Motor Testbench quick start guide

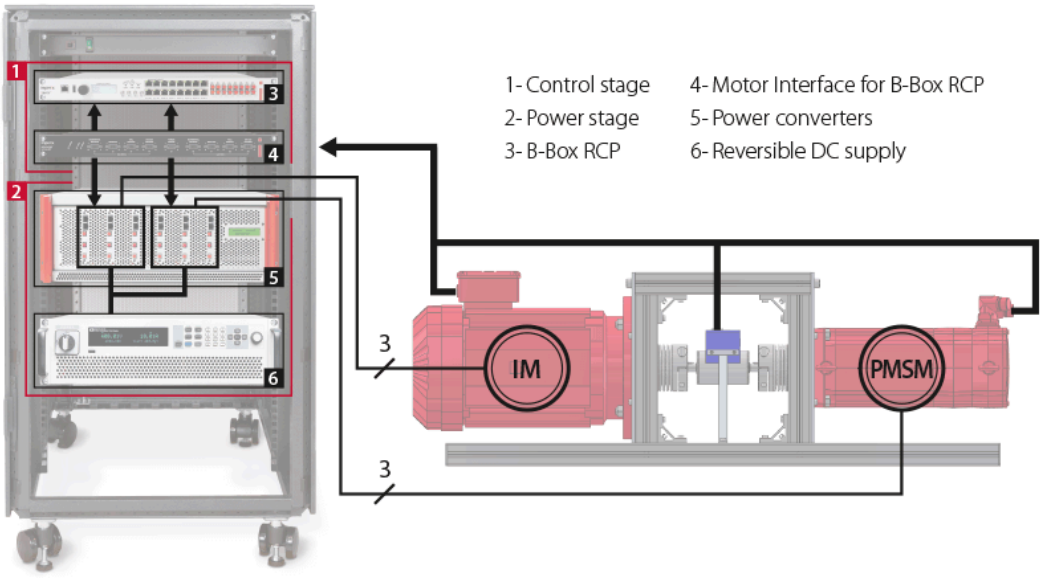
PN181 | Posted on November 14, 2022 | Updated on June 10, 2025

Simon STROBL
Product Director
imperix

Table of Contents

This page explains how to get started with the motor drive bundle and the motor testbench. It provides a comprehensive overview of the hardware configuration and step-by-step instructions to commission the equipment.

The bundle includes all the necessary equipment to operate the induction machine (IM) and the permanent magnet synchronous machine (PMSM) in a back-to-back configuration. It covers the entire motor-drive system from the digital controller to the power stage and the motors. In this type of setup, one machine can serve as the device under test (DUT), while the other one acts as a programmable mechanical load. The current page covers only the default configuration of the bundle, but a wide range of topologies can be explored thanks to the flexibility of imperix's solutions. For more details on other possible topologies, please refer to the page How to build a variable speed drive.



Default configuration of the motor drive bundle

Setting up the motor testbench

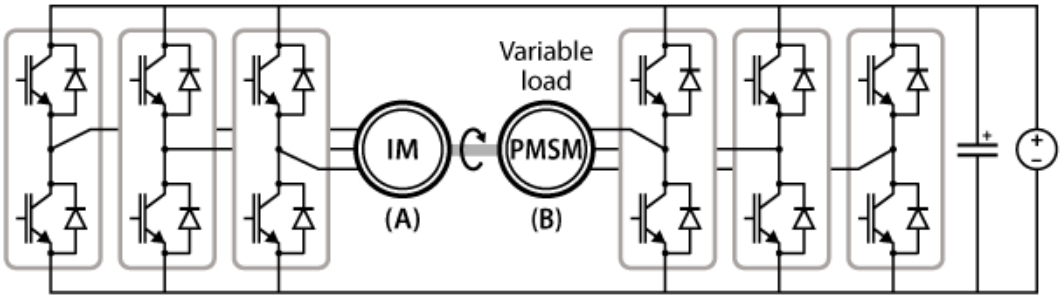
The default content of the electric motor drive bundle is listed below:

- Programmable controller (B-Box RCP)
- Motor Interface for B-Box RCP
- ACG SDK toolbox for automated generation of the controller code from Simulink or PLECS
- 6x phase-leg modules (PEB8038)
- Motor Testbench, including an induction machine and a permanent magnet synchronous machine (see the datasheet)
- Reversible DC source (third-party)
- All necessary RJ45 and fiber optic cables
- Laboratory safety cables (similar to this one)

The knowledge base provides multiple turn-key code examples for the **default configuration** of the motor drive bundle. If the user makes any modifications to the wiring schemes shown in this quick start guide, the code examples must be modified accordingly.

Wiring the motor testbench to the power stage

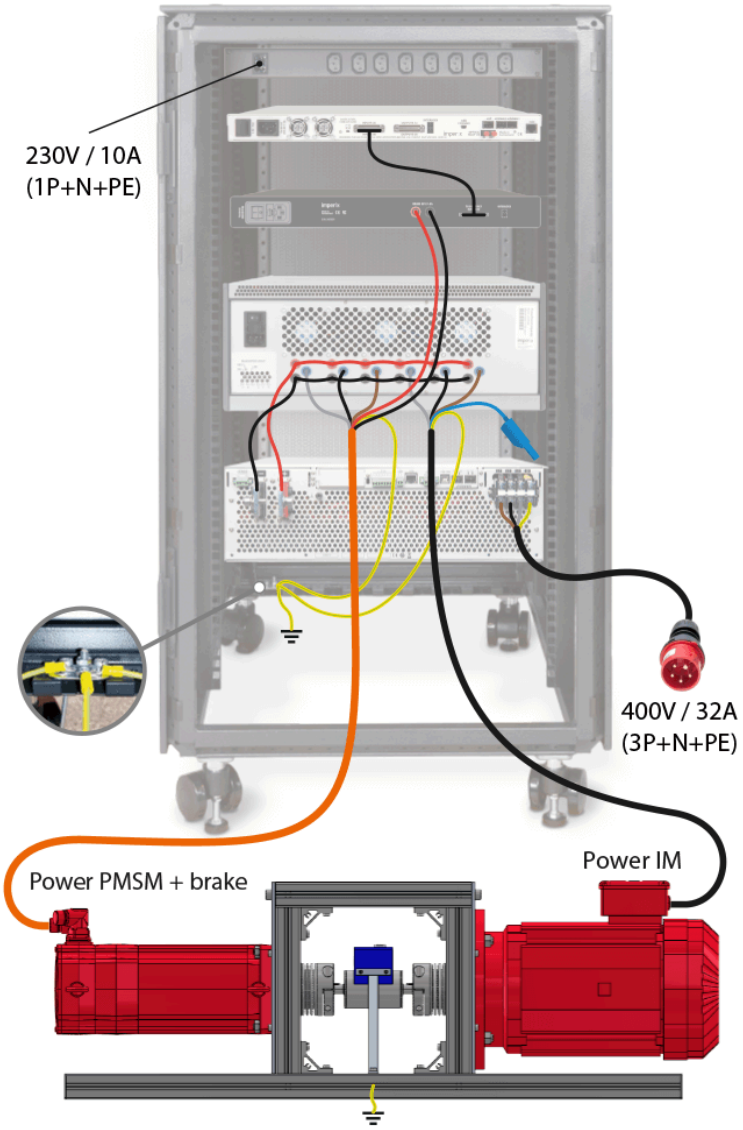
By default, imperix pre-wires the power stage as two inverters with a shared DC bus, as shown in the image below. With this topology, one machine operates as a motor and the other one as a generator. The power circulates in a loop through the DC bus and the shaft, and the DC source compensates for the losses of the system.



Default topology of the power stage

The DC bus is already connected to the DC source out of the box. However, the user must connect the machines to the cabinet themselves, as the motor testbench is shipped on a separate pallet. The image on the right illustrates how to connect the machines to their respective inverter. *(Note: the center point of the windings of the IM (blue connector) is left unconnected in the topology considered on this page).* Additionally, the built-in brake of the PMSM must be connected to the brake control unit of the motor interface. Make sure to observe the polarity of the brake.

The cabinet and the motor testbench must be connected to the earth to ensure the electrical safety of the system and optimal EMC performance. Additionally, earthing cables should be as short as possible and connected to a common ground to avoid ground loops.

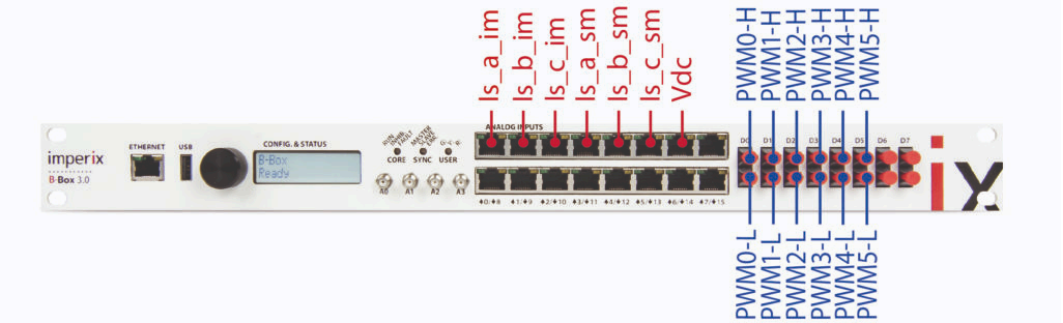


Wiring of the power stage and earthing connection.

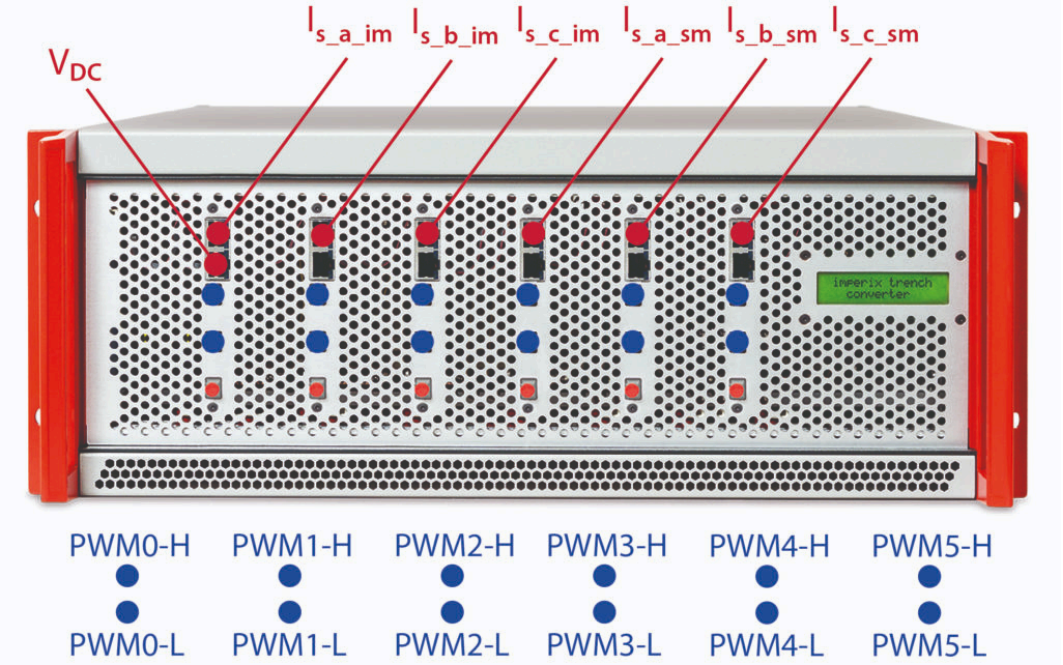
The third-party DC source included in the bundle ships with a 400V/32A IEC60309 plug (3P+N+PE). Additionally, the auxiliary power is supplied through a 230V/10A plug. Equivalent cables are provided depending on the destination country.

Wiring of the control stage

The B-Box controller generates the control signals for the converter (PWM gate signals), based on measured feedbacks (motor currents and DC bus voltage). The gate signals are transmitted to the PEB 8038 modules through optical fibers, whereas the motor currents and the DC bus voltage – measured by the sensors embedded on the PEB 8038 modules – are fed back to the controller through RJ45 cables. The following schematics indicate the assignation of the analog and PWM channels (pre-wired out of the box).

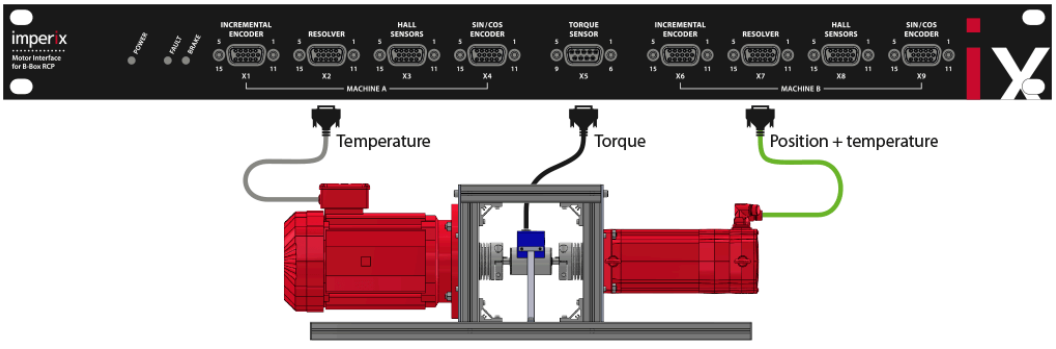


Channel assignation on the B-Box



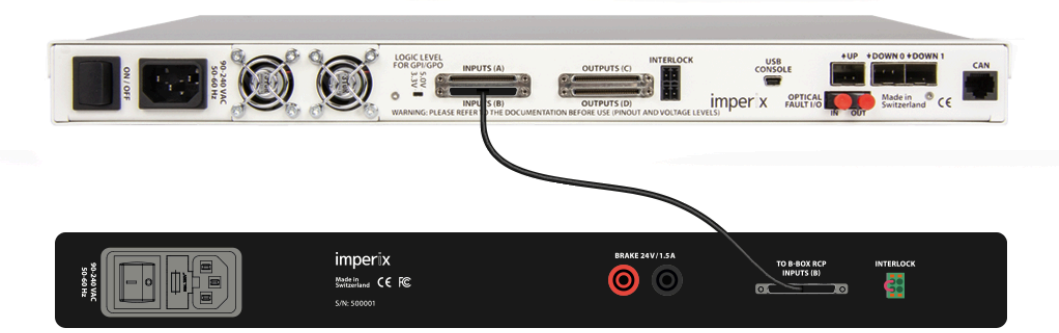
Channel assignation on the converter rack

The figure below illustrates how to wire the measurements from the motor testbench to the motor interface box. The green cable of the PMSM, that combines the resolver and temperature measurements, goes in connector X7, and the temperature measurement of the IM (grey cable) goes to the connector X2. Finally, the torque sensor must be connected to X5.



Connection of the motor measurements to the Motor Interface

The motor interface is connected to the B-Box through VHDCI Inputs (B) with the provided cable.



Connecting the motor interface to the B-Box

An emergency stop button can be connected to the motor interface through its INTERLOCK input. This button is optional and not included with the interface. If not used, it must be replaced by the included dummy plug to avoid reporting a fault to the B-Box (please refer to the motor interface datasheet for more details).

Commissioning the motor testbench

The motor testbench is a complex system to drive with multiple converters, control loops, and mechanical elements. Thus, imperix recommends proceeding step-by-step to simplify the commissioning.

The B-Box controller can be programmed either in C++ or with Automated Code Generation (ACG) tools from Simulink and PLECS. In this section, the code examples are implemented with Simulink. If you are not familiar with the code generation feature of Simulink, please refer to the Getting started with ACG SDK on Simulink. In any case, make sure to install the latest version of ACG SDK beforehand, available for download at imperix.com/downloads/.

Safety considerations

Before starting any experiment, it is essential to take all necessary precautions to operate the system safely. Please observe the safety measures detailed in the panes below.

Configure the overcurrent and overvoltage protections

To ensure that the ratings of the power converter and machines are never exceeded, the hardware protection limits of the B-Box RCP need to be configured properly. A detailed

explanation of how to compute these limits is given on the page Analog front-end configuration on B-Box RCP.

For the PMSM, the overcurrent protection threshold is set to 1.2 p.u. or 19 A (peak value). It leaves enough margin for the controller to regulate the currents when the machine is operated close to its nominal conditions. For the IM, the V/f control technique leads to a large start-up current due to the discontinuity of the V/f profile at low speed. Thus, the overcurrent protection has to be set to 2.5 p.u. or 30 A (peak value). Additionally, the DC bus overvoltage protection is set to 830V. Please note that the lower limit is set to -20V on the DC bus to avoid tripping the protection while the bus is discharged. The table below gives the complete configuration of the analog front-end of the B-Box according to the aforementioned limits.

Measured signal	Input channel number	Low impedance	Programmable Gain	Filter	Limit high [V]	Limit low [V]	Disable safety
$I_{s,im}$	0,1,2	no	x4	no	6.0	-6.0	no
$I_{s,pmsm}$	3,4,5	no	x8	no	7.6	-7.6	no
V_{dc}	6	no	x2	no	8.2	-0.2	no

Analog inputs configuration

The analog input protections have a tolerance of ±0.2V. For example, the limit high of the DC bus voltage is 8.2V in the previous table. However, the actual protection can be tripped anywhere between 8.0V and 8.4V. Considering the programmable gain and the sensitivity of the voltage sensor (PEB8038), this translates to 810-850V on the DC bus. It is essential to take tolerance into account when setting protection limits.

Secure the motor testbench against unwanted movements

The motor testbench must be horizontally installed on a vibration-free and rigid structure. It must be secured against unwanted movements caused by the vibrations of the machines.

Don’t remove the protective covers

Transparent acrylic plates surround the shaft to prevent accidental access while the machines are spinning and protect against flying parts in case of mechanical failure. Do not operate the motor testbench without the protective covers.

Release the mechanical brake

By default, the brake is active and locks the shaft. Supplying the brake’s coil with 24V releases the mechanism. Deactivating or reactivating the brake emits a very distinctive

and audible sound. The motor interface has a brake control unit to supply 24V to the coil whenever needed. In Simulink, the brake can be released using the BRK -Brake block.

It is essential to release the brake before operating the motors since the induction machine has its cooling fan attached to its rotor and may overheat when the rotor is blocked. Blocked rotor tests can be carried out with the brake applied, but this requires extra care to be taken when running the test to prevent any damage to the machine.

Limit the DC source output current

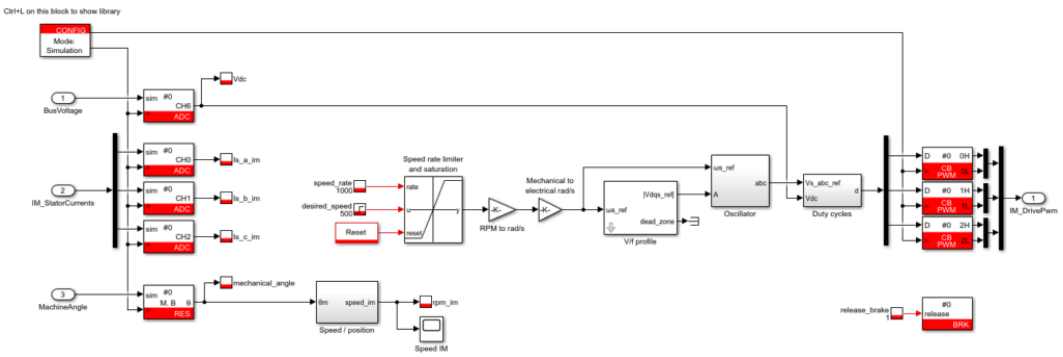
The laboratory DC source from the motor bundle can limit its output current to avoid damaging its load. In the topology considered on this page, the DC source only compensates for the losses and does not require a high output current in steady-state. However, the source needs enough margin to regulate the DC bus voltage under load. Setting the limits to ±5A leaves enough margin for regulation while protecting the DC bus capacitors from excessive inrush currents.

Test 1: no load test in open-loop

Many control techniques rely on a position sensor to accurately estimate the position of the rotor flux. It is also possible to calculate the angular speed by computing the derivative of the position and using it for closed-loop speed regulation. For this reason, it makes sense to check the correct operation of the resolver first. The Simulink model provided below implements an open-loop V/f control of the IM that sets its speed to a known value without relying on a speed measurement. Additionally, to avoid slowing down the rotor, the PMSM is not supplied (no load test). More details on the control can be found in V/f control of an induction machine.

The code example uses Simscape Electrical to model the motor testbench. Matlab R2023a (or later) is required to run the model in simulation mode.

TN138_Motor_TB_no_load_test_powerlibDownload



Overview of the Simulink control file for the open-loop V/f test

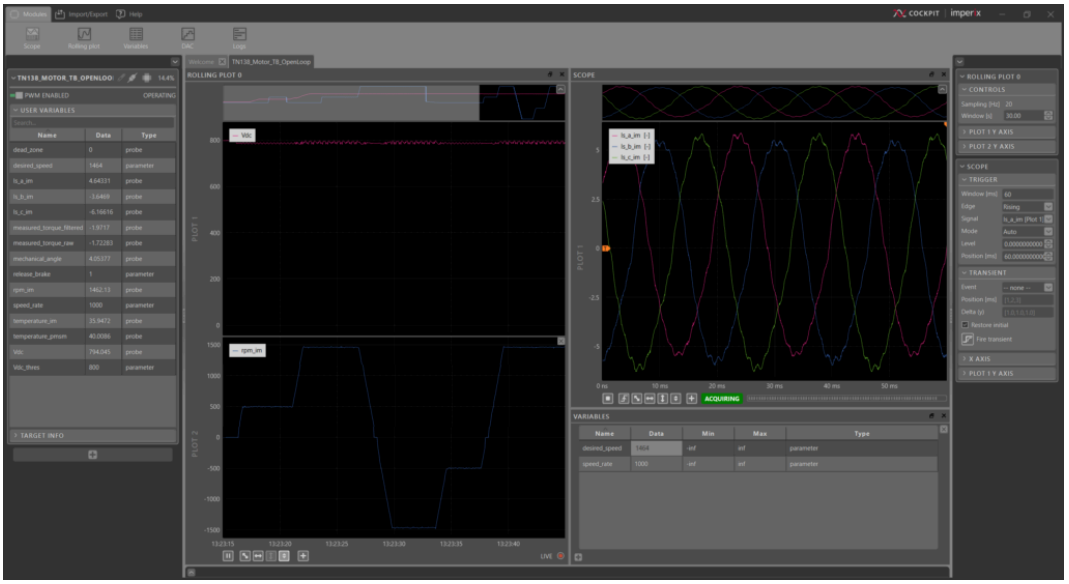
Loading the code and setting up the workspace in Cockpit

1. Open the Simulink model and set the mode to *Automated Code Generation* in the *CONFIG* block.
2. Build the model (Ctrl + B). It will automatically launch *Cockpit*.
3. Set the target IP in Cockpit and click on *Create* to generate a new project.
4. Add a new **rolling plot** module and drag-&-drop the *Vdc* variable to monitor the DC bus voltage.
5. Add a subplot to the rolling plot, and drag and drop the *rpm_im* variable. Using a rolling plot rather than a scope allows for monitoring the speed over a long period of time.
6. Add a new **scope** module and drag-&-drop the *Is_a_im*, *Is_b_im*, and *Is_c_im* variables. The scope can display each and every sample made available to the control.
7. Add a new **variables** module and drag-&-drop the *desired_speed* and *speed_rate* variables. This way, the main parameters of the speed control are easy to find.

Step-by-step test procedure

1. Turn on the laboratory DC source and gradually increase the DC bus voltage from 0 to 300V. Check with Cockpit that *Vdc* matches the voltage of the source.
2. Check that the *release_brake* variable is set to '1' (= brake released). The brake LED should turn orange on the motor interface when the brake is released.
3. Check that *desired_speed* = 500 rpm and *speed_rate* = 1000 rpm/s.
4. Enable the PWM outputs from Cockpit and observe the rotation of the shafts. The use of flexible couplings allows a small misalignment of the shafts. As a result, the torque sensor has some backlash by design and can vibrate a little bit.
5. Compare the speed measurement *rpm_im* with the speed reference *desired_speed*. The actual speed should be close to the reference due to the absence of load.
6. Disable the PWM outputs and increase the DC bus voltage to 800V.
7. Enable the PWM outputs again and gradually increase *desired_speed* to 1464 rpm (nominal speed). Again, compare *rpm_im* with *desired_speed*.
8. Try rotating the machine in the reverse direction, by setting *desired_speed* down to -1464 rpm. Check that the speed measurement also works for negative speeds.
9. At the end of the experiment, disable the PWM outputs first, and then reduce the output of the DC source to 0 V. Double-check in Cockpit that the DC bus is fully discharged.

The screenshot below shows how the workspace could look in the end, while running the test procedure.



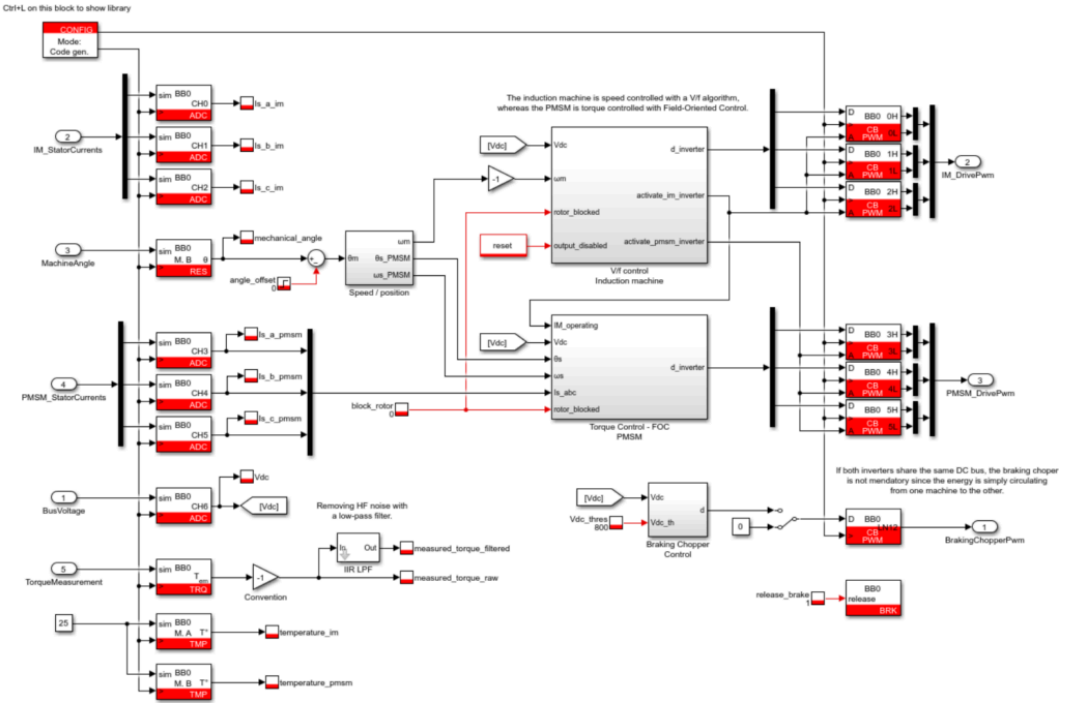
Running the no-load test using Cockpit

Test 2: load test

After validating the correct functioning of the resolver, the position measurement can be used by the control. For example, the Simulink model below implements a Field-Oriented Control (FOC) of the PMSM (see TN111 for more details). The goal is to load the IM and operate the motor testbench under nominal conditions. Additionally, the V/f control has been upgraded with a closed-loop speed controller for improved disturbance rejection. Operating the machines under load also gives the opportunity to check the torque and temperature sensors.

The code example uses Simscape Electrical to model the motor testbench. Matlab R2023a (or later) is required to run the model in simulation mode.

TN138_Motor_TB_load_testDownload



Overview of the Simulink control file for the closed-loop V/f under load test

Loading the code and setting up the workspace in Cockpit

1. Open the Simulink model and set the mode to *Automated Code Generation* in the *CONFIG* block.
2. Build the model (Ctrl + B). This will automatically launch *Cockpit*.
3. Set the target IP in *Cockpit* and click on *Create* to generate a new project.
4. Add a new **rolling plot** module and drag-&-drop the *Vdc* variable to monitor the DC bus voltage.
5. Add a subplot to the rolling plot, and drag and drop the *rpm_im* and *rpm_ref_im* variables.
6. Add another subplot to the rolling plot, and drag and drop the *Tem_pmsm*, *Tem_ref_pmsm*, and *measured_torque_raw* variables.
7. Add another subplot to the rolling plot, and drag and drop *temperature_im* and *temperature_pmsm*.
8. Add a new **scope** module and drag-&-drop the *Is_a_im*, *Is_b_im*, and *Is_c_im* variables.
9. Add a subplot to the scope and drag-&-drop the *Is_a_pmsm*, *Is_b_*pmsm, and *Is_c_*pmsm variables.
10. Add a new **variables** module and drag-&-drop the following variables: *desired_speed*, *speed_rate, desired_torque*, *torque_rate, block_rotor*, *mechanical_angle*, and *angle_offset*.

Torque and temperature measurements

The code example uses the theoretical sensitivity and offset of the torque sensor (53.3 mV/Nm and 5V, respectively). For optimal results, the sensor should be calibrated. Additionally, the backlash given to the sensor by the flexible couplings results in little

vibrations that appear in the torque measurement as small oscillations at the mechanical frequency of the rotor. A low-pass filter might prove helpful to filter out the vibrations. To this end, the code example includes the raw value *measured_torque_raw* and its filtered version *measured_torque_filtered*.

In addition to the torque, the temperature of the stator windings is available through the *temperature_im* and *temperature_pmsm* probes.

Calibration of the PMSM's rotor position

The built-in resolver of the PMSM measures the absolute position of the rotor. However, there is no guarantee that the sensor is aligned with the poles of the machine. Calibrating the rotor angle is essential to orient the field with a FOC control.

The Simulink model includes a calibration procedure that applies a DC current in phase A of the machine to force the alignment of the rotor and stator fluxes. The offset on the angle measurement can then be compensated.

1. Turn on the laboratory DC source and gradually increase the DC bus voltage from 0 to **50V**. Check with Cockpit that *Vdc* matches the voltage of the source.
2. Check that the *release_brake* variable is set to '1' (= brake released). The brake LED should turn orange on the motor interface when the brake is released.
3. Set *block_rotor* to '1' and enable the PWM outputs. The control code will apply a DC current in phase A of the PMSM. As a result, the rotor aligns itself on phase A.
4. At that moment, the value of *mechanical_angle* corresponds to the offset between the position measurement and the actual location of $\theta_r = 0\,\mathrm{rad}$, the absolute rotor angle.
5. Set the **initial value** of *angle_offset* equal to *mechanical_angl*e directly in the Simulink model. Disable the PWM outputs and regenerate the code (Ctrl + B). After this manipulation, the control code keeps compensating the offset permanently.

Since the PMSM has four pairs of poles, four mechanical positions correspond to $\theta_r = 0\,\mathrm{rad}$. Any of these positions can be used as the angle offset. Additionally, the angle offset is a constant value. Thus, the calibration procedure needs to be performed only once.

Step-by-step test procedure

1. Turn on the laboratory DC source and gradually increase the DC bus voltage from 0 to 300V. Check with Cockpit that *Vdc* matches the voltage of the source.
2. Check that the *release_brake* variable is set to '1' (= brake released). The brake LED should turn orange on the motor interface when the brake is released.
3. Check that *desired_speed* = 500 rpm and *speed_rate* = 1000 rpm/s.
4. Enable the PWM outputs from Cockpit and check that *rpm_im* and *Tem_pmsm* follow their respective references.
5. Check the reading *measured_torque_raw* from the torque sensor. It should be non-zero even after calibration since the weight of the shaft elements and the friction

at the bearings' level act as a passive mechanical load.

6. Set *desired_torque* to 5 Nm and verify that *Tem_pmsm* follows the new reference. Check that *measured_torque_raw* increases from 5 Nm.
7. Disable the PWM outputs and increase the DC bus voltage to 800V. Set *desired_torque* back to 0 Nm.
8. Enable the PWM outputs again and gradually increase the *desired_speed to* 1464 rpm.
9. Gradually increase *desired_torque* to 22.8 Nm. Check that *measured_torque_raw* increases accordingly. At this point, the motor testbench is operating at its nominal power of 3.5 kW.
10. Memorize the values of *temperature_im* and *temperature_pmsm*. Then, let the motor testbench run at nominal power for 5 to 10 minutes. You should observe the temperature of the machines increasing from several degrees.
11. At the end of the experiment, disable the PWM outputs first, and then reduce the output of the DC source to 0 V. Double-check in Cockpit that the DC bus is fully discharged.

The screenshot below shows how the workspace could look in the end, while running the test procedure.

Running the load test using Cockpit

To go further…

… with the motor testbench

The knowledge base presents standard control techniques for both the IM and the PMSM, such as:

- V/f Control of an induction machine (with turn-key code example)

- Rotor Field-Oriented Control of an induction machine (with turn-key code example)
- Field-Oriented Control of a PMSM (with turn-key code example)
- Direct Torque Control of a PMSM

Testing various control techniques is not the only purpose of the motor testbench. It can also serve as a reduced-scale prototype, functionally equivalent to a real-life system. The following articles present two examples of such applications:

- Electric car motor control
- Wind turbine generator control

… with the motor interface

The motor interface supports different position sensors, allowing the option to replace the motor testbench with another set of machines. The functionalities of the motor interface are available through the ACG and CPP SDKs. Since they are part of the imperix library, they do not require any additional installation procedure.

When using the Simulink or PLECS blocksets, the blocks specific to the Motor Interface are used like any other blocks. They mostly hide the hardware layer and provide only information relevant to the control, such as decoded angle of a machine, temperature measurements in °C, etc. More details can be found in the software documentation:

- INC – Incremental encoder
- RES – Resolver
- HAL – Hall sensor
- S/C – Sin/cos encoder
- TMP – Temperature sensor
- TRQ – Torque sensor
- BRK – Brake