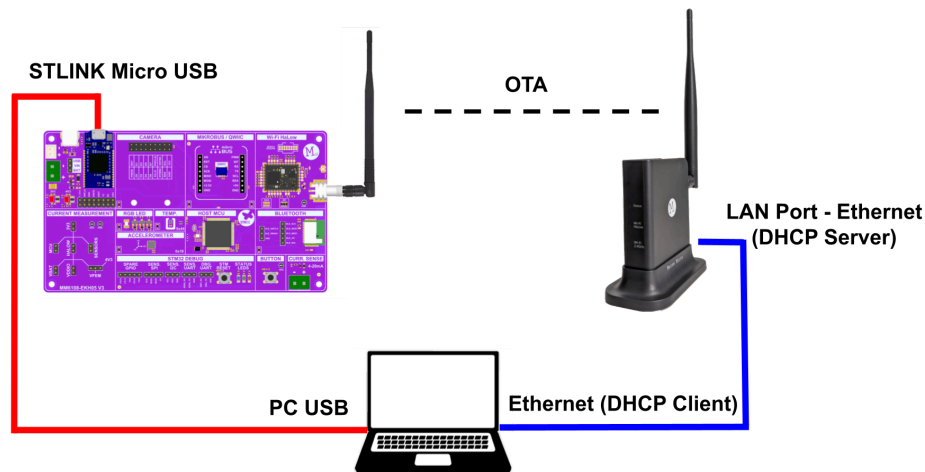


Figure 3. Hardware setup with antennas.

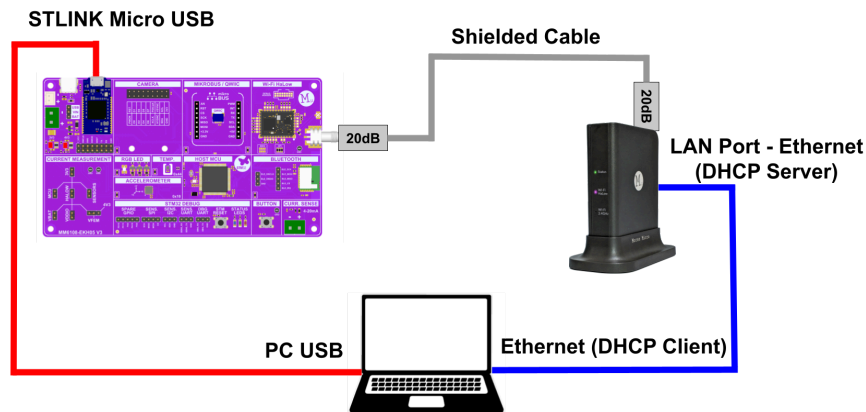


Figure 4. Hardware setup with shielded cable.

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

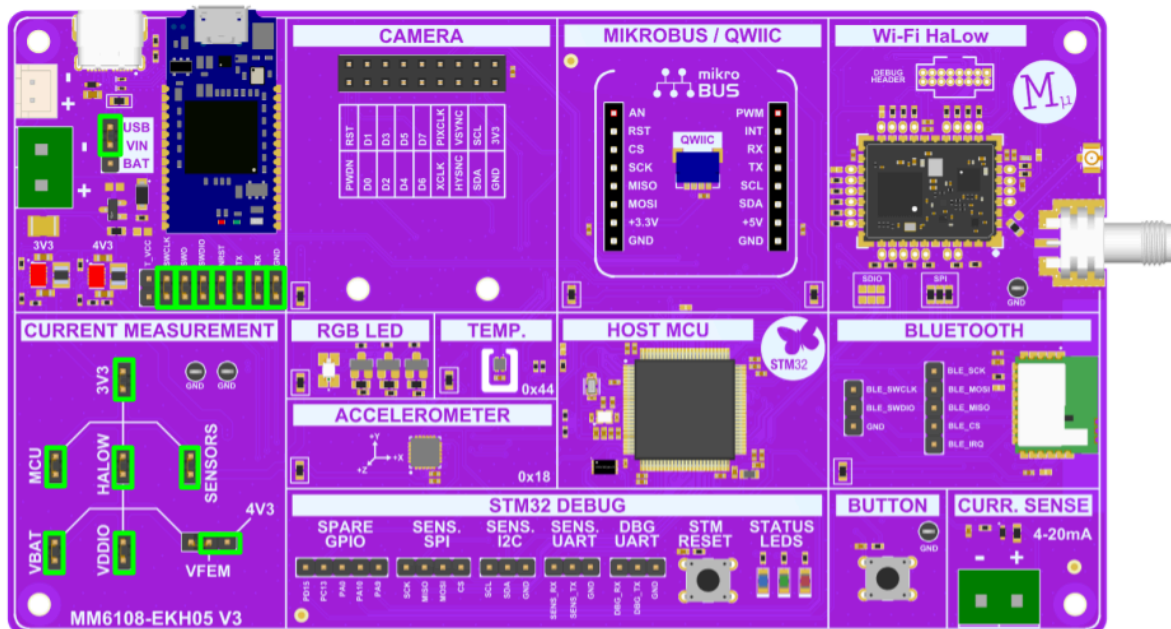


Figure 5. Default jumper placements on MM6108-EKH05.

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**.

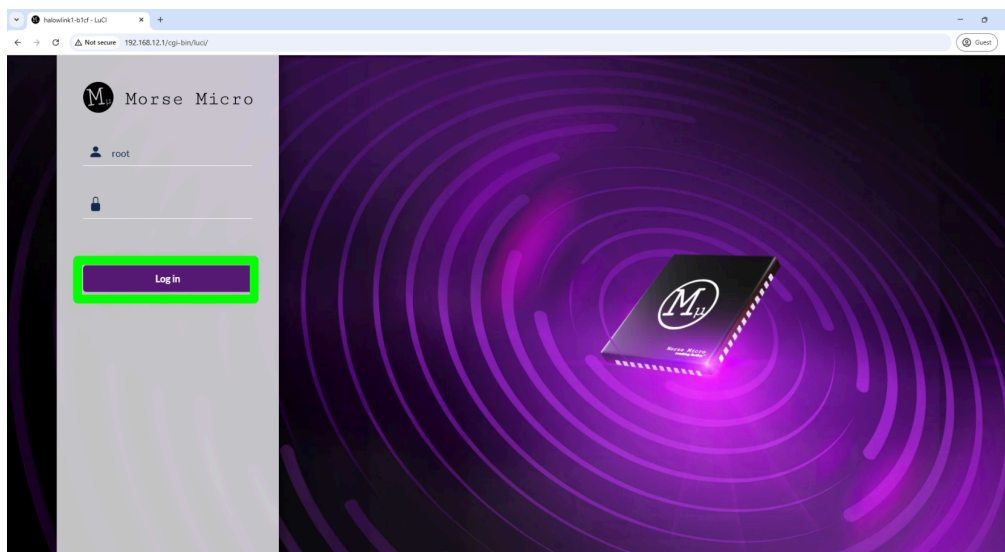


Figure 6. OpenWRT login screen.

Select **Quick Config**.

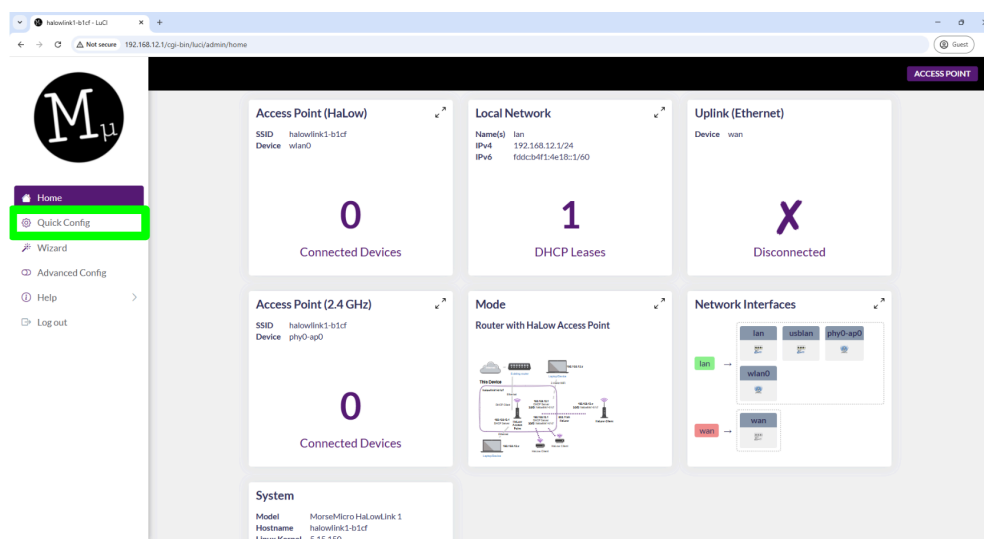


Figure 7. Highlighting the Quick Config menu.

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 MM6108-EKH05 example applications expect the following SSID and Passphrase. Set the following to ensure the MM6108-EKH05 associates with the AP.

SSID: MorseMicro

Passphrase: 12345678

The default **Country Code** loaded on the HaLowLink 1 and the MM6108-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 MM6108-EKH05, please see section [5.4](#).

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

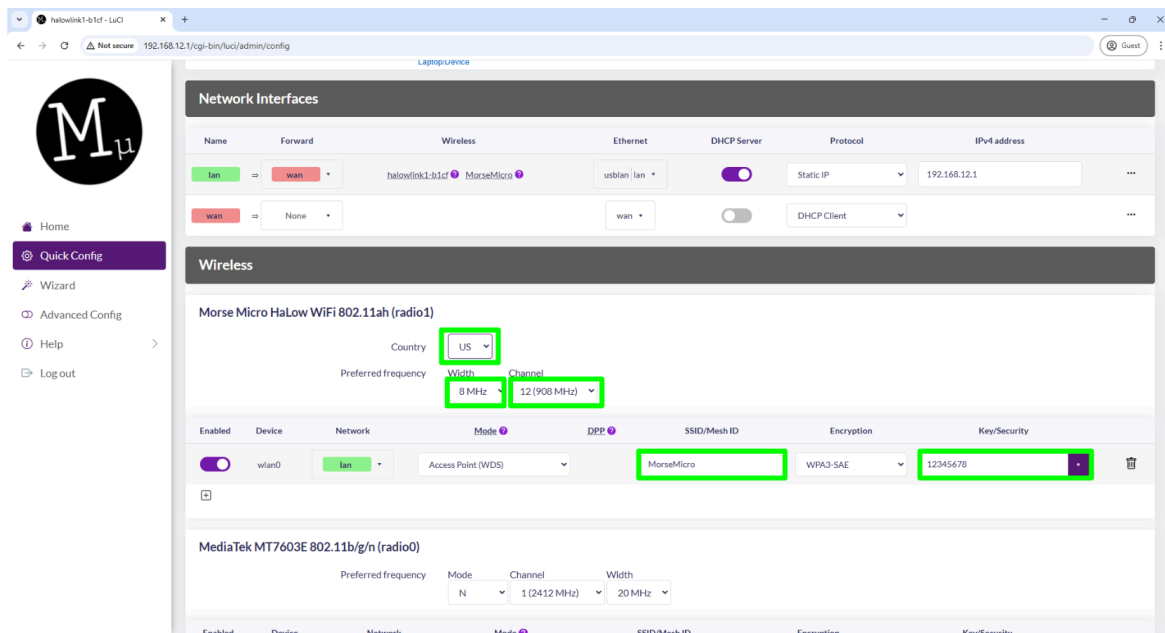


Figure 8. Quick config menu showing where to change country code, SSID, passphrase, bandwidth, and channel.

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.

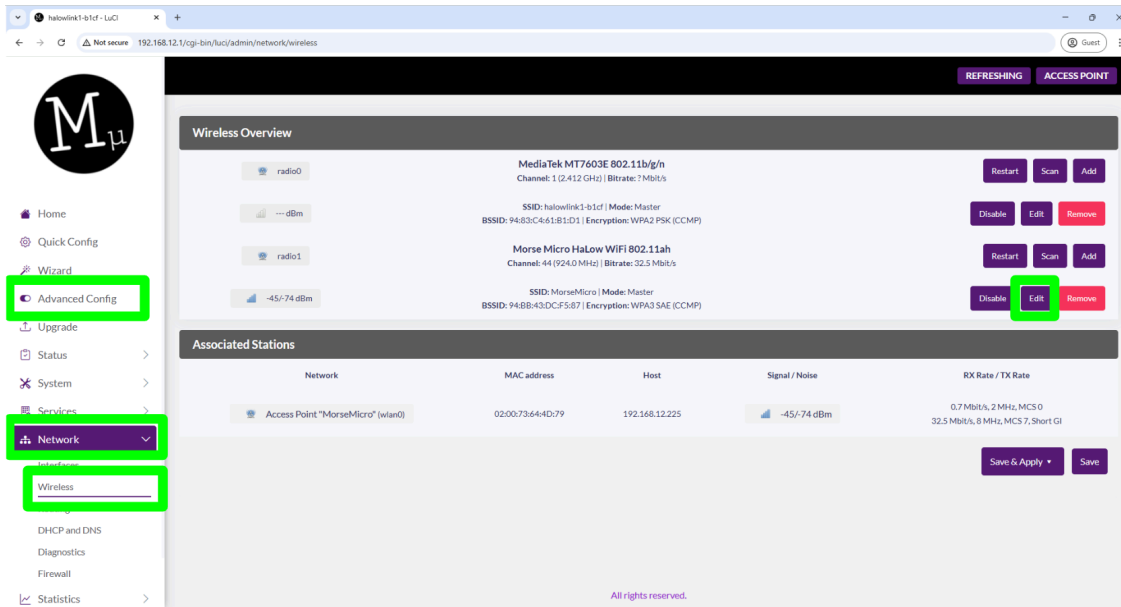


Figure 9. Wireless network configuration page on OpenWRT.

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.

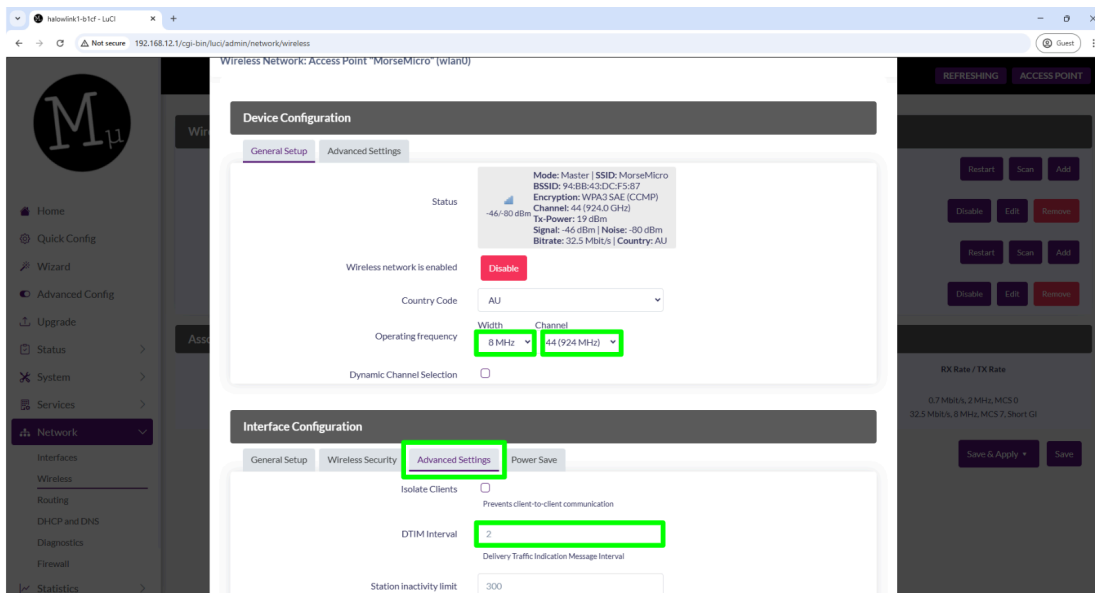


Figure 10. Advanced setup showing where to set DTIM interval, bandwidth, and channel.

4.3 Software Examples

The following table will describe the list of supported software examples that are included with the CMSIS pack for the MM6108-EKH05. The “EKH05 Demo” example is loaded on the device out of the box.

Software Example	Description
EKH05 Demo	As the default out-of-box example, the EKH05 Demo showcases the operation of all on-board peripherals of the MM6108-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.
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.

4.4 Viewing MM6108-EKH05 Demo HTTP Server

To view the HTTP server running on the MM6108-EKH05 device from a PC, the IP address of the MM6108-EKH05 needs to be found. To do this, navigate back to the **Home** section, and click on the **Connected Devices** link.

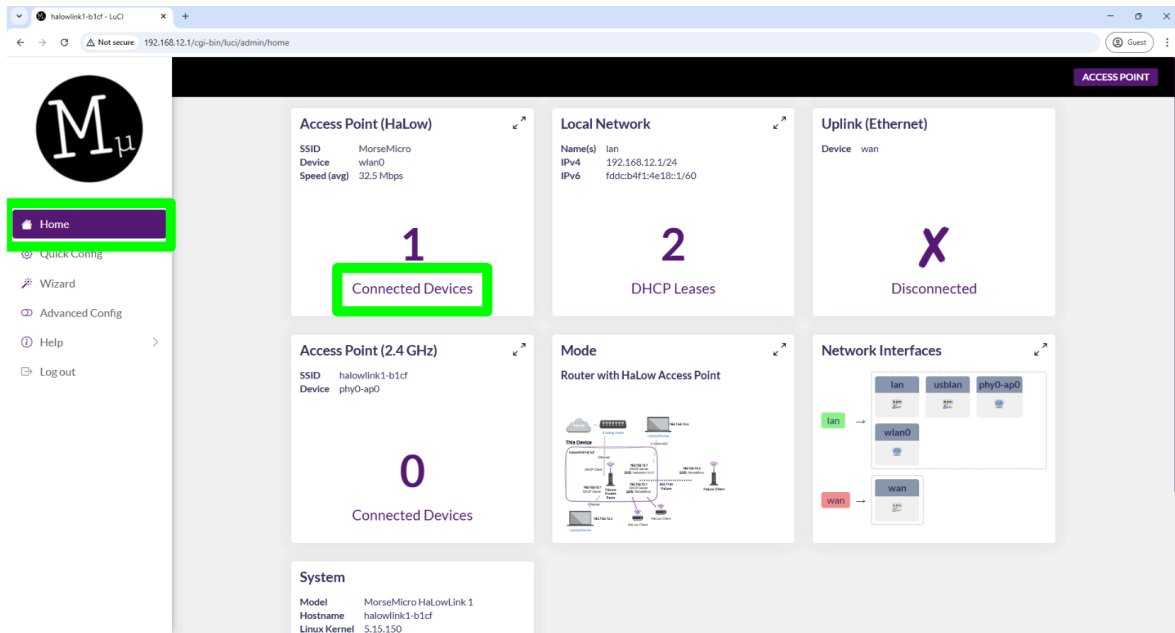


Figure 11. Home OpenWRT menu showing connected devices.

This will bring up a list of DHCP leases the AP has distributed to connected stations. In this case, the MM6108-EKH05 connected to the AP has an IP address of 192.168.12.225.

Access Point (HaLow)

SSID MorseMicro
Device wlan0
IPv4 192.168.12.1
IPv6 fdcc:b4f1:4e18::1/60
Encryption WPA3 SAE (CCMP)

Associated devices

► Connect a new device

MAC Address	IPv4	IPv6	Time connected	Noise	Signal
02:00:73:64:4D:79	192.168.12.225	fe80::73ff:fe64:4d79	7 min(s)	-78 dBm	-45 dBm

Figure 12. Connected devices and their connection details.

Navigate to the IP address of the MM6108-EKH05 (192.168.12.225 in this case) in a web browser to view the contents of the HTTP server.

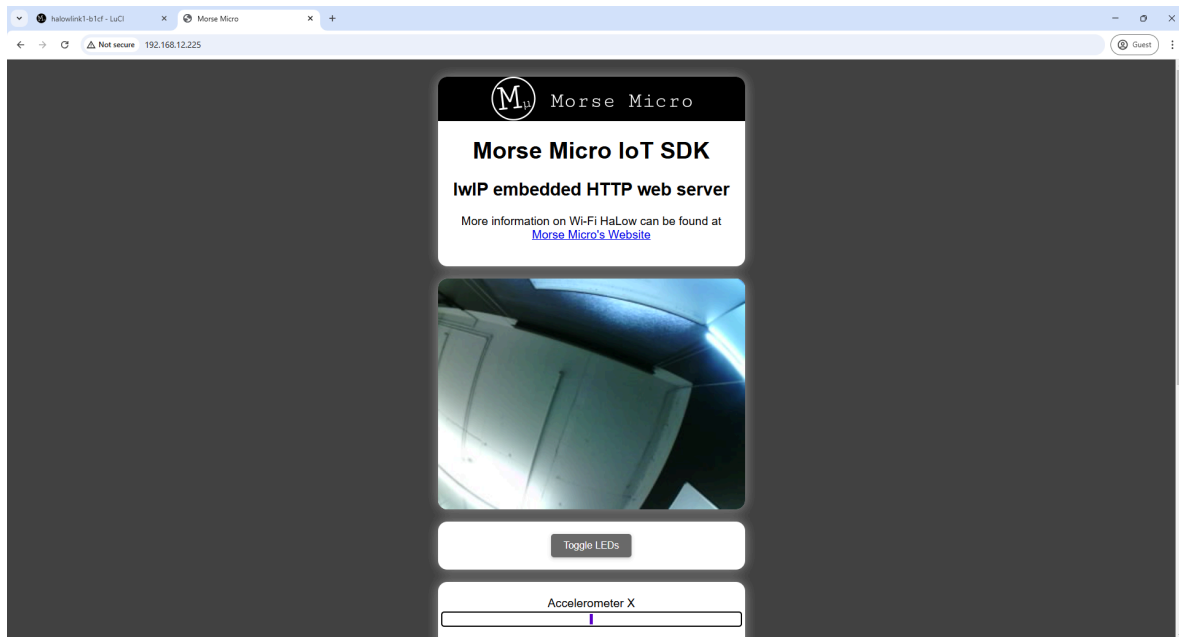


Figure 13. HTTP server running on MM6108-EKH05 demo application.

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:

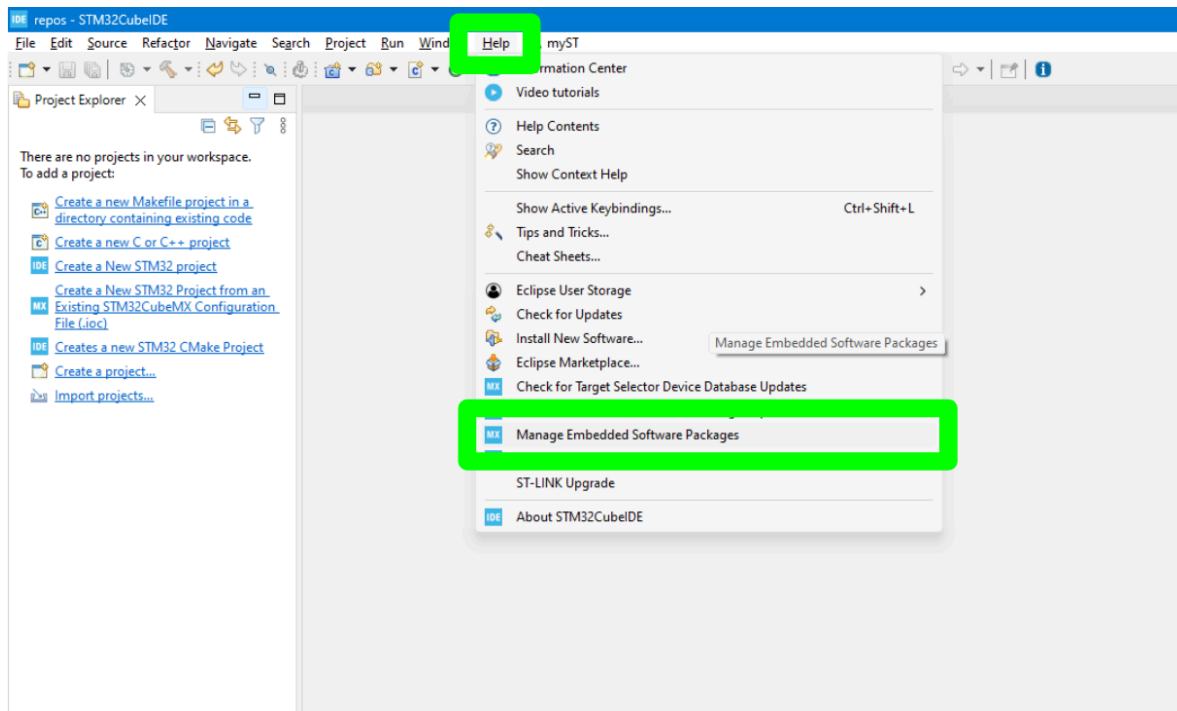


Figure 14. STM32CubeIDE help menu dropdown.

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

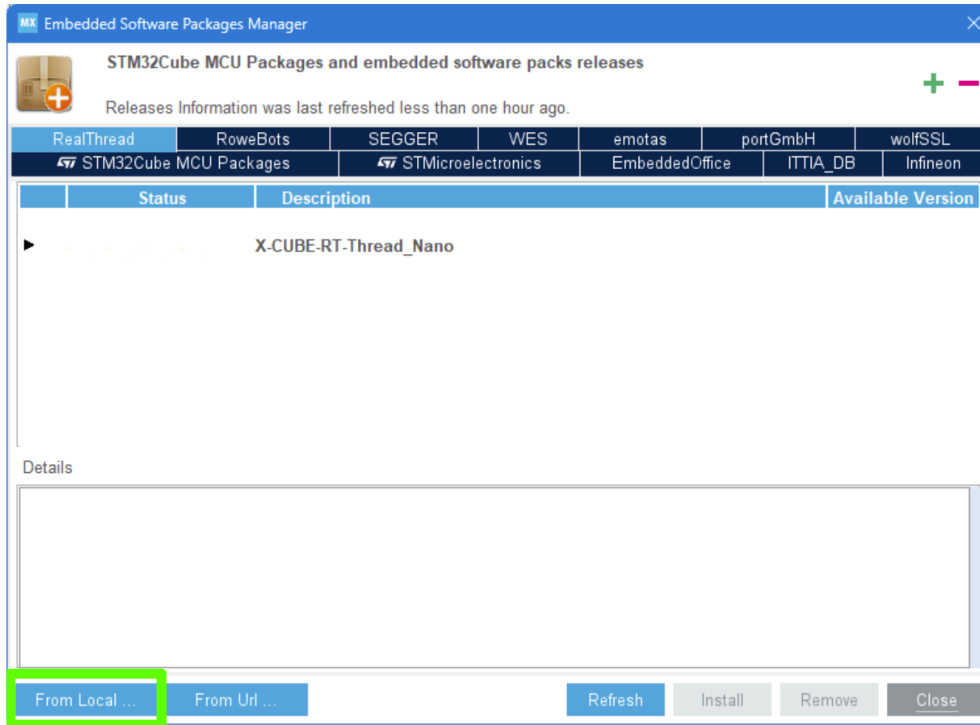


Figure 15. Embedded software packages manager menu.

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

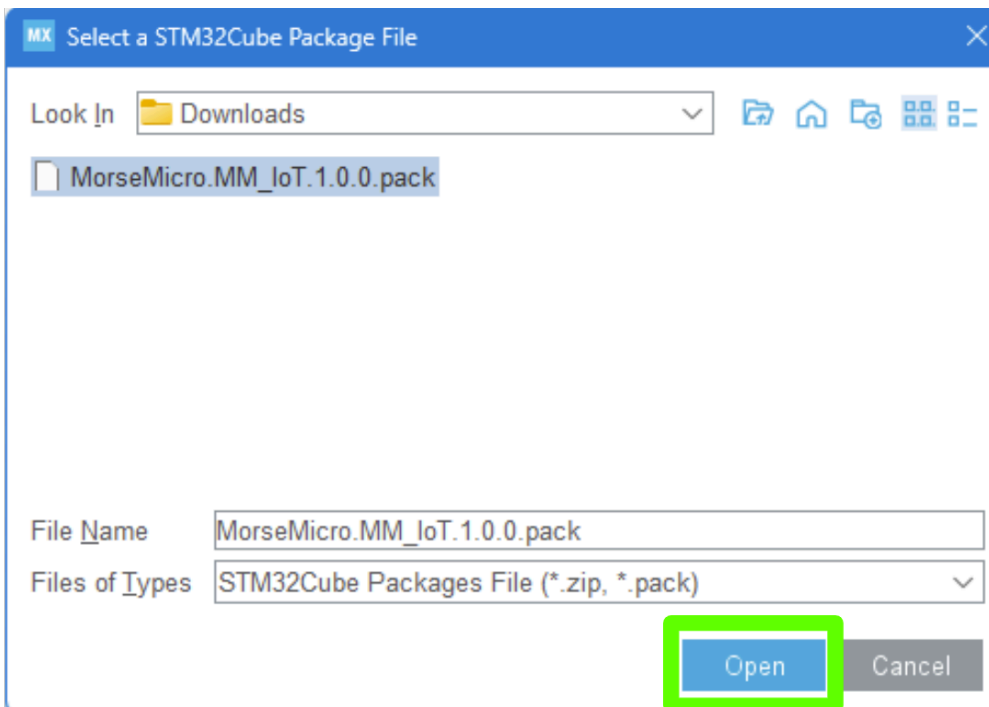


Figure 16. Opening CMSIS pack from downloaded location.

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

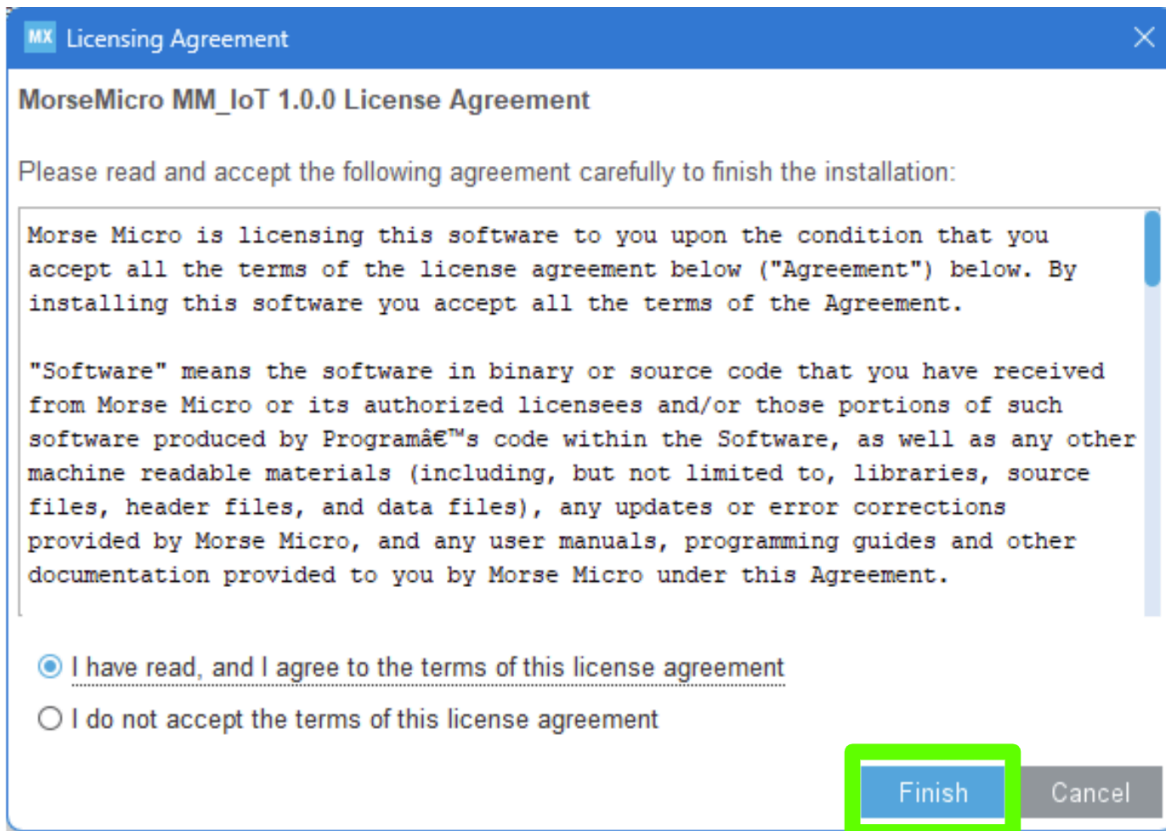


Figure 17. License agreement menu for installing CMSIS pack.

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.

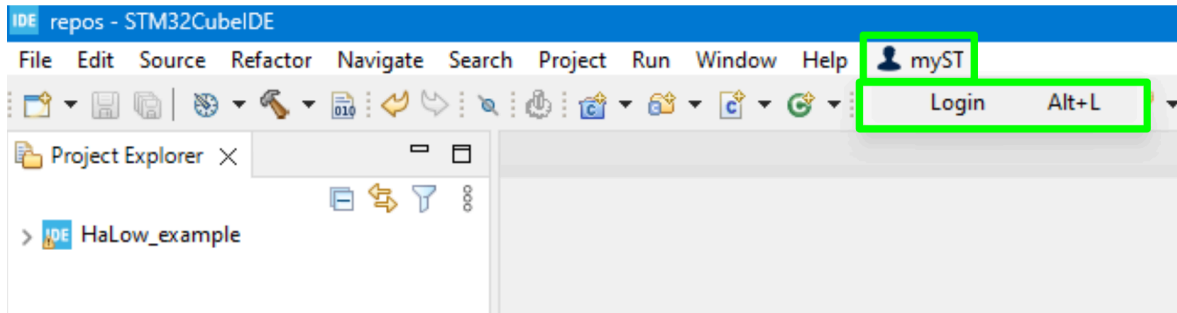


Figure 18. Where to login to ST account in STM32CubeIDE.

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

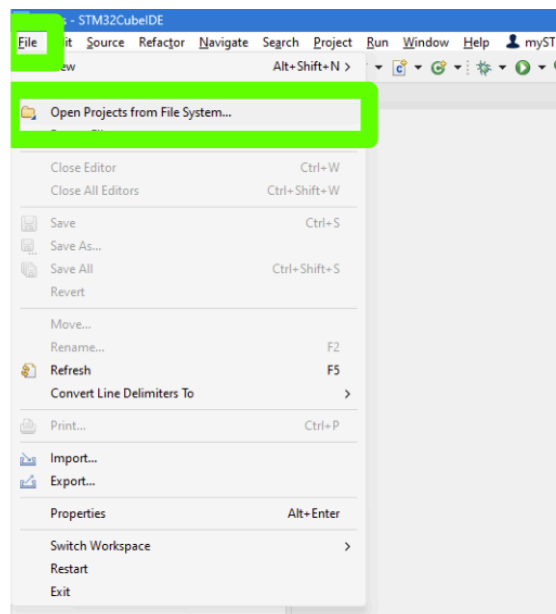


Figure 19. Where to open project in STM32CubeIDE.

Click on **Directory...**:

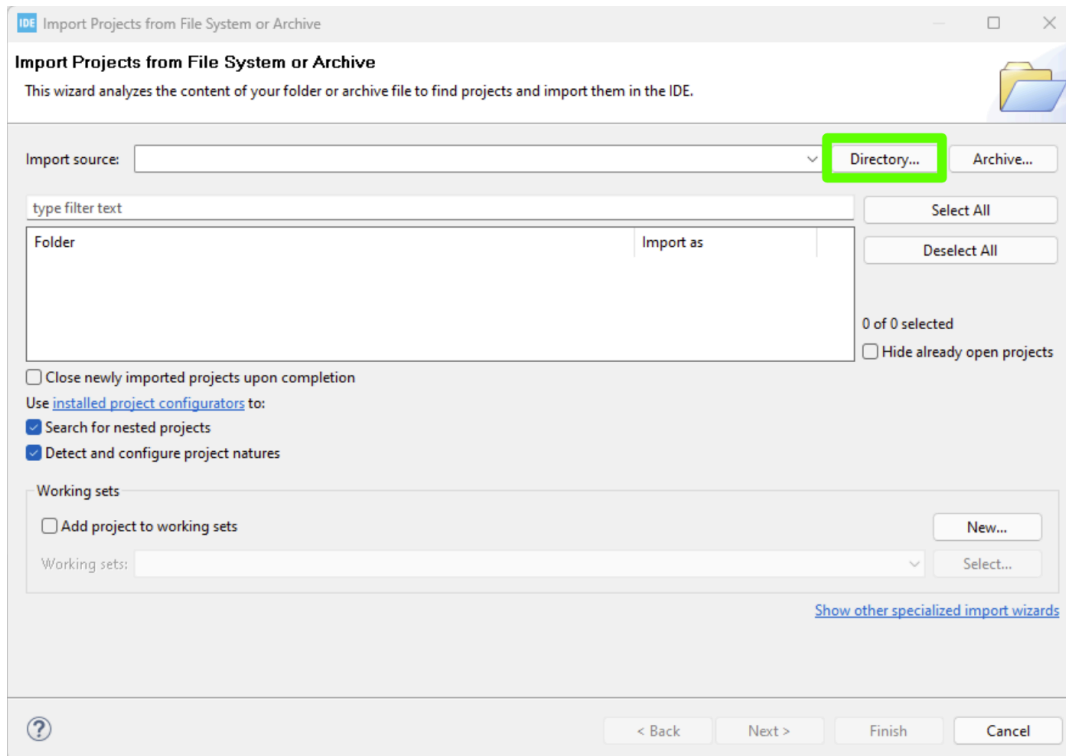


Figure 20. Highlighting directory button to search for project.

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\1.5.0\HaLow_example_spi
```

On Linux, the default path is:

```
/home/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.5.0/HaLow_example_spi
```

On Mac, the default path is:

```
/Users/USER_NAME/STM32Cube/Repository/Packs/MorseMicro/MM_IoT/1.5.0/HaLow_example_spi
```

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

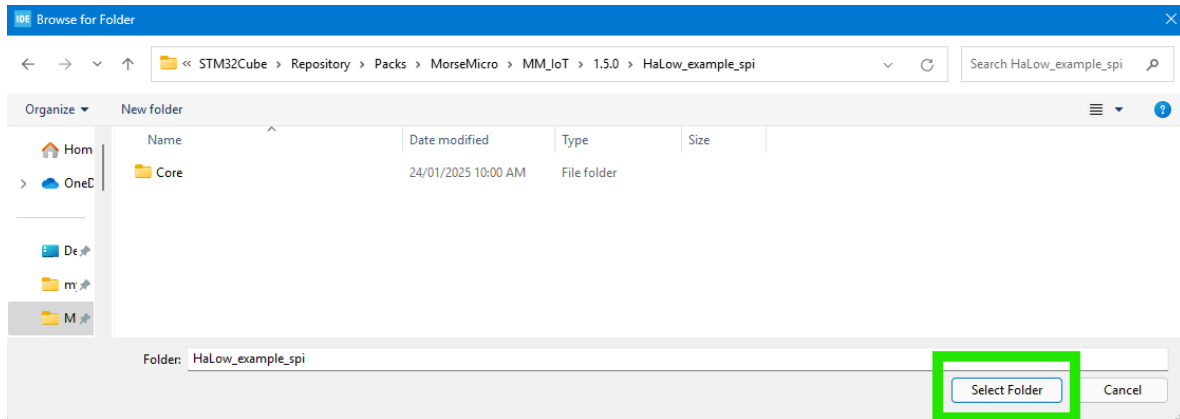


Figure 21. Example location of where to find the example application on a windows machine.

Click **Finish**:

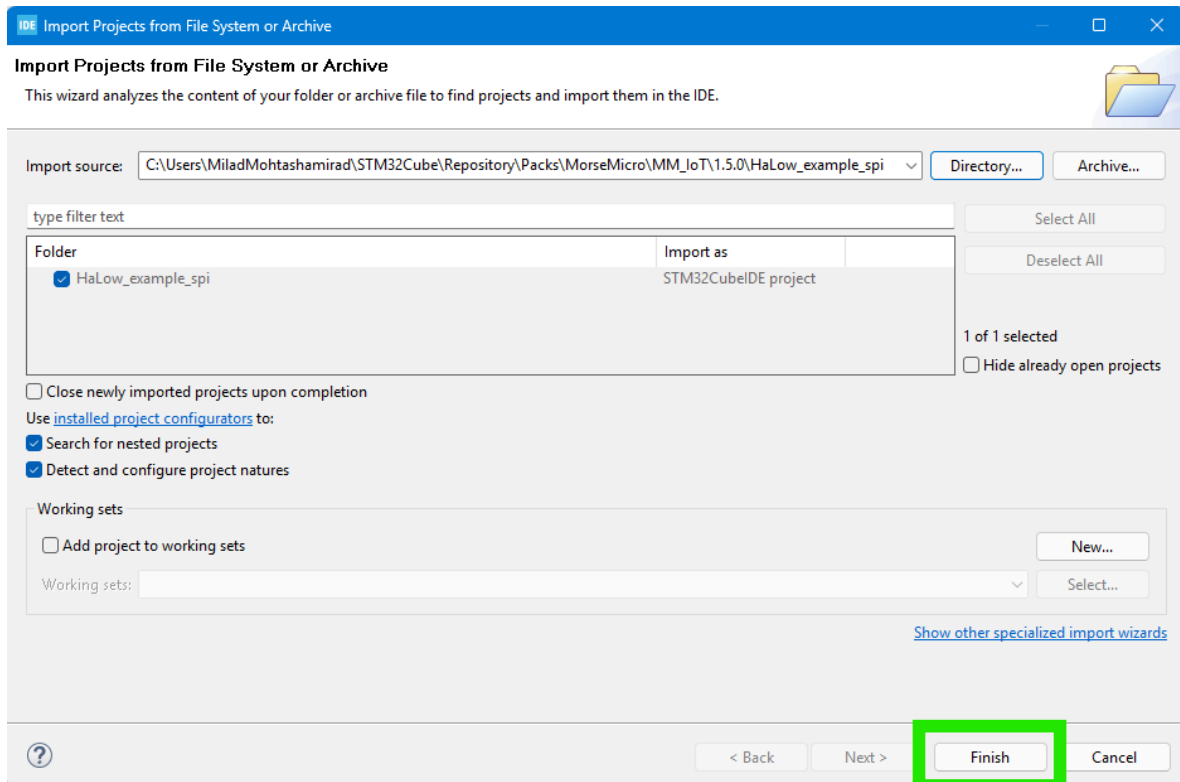


Figure 22. Finish menu importing the example application.

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

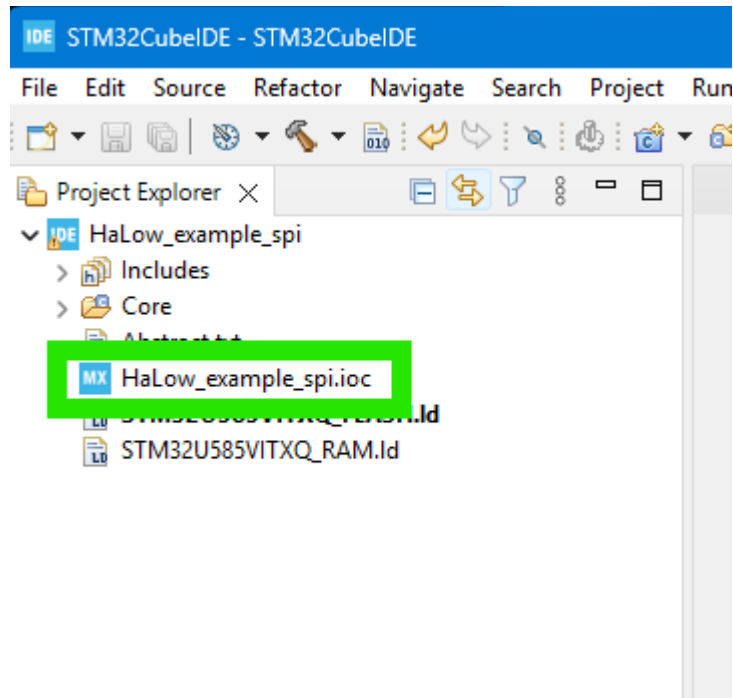


Figure 23. Highlighting .ioc file in project explorer.

Click **Continue**.

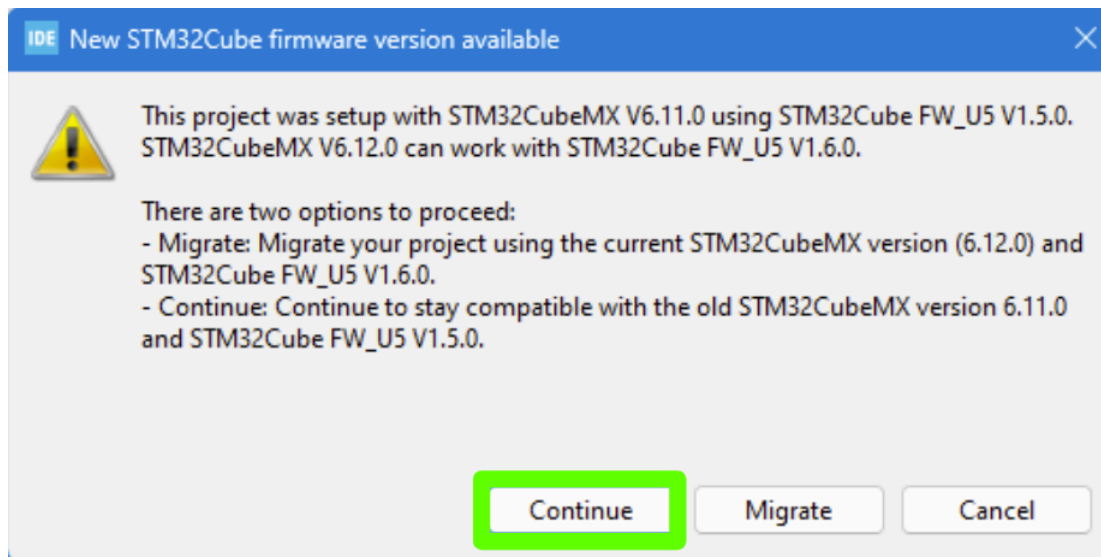


Figure 24. Pop-up menu after opening .ioc file.

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

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

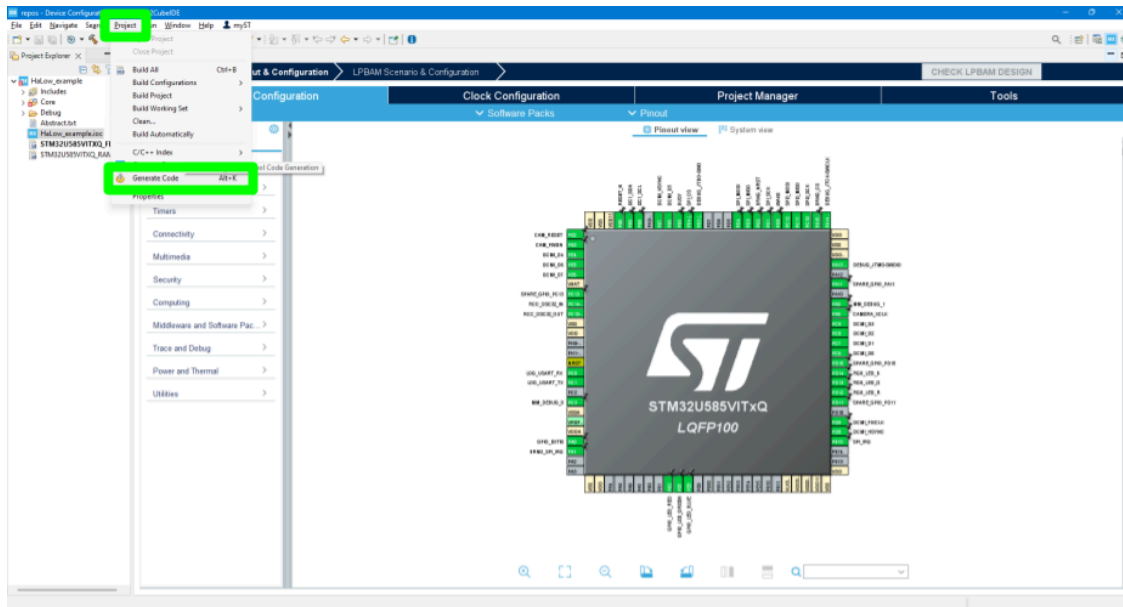


Figure 25. Highlighting where to generate code from after opening the .ioc file.

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:

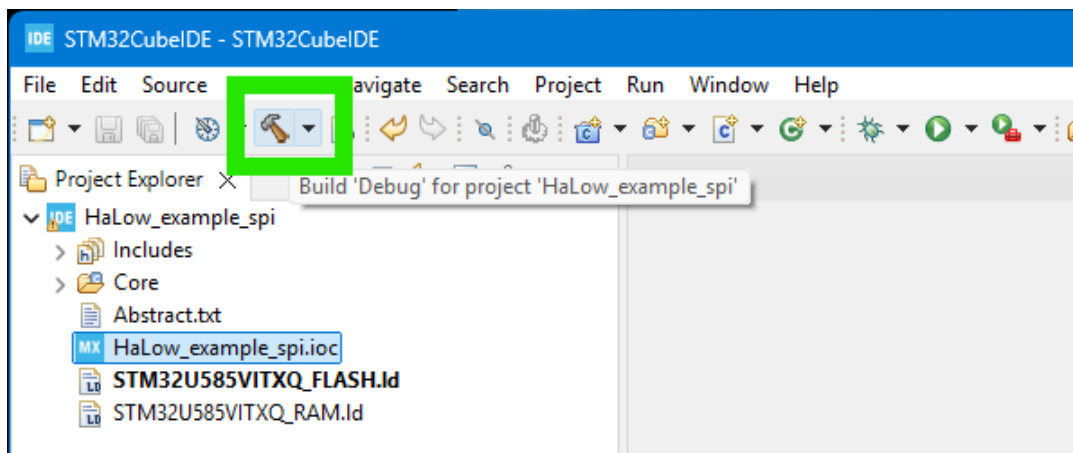


Figure 26. Icon in STM32CubeIDE for building application.

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

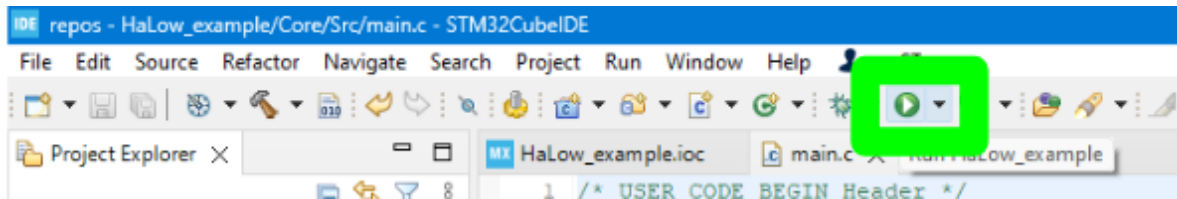


Figure 27. Icon in STM32CubeIDE for running application.

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

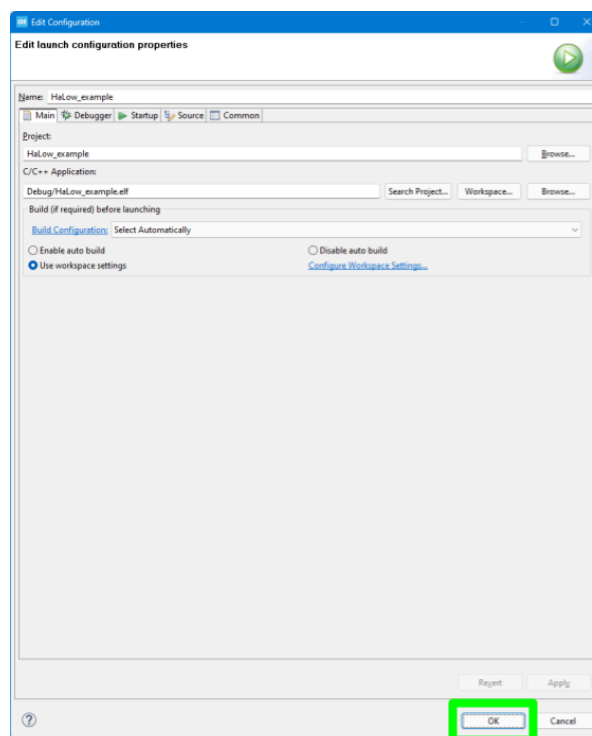


Figure 28. Run configuration menu.

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.

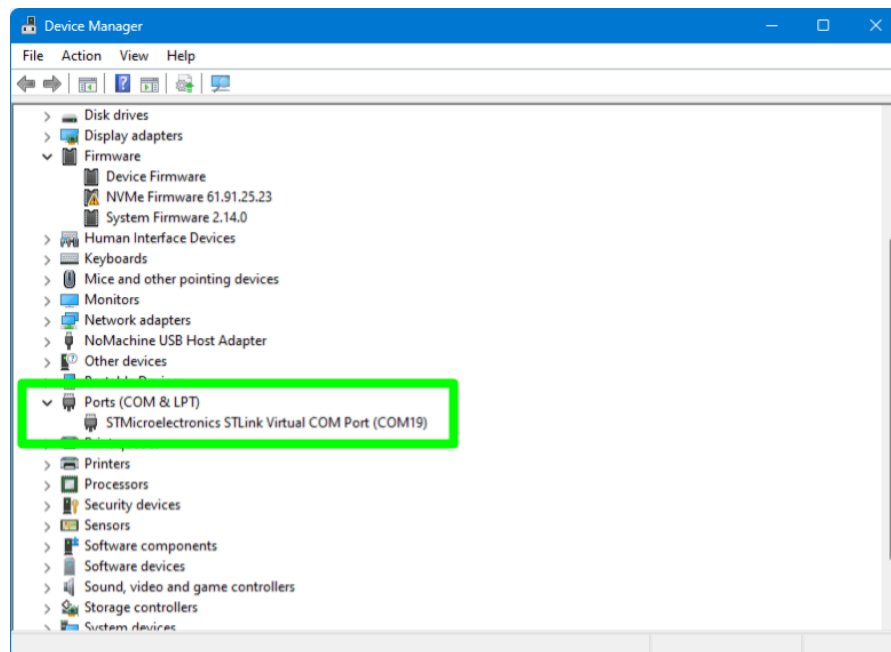


Figure 29. Device manager on windows machine showing where the STLINK-V3 COM port is located.

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.

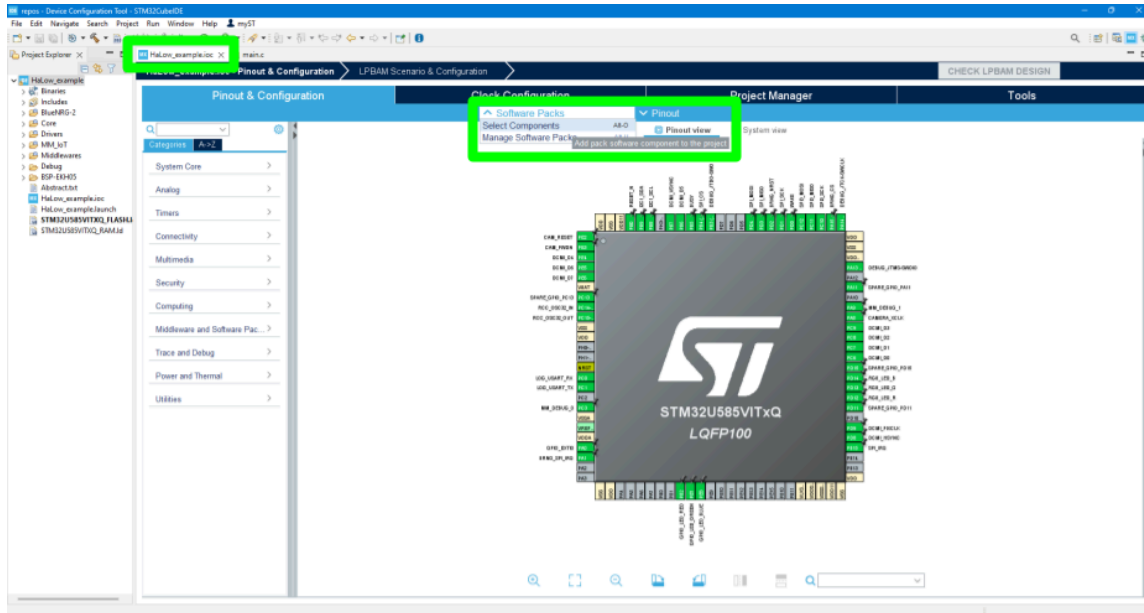


Figure 30. Select components menu in the .ioc file.

Click on the MorseMicro.MM_IoT arrow, and then select the desired application from the drop down list.

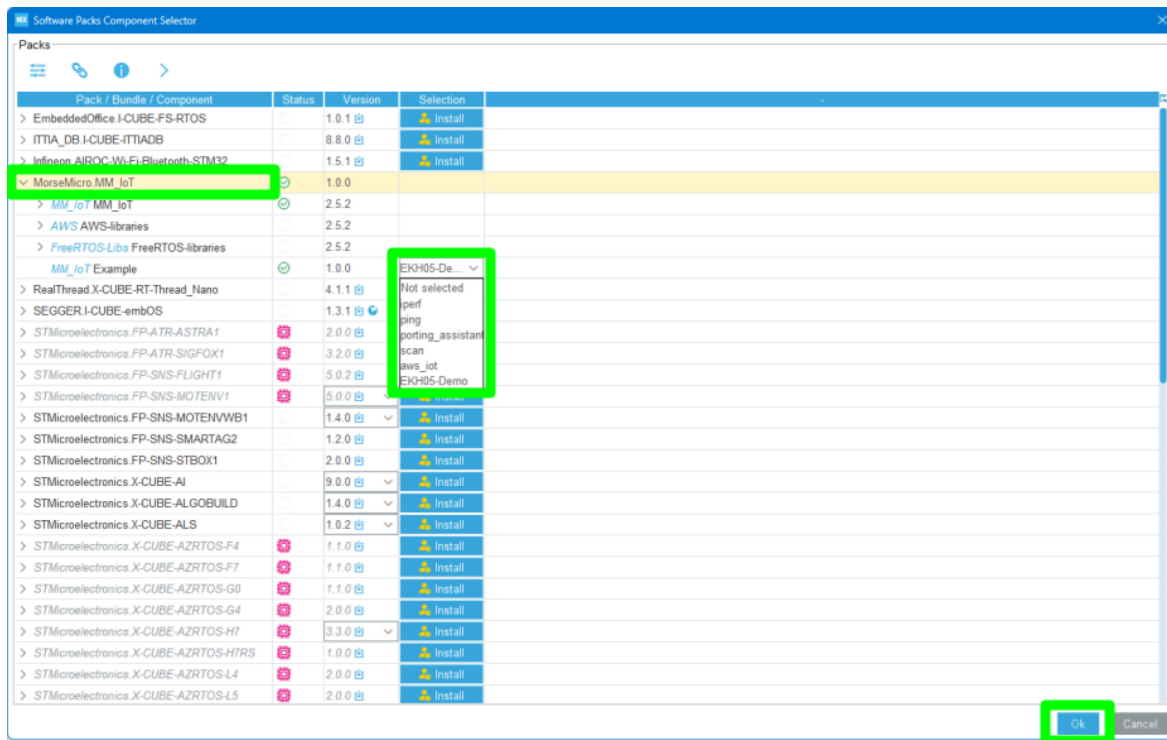


Figure 31. List of example applications within the CMSIS pack.

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**:

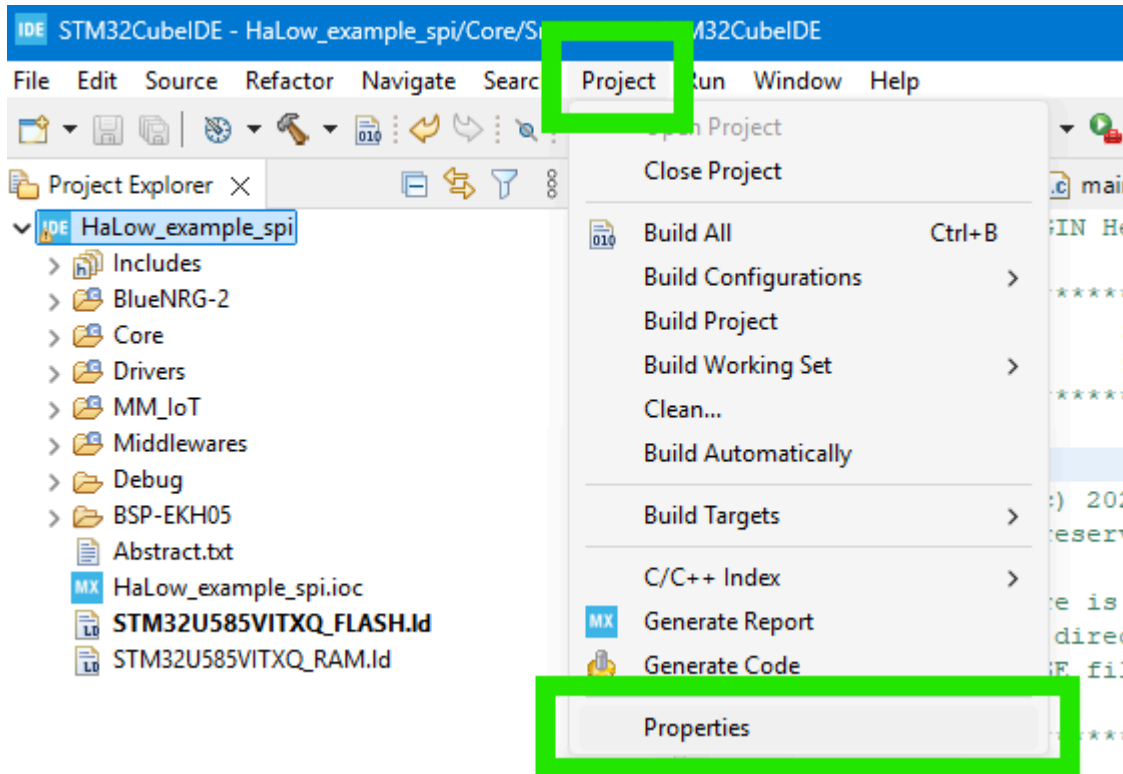


Figure 32. Project properties menu in STM32CubeIDE.

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:

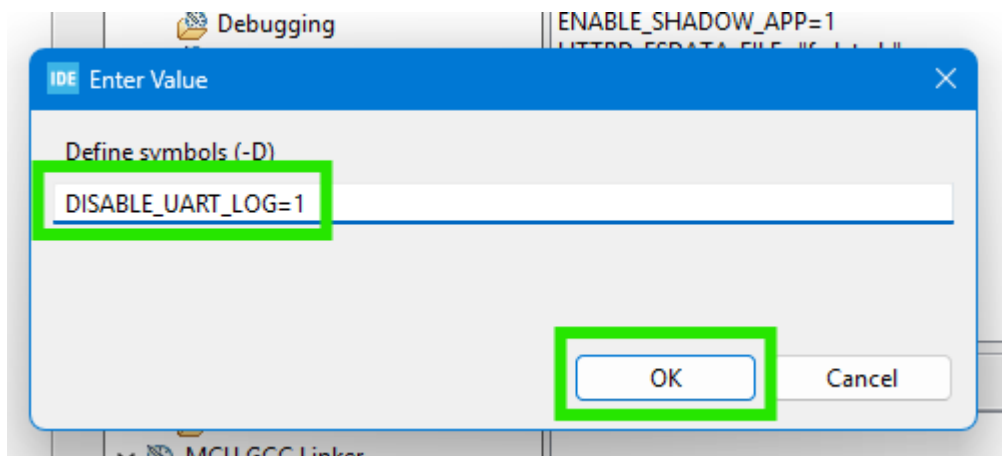


Figure 33. Adding the symbol necessary to build the rf-test application.

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.

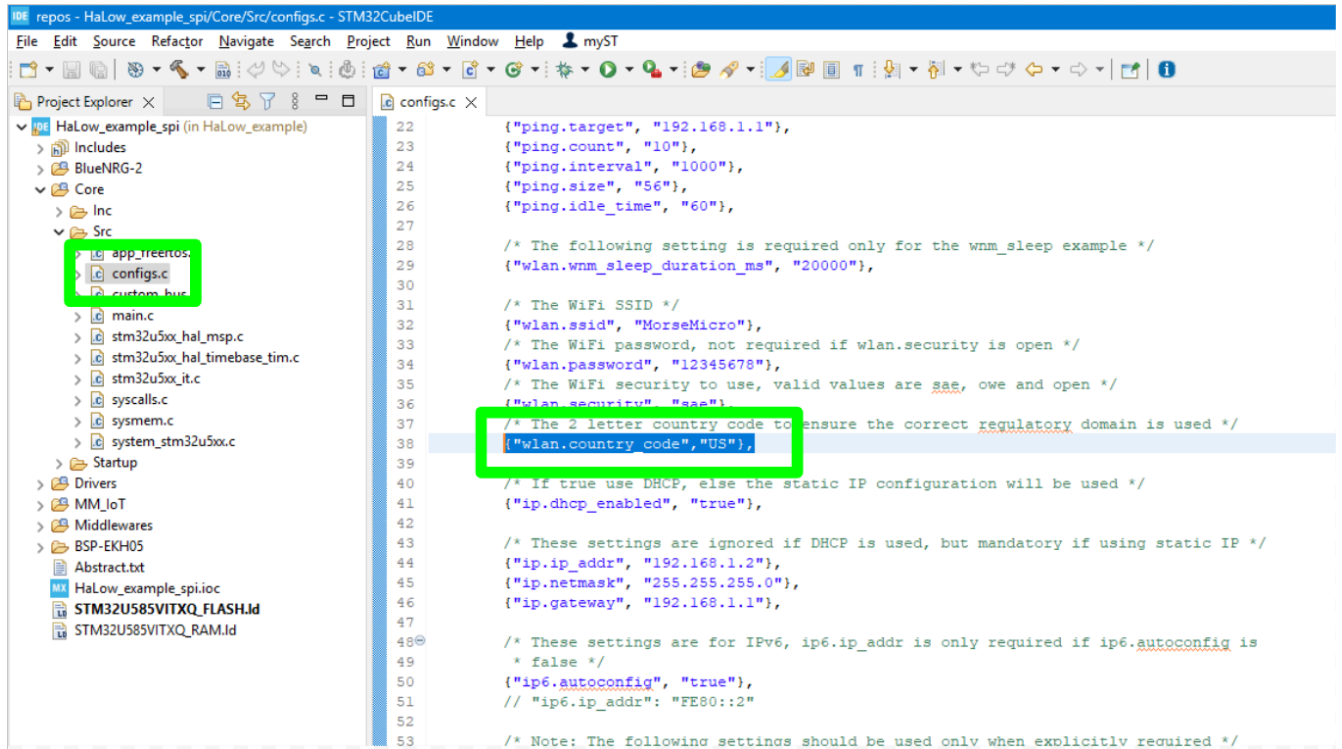


Figure 34. Source code location to change the country code.

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.

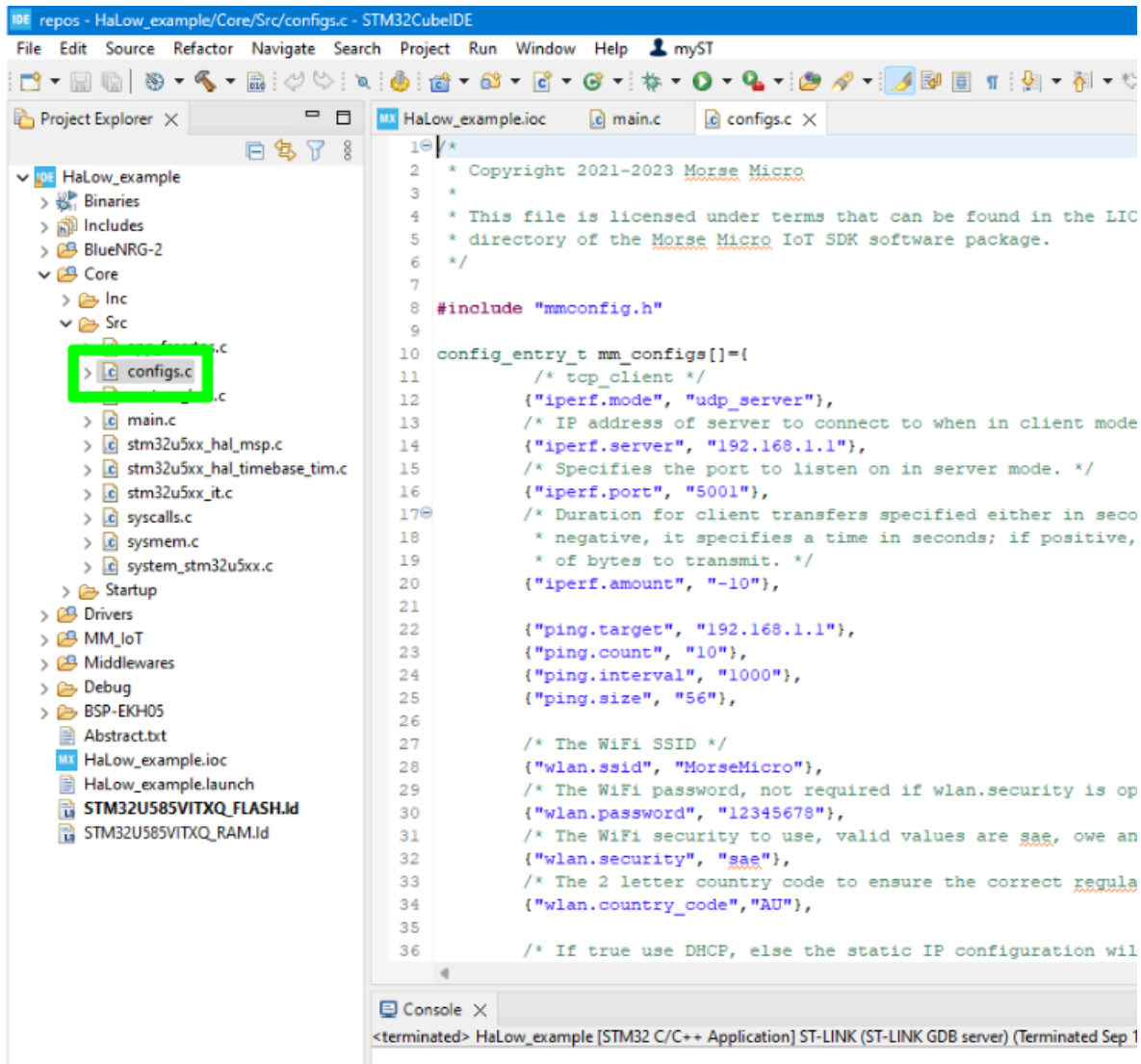


Figure 35. Showing all configurations available.

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.

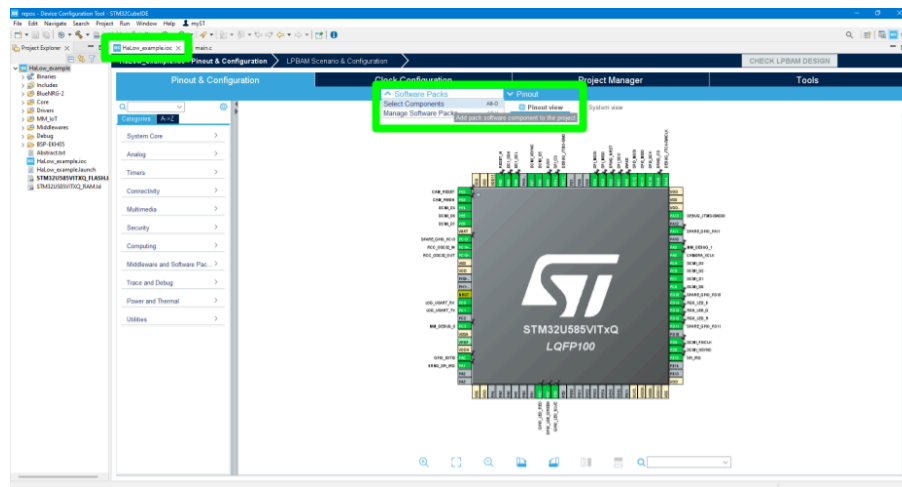


Figure 36. Select components menu in the .ioc file.

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.

▼ MorseMicro_MM_IoT	✓	1.0.0	
▼ MM_IoT_MM_IoT	✓	2.5.2	
morselib	✓	2.5.2	☒
mmconfig	✓	2.5.2	☒
mmipal	✓	2.5.2	☒
mmutils	✓	2.5.2	☒
mmLwIP	✓	2.2.0	☐
mmping	✓	2.5.2	☒
mmiperf	✓	2.5.2	☐
mbedtls	✓	2.5.2	☐
mm-knycbor	✓	2.5.2	☐
> BSP-EKH05	✓		
> AWS-AWS-libraries	✓	2.5.2	
▼ FreeRTOS-Libs_FreeRTOS-libraries	✓	2.5.2	
backoffAlgorithm	✓	2.5.2	☐
freertos-lbs-common	✓	2.5.2	☐
coreJSON	✓	2.5.2	☐
coreMQTT	✓	2.5.2	☐
coreMQTT-Agent	✓	2.5.2	☐
coreSNTP	✓	2.5.2	☐
FreeRTOS-Plus-TCP	✓	4.1.0	☒
MM_IoT_Example	✓	2.5.2	ping
> RealThread.X-CUBE-RT-Thread_Nano	✓	4.1.1	Install
> SEGGER-I-CUBE-embOS	✓	13.1	Install
> STMicroelectronics.FP-ATR-ASTRA1	✓	2.0.0	Install
> STMicroelectronics.FP-ATR-SIGFOX1	✓	3.2.0	Install

Figure 37. 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 FreeRTOS-libraries** checkbox is ticked.

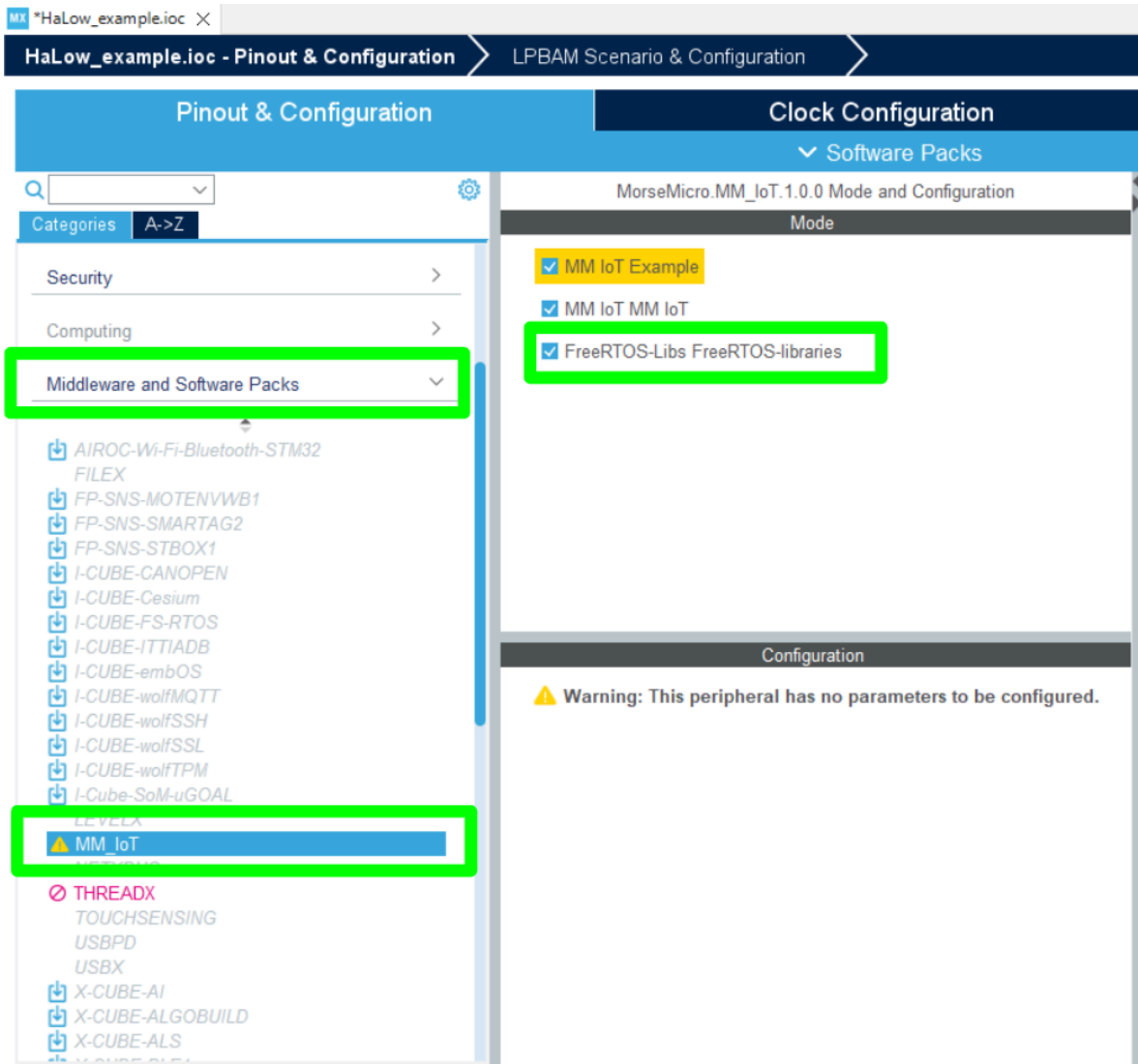


Figure 38. Middleware and software packs menu to add FreeRTOS libraries.

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.

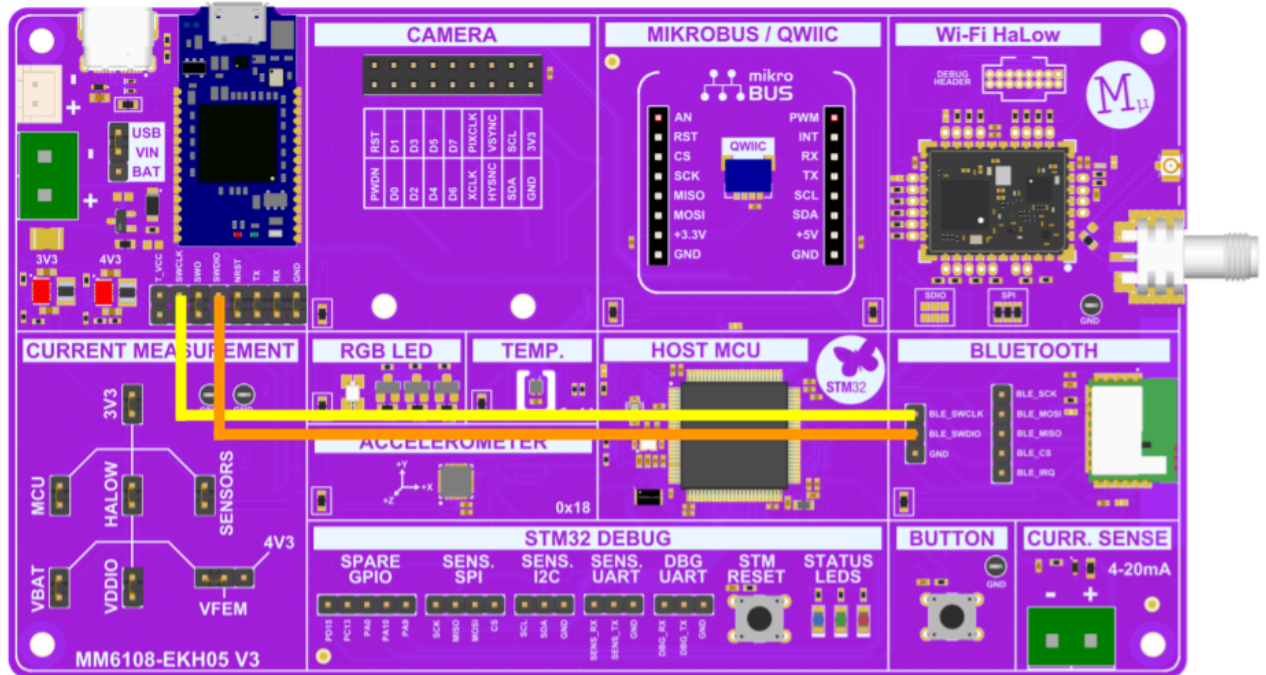


Figure 39. Connecting the STLINK V3 SWCLK and SWDIO pins to the BLUENRG-M2SP module.

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

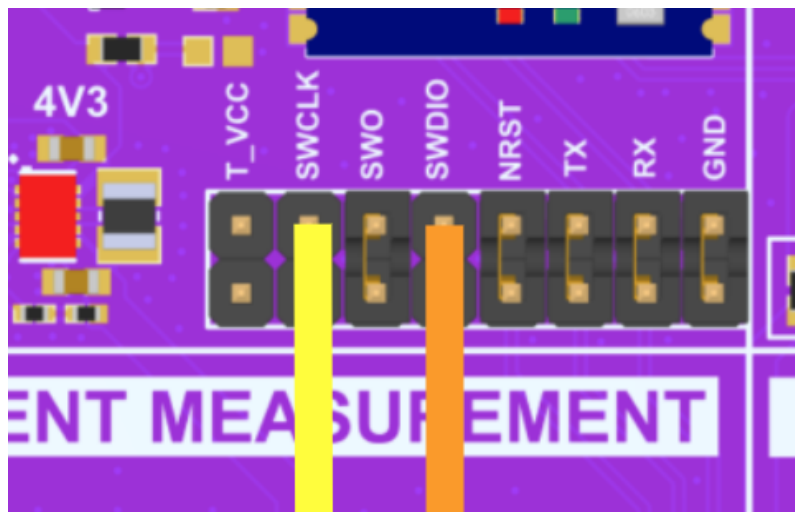


Figure 40. Close up view of connections, wires must be placed on top pins of the header.

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**.

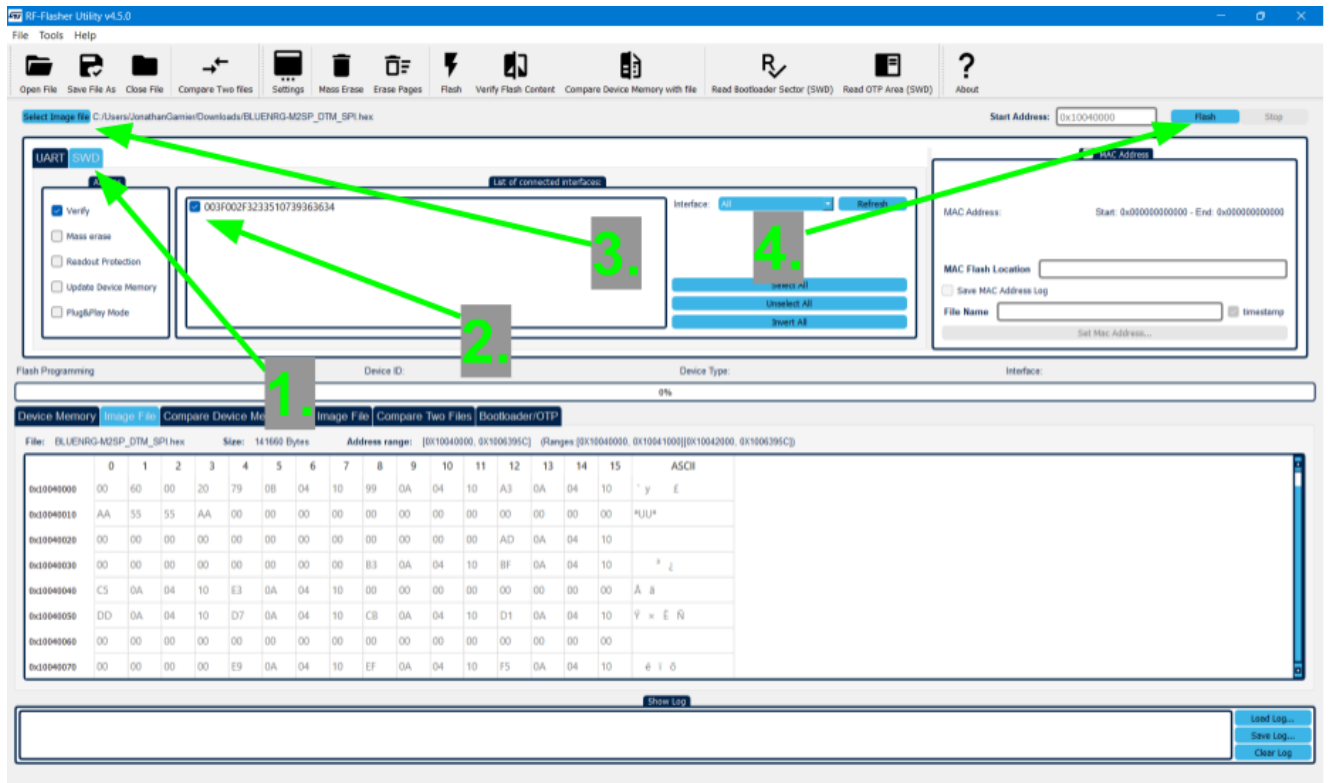


Figure 41. Steps in the RF-Flasher utility to program new firmware to the BLUENRG-M2SP module.

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.

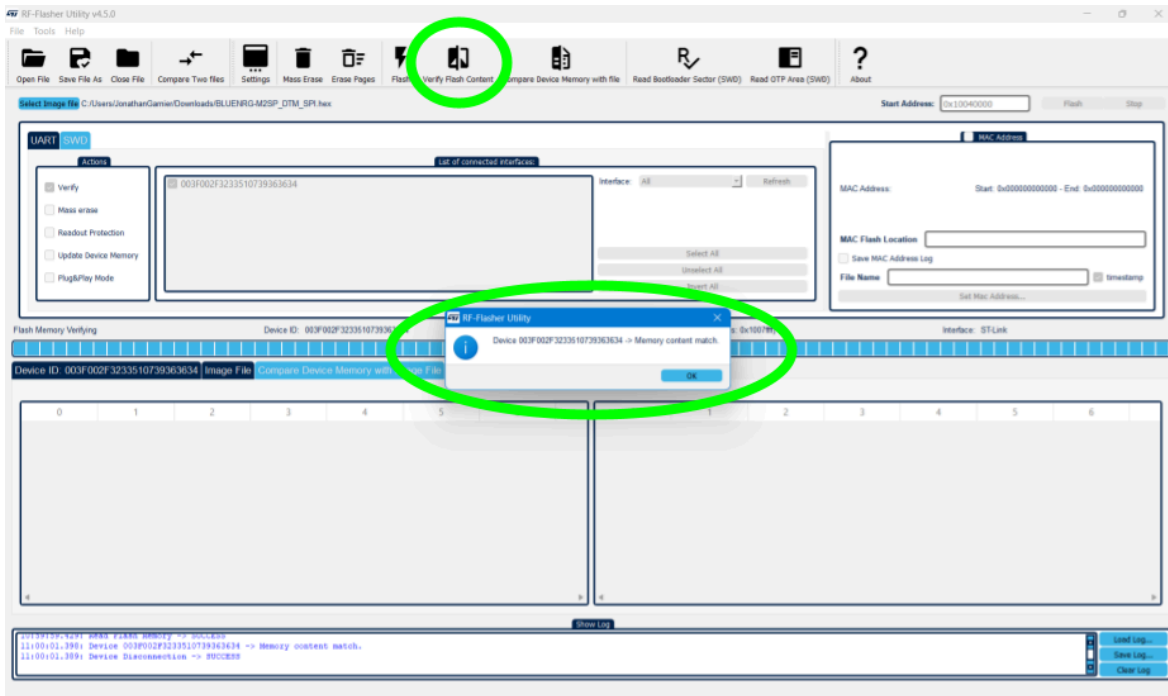


Figure 42. Verify firmware flash was successful.

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.

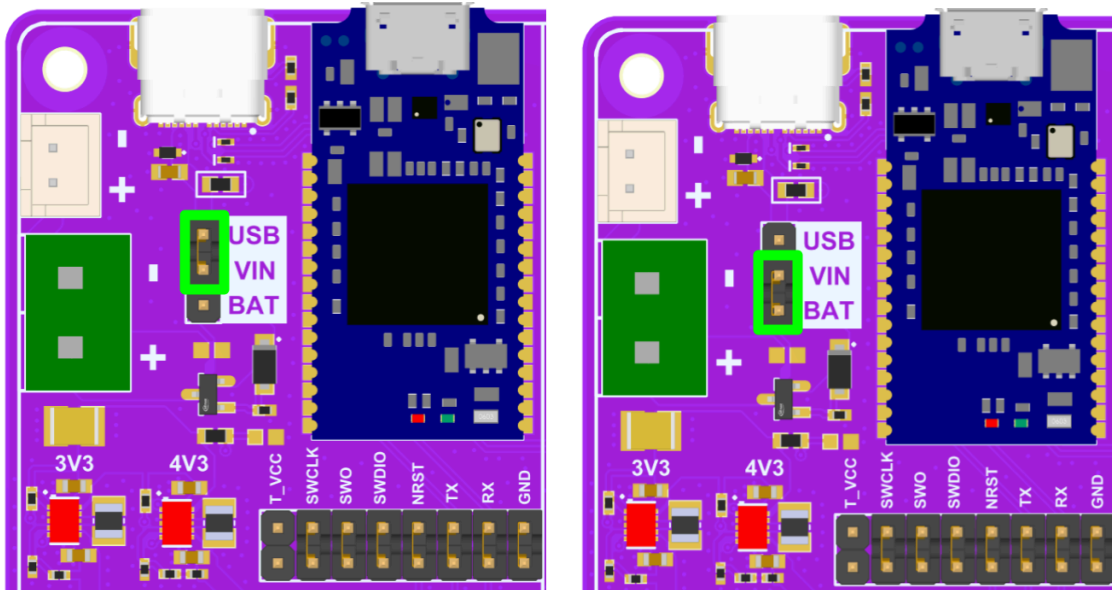
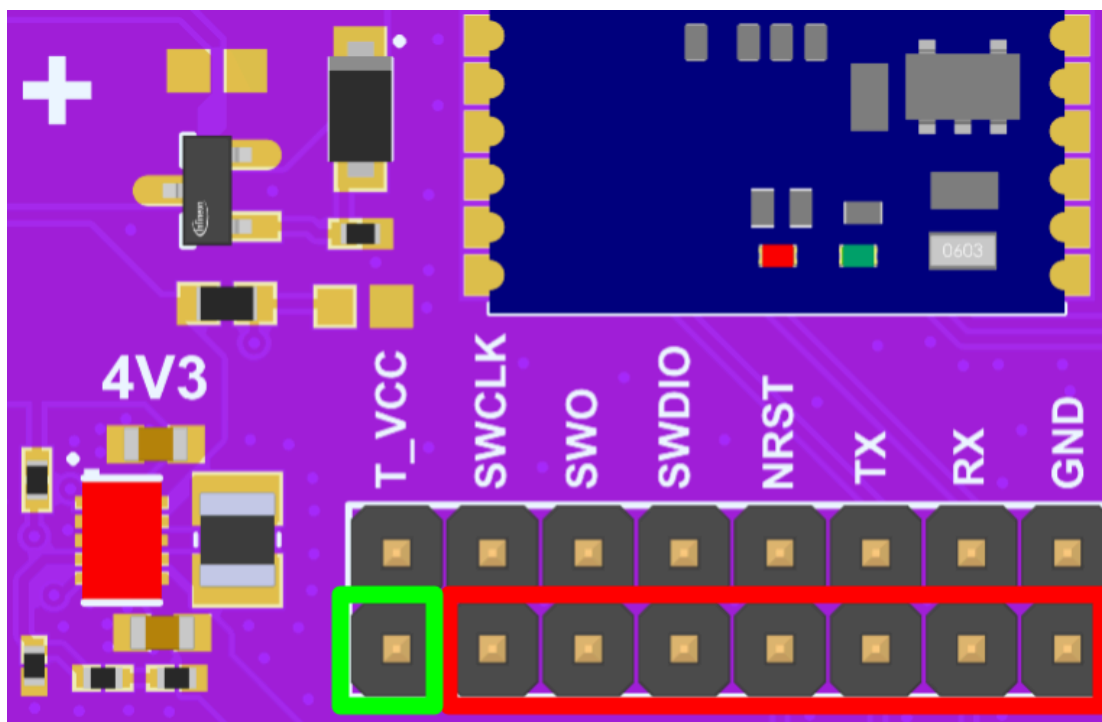


Figure 44. Changing power input between 5V USB on left (from USB-C or STLINK-V3) and battery inputs on right (JST connector or battery screw terminal).

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 can be attached to with an external debug tool to connect to the relevant pins on the STM32U585 microcontroller.



Connects to 3.3V.
Attach target voltage
pin of external
debugger here.

This side connects to
STM32U585
Microcontroller. Connect
external debug tool to this
side of header.

Figure 45. How to attach an external programmer.

6.3 Changing VFEM Voltage

The voltage supplied to the Front End Module (FEM) on the HaLow module can be changed between 3.3V and 4.3V. Setting to 4.3V will increase the power output from the Power Amplifier, and increase the maximum range. This will also increase power consumption of the VFEM rail when actively transmitting. Note that different Board Configuration Files (BCF) are required for each of the settings. In the Morse Micro examples, an ADC from the STM32U585 is connected to VFEM and will automatically select the correct BCF. For user applications, please take care to ensure the correct BCF is being used.

To switch between the two, the jumper can be placed on the leftmost header and VFEM (3.3V), or the rightmost header and VFEM (4.3V). ⚠ A power cycle is necessary after changing the jumper for automatic BCF detection to take effect.

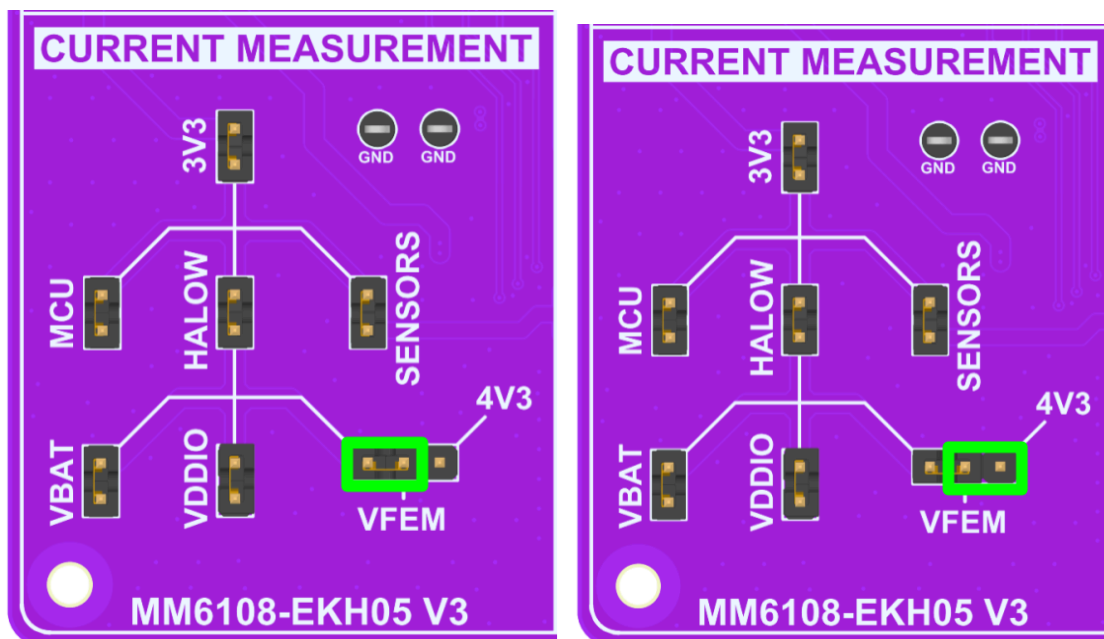


Figure 47. Changing between setting VFEM to 3.3V (left) and 4.3V (right).

6.4 Switching Between SDIO and SPI

The MM6108-EKH05 gives users the option to interface the STM32U585 Microcontroller with the HaLow module either through SDIO or SPI. SPI is the default and has been optimised for lowest power consumption. If using SDIO however, the 47 kOhm pull-up resistor **must be populated** as pointed out by the arrow below. For SPI, this pull up resistor will cause **extra leakage current** on the VDDIO rail and should be removed.

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

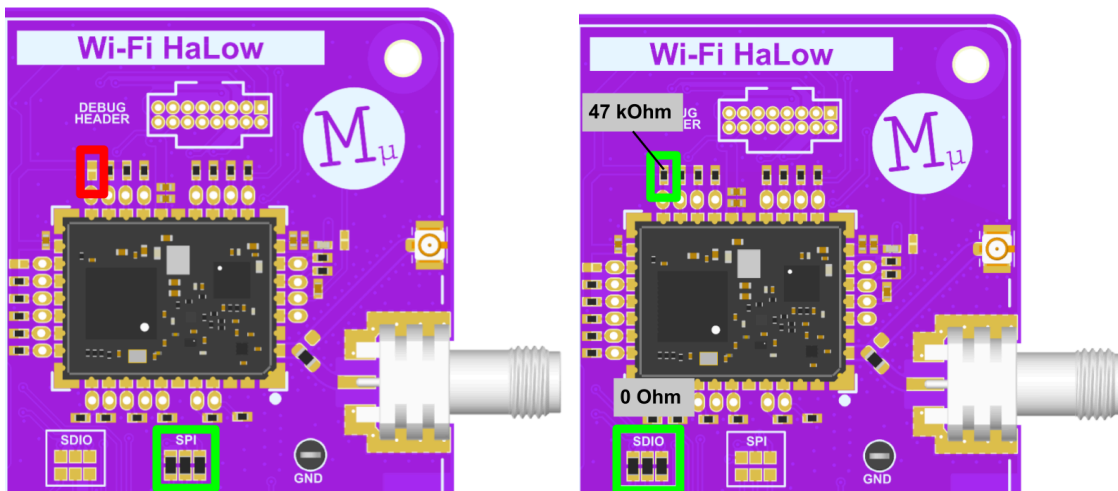


Figure 49. SPI configuration (left) and SDIO configuration (right).

6.5 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.

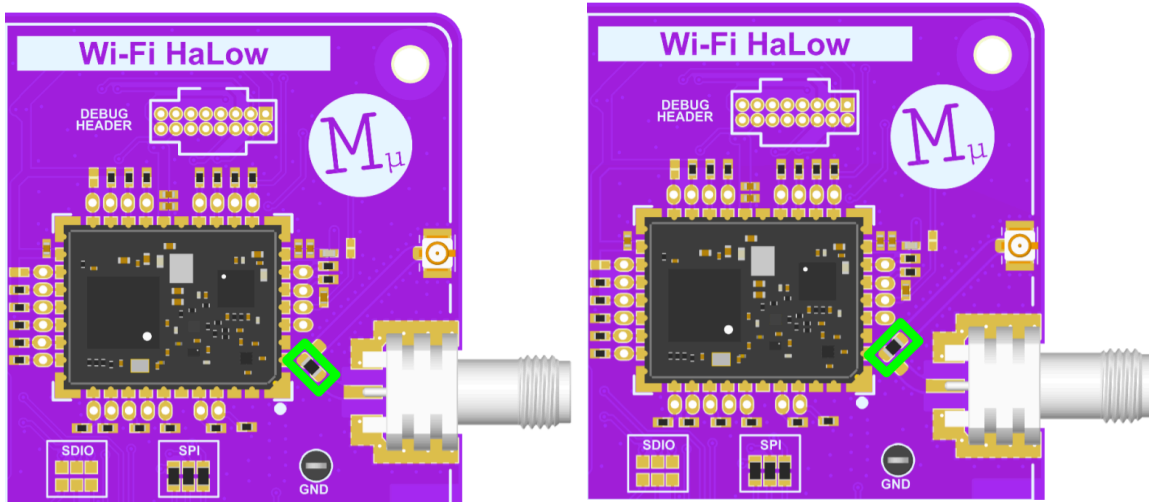


Figure 51. SMA connection (left) and U.FL connection (right).

6.6 Disconnecting Sensors

The sensor blocks on the MM6108-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.

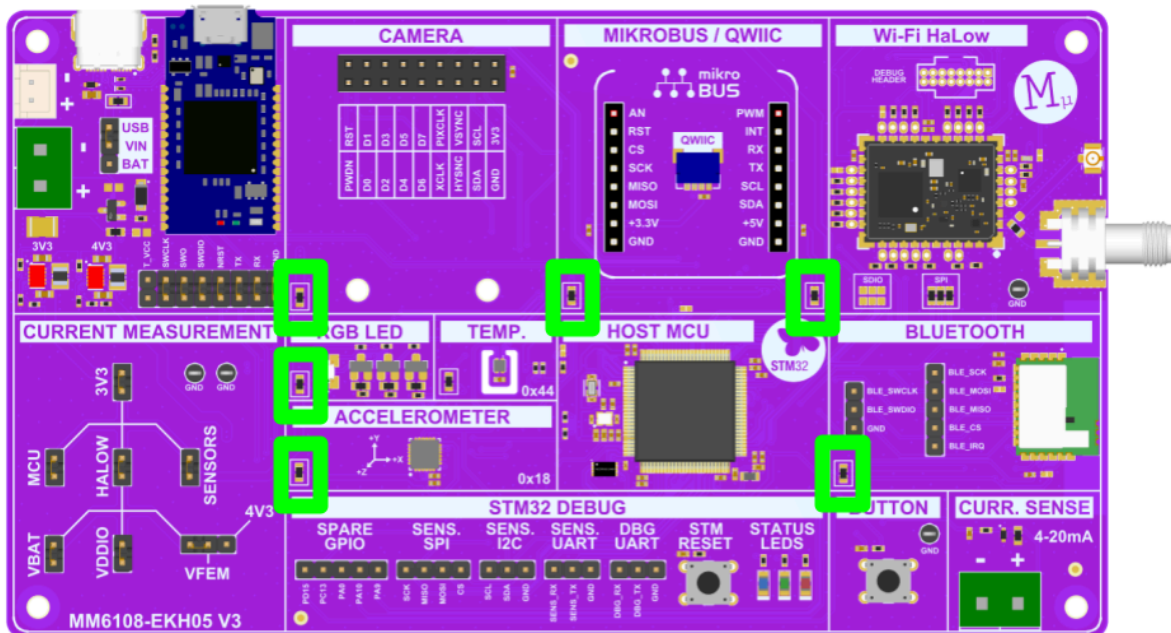


Figure 52. 0 ohm links which connect power to each sensor.

7 Power Consumption Measurements

7.1 Power Consumption Measurement Points

7.1.1 General Structure

The MM6108-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.

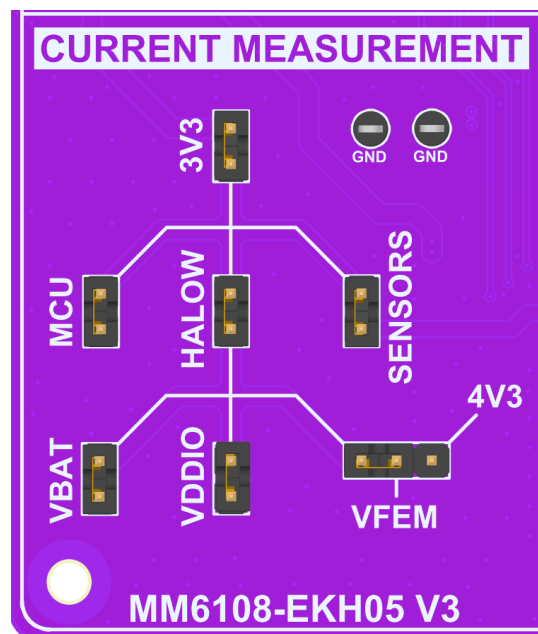


Figure 53. Current measurement tree on MM6108-EKH05.



WARNING: Never disconnect the VBAT jumper while the VDDIO jumper is on and powered, as this may damage the HaLow module.

7.1.2 Power Measurement Considerations

7.1.2.1 HaLow and VFEM

The EKH05 power architecture supplies 3.3V VFEM and 4.3V VFEM from separate power rails. The 3.3V VFEM rail is shared with the HaLow power rail, while the 4.3V VFEM rail is dedicated. Therefore, selecting 3.3V VFEM will include VFEM current in the HaLow power measurement. Alternatively, selecting 4.3V VFEM requires VFEM current to be measured separately or using a second ammeter.

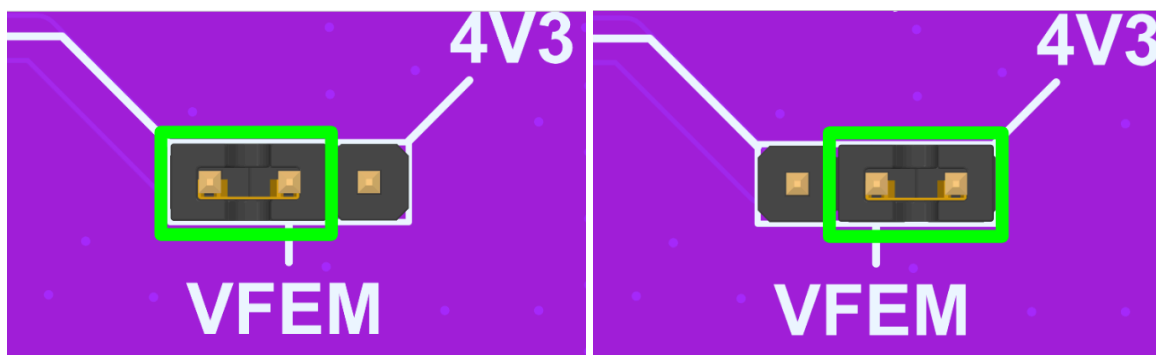


Figure 55. VFEM included in measurements (left), VFEM not included and a separate measurement of VFEM is needed (right).

7.1.2.2 Whole System Power Consumption

To measure the whole system power consumption, including efficiency losses of both 3.3V and 4.3V on-board buck-boost regulators, connect ammeter as per [Figure 56](#).

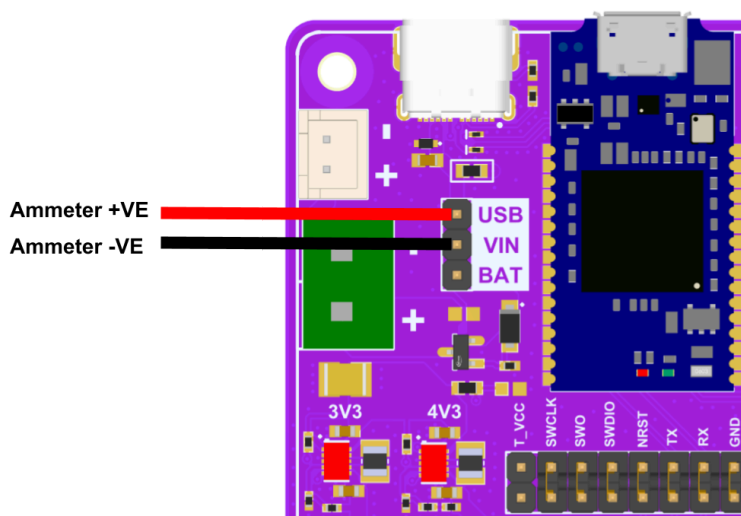


Figure 56. Ammeter positions to measure the entire board including the on-board regulators.

When VFEM is set to 3.3V, the 4.3V on-board buck-boost regulator can be disabled by moving the 0 ohm resistor as shown in [Figure 58](#). This reduces its power consumption to negligible levels, enabling more accurate power consumption measurements.

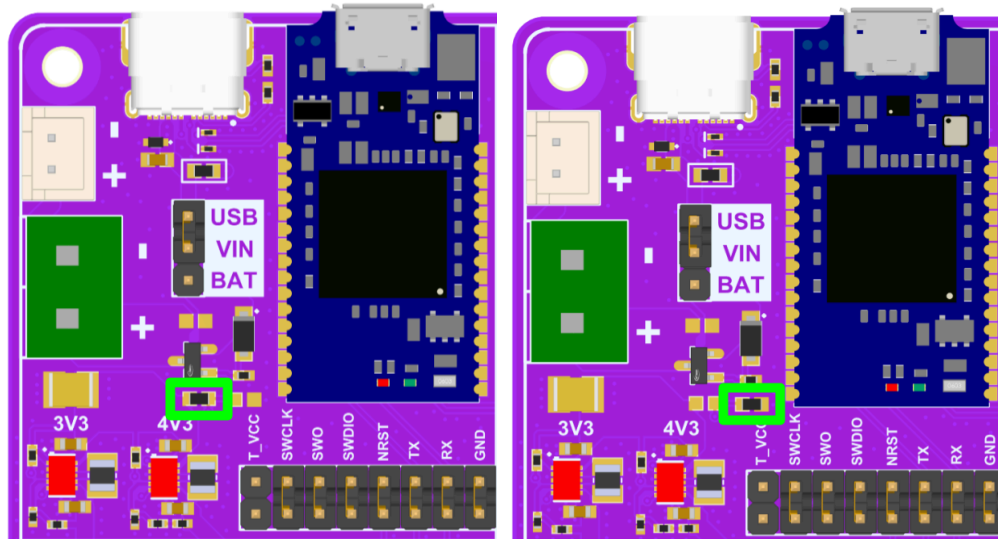


Figure 58. 4.3V regulator enabled (left), 4.3V regulator disabled (right).

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.

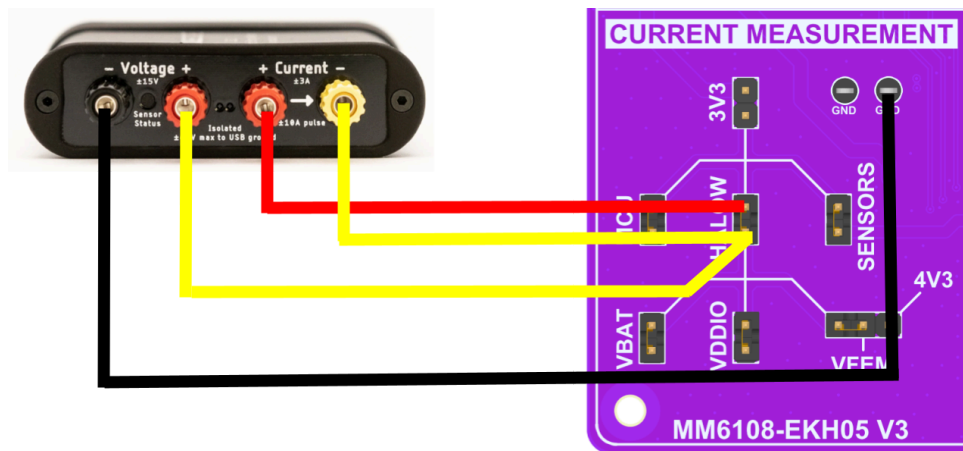


Figure 59. Example joulescope connections measuring HALOW header on MM6108-EKH05.

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.

For VFEM, the positive ammeter position should be placed on either the leftmost header (3.3V), or the rightmost header (4.3V) depending on which voltage is desired for VFEM. The negative ammeter position should always be connected to the middle header as shown below.

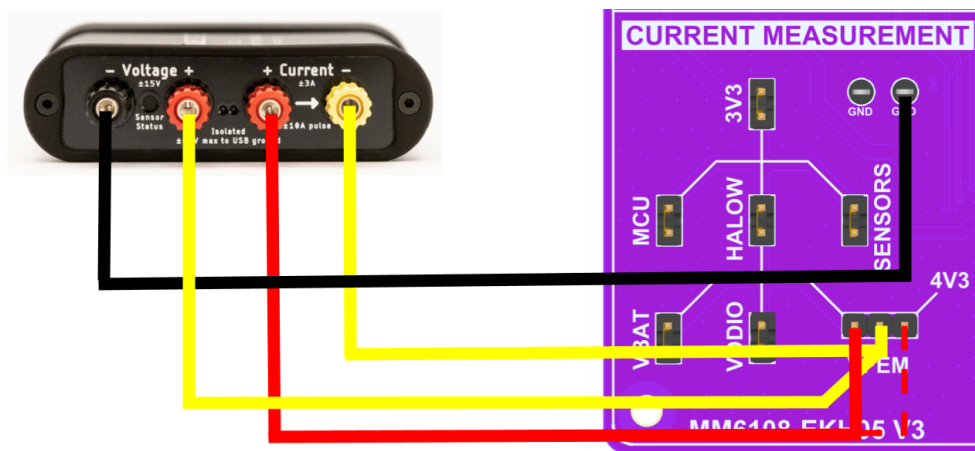


Figure 60. Example joulescope connections measuring VFEM header on MM6108-EKH05.

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 MM6108-EKH05 and can be connected to the Joulescope as follows.

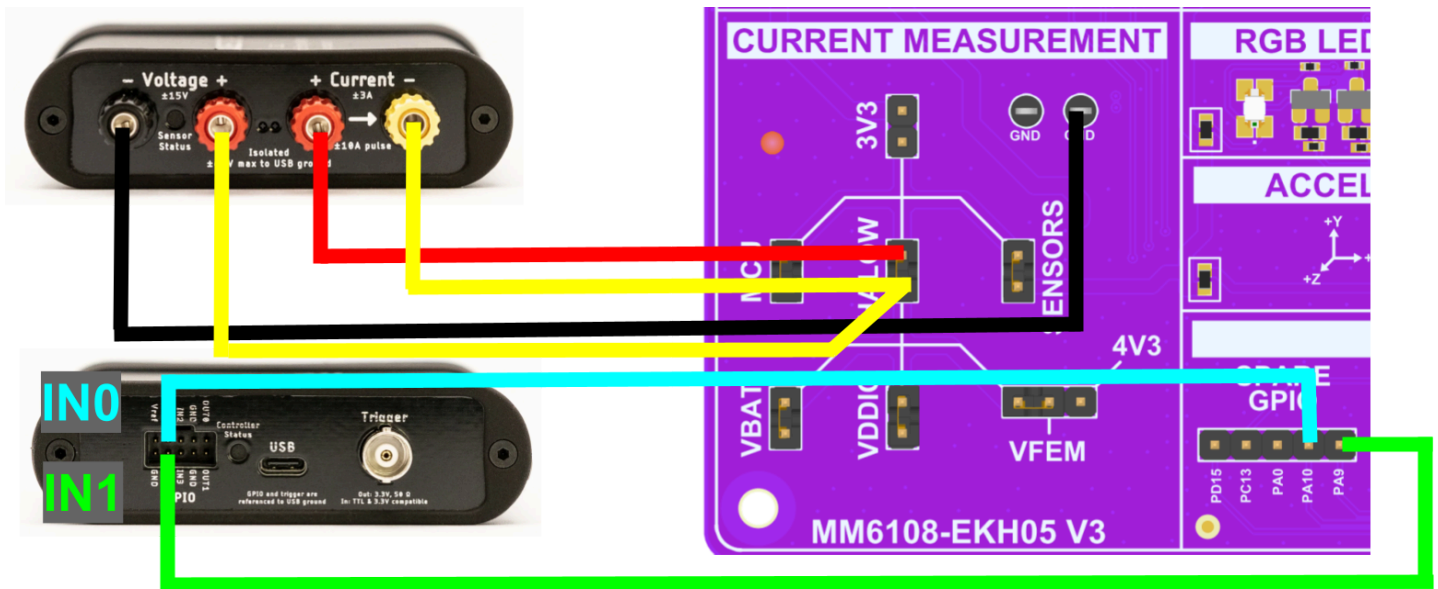


Figure 61. Showing where to connect debug GPIO's from MM6108-EKH05 to a joulescope.

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.

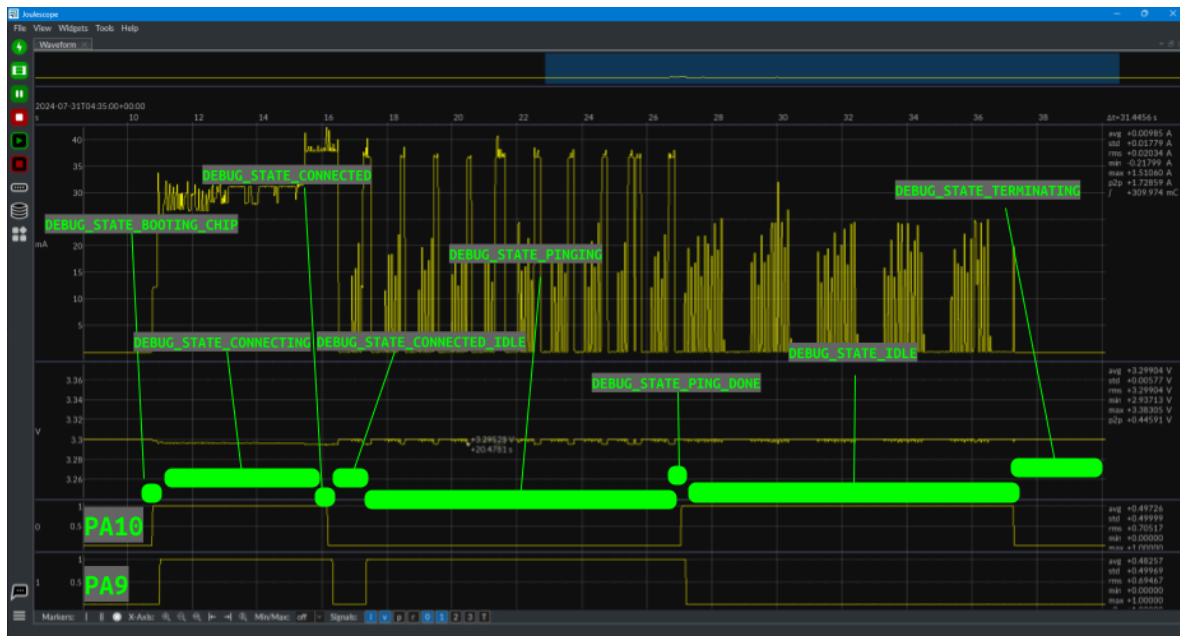


Figure 62. Power consumption profile of the ping app with the states from the debug GPIO's.

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\1.5.0\Example\ping\ping.c
```

Linux:

```
/home/USER_NAME/STM32Cube\Repository/Packs/MorseMicro/MM_IoT/1.5.0/Example/ping/ping.c
```

Mac:

```
/Users/USER_NAME/STM32Cube\Repository/Packs/MorseMicro/MM_IoT/1.5.0/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 MM6108-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 MM6108 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 63. Example UART output from ping application.

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



Figure 64. 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.

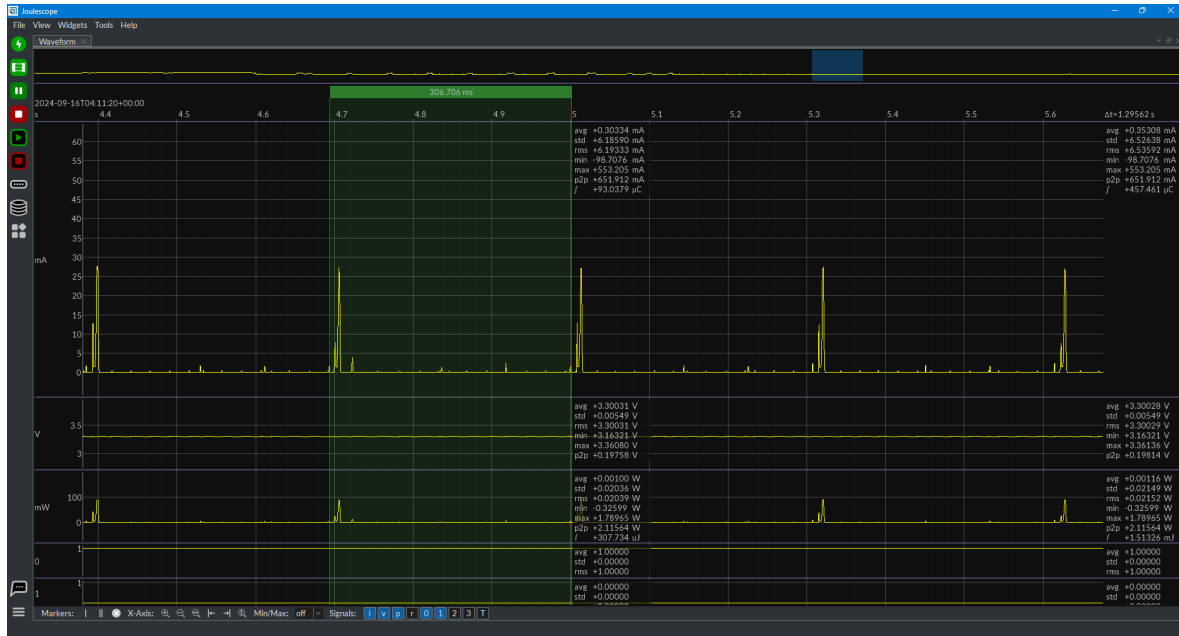


Figure 65. Zoomed in view of a DTIM3 measurement.

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 MM6108-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 MM6108 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 MM6108-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 66. 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, 0.00% 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 67. Example UART output from wnm_sleep application continued.

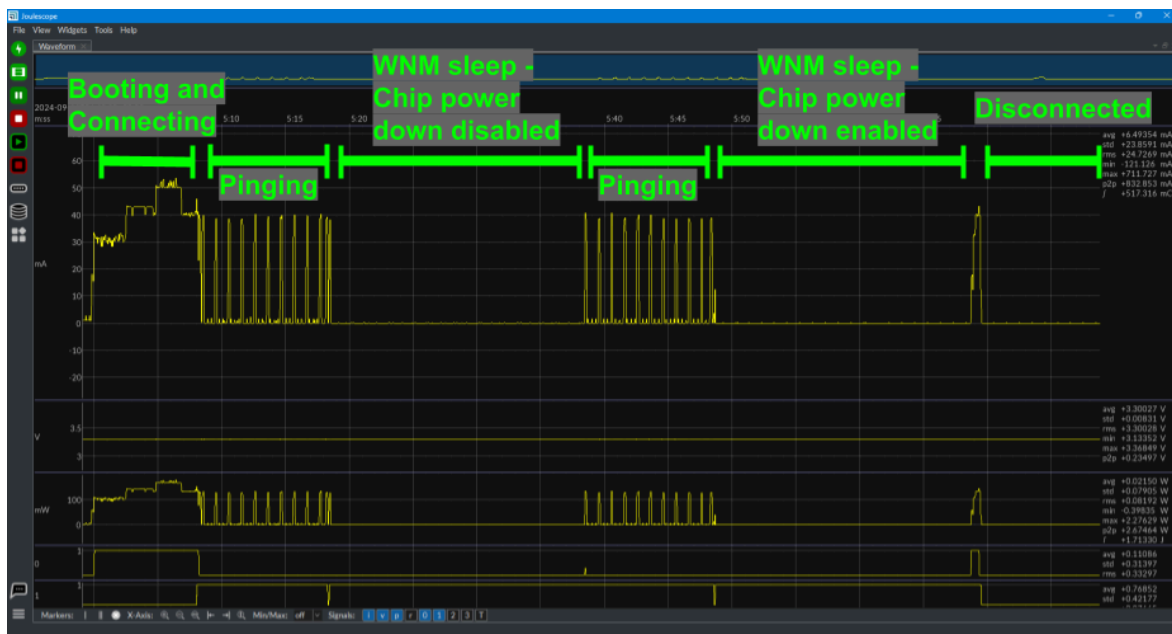


Figure 68. Example current consumption profile from wnm_sleep application.

7.6 Current Spikes When Waking Up

Users may observe a momentary current spike when MM6108 based devices wake up from sleep modes, sometimes reaching or exceeding 1 A. This spike is expected, and results 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. This behavior is illustrated in the image below.

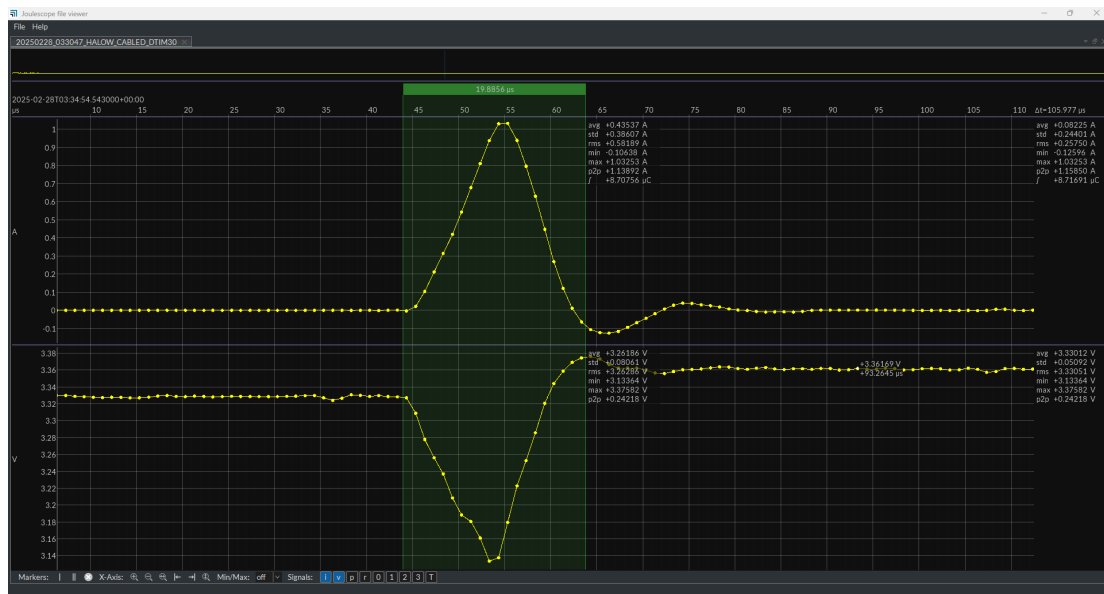


Figure 69. Joulescope capture showing zoomed in view of wakeup current spike.

Although the peak of the current is large, the duration is short, so the energy is small (in this instance less than $10 \mu\text{C}$) 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 MM6108. 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 Memory Retention Current Spikes

On MM6108 based devices, users may observe regular spikes of current roughly every 85ms (this period will change with device capacitance, and with temperature). These are memory retention current spikes. When the MM6108 goes into its default low power mode, the internal memory in the chip is still being powered. The buck regulator on the MM6108 will operate to keep the CAPD and DVDD pin of the chip at a voltage where memory is retained. The buck regulator will then pulse when this voltage goes below a certain threshold. This is the spike in current that is being measured, and the period will depend on temperature and the capacitance of the board.

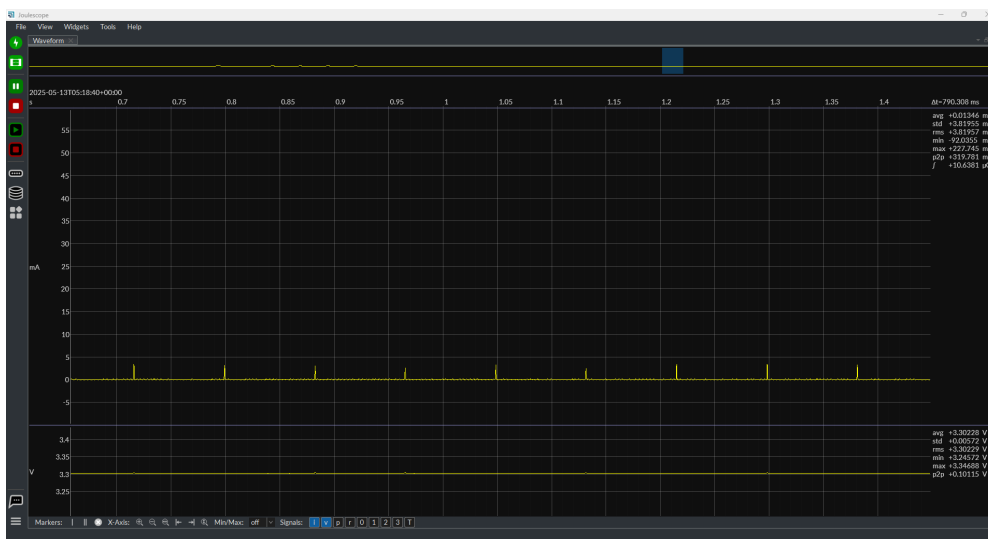


Figure 70. Joulescope capture showing memory retention current spikes.

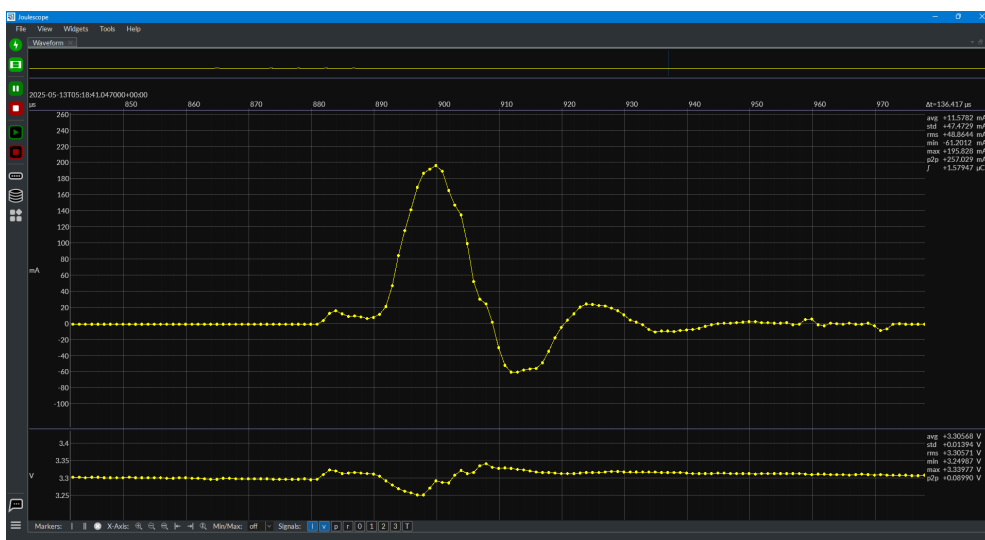


Figure 71. Joulescope capture showing zoomed in view of memory retention current spike.

7.8 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.

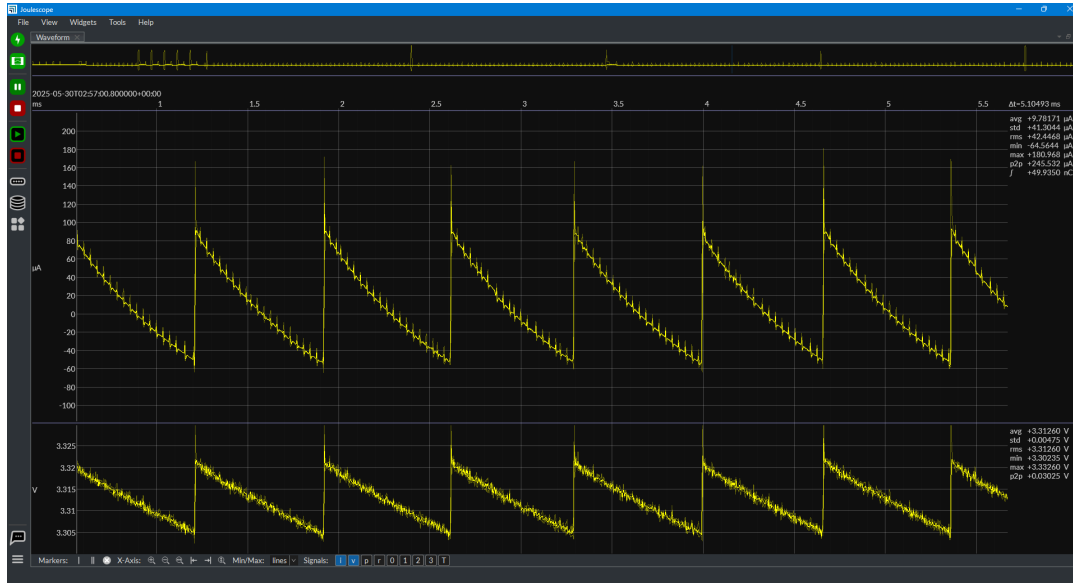


Figure 72. Zoomed in view of a snooze measurement showing negative current spikes.

This behavior is caused by the on-board buck-boost regulator, the [TPS63802](#). In the MM6108-EKH05 design, the MODE pin is tied low, which places the regulator in power-save mode (PFM) to optimize efficiency at light loads:

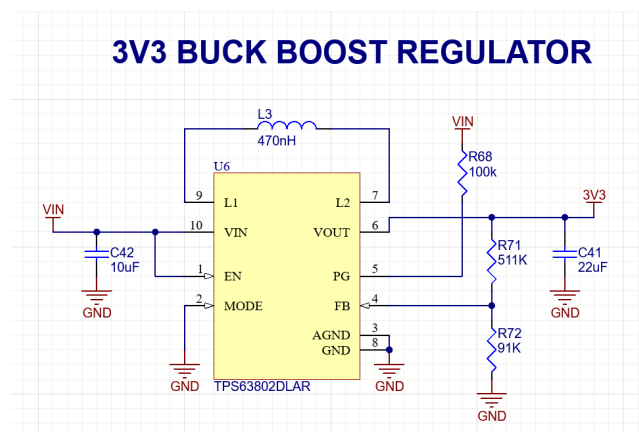


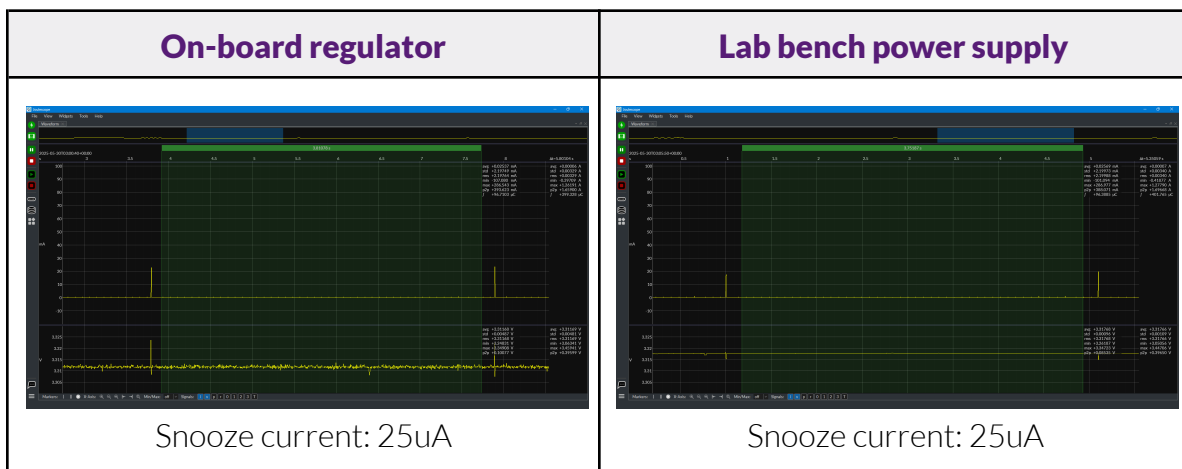
Figure 73. Schematic of TPS63802 in the MM6108-EKH05.

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.

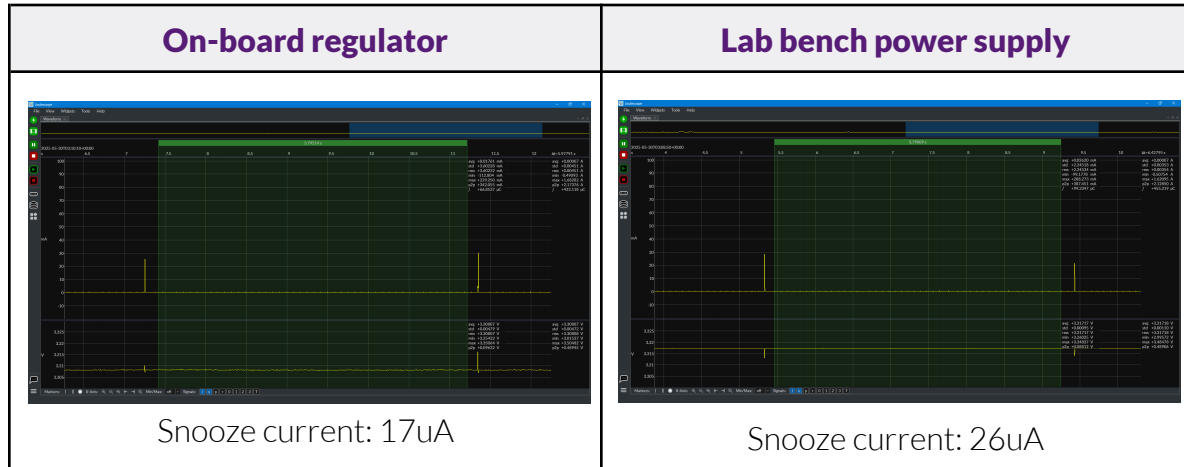
The simplest way to eliminate the reverse current effect is to bypass the on-board regulator and instead power the 3V3 rail using a lab bench power supply. However, under light load conditions, the difference in measured HaLow current between the on-board regulator and an external supply is negligible.

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.



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.



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.

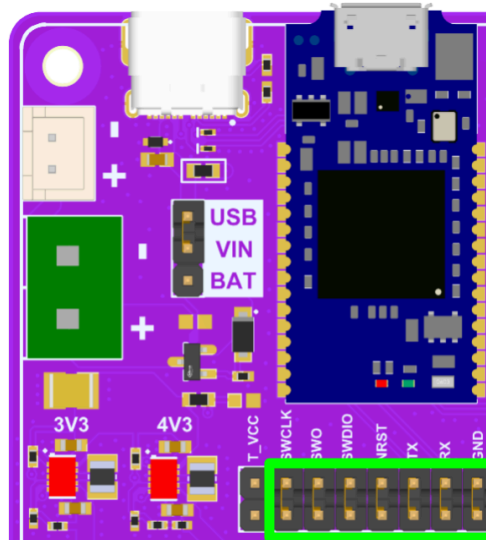


Figure 78. Troubleshooting step checking if the highlighted jumpers are connected.

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

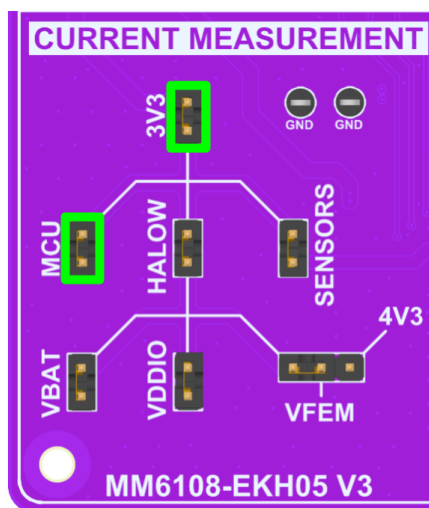


Figure 79. Troubleshooting step checking if the 3V3 and MCU jumpers are connected.

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

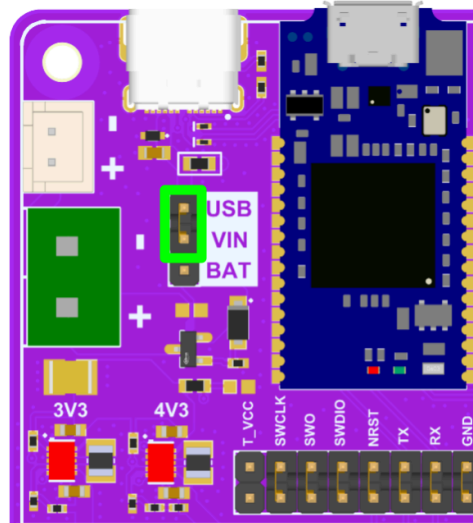


Figure 80. Troubleshooting step checking the input power supply.

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.

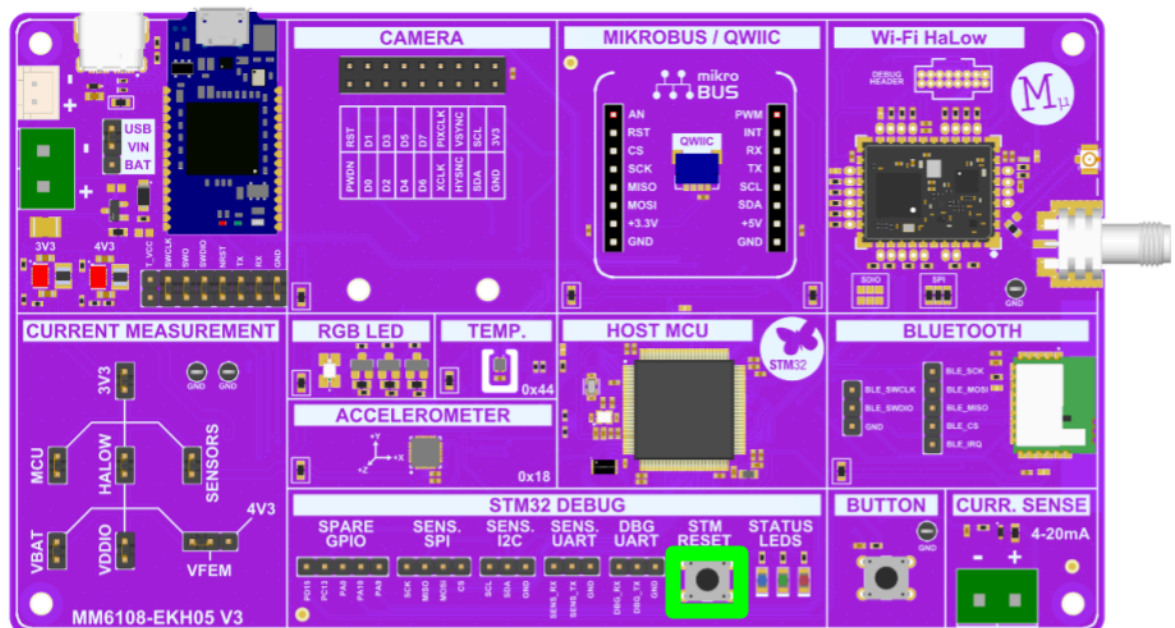


Figure 81. Reset button on the MM6108-EKH05. Resets the host MCU.

8.2 Poor HaLow Performance

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

1. Confirm the BCF loaded on to the device matches the VFEM setting. The bcf_aw_hm593.bin file should be used for 3.3V, and the bcf_aw_hm593_4v3.bin file should be used for 4.3V. If using a Morse Micro example from the CMSIS pack, the software will automatically detect which setting is used and select the correct BCF. If this is the case, the device might just need to be reset with the STM RESET button.
2. Ensure the AP configuration is expected. I.e. the correct channel and bandwidth are configured.
3. Ensure there are no other HaLow devices on the same channel causing in-band interference.
4. Ensure antenna connection is on securely.
5. 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:

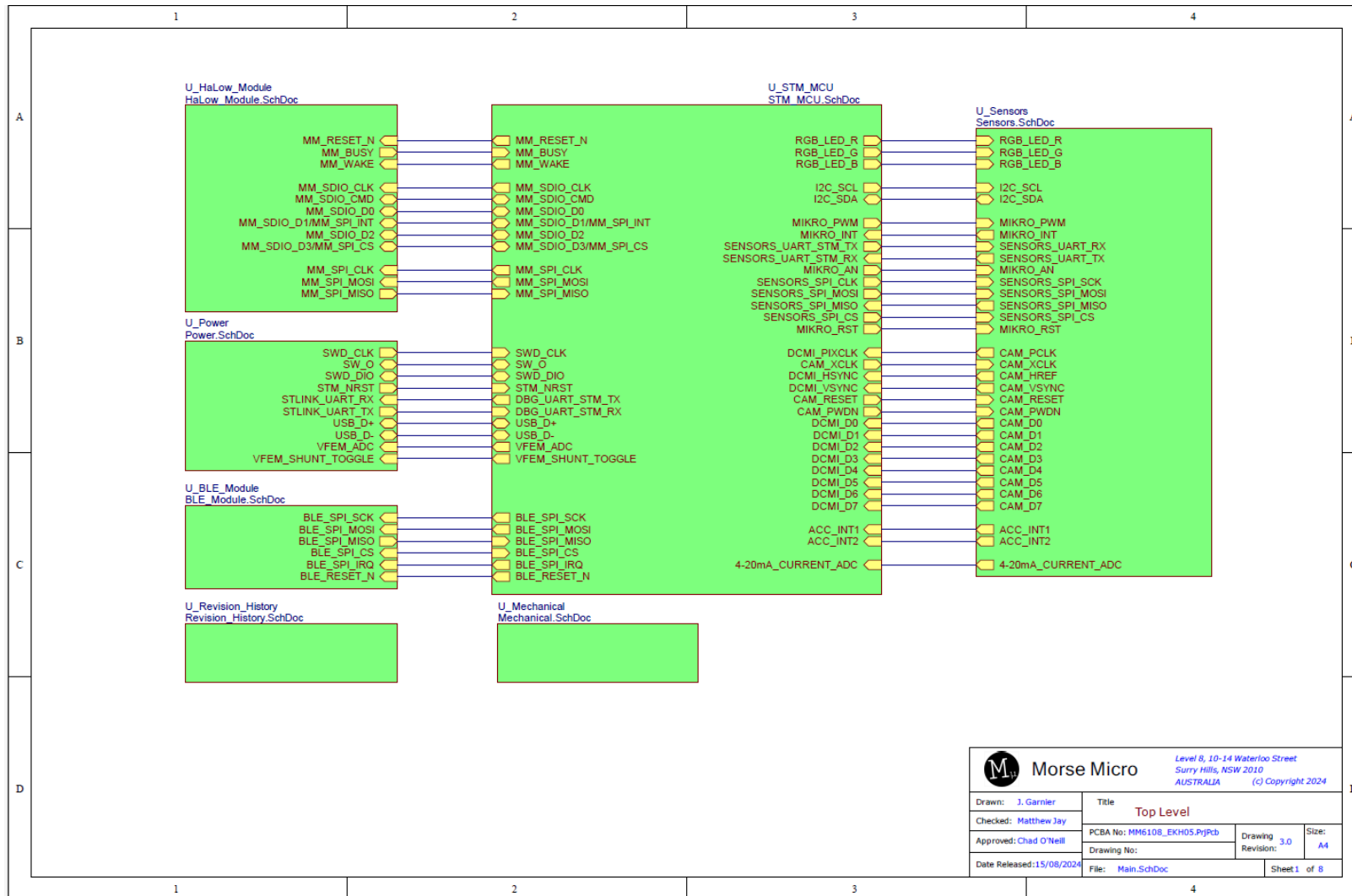
1. If the T_VCC jumper is populated, this will add ~300uA to the 3V3 power consumption.
2. For lowest power consumption, ensure the network stack is set to FreeRTOS plus TCP instead of LWIP. See section [5.7](#) for details.
3. To replicate Morse Micro DTIM power consumption numbers, the devices should 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 MM6108-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 MM6108-EKH05. Longer leads will add equivalent series resistance (ESR) and 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 MM6108-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.

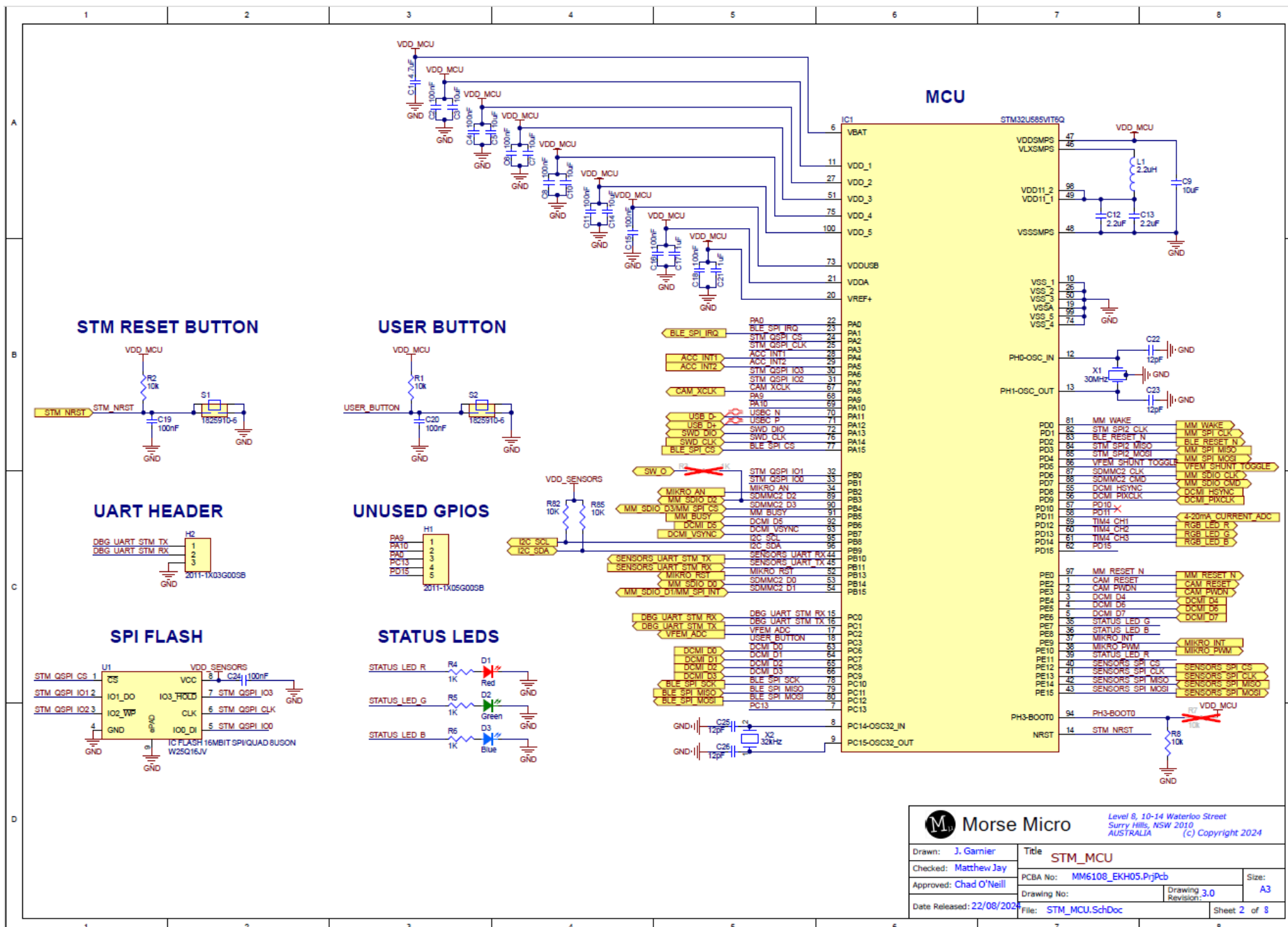
9 Errata

10 Revision History

Release Number	Release Date	Release Notes
01	19/11/2024	Initial release.
02	12/03/2025	- Updated instructions from using EKH01 to HaLowLink 1 as preferred AP. - Updated instructions to accommodate CMSIS pack 1.5.0 changes. - Added notes about rf-test example. - Added section explaining current spikes.
03	30/05/2025	Added section 7.8 explaining negative current from regulator.

Appendix A: Schematics





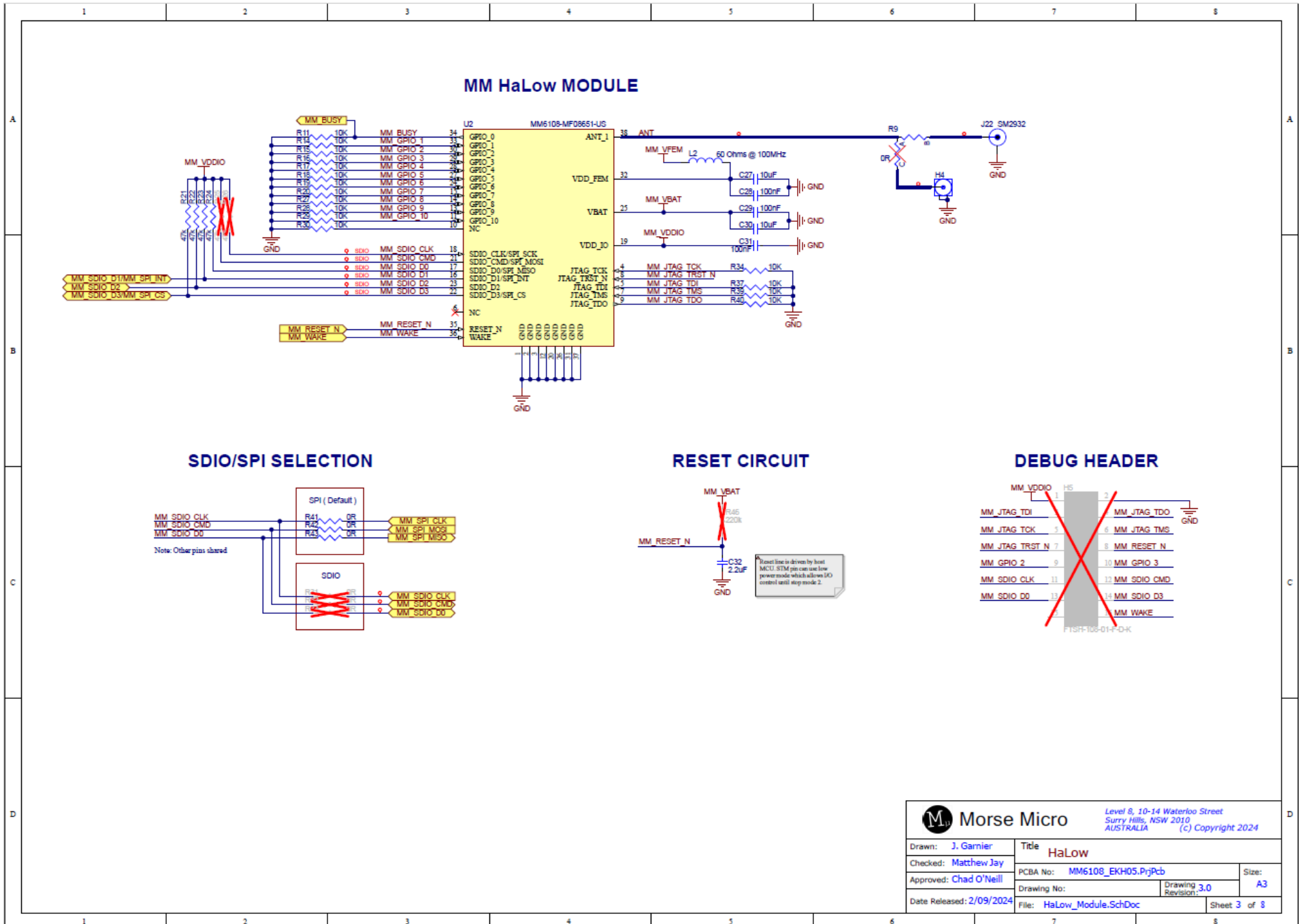
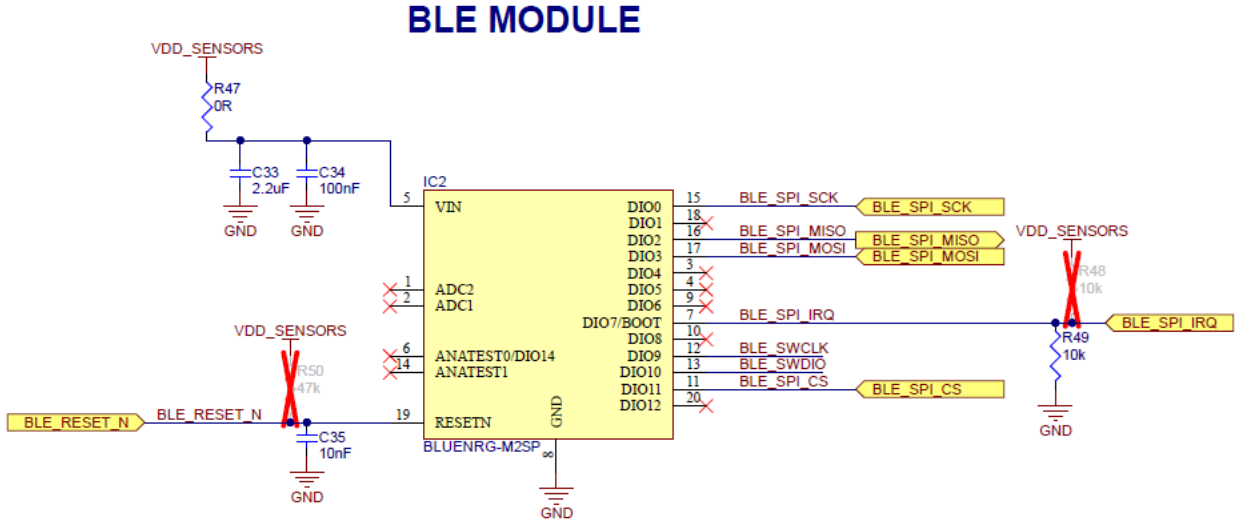


Table 8. Pin connection

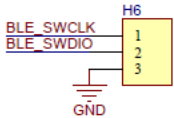
Pin #	Name	Function			
		Mode "000"	Mode "001"	Mode "100"	Mode "010"
1	ADC2	ADC Input 2			
2	ADC1	ADC Input 1			
3	DIO4	GPIO4	UART_RXD	I2C2_CLK	PWM0
4	DIO6	GPIO6	UART_TXD	I2C2_DAT	PWM1
5	VIN	Power Supply			
6	ANATEST0/DIO14	GPIO14	I2C1_CLK	SPI_CLK	ADC_DAT
7	DIO7/BOOT (1)	GPIO7	UART_CTS	I2C2_DAT	PDM_CLK
8	GND	Ground			
9	DIO8	GPIO8	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 (2)	
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 (3)		I2C1_CLK	
21 (1)	GND	Ground or leave unconnected			
22 (1)	N.C.	Leave unconnected			
23 (1)	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 acts 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 acts as MISO.
4. DIO12 can only be general purpose input pins (not output), or I2C1 clock pin.
5. BLUENRG-M2SA only.

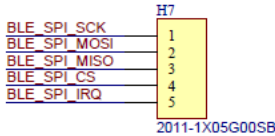


- Note:
- All unused pins should be left floating; do not ground
 - All GND pins must be well grounded
 - The area around the module should be free of any ground planes, power planes, trace routings, or metal for 6 mm from the module antenna position, in all directions.
 - Traces should not be routed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

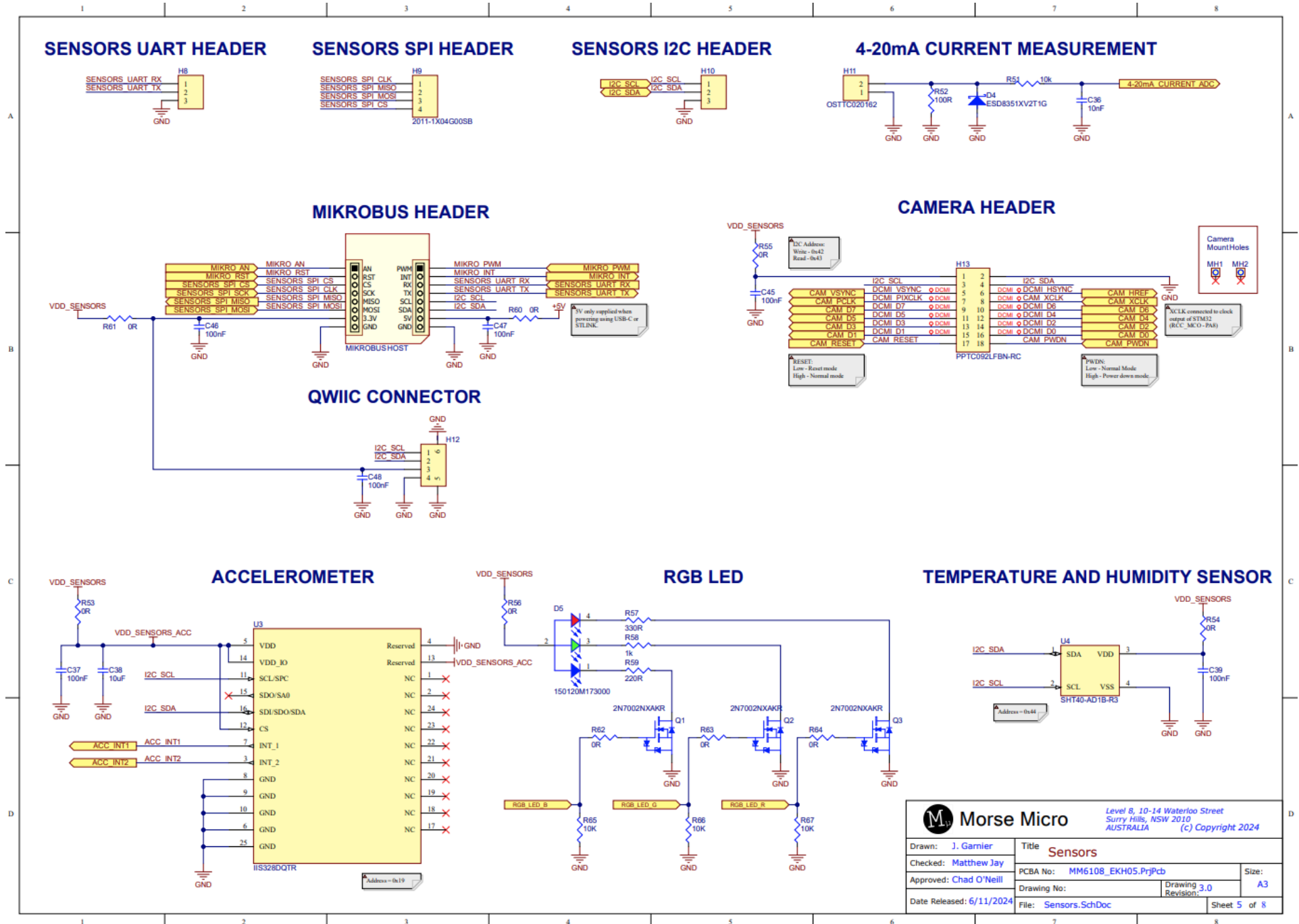
BLE SWD HEADER



BLE SPI HEADER

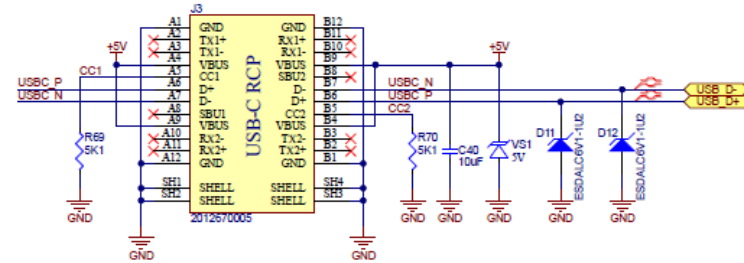


		Morse Micro		Level 8, 10-14 Waterloo Street Surry Hills, NSW 2010 AUSTRALIA (c) Copyright 2024	
Drawn: J. Garnier		Title BLE			
Checked: Matthew Jay		PCBA No: MM6108_EKH05.PjPcb		Drawing Revision: 3.0	Size: A4
Approved: Chad O'Neill		Drawing No:			
Date Released: 20/08/2024		File: BLE_Module.SchDoc			Sheet 4 of 8

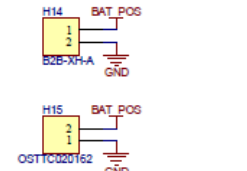


1 2 3 4 5 6 7 8

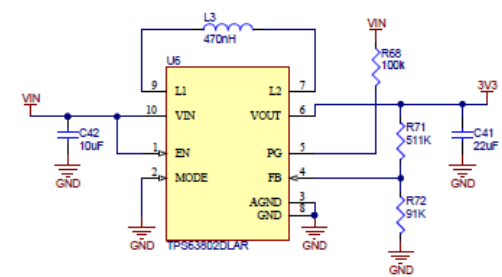
USB TYPE-C



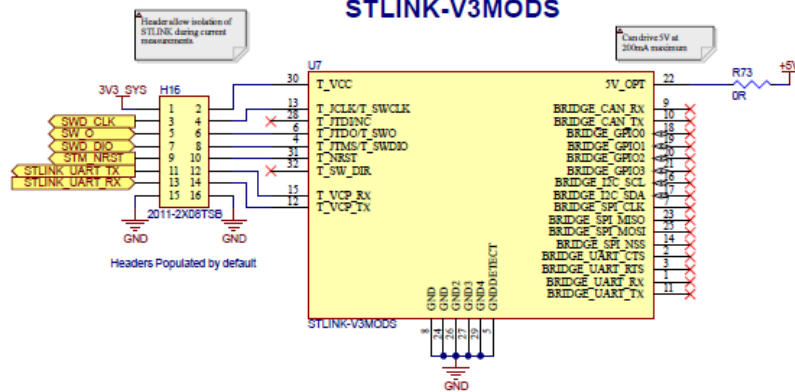
BATTERY CONNECTORS



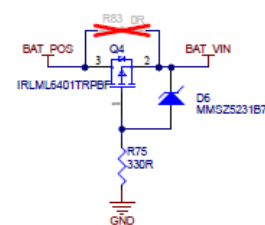
3V3 BUCK BOOST REGULATOR



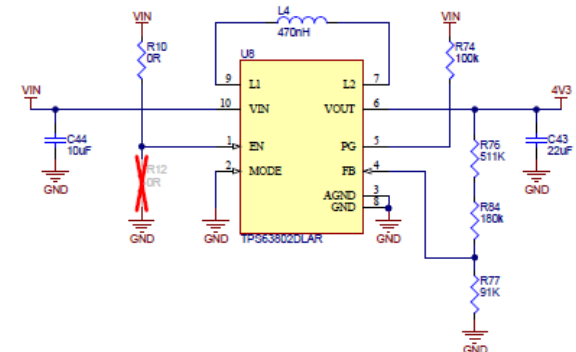
STLINK-V3MODS



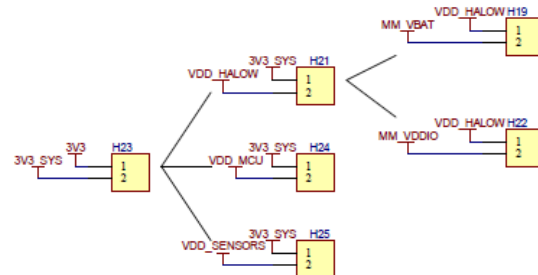
BATTERY REVERSE POLARITY PROTECTION



4V3 BUCK BOOST REGULATOR

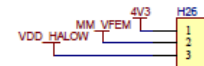


CURRENT MEASUREMENTS

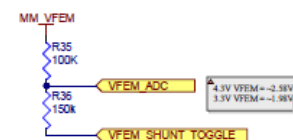


HaLow RF POWER SELECT

Options: 3V3 or 4V3

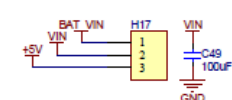


VFEM ADC



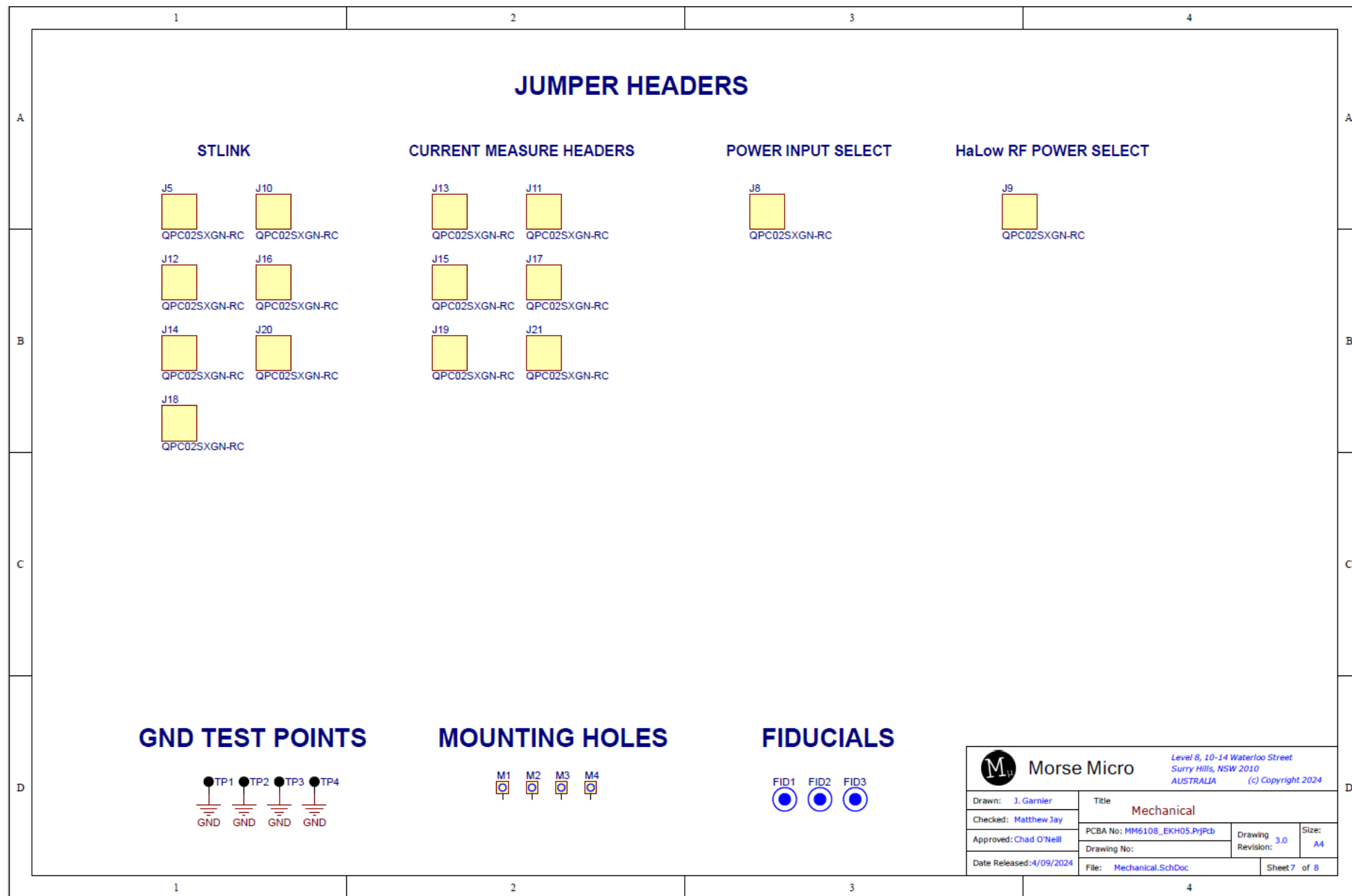
POWER INPUT SELECT

Options: Battery or USB



 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: MM6108_EKH05.PrjPcb	Size: A3	
Approved: Chad O'Neill	Drawing No:	Drawing Revision: 3.0	
Date Released: 4/09/2024	File: Power.SchDoc	Sheet 6 of 8	

1 2 3 4 5 6 7 8

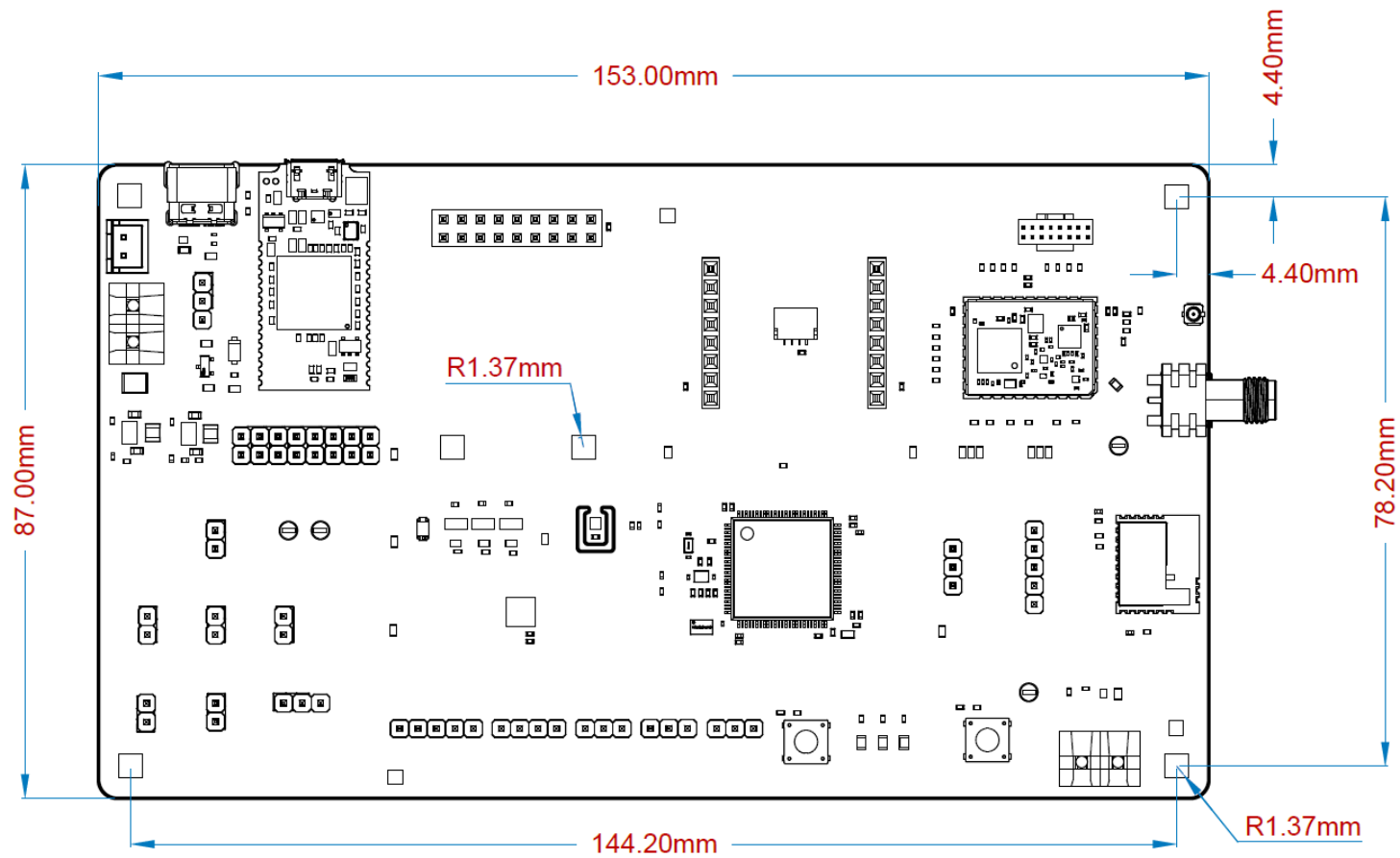


A	1	2	3	4
B				
C				
D				

Revision History	
1.0	- Initial release.
2.0	- Connect USB data to STM - Connect RTS/CTS to BLE Module. - Swap UART pins of BLE Module. - Update buck boost output cap to 22uF - Reversed UART RX/TX connections on STLINK - Moved power supply for QSPI Flash to VDD_SENSORS - Updated battery connector to JST - Added bulk cap on Power select - Removed load switch for sensors power supply - Added option to change VFEM to 4.3V
3.0	- Updated SMA footprint. - Updated BLE module communication protocol from UART to SPI. Updated pinout as needed. - Removed 0ohm links to unneeded SPI/SDIO connections. - Combined the MM_SPI_CS and MM_SPI_INT pins with SDIO - Pin assignment updates to red status LED, MIKRO_AN and MIKRO_PWM - Updated VFEM High Power Select to 4.3V - Removed Vin & VFEM from power measure tree

 Morse Micro		Level 8, 10-14 Waterloo Street Surry Hills, NSW 2010 AUSTRALIA (c) Copyright 2024	
Drawn: J. Garnier	Title Revision History		
Checked: Matthew Jay	PCBA No: MM6108_EKH05.PrjPcb		Drawing Revision: 3.0
Approved: Chad O'Neill	Drawing No:		Size: A4
Date Released: 2/09/2024	File: Revision_History.SchDoc		Sheet 8 of 8

Appendix B: Mechanical Drawing





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

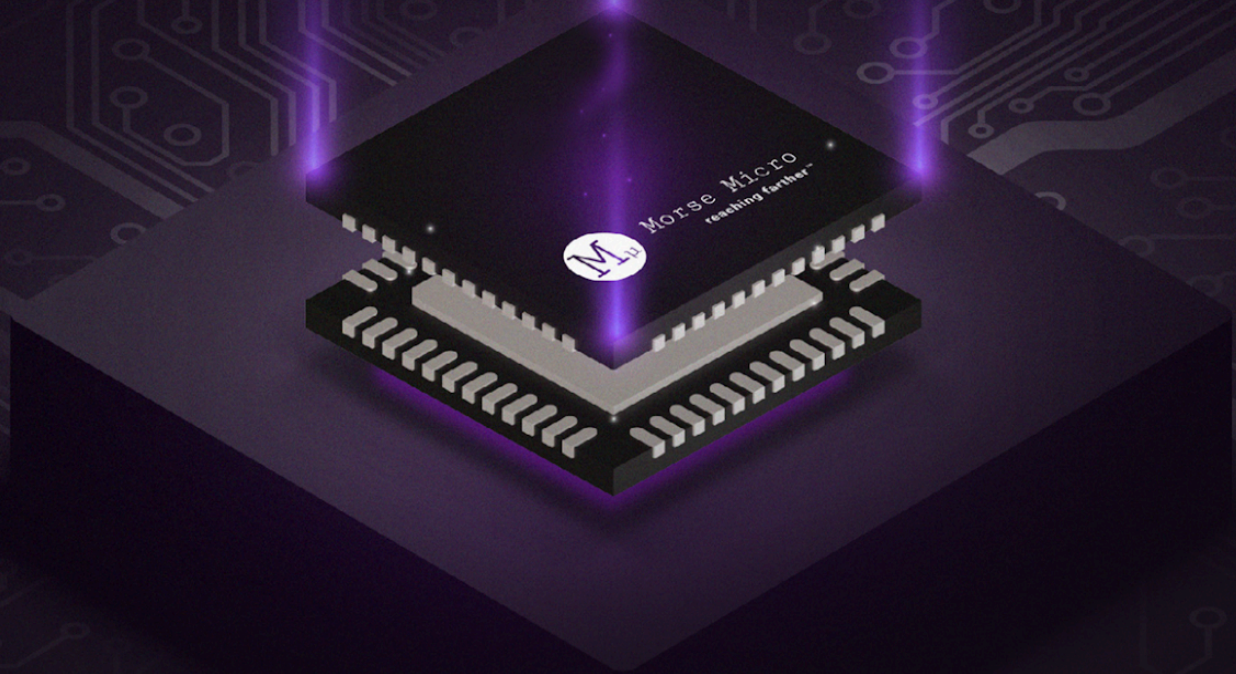
www.morsemicro.com

© 2025 Morse Micro Pty. Ltd.

All rights reserved. Morse Micro and the Morse Micro logo are trademarks of Morse Micro Ltd. All other trademarks and service marks are the property of their respective owners.

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™