Getting started with the STM32Cube function pack FP-IND-MCAI1 and the EVLSPIN32G4-ACT reference design based on the STSPIN32G4 for edge AI integration

Introduction

The FP-IND-MCAI1 function pack is a reference code to control a low voltage three-phase brushless motors with EVLSPIN32G4-ACT reference design, based on the STSPIN32G4, ST's STSPIN flagship device for 3-phase brushless DC, an extremely integrated and flexible motor controller with an embedded Cortex®-M4 microcontroller, monitoring the vibrations using the STEVAL-C34KAT1 sensor kit, and classifying the operating conditions through an optimized AI algorithm.

The function pack comes with an implementation example of a motor fault classification based on a machine learning (ML) solution developed through NanoEdge™ AI studio, working in parallel with the field-oriented control motor control algorithm (FOC) generated with the STM32 motor control software development kit (X-CUBE-MCSDK).

The firmware collects data on the motor current and the IIS3DWB vibrometer (mounted on the STEVAL-C34KAT1 board). This data stream feeds the ML model allowing an accurate classification of motor behavior among normal operation (no faults) and two possible fault conditions.

The user can customize the ML model by changing motor configuration and adding their own classes.

1 1. FP-IND-MCAI1 software expansion for STM32Cube

1.1 Overview

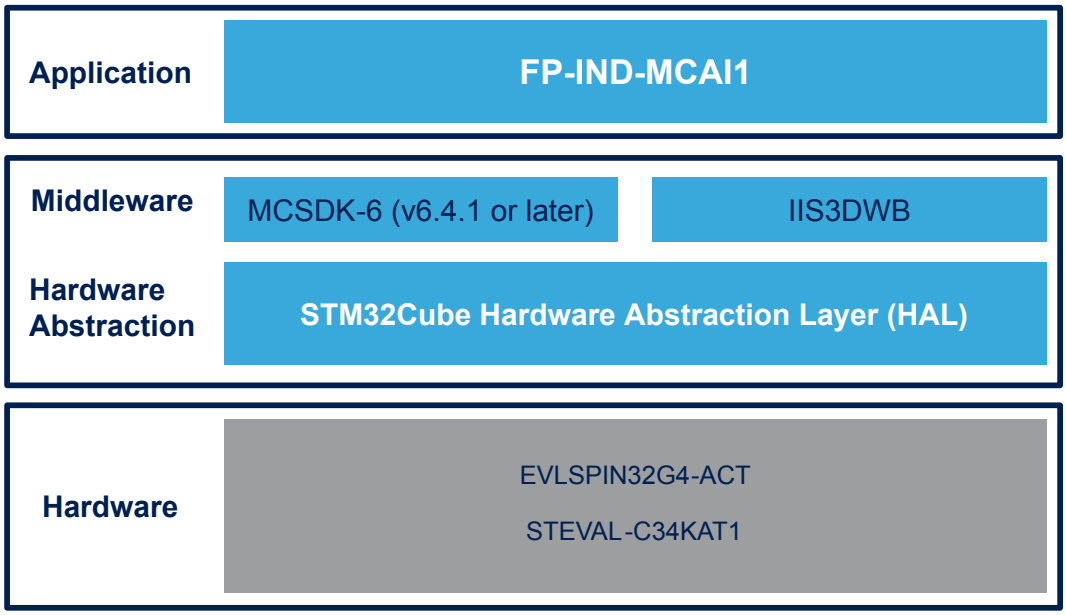
The FP-IND-MCAI1 features:

- a sample application with FOC algorithm to drive a low voltage three-phase brushless motor;
- an ML model for motor behaviour classification;
- Motor Control Protocol (MCP) implementation to interact with the evaluation board through the motor pilot tool included in the MCSDK;
- free, user-friendly license terms.

1.2 Architecture

The FP-IND-MCAI1 package was developed for the EVLSPIN32G4-ACT board.

Figure 1. FP-IND-MCAI1 software architecture



The FP-IND-MCAI1, compliant with STM32Cube architecture, is structured into a set of layers of increasing abstraction.

The hardware abstraction layer (HAL) interfaces with the hardware and provides the low-level drivers and the hardware interfacing methods to interact with the upper layers (application and libraries). It provides APIs for communication peripherals (SPI, UART, etc.) for initialization and configuration, data transfer, and communication errors. There are two types of HAL driver APIs:

- generic APIs, which provide common and generic functions to the entire STM32 series,
- extension APIs, which provide specific, customized functions for a particular family or a specific part number.

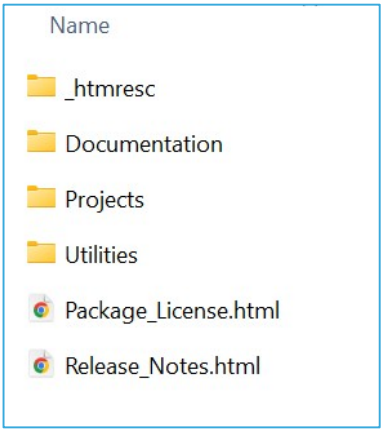
The package extends STM32Cube by providing a board support package (BSP), which deals with the board specific peripherals and functions (LED, user button, etc.). The BSP structure follows the hardware structure, including a component management layer as well as the specific layers of the boards used.

On top of such features, which are inherited from STM32Cube, FP-IND-MCAI1 adds the code reusability at application level that is based on the STM32 motor control software development kit (MCSDK) and NanoEdge™ AI library.

1.3 Folder structure

The FP-IND-MCAI1 firmware package folder structure follows a layer-based approach similar to the one of the STM32Cube architecture.

Figure 2. FP-IND-MCAI1 package folder structure



The folders included in the software package are:

1. **Documentation**: contains a compiled HTML file generated from the source code, which details the software components and APIs.
2. **Projects**: contains a sample application for motor control with AI data classification. This application is provided for the EVLSPIN32G4-ACT with three development environments:
   a. STM32CubeIDE
   b. IAR Embedded Workbench for ARM (EWARM)
   c. ARM® RealView™ Microcontroller Development Kit (MDK-ARM)
   Differently from the standard STM32Cube architecture, the Drivers and the application-specific libraries are included in the Projects folder. In particular:
   a. **Drivers**: contains the HAL drivers and the board-specific drivers for each supported board or hardware platform, including the on-board components and the CMSIS vendor-independent hardware abstraction layer for ARM Cortex®-M processor series.
   b. **NanoEdgeAI**: contains the NanoEdge™ AI library used for data classification
   c. **MCSDK_v6.4.1-Full**: contains the motor control library with field orient control algorithm, delivered with the 6.4.1 version of the ST Motor Control Software Development
3. **Utilities**: contains complementary project files (MC Data logging source files and the dataset).

2 Getting started

2.1 System configuration

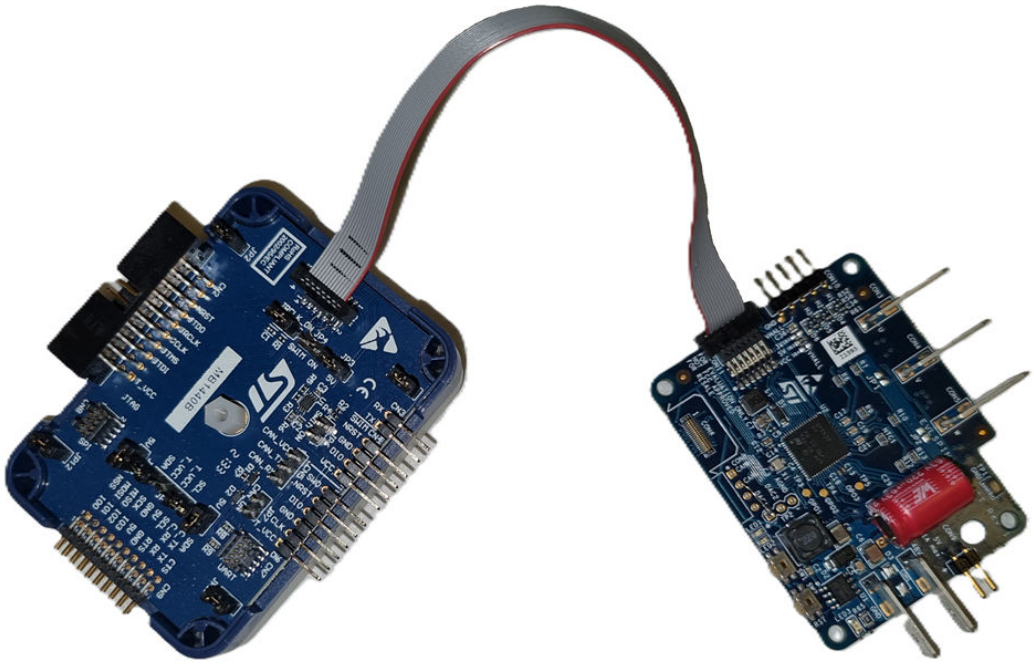
The FP-IND-MCAI1 requires the following hardware to run:

- EVLSPIN32G4-ACT: STSPIN32G4 reference design for next-generation smart actuators.
- STEVAL-C34KAT1: Digital accelerometer sensor kit based on IIS3DWB.
- STLINK-V3SET or STLIN-V3MINIE: In-circuit debugger and programmer for STM32/STM8.
- three-phase brushless motor (the example is based on the MAXON EC-i 40 100 W included in the EVALKIT-ROBOT-1 reference design kit).
- power supply providing 36 V / 3 A output.

2.2 How to program EVLSPIN32G4-ACT

The EVLSPIN32G4-ACT board can be programmed with an external programmer such as the STLINK-V3SET or STLIN-V3MINIE.

Figure 3. EVLSPIN32G4-ACT and STLINK-V3SET programmer



Once the hardware connections are in place, you can either:

- download the sample application binary provided using, for example, the STM32CubeProgrammer,
- recompile and flash memory one of the provided projects with your preferred IDE as, for example, STM32CubeIDE.

3 Firmware

3.1 Overview

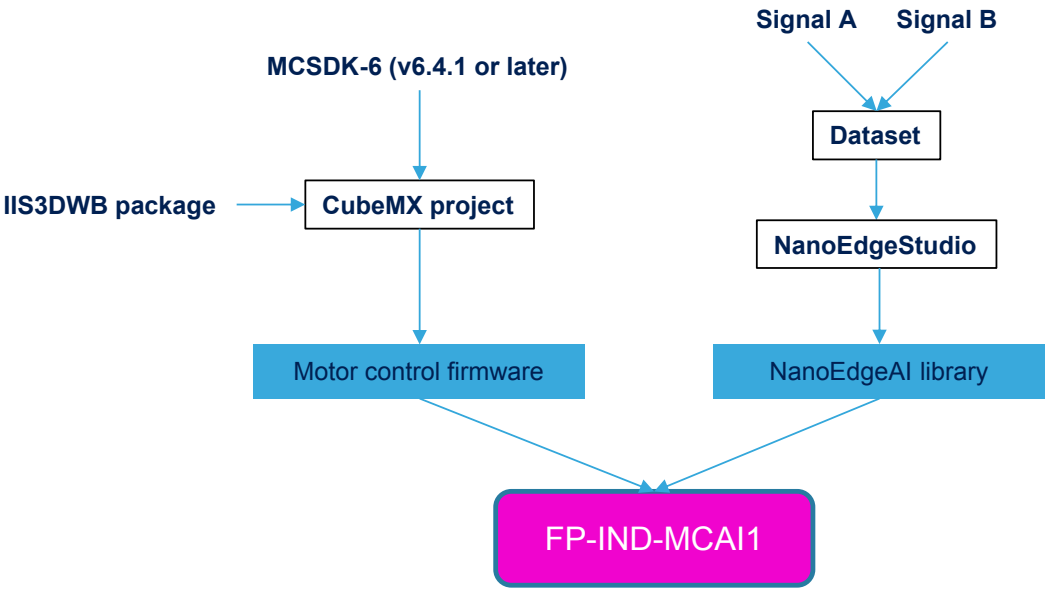
The FP-IND-MCAI1 is a reference example about how to integrate the NanoEdge™ AI library into a motor control application, with both the elements running in parallel onto the same microcontroller.

The motor control application is based on the motor control library configured through the motor control workbench, included in the STM32 MCSDK, to work with the combination of the EVLSPIN32G4-ACT inverter board and the Maxon EC-i 40 100 W brushless motor.

The IIS3DWB package was added to the STM32CubeMX project allowing the data acquisition form a vibrometer embedded on the STEVAL-C34KAT1.

In parallel to the FOC algorithm, the ML algorithm classifies the data stream from runtime acquisitions and eventually identifies faults based on a previously performed training process whose output is a dataset and a library generated with the NanoEdge™ AI studio.

Figure 4. FP-IND-MCAI1 firmware overview



The integration of the NanoEdge™ AI library into the motor control firmware relies on the parallel execution of a dedicated state machine whose purpose is to analyze the data continuously acquired and buffered by the motor control algorithm.

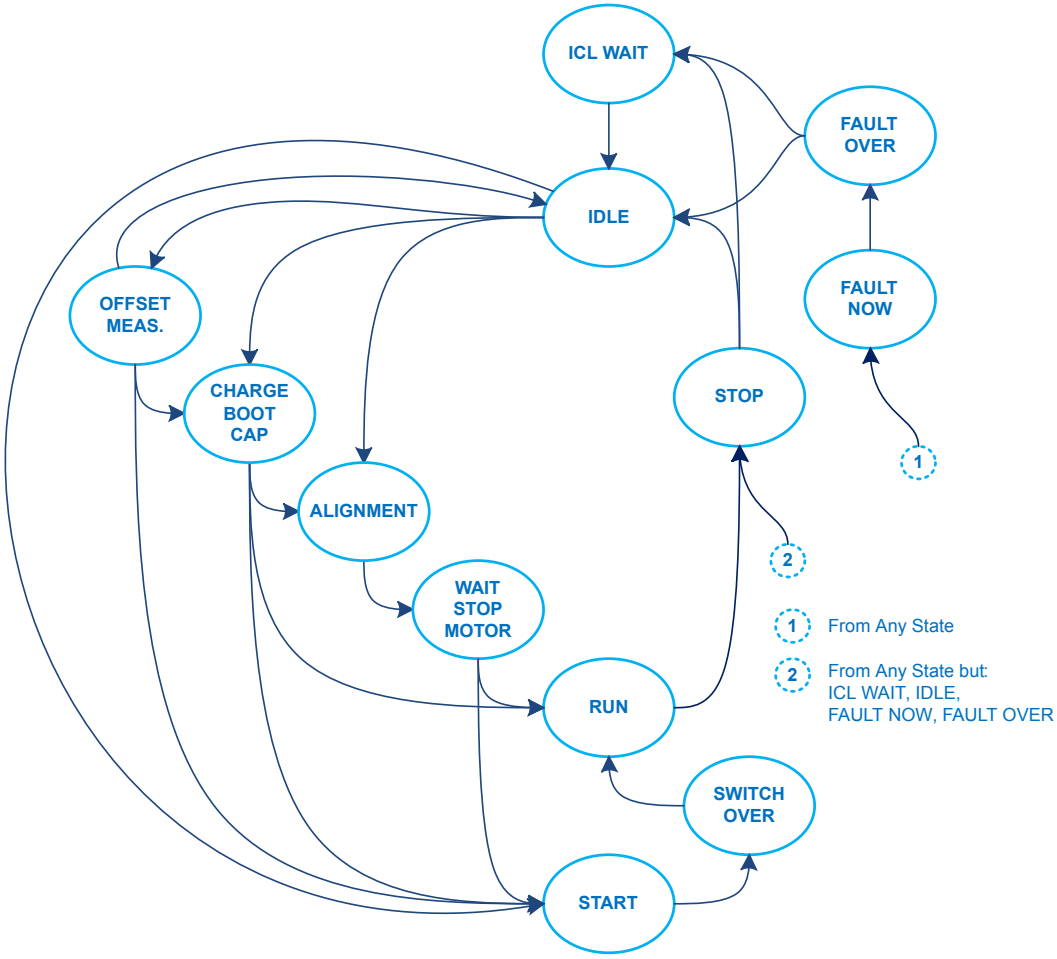
The dataset used by NanoEdge™ AI studio to generate the library is provided with the package. The dataset was generated in the following conditions (knowing the dataset creation conditions are essential to perform a proper fault identification, see Section 4.2):

- signal A: I_q (field-oriented control quadrature current), sampling 20 kHz, decimation factor 4
- signal B: X-axis accelerometer value, sampling 20 kHz, decimation factor 4.

3.2 Motor control state machine

The operation of the motor control FOC library is driven by the state machine described in the Figure 5 chart. Refer to the MCSDK documentation for the detailed description of all the states.

Figure 5. Motor control state machine



This state machine handles the motor start, motor stop, and fault management.

In the example, it has been modified to:

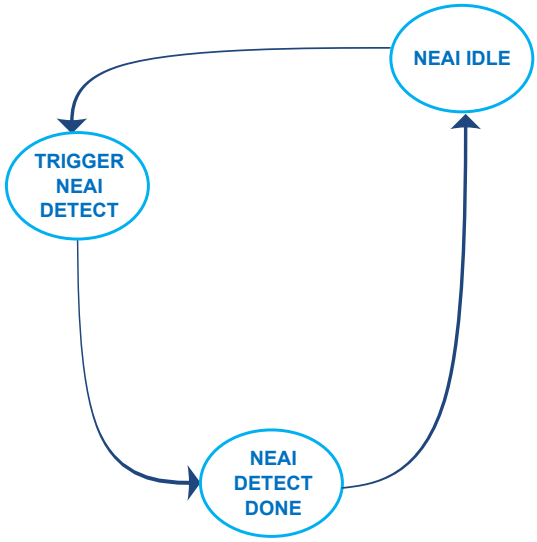
- initialize the IIS3DWB digital vibrometer,
- collect the motor telemetries and the vibrometer data,
- buffer the data in the array, which made available to the NanoEdge™ AI library,
- check the results of the data classification and, in case of a detected fault, enables an LED blinking on the EVLSPIN32G4-ACT board (LED1).

The motor control state can be changed by the user with the motor pilot, sending a RUN/STOP command to the board via USART. For additional information about the motor pilot application, please refer to the MCSDK documentation.

3.3 NanoEdge™ AI library state machine

The NanoEdge™ AI library state machine works in parallel with the motor control one.

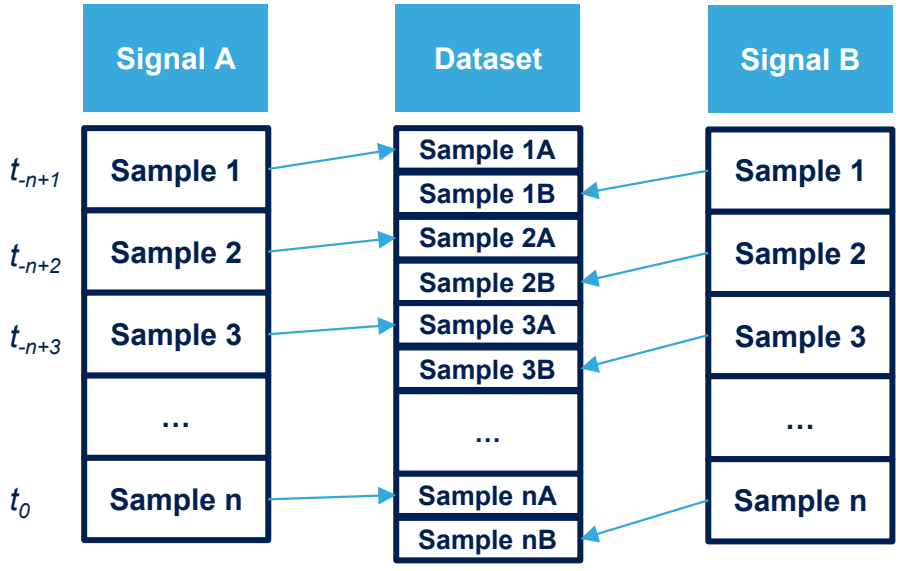
Figure 6. NanoEdge™ AI library state machine



It is composed by only three states:

- NEAI IDLE: The data are continuously acquired and the system is waiting for the request of its analysis. The request is carried out by the expiring of a countdown (AI_DELAY_TICKS in mc_tasks.c) and its value can be changed by the user according to its application (the default value is 2 seconds).
- NEAI TRIGGER DETECT: The firmware enters this state at data analysis request. In this state, the NanoEdge™ AI library sorts the collected and invokes the classification function.
  The sorting of the data must be the same as the dataset used during the training process.
  The position of the signals inside the array passed to the classification function must be the same as the one used for the training dataset and the samples must be sorted in chronological order, from the oldest to the newest one.

Figure 7. NanoEdge™ AI library data sorting



- NEAI DETECT DONE. This is a transition state. Once the classification function returns the result, the firmware enters this state, it starts a new countdown and switches to NEAI IDLE state.

4 Configuration customization

The package is provided with a pre-configured motor control algorithm (3-shunt sensorless FOC) generated for MAXON EC-i 100 W, a 3-phase brushless motor. On the other hand, the dataset for the data classification was generated to identify normal condition, a fault related to an increase of the motor vibrations and a fault related to an unstable motor load.

The package is intended to be a reference of how to integrate the NanoEdge™ AI library into the MCSDK algorithm. However, the user can exploit and customize it according to its needs and its application requirements.

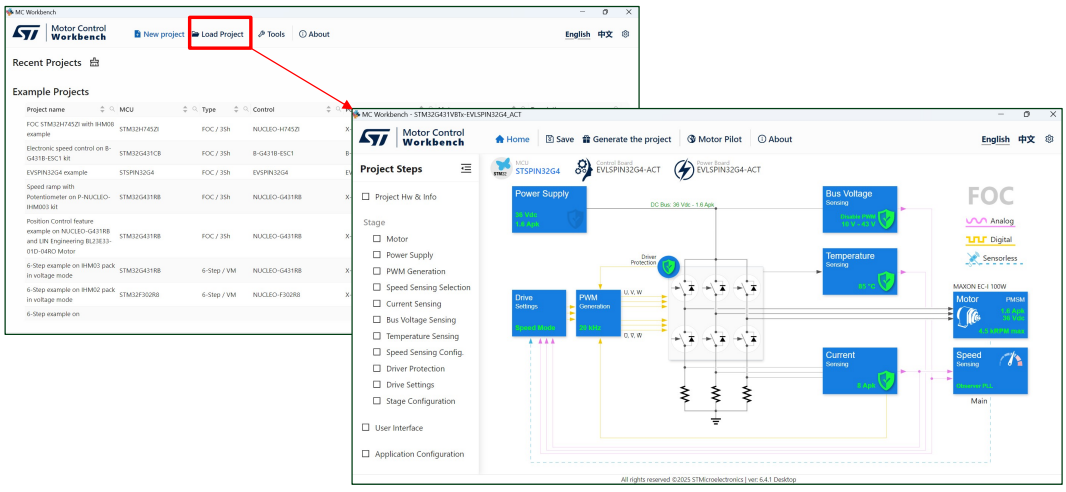
4.1 Motor control parameters

In this section, the user is taught how to customize the following parameters:

- motor parameters
- power supply
- PWM generation frequency
- FOC execution rate
- shunt resistor value and current sensing gain
- sensorless start-up parameters
- PI controller gains (current regulator, speed regulator)
- bus voltage and temperature protection thresholds

The user can customize the motor control parameters by loading the provided *STM32G431VBTx-EVLSPIN32G4_ACT.ioc* file into the motor control workbench (version 6.4.0 or later).

Figure 8. Opening the ioc file with the motor control workbench



Once loaded the project, the customizable parameters can be found in the corresponding tab of the tool.

See the STM32 MCSDK technical documentation for further details about the configuration of the motor control library.

After saving it, the new code can be generated by clicking on the “generate the project” button and choosing the preferred toolchain.

Note: The new parameters overwrite the old ones. It is recommended to create a backup copy before proceeding.

Please also note that the new code MUST be generated in the same folder location of the previous one since this process needs the original configuration files that are included in the package.

Otherwise, the generated code gives compilation errors.

Figure 9. Generation of the new code with the motor control workbench

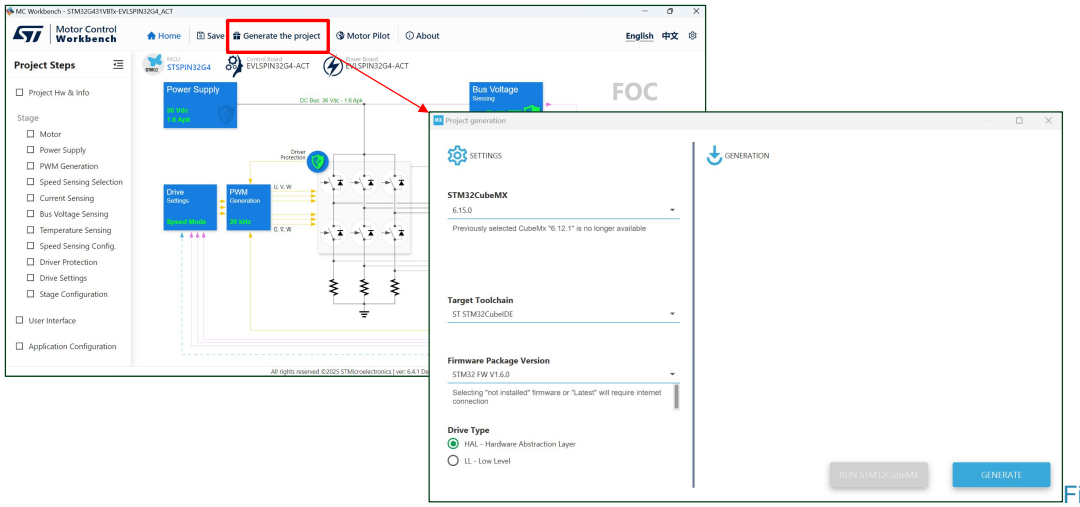
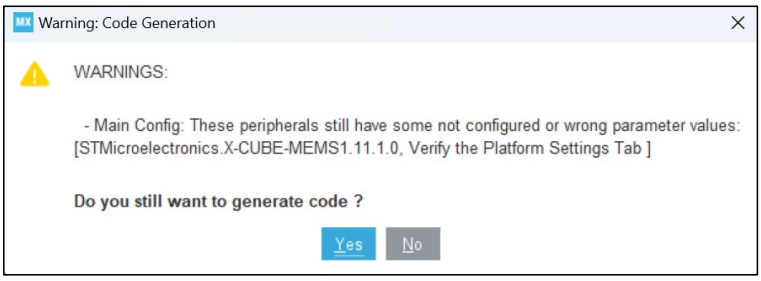


Figure 10

If the generation process shows the message in, click YES.

The warning is related to the IIS3DWB device whose configuration will not affect the goodness of the result.

Figure 10. IIS3DWB warning



4.2 EdgeAI dataset

The FP-IND-MCAI1 package comes with a dataset whose role in the algorithm was described in Section 3.1.

The dataset was created using two signals (FOC quadrature current and x-axis accelerometer) to identify 3 classes: normal motor operation, fault1 (high motor vibration) and fault2 (unstable motor load).

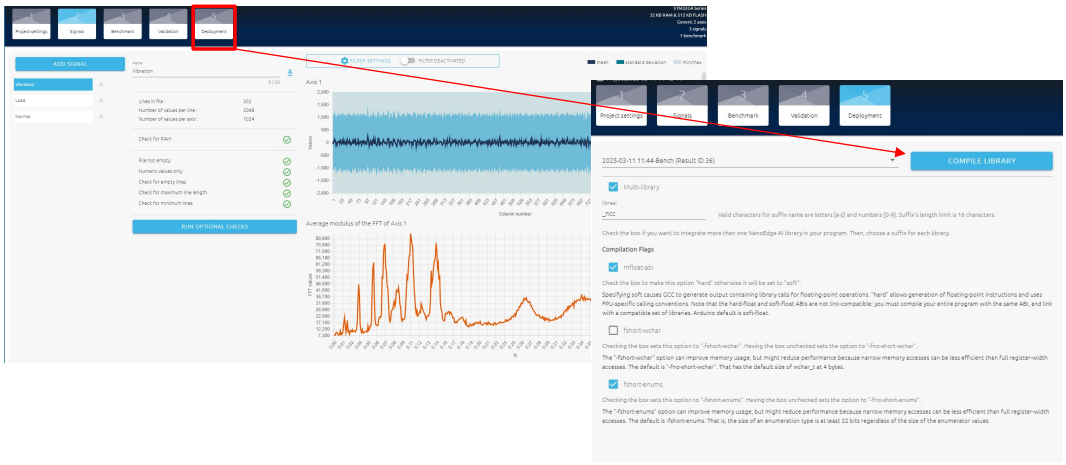
Note: ***It is important to note that, to properly classify the data, the sampling conditions with which the dataset is created must reflect the sampling conditions that will be used in the application.***

In particular, the two signals are collected during the high frequency task of the FOC algorithm whose frequency is the PWM frequency divided by the FOC execution rate (respectively 20 kHz and 2 in the example). Therefore, the dataset was created by acquiring data at 10 kHz (20 kHz / 2) in the three operative conditions described before (normal, fault1, and fault2).

To increase the time window of the sampling after buffering the data was then decimated by a factor 4. Again, this was done during the creation of the dataset and during application. The decimation factor (AI_DECIMATION) is defined in the file *mc_tasks_foc.c*. The decimation order does not have to affect the ability to recognize sharp and quick variation of the signals that may prevent a good classification.

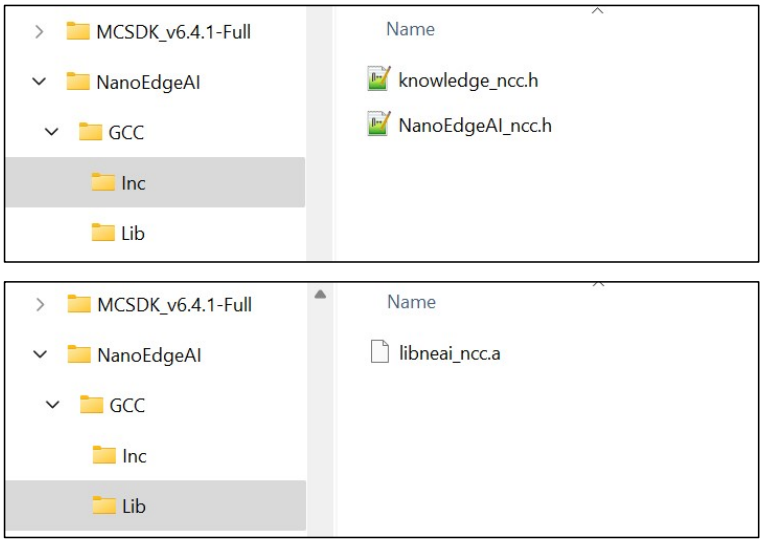
Finally, the dataset is imported into the NanoEdge™ AI studio to create the library used by the firmware.

Figure 11. Generation of a new library with NanoEdge™ AI studio



The files generated by NanoEdge™ AI studio must replace the ones in the subfolders of the “NanoEdgeAI” folder of the FP-IND-MCAI1 package.

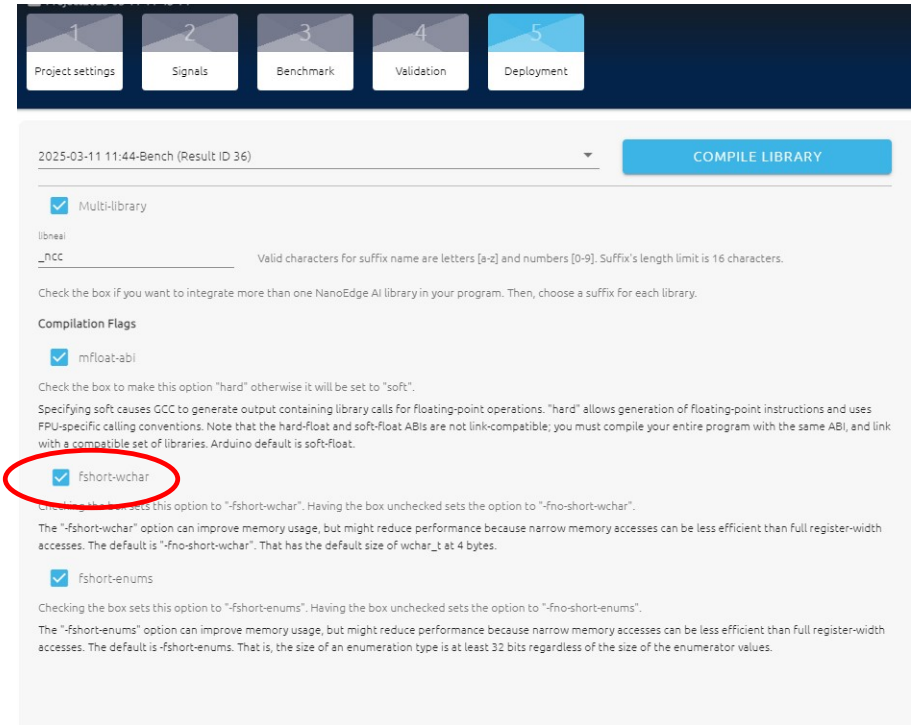
Figure 12. “NanoEdgeAI” folder update



4.2.1 NanoEdgeAI library generation for Keil® MDK-ARM toolchain

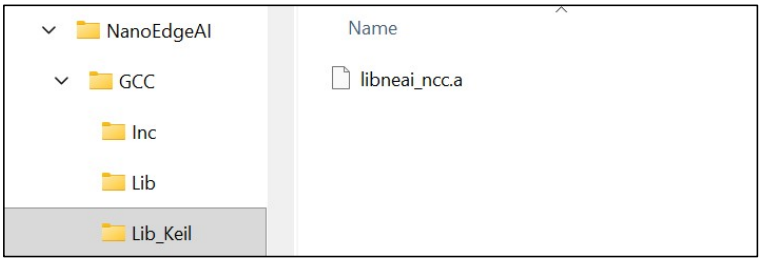
After having created the dataset and imported the signals in the NanoEdge™ AI studio, the generation of the NanoEdgeAI library for Keil® MDK-ARM toolchain requires the enabling of the flag *fshort-wchar* in the window shown below.

Figure 13. Library generation for Keil® MDK-ARM toolchain



While the files *.h* generated by NanoEdge™ AI studio, they must replace the ones in the *Inc* subfolder of the “NanoEdgeAI” folder of the FP-IND-MCAI1 package, as already explained in the previous section, the file *libneai_ncc.a* must replace the one in the dedicated folder *Lib_Keil*.

Figure 14. “NanoEdgeAI” Keil® MDK-ARM folder update



Revision history

Table 1. **Document revision history**

Date	Version	Changes
15-Dec-2025	1	Initial release.

Contents

List of tables

List of figures