

INTRODUCTION

The Movesense sensor platform allows OEM customers to implement their own solutions quickly and easily and to shorten the time to market for new sensor products. Because the Movesense sensor allows companies to deploy their own custom sensor firmware, companies should also be aware of things their firmware affects outside of their own customers.

For customers that opt to use production ready Movesense sample apps instead of their own firmware, such as Movesense “Default firmware”, the requirements are less strict, but it is still good to know the rest of the details.

This document provides instructions for Movesense OEM Customers to prepare the sensor firmware to production. The same content is available at: [Preparing for Volume Orders/Production](#) - Movesense.

FROM FACTORY TO CUSTOMERS

There are many steps and details in the manufacturing of the Movesense sensors that may not be obvious. They however pose a set of requirements that must be fulfilled for the process to be successful.

Production Line

In the factory production line, the Movesense sensor hardware is first programmed with our internal production firmware. This firmware allows the production line equipment to interact with the sensor, testing all the hardware and programming the necessary data into it, such as product information.

When the sensor has passed all the test steps, it is programmed with the firmware chosen by the customer. To program the firmware, a .hex file format is used. Depending on the base Movesense framework version used (1.9.x or 2.x series), a different file used. For 1.9.x the firmware is typically provided as three separate hex-files:

- `bootloader.hex`
- `s132_nrf52_4.0.5_softdevice.hex`
- `Movesense_with_settings.hex`

For newer 2.x series firmwares the build system combines all the necessary data into `Movesense_combined.hex` file which is produced by the firmware build process.

After the firmware is programmed, the sensor is set to “power off mode with stud contact wakeup”. This internal power-off state is a one-time power-off state to guarantee that the sensor is in power off when leaving the production line. However, it is possible that further activity such as packaging or quality control steps wake up the sensor from this power off state.

After the power off, the sensors are packaged in the packaging chosen by the customer and prepared for the shipping.

Movesense Default Firmware from BitBucket

As explained before, at the time of manufacturing, the firmware version chosen by the customer is programmed to the Movesense sensor. If the customer has used Movesense Default Firmware to test and develop their application, it is important that they are aware of the firmware and the version they have been using. Multiple Movesense sample firmware versions, including simple demos, such as “Blinky”, are available at:

<https://bitbucket.org/movesense/movesense-device-lib/src/master/samples/bin/release/>

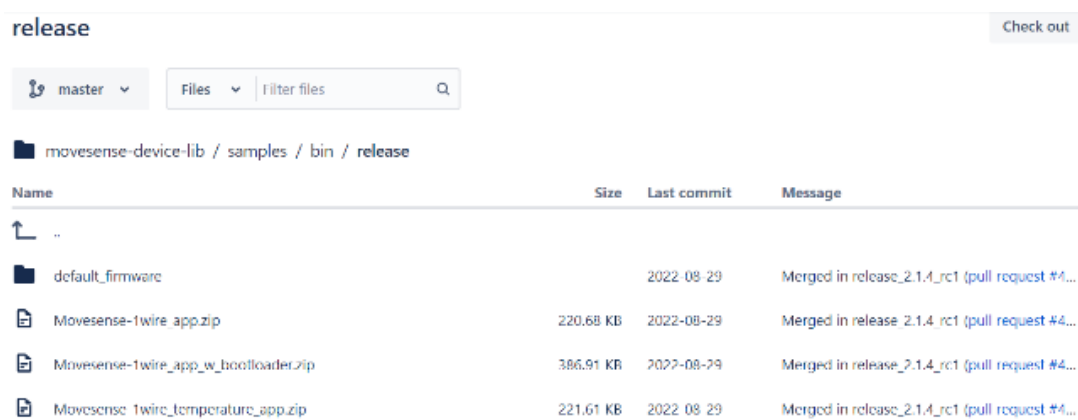


Image 1: Folder containing sample firmware list

All the Default Firmware versions are in the default_firmware folder:

https://bitbucket.org/movesense/movesense-device-lib/src/master/samples/bin/release/default_firmware/

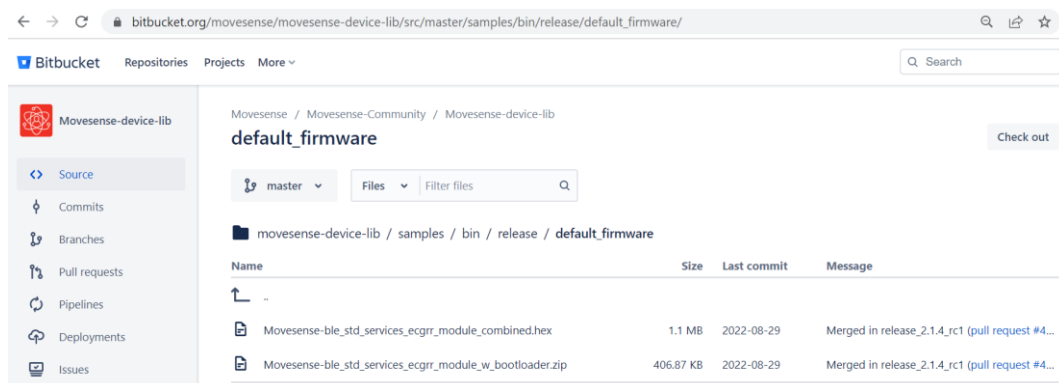


Image 2: Folder containing Default Firmware only

When opening the Default Firmware folder, only the latest Default Firmware version is visible. Older firmware versions are found under master button (see image 2).

Before starting to manufacture OEM sensors, each customer needs to define and provide to Movesense the exact firmware to be programmed to the sensors. The same process applies even if you desire to use the Movesense Default Firmware. You can download the default firmware from the Movesense Bitbucket in .hex file format and sent to the Movesense Team.

OEM Customers are responsible for verifying and testing the suitability of the chosen firmware to the customer specific use case and, once happy with its performance, for providing the desired firmware version to Movesense for pre-programming into customer specific OEM sensors.

Movesense programs the sensors on the production line, using the firmware specified and delivered by the customer, but Movesense is not able to verify which firmware version is suitable for each customer specific use case. This verification is the responsibility of each OEM customer. However, the Movesense team is happy to help if guidance is needed.

In testing and development phase, the sensor firmware can be updated by a smartphone application or a programming jig that can be purchased from Movesense.com: <https://www.movesense.com/product/movesense-jig/>.

Step by step guide on how to locate and download Movesense Default Firmware in .hex file format from the Movesense Bitbucket

1. Go to: https://bitbucket.org/movesense/movesense-device-lib/src/master/samples/bin/release/default_firmware/
 - a. The latest Movesense Default Firmware is listed on the page (Image 3). There are two files: raw .hex and a compressed .zip pack. Both files have a message text after them, which can be used to check the file version.
 - b. If you are happy with the latest firmware and its version shown, skip directly to step 3) below.

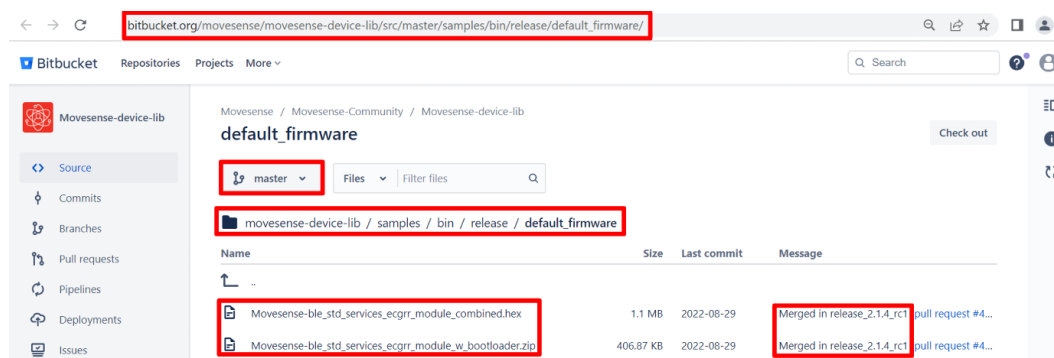


Image 3. The latest version of default firmware (Master) is located here. However, when the new firmware version releases the content here changes to that.

2. To find earlier firmware versions:

- Click “Master” and select “Tags” (Image 4)
- The list shows the latest firmware version (“Master”) and previous versions
- Select the intended firmware version to open its folder.

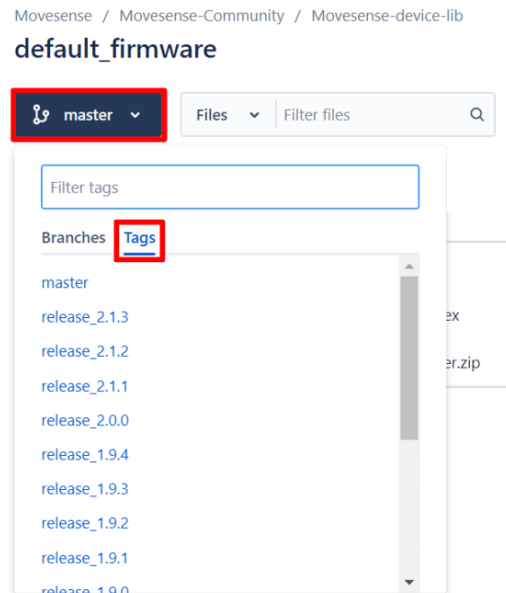


Image 4. The latest version is referred as “master” version. All older versions are visible under “Master” -> “tags”.

3. After the wanted firmware version folder is opened, click the .hex file. New web page opens.

4. Click the three dots on the right side of the page and click “Open raw”. (Image 5)



Image 5. After selecting the desired firmware version, click the .hex file to the view shown. Next click the three dots near to “Edit” link (on the right side in the image) and select “Open raw”. This opens the file in a browser window.

5. Right click the page to open the context menu and click “Save as”. (Image 6)

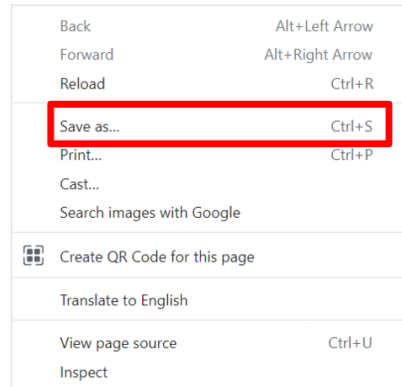


Image 6. Right clicking the page opens context menu. Click “save as” to save the file.

6. Remove the last four (4) characters from the file name (“.txt”) to save the file as .hex. (Image 7)

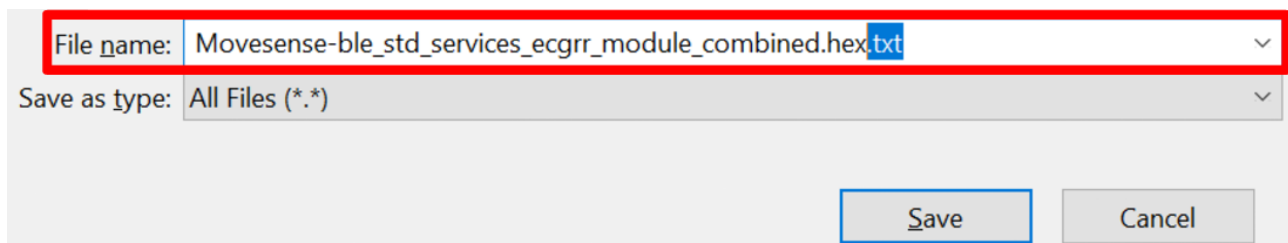


Image 7. When saving the .hex file to the computer, it is possible that its name needs to be modified. Remove the “.txt” suffix from the file name, so that the file name ends with “.hex”. After selecting the wanted path for the file, press Save.

7. Once the file has been saved to the computer, you are ready to send it to the Movesense Team, for example via email.

Shipping

The sensors are shipped first to our intermediate quality control and forwarded from there to their final destination around the world using the agreed-on shipping methods. Since most shipping methods include air cargo, the IATA rules for air freight also apply. The most important of those is the requirement to not to use radio transmitters during the flight. To comply with the requirement, it is necessary to implement a “shipping mode” in the custom firmware unless the firmware is such that it automatically goes to “non-transmitting” mode like Movesense Default firmware.

Shipping mode in firmware

If the sensor firmware is meant to stay powered on even without Bluetooth connection, the easiest way to implement the shipping mode is:

1. When the user is first time taking sensor into use (real mobile connection from your own app), write a piece of information in the sensors EEPROM memory. A small area of EEPROM can be dedicated for this kind of information using

the `DATALOGGER_MEMORY_AREA` -macro that is found at https://movesense.com/docs/test_env/esw/data_storage/#datalogger-and-logbook-memory-area

2. In the bootup, check the EEPROM content written in step 1 to see if sensor was taken into use. If the sensor was not taken into use:
 - wait for a while (1-2 minutes), in case mobile app makes a connection.
 - If connection was made, see step 1 above
 - If there was no connection (or the connection did not result start of operations), set wakeup to stud contact using `/Component/MAX30004` -API and shutdown the sensor. For an example, see `hr_wakeup_app` in `movesense-device-lib/samples`.

Device Firmware Update (DFU) on the field

When the sensor is in its final use, it may need to be updated to a new firmware. The firmware update should be integrated as a part of your own mobile software so that the process is as easy as possible for the final user. It is very important to test the DFU functionality of each custom firmware version before it is sent to customers for updating existing sensors or sent to be programmed on the production line to new sensors.

The minimum list of tests that need to be performed for each new firmware:

- Test that the old firmware can be updated to the new one using your mobile application.
- Test that the new firmware can be updated to another firmware (or the same) using the mobile application to ensure that the DFU works also with the next version of firmware.
- Test the current consumption of the sensor right after the DFU update and during and after possible power off.

INFORMATION FOR MOVESENSE

The following information must be provided to Movesense when making a bulk OEM order of Movesense sensors:

Information	Description
Firmware file(s)	Either Movesense_with_settings.hex (Movesense 1.9.x) or Movesense_combined.hex file. Preferably named: <i>CompanyAppName_FW_AppVersion_Date.hex</i> ("SuperTzap_FW_1.2.3_2022-10-11.hex")
Product Name	Name of the product that is returned by /Info -service. This string is used as a basis of BLE advertising device name (first 9 characters). Example: "SuperTzap".
App Name	Exactly as written in the App.cpp
App Company	Exactly as written in the App.cpp
App Version	Exactly as written in the App.cpp
Confirmation of shipping mode	"Yes, shipping mode has been implemented" / "Shipping mode is not needed (goes to sleep in normal operation)"
Movesense framework version	As reported by the /Info -service. (e.g. "2.1.1")