Freescale Semiconductor
Application Note

Document Number: AN4912
Rev. 0, 03/2014

Tuning 3-Phase PMSM Sensorless Control Application Using MCAT Tool

Devices Supported: Kinetis KV, DSC

Contents

1 Introduction

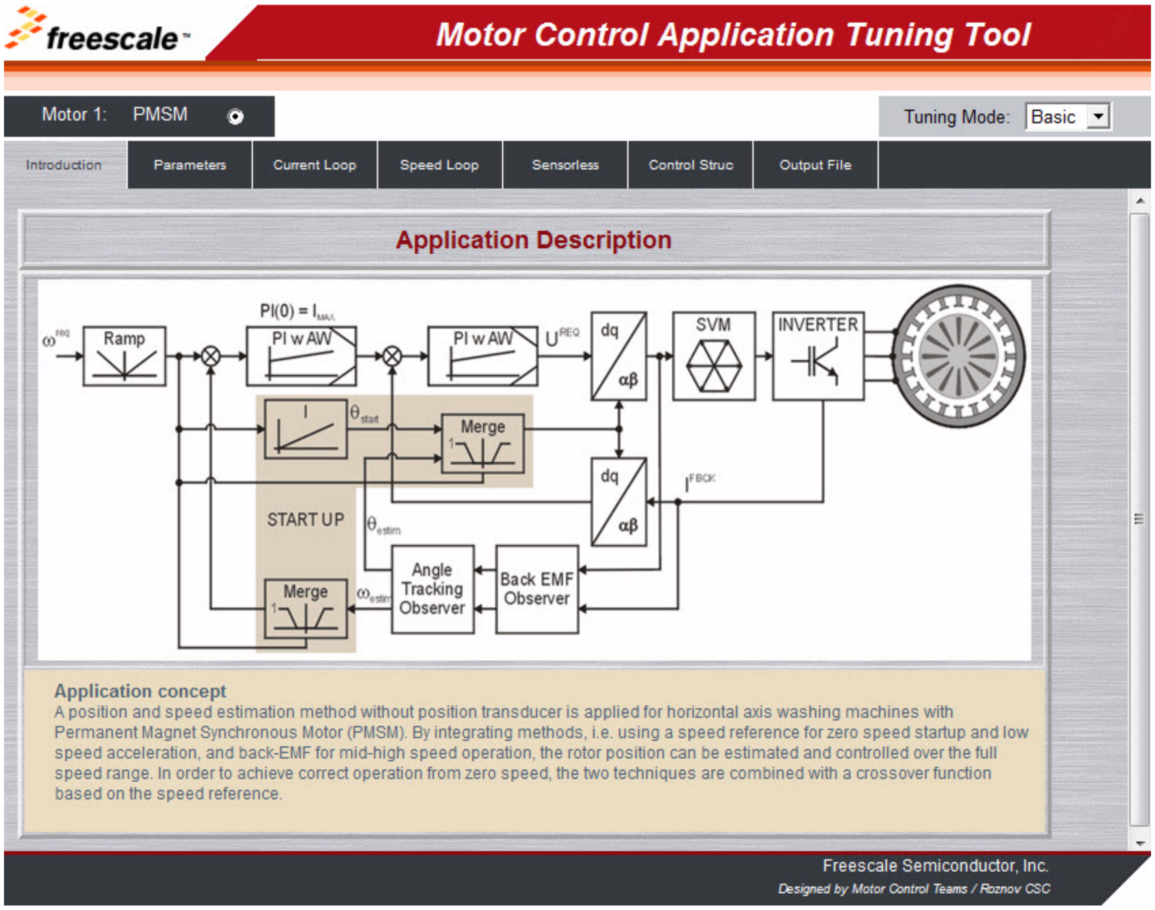
This document provides a guide on how to set-up, tune, and run a Permanent Magnet Synchronous Motor (PMSM) using the Freescale PMSM sensorless reference design. The reference design uses the Motor Control Tuning (MCAT) tool to simplify and speed-up the process of customizing the sensorless control application to run using a different motor or power stage.

The following requirements must be accomplished to run a new motor properly:

1. The reference design HW configuration works correctly and is set up according to the dedicated application note. See Section 6, "References."
2. The motor attached to the demo runs properly over a defined speed range

2 MCAT tool for PM synchronous motors

The MCAT tool is a graphical interface application with a friendly environment and intuitive control. As shown in Figure 1, the tool consists of a motor selector bar, a tab menu, and the workspace. The MCAT tool represents a modular concept consisting of several sub-modules. Each sub-module represents one tab in the tab menu. The arrangement of the sub-modules is flexible according to the needs of the embedded application. Prior to modification and set-up of the MCAT tool parameters, read AN4642 which describes the PMSM constant calculation and settings in more detail.



Application concept
A position and speed estimation method without position transducer is applied for horizontal axis washing machines with Permanent Magnet Synchronous Motor (PMSM). By integrating methods, i.e. using a speed reference for zero speed startup and low speed acceleration, and back-EMF for mid-high speed operation, the rotor position can be estimated and controlled over the full speed range. In order to achieve correct operation from zero speed, the two techniques are combined with a crossover function based on the speed reference.

Figure 1. MCAT Tool Screen

The predefined MCAT tool is a part of the reference software for a particular microcontroller. Because the tuning tool cannot be used as a standalone tool, it is included in the FreeMASTER project by default.

The MCAT tool runs under the FreeMASTER online monitor, which enables real-time tuning of the motor control application. With respect to the parameters of the controlled drive, the correct values of the control structure parameters are calculated and can be directly updated in the target application or stored in an application static configuration file. The electrical subsystems are modeled using physical laws and the parameters estimation algorithms are based on the Pole-placement method.

The tool enables output header file generation with the calculated constants required for the control algorithms, and also enables online update of those selected application control variables to be tuned for example control loop, speed ramp and so on. The variables are updated by clicking the *Update Target* button on each control tab.

3 Application Parameters

To ensure proper MCAT functionality and accuracy of the control parameter calculations, the PM synchronous motor parameters must be entered into the MCAT tool Parameter tab.

Input parameters are divided into five groups:

- Motor parameters—a mandatory set of parameters used for constant calculation of the PI controllers, observers, limits, and so on, in the required arithmetic type (Fractional 16- or 32-bit, Float)
- Hardware scales—the maximum measurable board voltage and current analog quantities
- Fault limits—a set of limits and thresholds used for fault detection
- Application scales—the maximum values used for variable scaling in fractional arithmetic type
- Alignment—initial rotor position alignment parameters

4 How to run a new motor

This chapter provides a guide on how to run your motor in several steps. It is highly recommended to go through all the steps carefully to eliminate any possible issues during the tuning process. The state diagram shown in Figure 2 depicts a typical PMSM sensorless control tuning process. Each tuning phase is described in the following chapters in more detail.

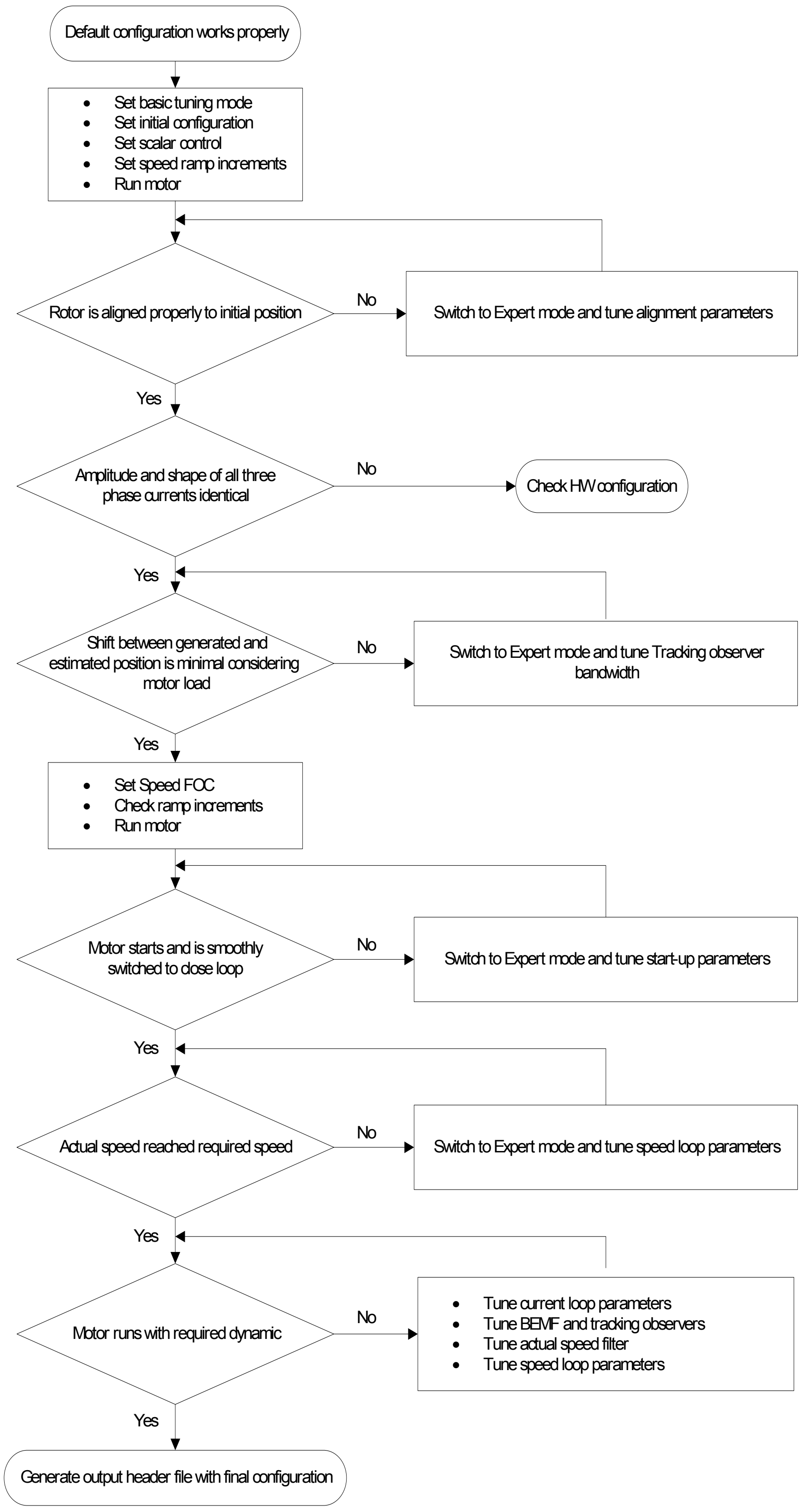

Default configuration works properly
Set basic tuning mode
Set initial configuration
Set scalar control
Set speed ramp increments
Run motor
Rotor is aligned properly to initial position
No
Switch to Expert mode and tune alignment parameters
Yes
Amplitude and shape of all three phase currents identical
No
Check HW configuration
Yes
Shift between generated and estimated position is minimal considering motor load
No
Switch to Expert mode and tune Tracking observer bandwidth
Yes
Set Speed FOC
Check ramp increments
Run motor
Motor starts and is smoothly switched to close loop
No
Switch to Expert mode and tune start-up parameters
Yes
Actual speed reached required speed
No
Switch to Expert mode and tune speed loop parameters
Yes
Motor runs with required dynamic
No
Tune current loop parameters
Tune BEMF and tracking observers
Tune actual speed filter
Tune speed loop parameters
Yes
Generate output header file with final configuration

Figure 2. Motor Tuning State Diagram

4.1 Initial configuration setting and update

1. Open the PMSM sensorless control application FreeMASTER project containing the dedicated MCAT plug-in module.
2. Select Basic mode—recommended for users who are not experienced in motor control theory. The number of required input parameters is reduced.
3. Select the Parameters tab.
4. Set the correct motor parameters (see Table 1)—The correct parameters can be obtained from the motor data sheet or by using the PMSM parameters measurement procedure described in AN4680. All parameters provided in Table 1 are accessible in both Basic and Expert modes. Motor inertia *J* expresses the overall system inertia that is very often difficult to obtain. Additional methods to identify drive inertia can be found other than in Freescale publications, for example, through IEEE. The *J* parameter is used to calculate the speed controller constant, however, manual controller tuning can also be used to calculate this constant.

Table 1. Motor Parameters

Parameter Name	Units	Description	Typical Range
pp	[-]	Motor pole-pair number	1–10
Rs	[Ohm]	One phase stator resistance	0.5–50
Ld	[H]	One phase direct inductance	0.00001–0.1
Lq	[H]	One phase quadrature inductance	0.00001–0.1
ke	[V.sec/rad]	Back-emf constant	0.00001–0.1
J	[kg.m2]	System inertia	0.000001–1
Iph nom	[A]	Motor nominal phase current	0.5–8
Uph Nominal	[V]	Motor nominal phase voltage	10–300
N nom	[rpm]	Motor nominal speed	1000–20,000

5. Set the hardware scales—modification of these two fields is not required when a reference to the Freescale Power Stage Board is used. These scales express the maximum measurable current and voltage analog quantities.
6. Check the fault limits—these fields are not accessible in Basic mode and are calculated using the motor parameters and hardware scales, see Table 2.

Table 2. Fault Limits

Parameter Name	Units	Description	Typical Range
U DCB trip	[V]	Voltage value of the external braking resistor when the switch is turned on	U DCB Over–U DCB max
U DCB under	[V]	Trigger value when the under voltage fault is detected	0–U DCB Over
U DCB over	[V]	Trigger value when the over voltage fault is detected	U DCB Under–U max
N over	[rpm]	Trigger value when the over speed fault is detected	N nom–N max
N min	[rpm]	Minimal actual speed value for sensorless control	(0.05–0.2) × N max

7. Check the application scales—these fields are not accessible in Basic mode and are calculated using the motor parameters and hardware scales.

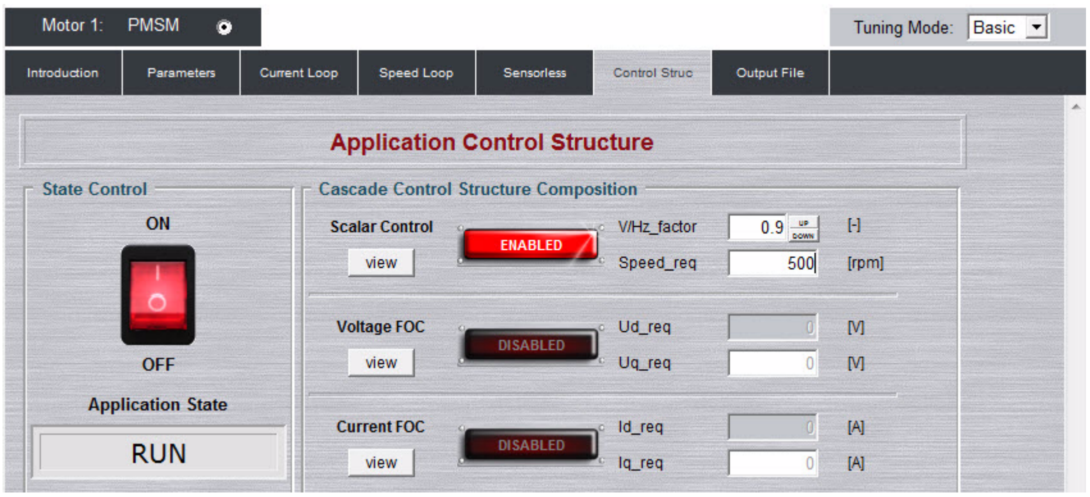
Table 3. Application Scales

Parameter Name	Units	Description	Typical Range
N max	[rpm]	Speed scale	> 1.1 × N nom
E max	[V]	Back- emf scale	ke × N max [rad/sec]
kt	[Nm/A]	Motor torque constant	—

7. Check the alignment parameters—these fields are not accessible in Basic mode and are calculated using the motor parameters and hardware scales. Parameters express the required voltage value applied to the motor during rotor alignment and its duration.
8. Click the *Store Data* button to save modified parameters in the inner file
9. Select the Output File tab where the header file preview is available containing a set of defines required for the target application.
10. Click *Generate Configuration File* to update the *PMSM_appconfig.h* file
11. Rebuild the application code and load the code into the target. This step is required only once to update the static motor constants. All other setting and updating can be done online using the FreeMASTER interface.

4.2 Control structure modes

1. If the target code is properly updated, select the Control Structure tab.
2. Select scalar control by clicking the DISABLED button in the *Scalar Control* section. The button color changes to red, and the text to ENABLED
3. Turn the application switch on. The application state should change to RUN.
4. Set the required speed value in the *Speed_req* field for example, 500 rpm in the Scalar control section. The motor should start running, see Figure 3.


Motor 1: PMSM

Tuning Mode: Basic

Application Control Structure

State Control

Application State: RUN

Cascade Control Structure Composition

Scalar Control	ENABLED	V/Hz_factor	0.9	[-]
		Speed_req	500	[rpm]
Voltage FOC	DISABLED	Ud_req	0	[V]
		Uq_req	0	[V]
Current FOC	DISABLED	Id_req	0	[A]
		Iq_req	0	[A]

Figure 3. Phase Currents

5. Select the *Phase Currents* recorder from FreeMASTER project tree Scalar & Voltage Control
6. Change the V/Hz factor directly or use the UP/DOWN buttons to identify the optimal ratio for the V/Hz profile. The shape of the motor currents should be close to the sinusoidal shape, see Figure 4.

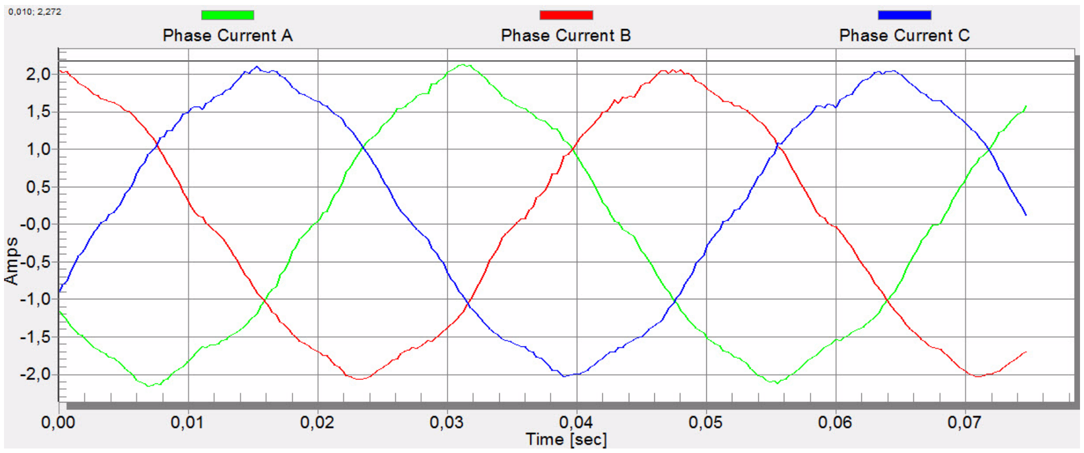


Figure 4. Phase Currents

7. Select the *Position* recorder to check the observer functionality. The difference between the Position Electrical Scalar and the Position Estimated should be minimal in order for the Back-emf position and speed observer to work properly (see Figure 5). The position difference depends on the motor load. The higher the load, the bigger the difference between positions because of the load angle.

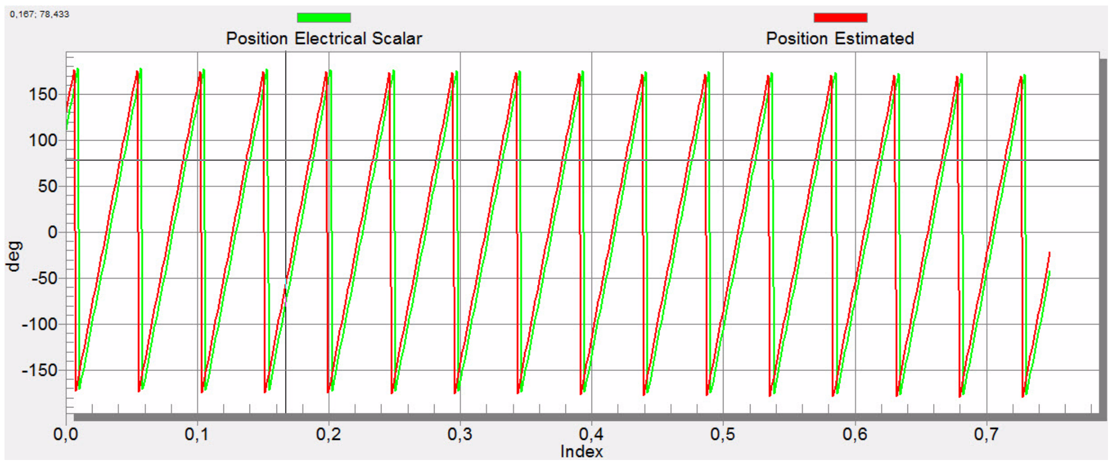


Figure 5. Generated and Estimated Positions

8. If an opposite speed direction is required, set a negative speed value of *Speed_req*.
9. At this step, proper observer functionality and measurement of analog quantities is expected.
10. Enable *Voltage FOC* mode by clicking the disabled button in the Voltage FOC section while the main application switch is turned off.
11. Switch the main application switch on and set a non-zero value of *Uq_req*. The FOC algorithm uses the estimated position to run the motor.

4.3 Alignment tuning

The alignment procedure sets the rotor to an accurate initial position and enables applying a full start-up torque to the motor. Rotor alignment parameters are available for editing in Expert mode. A correct initial position is required primarily for high start-up loads (compressors, washers, and soon). The aim of alignment is to have the rotor in a stable position without oscillations prior to start-up.

1. Alignment voltage is a value applied to the d-axis during alignment. Increase this value for a higher shaft load.
2. Alignment duration expresses the time the alignment routine is called. Tune this parameter to have the rotor without oscillations or movement at the end of the alignment process.

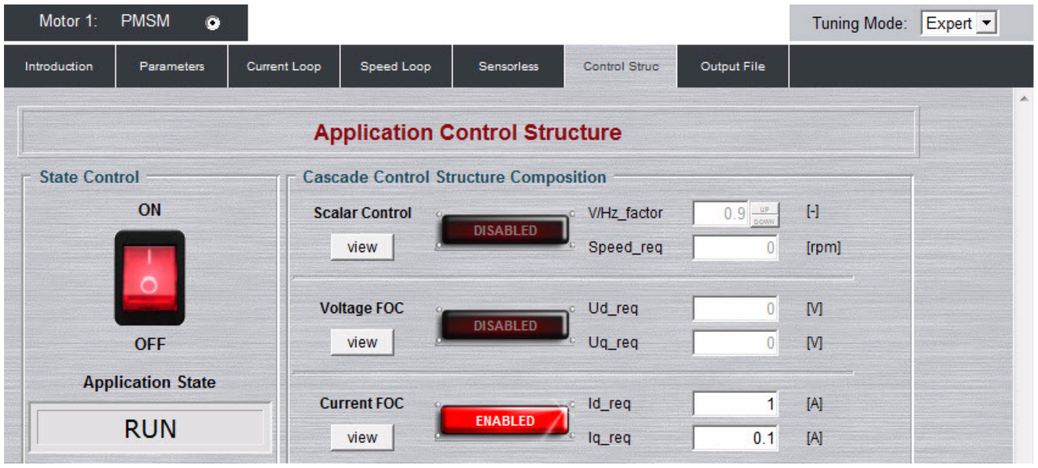
4.4 Current loop tuning

Parameters for the current D and Q PI controllers are fully calculated in Basic mode using the motor parameters and no action required in this mode.

If the calculated loop parameters do not correspond to the required response, the bandwidth and attenuation parameters can be tuned.

1. Switch the tuning mode to *expert*.
2. Set the required loop bandwidth and attenuation and click the *Update Target* button in the Current Loop tab. The tuning loop bandwidth parameter defines how fast the loop response is while the tuning loop attenuation parameter defines the actual quantity overshoot magnitude.
3. Select the *Current Controller Id* recorder.

4. Select the Control Structure tab, switch to Current FOC, set Iq_req to a very low value, for example 0.1, and set the required step to *Id_req*. The control loop response is shown in the recorder, see Figure 6.



Application State: RUN

V/Hz_factor 0.9 [-]
Speed_req 0 [rpm]
Ud_req 0 [V]
Uq_req 0 [V]
Id_req 1 [A]
Iq_req 0.1 [A]

Figure 6. Generated and Estimated Positions

5. Tune the loop bandwidth and attenuation until the required response is received. The following example waveforms show the correct and incorrect settings of the current loop parameters:
   — Loop bandwidth is low (100 Hz) and settling time of the Id current is long, see Figure 7.

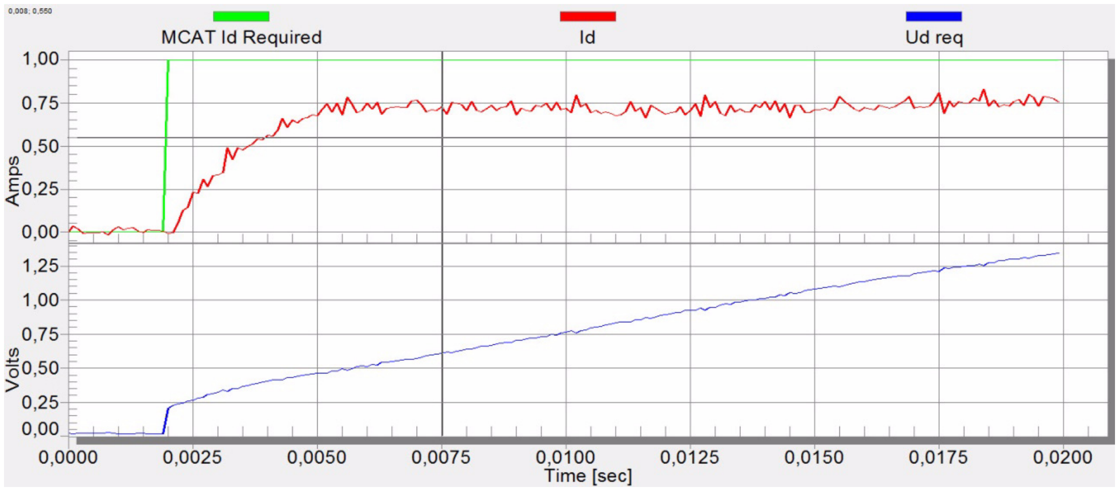


Figure 7. Slow Step Response of Id Current Controller

   — Loop bandwidth (300 Hz) is optimal and the response time of the Id current is sufficient, see Figure 8.

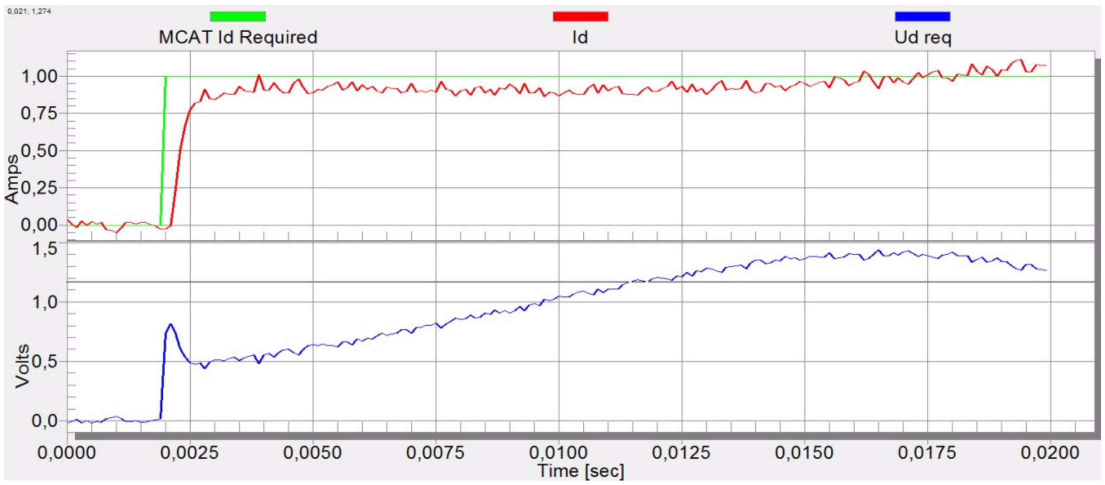


Figure 8. Optimal Step Response of Id Current Controller

— Loop bandwidth is high (500 Hz) and the response time of the Id current is very fast, but with oscillation and overshoot, see Figure 9.

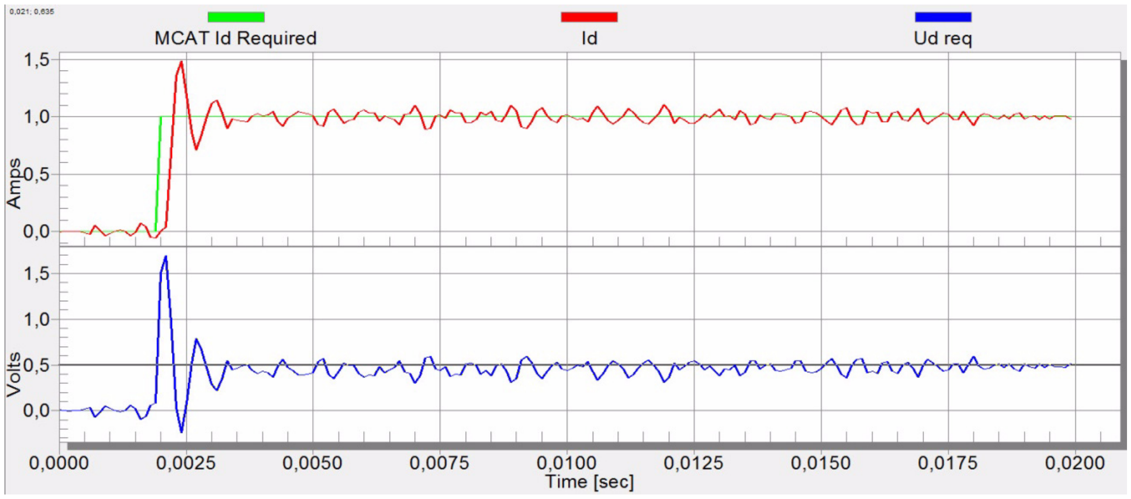


Figure 9. Fast Step Response of Id Current Controller

4.5 Actual speed filter

The estimated speed from the BEMF observer is fed into the speed PI controller through the IIR filter. The filter cut-off frequency can be modified in Expert mode within the Speed loop tab. The Speed loop sample time is typically several milliseconds, so the actual speed filter cut-off frequency is mostly in the range from 5 Hz to 100 Hz.

The filter output can be tracked in the *Speed* scope. The modified filter cut-off frequency value needs to be applied to the MCU by clicking the *Update Target* button.

4.6 Speed ramp tuning

The Speed command is applied to the speed controller through a speed ramp. The ramp function contains two increments, up and down, that express motor acceleration and deceleration per second. If increments are very high it can cause an over current fault during acceleration and an over-voltage fault during deceleration. It can be seen in the *Speed* scope whether the *Speed Actual Filtered* waveform shape equals the *Speed Ramp* profile.

Increments are common for Scalar and Speed control. Increment fields are located in the Speed Loop tab and are accessible in both tuning modes. Clicking the *Update Target* button applies the changes to the MCU. An example speed profile is shown in Figure 10. Ramp increment down is set to 500 rpm/sec, while increment up is set to 3,000 rpm/sec.

The start-up ramp increment is located in the Sensorless tab and its value is usually higher than for the speed loop ramp.

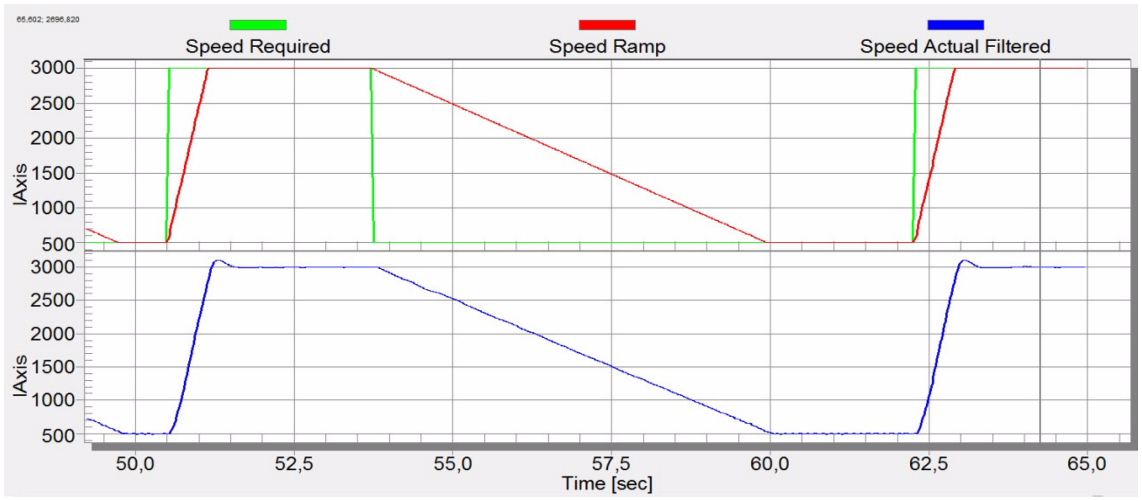


Figure 10. Speed Profile

4.7 Open-loop start-up

The start-up process can be tuned by a set of parameters located in the Sensorless tab. Two of these parameters, ramp increment and current, are accessible in both tuning modes. Start-up tuning can be processed in all control modes apart from the Scalar control. Setting optimal values achieves a proper motor start-up. An example start-up state of low dynamic drives (fans, pumps) is shown in Figure 11.

1. Select the Startup recorder from the FreeMASTER Project Tree.
2. Set the start-up ramp increment typically to a higher value than the speed loop ramp increment.
3. Set the start-up current according to the required start-up torque. For drives such as fans or pumps the start-up torque is not very high and can be set to 15% of nominal current.
4. Set the required merging speed—that is, the threshold when the open-loop and estimated position merging starts, mostly set in the range 5%–10% of nominal speed.

5. Set the merging coefficient—the position merging process duration, 100% corresponds to a half electrical revolution. The higher the value, the faster the merge is done. Values close to 1% are set for drives where high start-up torque and a smooth transition between open-loop and closed-loop are required.
6. Click the *Update Target* button to apply the changes to the MCU.
7. Switch to the Control Structure tab and enable Speed FOC.
8. Set the required speed higher than the merging speed.
9. Check the start-up response in the recorder.
10. Tune the start-up parameters until the optimal response.
11. If the rotor does not start running, increase the start-up current.
12. If the merging process fails (the rotor is stuck or stopped), decrease the start-up ramp increment, increase the merging speed and set the merging coefficient to 5%.

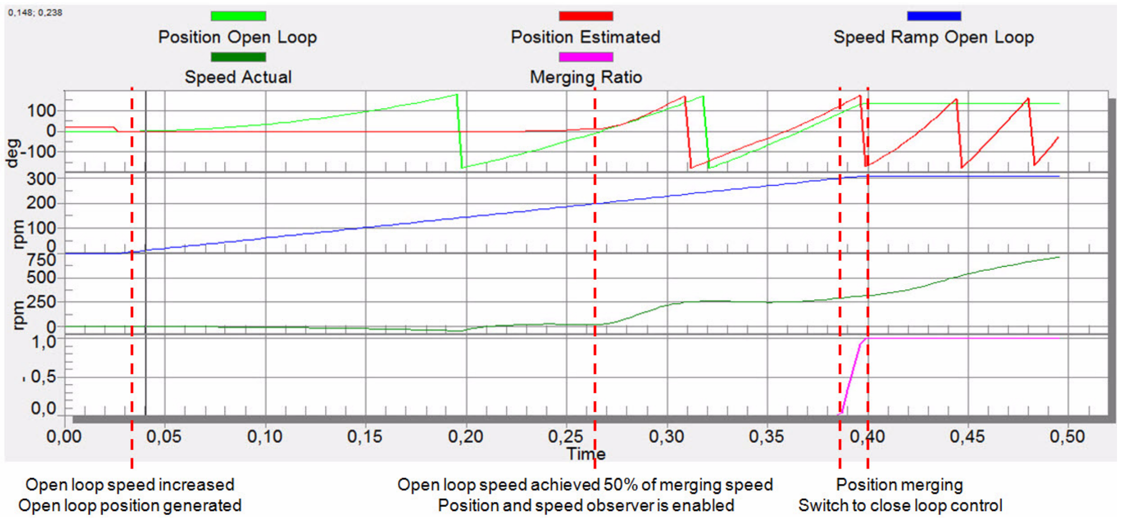


Figure 11. Motor Start-up

4.8 BEMF Observer Tuning

The parameters BEMF Observer and Tracking observer are fully calculated in Basic mode using the motor parameters and no action is required in this mode.

If the calculated loop parameters do not correspond to the optimal response, bandwidth and attenuation parameters can be tuned.

1. Switch tuning mode to *expert*.
2. Select the Observer recorder from the FreeMASTER Project Tree.
3. Set the required bandwidth and attenuation of the BEMF observer—the bandwidth is typically set to a value close to the Current Loop bandwidth.
4. Set the required bandwidth and attenuation of the Tracking observer—typically the bandwidth is set in the range 10–20 Hz for most low dynamic drives (fans, pumps).
5. Click the *Update Target* button to apply the changes to the MCU.

6. Check the observer response in the recorder.

4.9 Speed PI controller tuning

The motor speed control loop is a first order function with a mechanical time constant that depends on the motor inertia and friction. If these mechanical constants are available, the PI controller constants can be tuned using loop bandwidth and attenuation.

In practice, the values of the motor and load inertias and frictions are very often unknown and it is quite difficult to obtain them. Therefore, a manual tuning of the P and the I portions of the speed controllers is available to obtain the required speed response, see the example response in Figure 8.

There are available dozens of approaches to tuning the PI controller constants. The following steps provide one of these examples to set tune the speed PI controller for a PM synchronous motor.

1. Select the Speed Controller from the FreeMASTER Project Tree.
2. Select the Speed loop tab.
3. Check *Manual Constant Tuning*—that is, the bandwidth and attenuation fields are disabled and *SL_Kp* and *SL_Ki* are enabled.
4. Tune the proportional gain:
   a) Set the *SL_Ki* integral gain to zero.
   b) Set the speed ramp to 1000 rpm/sec or higher.
   c) Switch to the Control Structure tab and run the motor at a convenient speed, about 30% of nominal speed.
   d) Set a step in the required speed to 40% of N nom.
   e) Switch back to the Speed loop tab.
   f) Adjust the proportional gain *SL_Kp* until the system responds properly to the required value without oscillations or excessive overshoot:
      – If *SL_Kp* is set low, then the system respond will be slow.
      – If *SL_Kp* is set high, then the system respond will be tighter.
      – When *SL_Ki* = 0, then the system will probably not achieve the required speed.
   g) Click the *Update Target* button to apply the changes to the MCU.
5. Tune the integral gain:
   a) Increase slowly *SL_Ki* to minimize the difference between the required and actual speeds to zero.
   b) Adjust *SL_Ki* in such a way as to not see any oscillation or large overshoot of the actual speed value while the required speed step is applied.
   c) Click the *Update Target* button to apply the changes to the MCU.
6. Tune the loop bandwidth and attenuation until the required response is received. Example waveforms are shown in Figure 12 with the correct and incorrect settings of the current loop parameters:

   — *SL_Ki* value is low, and *Speed Actual Filtered* does not achieve *Speed Ramp*, see Figure 12.

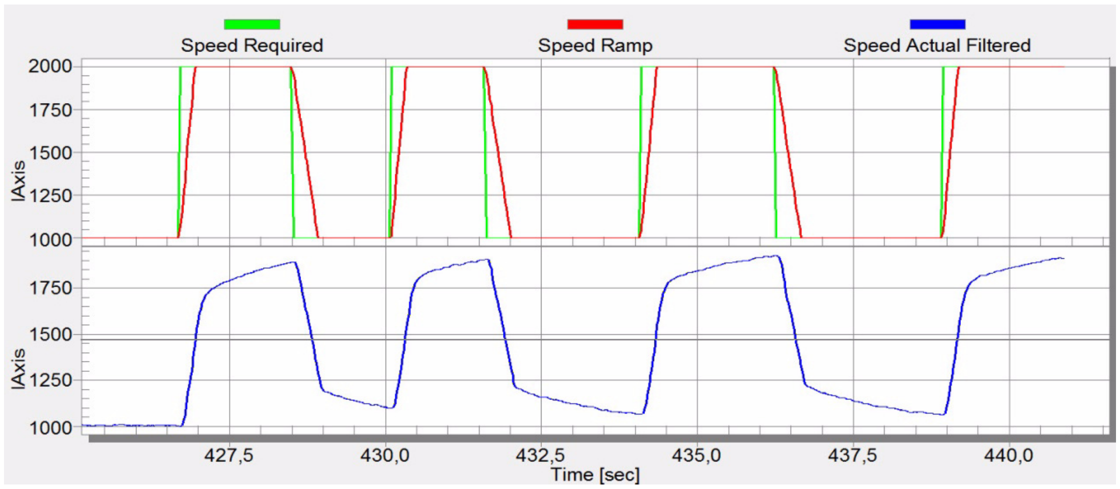


Figure 12. Speed Controller Response—*SL_Ki* value is low, *Speed Ramp* not achieved

— *SL_Kp* value is low, and *Speed Actual Filtered* greatly overshoots and the long settling time is unwanted, see Figure 13.

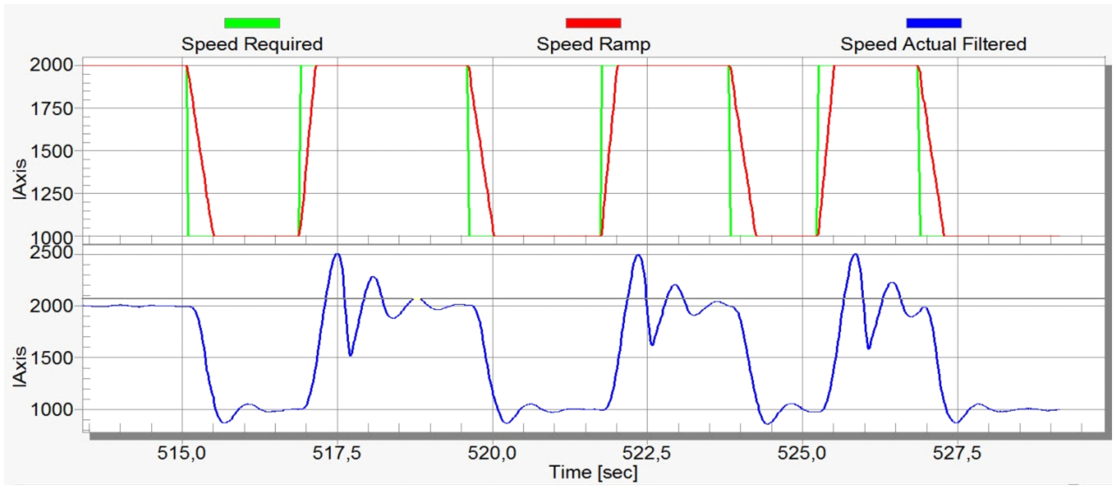


Figure 13. Speed Controller Response—*SL_Kp* value is low, *Speed Actual Filtered* greatly overshoots

— Speed loop response is with a small overshoot, and the *Speed Actual Filtered* settling time is sufficient. Such a response can be considered as an optimal, see Figure 14.

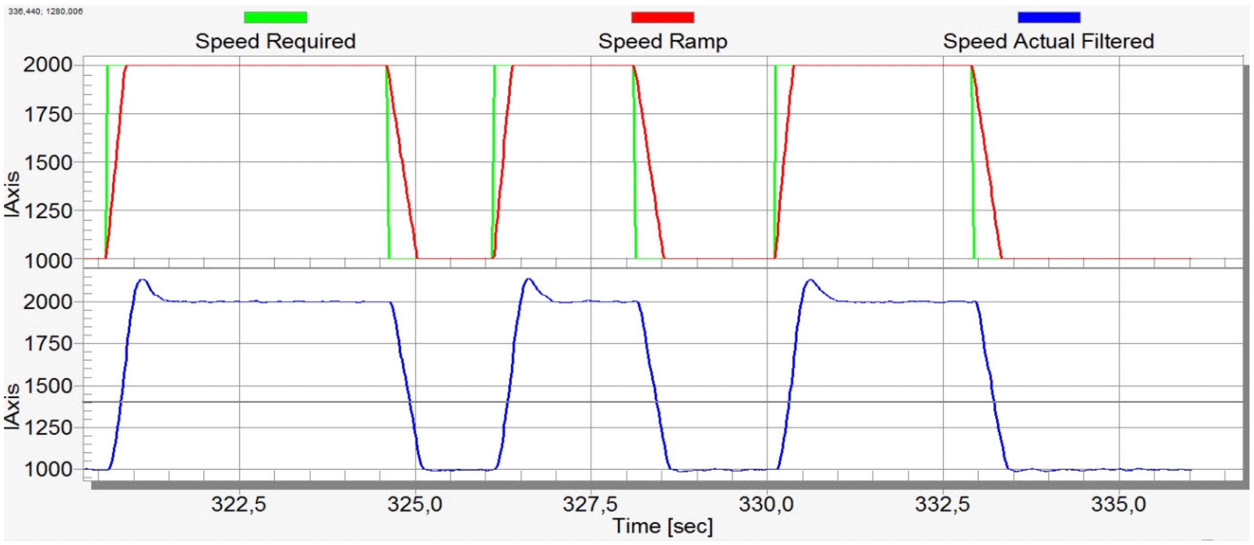


Figure 14. Speed Controller Response—Speed loop response with small overshoot

5 Conclusion

The tuning process of electric drives using the PMSM sensorless control application strongly depends on the motor and load type used. Low dynamic applications such as fans and pumps run mostly with a slow speed ramp and the control process does not require precise parameter estimation and constant controller setting. Such types of applications can be tuned in a reasonable time using the MCAT and FreeMASTER tools.

Applications such as servos, washer drives, and so on, require more expertise and experience to be set correctly according to the specified speed profiles and dynamics.

6 References

1. *Motor Control Application Tuning (MCAT) Tool for 3-Phase PMSM,* AN4642, by Freescale Semiconductor, Inc., 2013.
2. *PMSM Electrical Parameters Measurement,* AN4680, by Freescale Semiconductor, Inc., 2013.

For a current list of documentation, refer to www.freescale.com.

7 Revision history

Revision 0 is the initial release of this document.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4912
Rev. 0
03/2014