

Tuning Three-Phase BLDC Motor Sensorless Control Application Using the MKV10x

by: Petr Staszko, Pavel Sustek, Ivan Lovas

1 Introduction

This application note describes setup and tuning various three-phase Brushless DC (BLDC) motors using application described in [DRM144](#) and [AN4862](#). This application is based on Freescale's 32-bit Kinetis MKV10x device. Application software uses Motor Control Application Tuning Tool (MCAT) to easily tune various motors in very short time.

The concept of the application is a speed closed-loop motor control algorithm using Back-EMF voltage integration method for sensorless motor control.

Contents

1	Introduction	1
2	Basic information	2
3	Hardware	2
3.1	TWR-MC-LV3PH.....	2
3.2	TWR-KV10Z32	3
4	Software.....	4
4.1	Configure debugger and flashing software.....	4
4.2	Remote control using FreeMASTER .	4
5	MCAT tool for BLDC motors.....	6
5.1	Introduction tab	8
5.2	Parameters tab	8
5.3	Control Loop tab	10
5.4	Sensorless tab	12
5.5	Output File tab.....	13
5.6	Application Control tab.....	14
5.7	Procedure to run a new motor	14
6	References	20
7	Revision history.....	20

2 Basic information

This application requires BLDC motor with trapezoidal Back-EMF shape for proper operation. The following limitations must be considered before tuning the motor using this application and tower system:

- Maximal Voltage: 36 V
- Maximal motor current: 7 A
- Maximal motor power: 250 W
- Maximal motor speed: 40 000 rpm (1-pole pair motor; 20kHz PWM frequency)

Complementary bipolar switching PWM is used in the BLDC motor control application. With complementary bipolar switching, two phases are powered with the complementary PWM signals (Bottom MOSFET is switching in complement to the top MOSFET within a phase), one phase with the duty-cycle greater than 50%, second phase with complement duty-cycle value (less than 50%). This application is not suitable for low impedance motors because the PWM used can cause high current ripples for low impedance motors.

Use default motor, Linix 45ZWN24-40 (delivered with TWR-MC-LV3PH), and software setting to ensure that the hardware and software are working fine. For more details see AN4862: *Three-Phase BLDC Sensorless Control Using the MKV10x*, available on freescale.com.

Always use power supply with current limitation to protect motor and board.

3 Hardware

The following hardware modules are required to function this application properly:

- TWR-KV10Z32
- TWR-MC-LV3PH
- TWR-Elevator

Jumper settings of all the boards are described in following subsections.

3.1 TWR-MC-LV3PH

The three-phase Low-Voltage Motor Control board (TWR-MC-LV3PH) is a peripheral Tower System Module, interchangeable across the Tower development platform. Phase voltage and current feedback signals are provided; that allow a variety of algorithms to control 3-phase PMSM and BLDC motors. High level of board protection (overcurrent, undervoltage, overtemperature, and others) provided by MC33937 predriver.

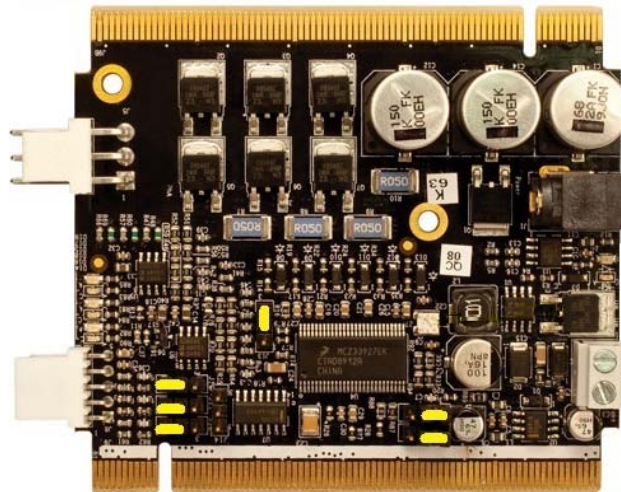


Figure 1. TWR-MC-LV3PH jumpers' settings

3.2 TWR-KV10Z32

The TWR-KV10Z32 microcontroller module is part of Freescale Tower System, a modular development platform that enables rapid prototyping and tool reuse through reconfigurable hardware.

Jumpers' settings for BLDC application is shown in [Figure 2](#):

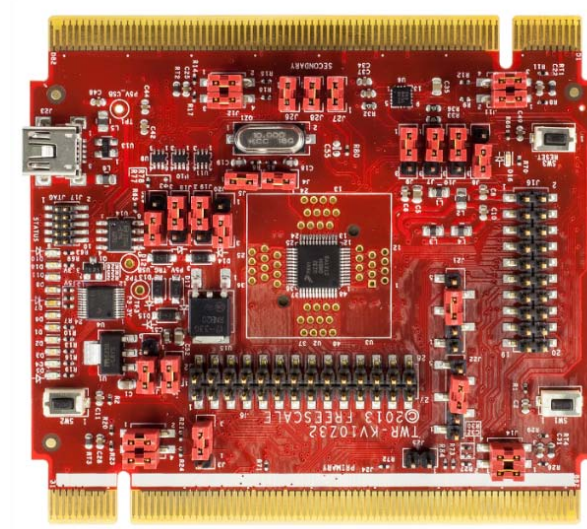


Figure 2. TWR-KV10Z32 jumper settings

4 Software

4.1 Configure debugger and flashing software

An onboard open SDA debugger can be used for debugging application and flashing software. To select OpenSDA debugger, proper setting in IAR options must be done. Go to project Options->Debugger->Setup and select “PE micro”. Then specify detailed setting of PE micro is shown in Figure 3. When the hardware is configured and the setup of debugger is completed, then the software can be downloaded into board by pressing Download button (in green color) in the IAR environment.

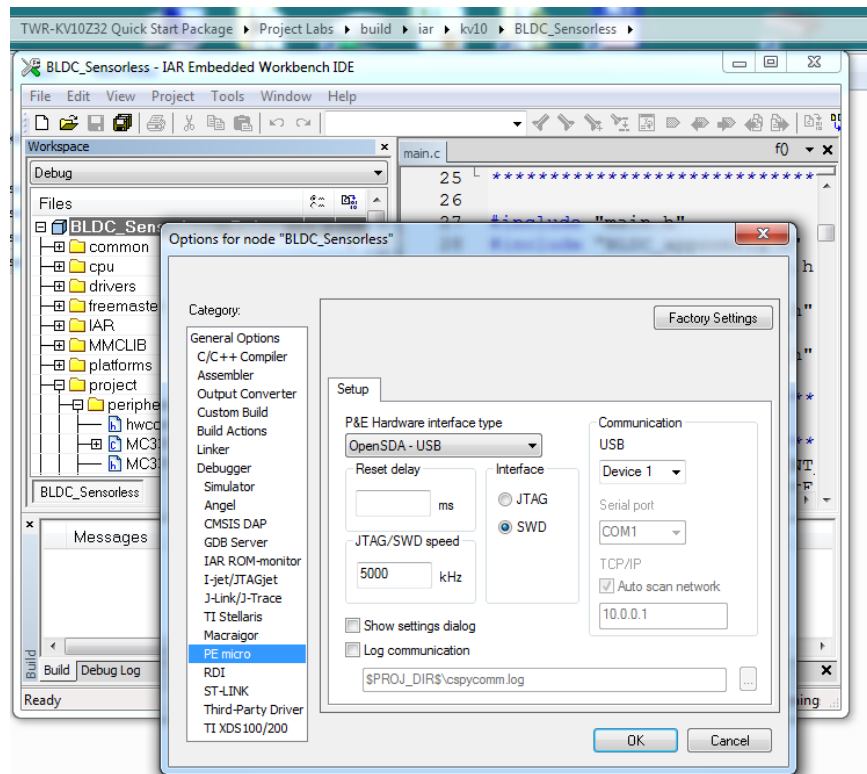


Figure 3. Debugger setting

4.2 Remote control using FreeMASTER

The MCAT tuning tool is based on the operation with FreeMASTER tool. The FreeMASTER tool is running on personal computer and connected to TWR-KV10Z32 via mini-USB port. Speed can be increased or decreased manually using the SW1 and SW2 switches (placed on the TWR-KV10Z32 board).

To establish communication with FreeMASTER the following steps are necessary:

1. Open FreeMASTER and go to Project->Options->Comm and set communication via to the Direct RS232.

2. Select the virtual COM port where the TWR-KV10Z32 board is connected - see System Properties->Device Manager->Ports and look for “OpenSDA – CDC Serial Port ([pemicro.com/opensda](http://www.pemicro.com/opensda)) (COMxx)”. Set communication speed to 9600 bps.
3. The next step is to toggle the communication button (Red STOP button). After that, in the bottom right-hand corner, there should be RS232;COMxx;9600, which means that communication has been established.
4. If not, toggle (STOP) the communication and unplug/plug the FreeMASTER USB cable. Then toggle (START) the communication button.

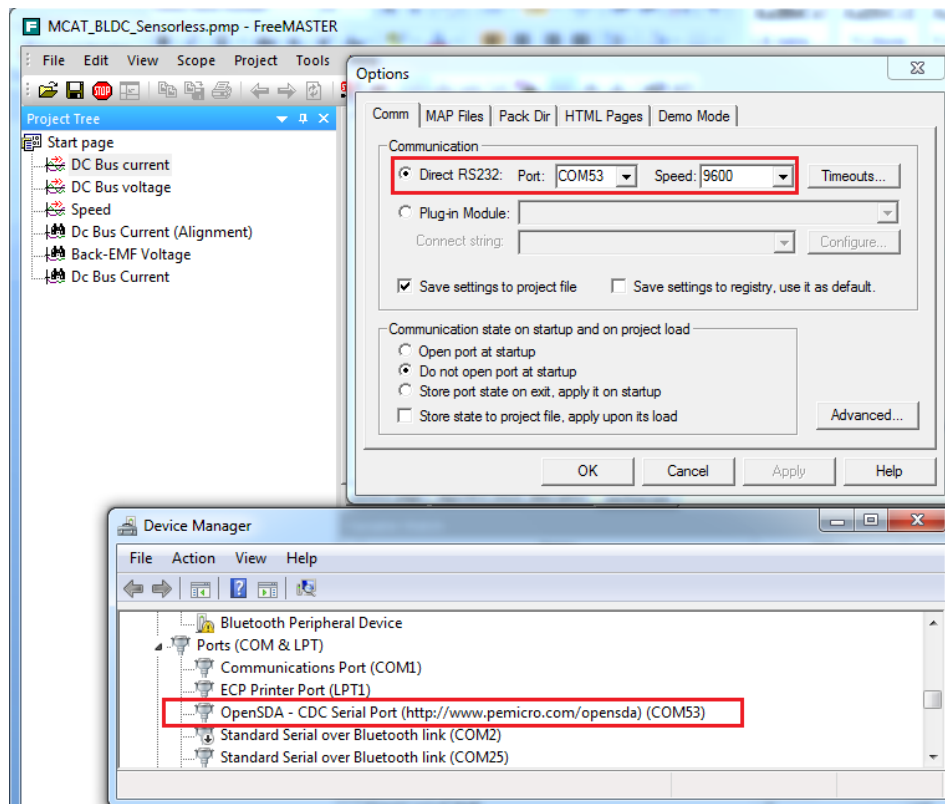


Figure 4. FreeMASTER settings

After launching the application and performing all necessary settings, click the “App Control” page in the main window (Motor Control Application Tuning Tool), as shown in [Figure 5](#). In this view, variables used for the application state, speed, PI controller, and ramp settings are visible at the bottom window; most important variables and settings are displayed using the graphics representation. Application can be switched on or off using “App switch” button or selecting state of “Application Switch” variable in the variable list window. You can enter required speed (in range -5000 to 5000) or click on the speed tachymeter to set the required speed. DC Bus current limiting value can be also set by clicking on the Ampere meter. In case any fault is detected it has to be cleared manually by entering zero value to “Fault” variable; then the application can be switched on again.

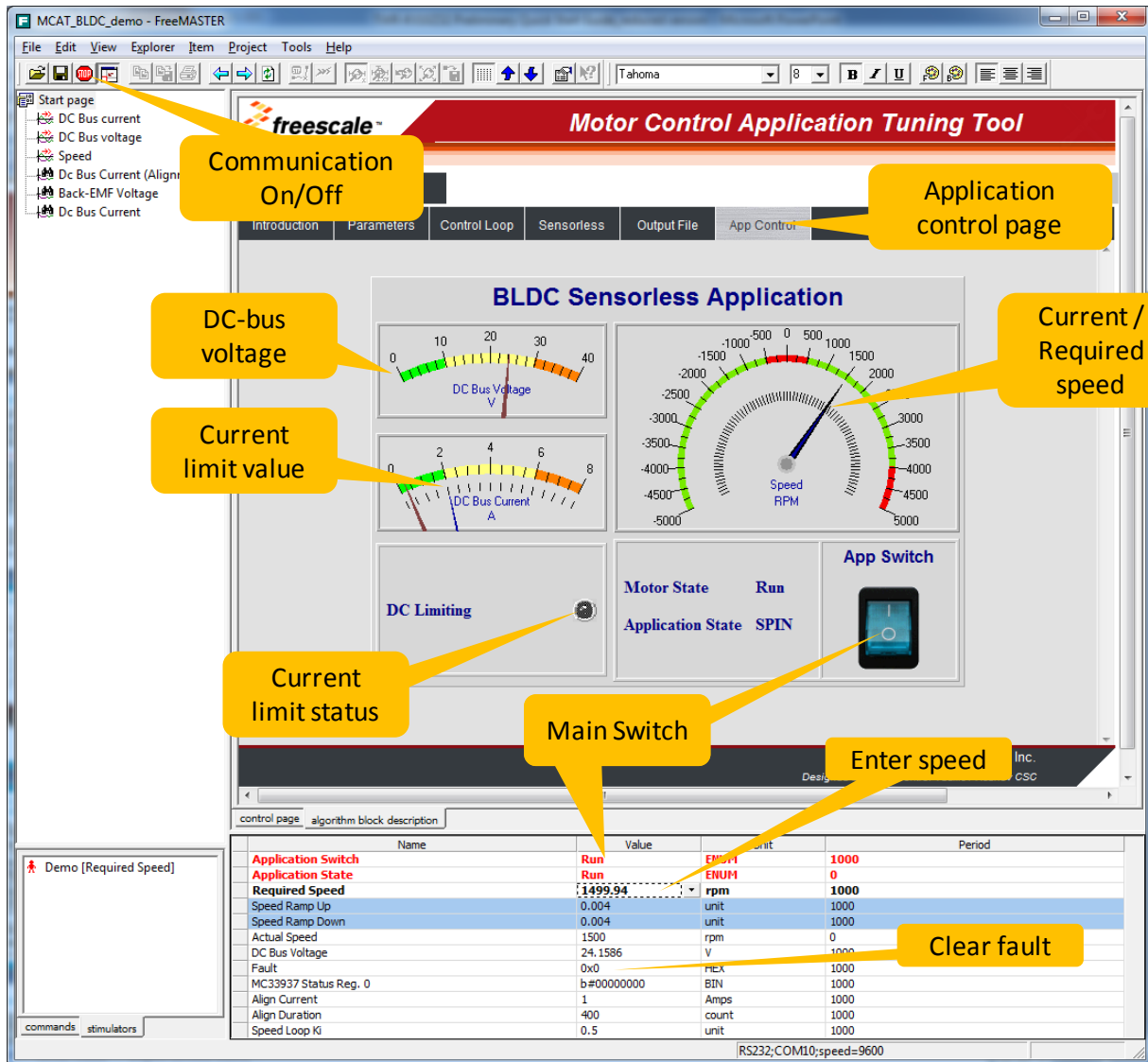


Figure 5. FreeMASTER project page

5 MCAT tool for BLDC motors

The MCAT tool is a graphical tool with friendly environment and intuitive control. As shown in Figure 6, the tool consists of motor selector bar, tab menu, and workspace. The MCAT tool represents a modular concept that consists of several sub-modules. Each sub-module represents one tab in the tab menu. The arrangement of the submodules is flexible according to the needs of embedded application.

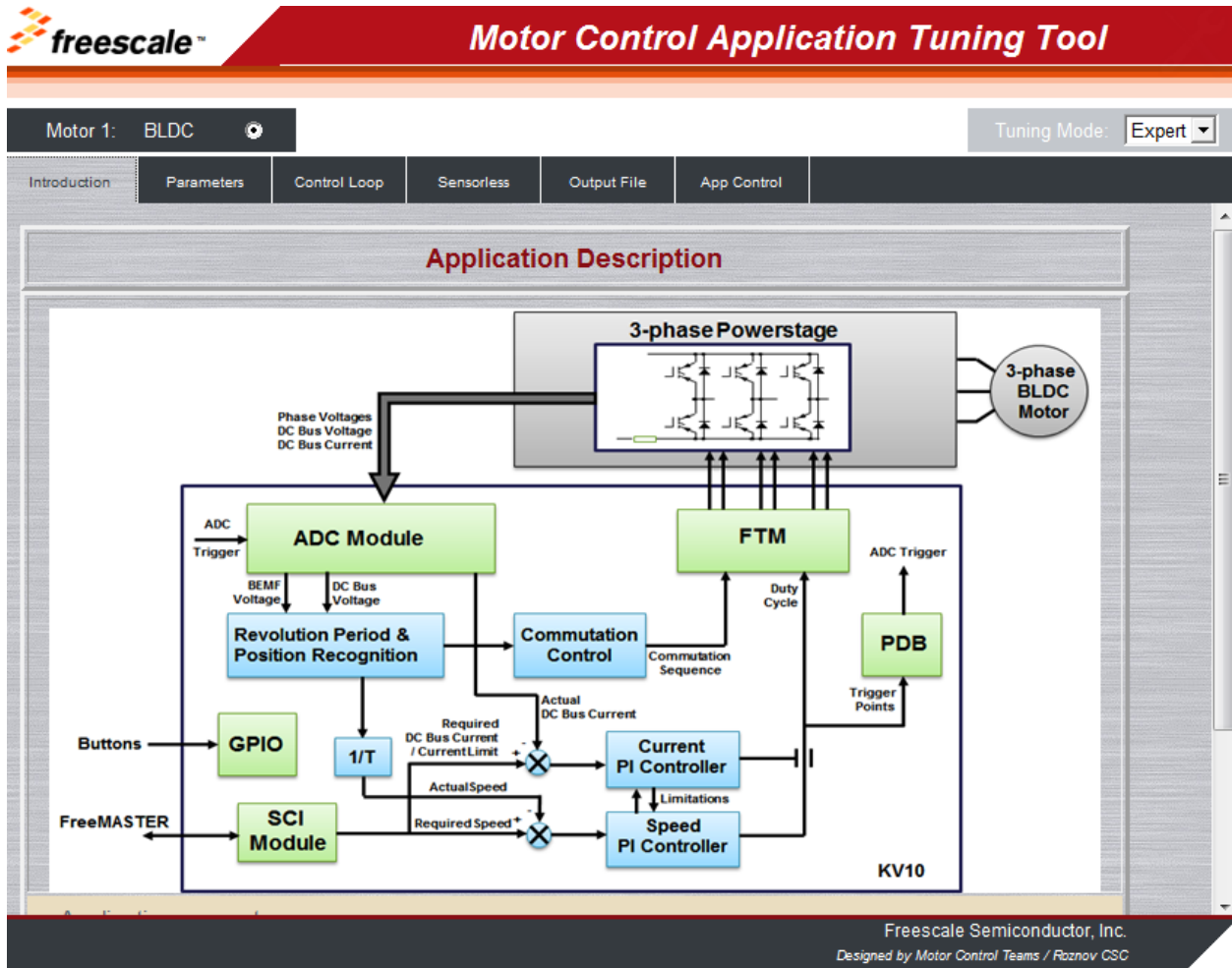


Figure 6. MCAT project page

Predefined MCAT tool is a part of reference software for dedicated MCU. Since the tuning tool cannot be used as a standalone, it is included in the FreeMASTER project by default.

The tool support output header file generation with calculated constants required for control algorithms and also enables on-line update of selected application control variables to tune, for example control loop, speed ramp, and so forth. The variables are updated clicking “Update target” button on each control tab.

Set motor parameters can be stored in internal MCAT file clicking “Store Data” button or the data can be reloaded clicking “Reload Data” button.

Each parameter and constant contains a short hint that can be activated on parameter name mouse focus; see [Figure 7](#) for example of hint information.

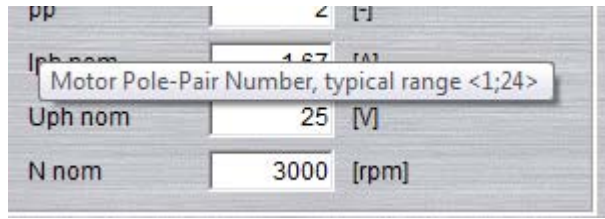


Figure 7. Parameter hint information

The MCAT tool workspace is unique for each tab and detailed overview of each available tab is provided in the subsequent sections.

5.1 Introduction tab

The Introduction tab can be considered as a voluntary tab. It provides a room for describing or introducing the targeted MC application, as shown in [Figure 6](#).

5.2 Parameters tab

The Parameters tab is dedicated for entering the input application parameters, as shown in [Figure 8](#). It is a mandatory tab due to its high-level dependency with other tabs. Please take care while filling an application parameter into a cell. These parameters are used across the MCAT calculations and improper value filled in the cells can cause unexpected behavior of application running on target. The impact of each required input is described in [Table 1](#). Number of input parameters need to be filled depends on the selected application tuning mode.

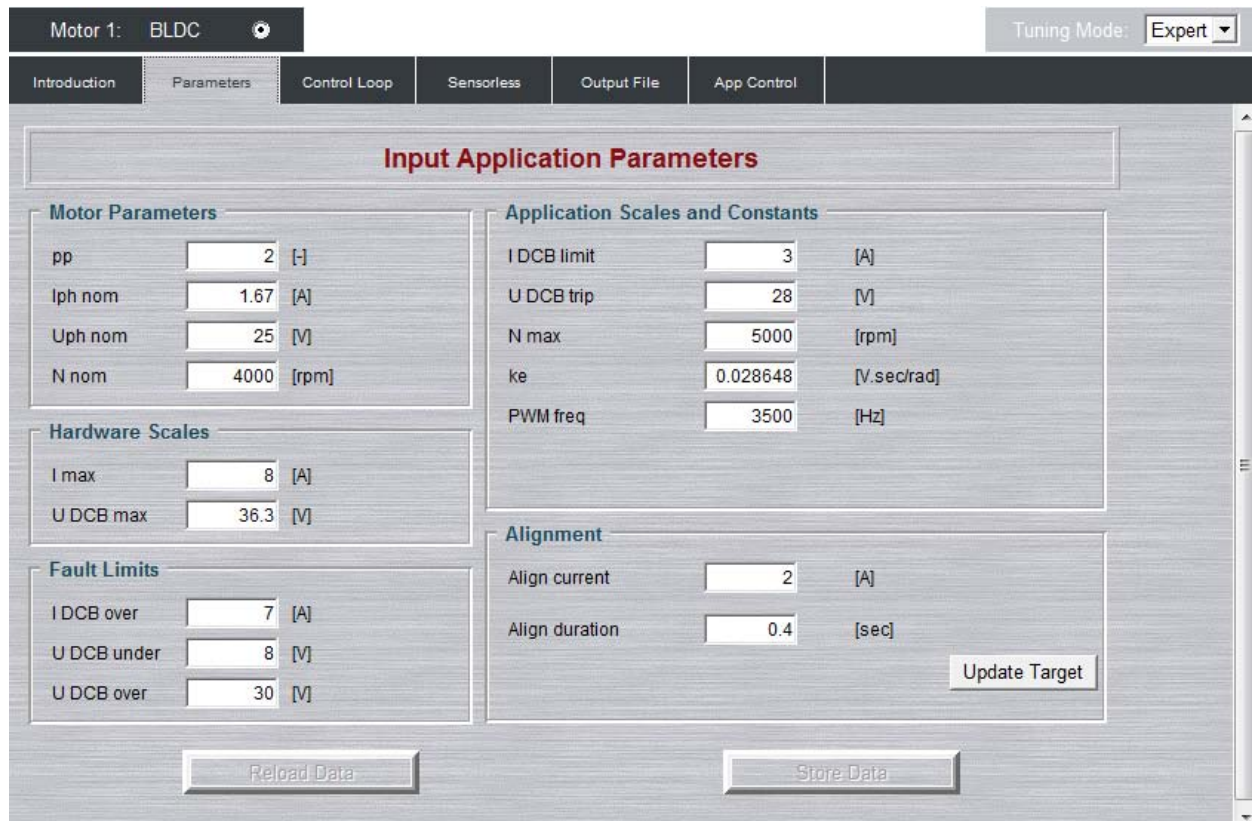


Figure 8. Parameters tab

The [Table 1](#) shows the list of MCAT tool input parameters with their physical units, brief description, typical range, and accessibility status in basic mode.

Table 1. Basic parameters

Parameter Name	Units	Description	Typical Range	Basic Mode Accessibility
pp	[-]	Motor pole-pair number	1 ~ 10	Yes
Iph nom	[A]	Motor nominal phase current	0.5 ~ 8	Yes
Uph Nominal	[V]	Motor nominal phase voltage	10 ~ U DCB max	Yes
N nom	[rpm]	Motor nominal speed	1000 ~ 40,000	Yes
I max	[A]	HW board current scale	2 ~ 20	Yes
U DCB max	[V]	HW board DC-bus voltage scale	20 ~ 450	Yes
I DCB over	[A]	SW trigger of DC-bus current limit	0.5 ~ I max	No
U DCB under	[V]	SW trigger of DC-bus voltage lower limit	0 ~ U DCB Over	No
U DCB over	[V]	SW trigger of DC-bus voltage higher limit	U DCB Under ~ U max	No
I DCB limit	[A]	Output current limit of control loop	0.5 ~ I max	No
U DCB trip	[V]	Braking resistor threshold value	U DCB Over ~ U DCB max	No
N max	[rpm]	Speed scale	>1.1 * N nom	No
ke	[V.sec/rad]	Back-EMF constant	0.0005 ~ 0.1	No
PWM freq	[Hz]	Frequency of PWM output signal	4,000 ~ 20,000	No
Align current	[A]	Current limit during rotor alignment	I ph nom ~ 2.5 * I ph nom	No
Align duration	[s]	Duration of rotor alignment	0.1 ~ 5 sec	No

The parameters of controlled motor can be acquired from a motor datasheet provided by the motor manufacturer or laboratory measurement.

Clicking “Update Target” button effects update of rotor alignment dedicated variables in target using actual inputs from the tab, see [Table 2](#).

Table 2. Updated basic parameters

Variable Name in FreeMASTER	Variable Name in Source Code	Control Algorithm Part
Align Current	gsM1_Drive.f16DcBusCurrentAlign	Rotor alignment
Align Duration	gsM1_Drive.uw16McatAlignmentPeriodMs	Rotor alignment

5.3 Control Loop tab

The Control Loop tab is designed for speed and torque loop tuning. The torque and speed PI controllers run in parallel with common output limitation. The tab contains input parameters for torque and speed control loop that are used for PI controller, speed ramp, and speed filter constant calculations, as shown in [Figure 9](#).

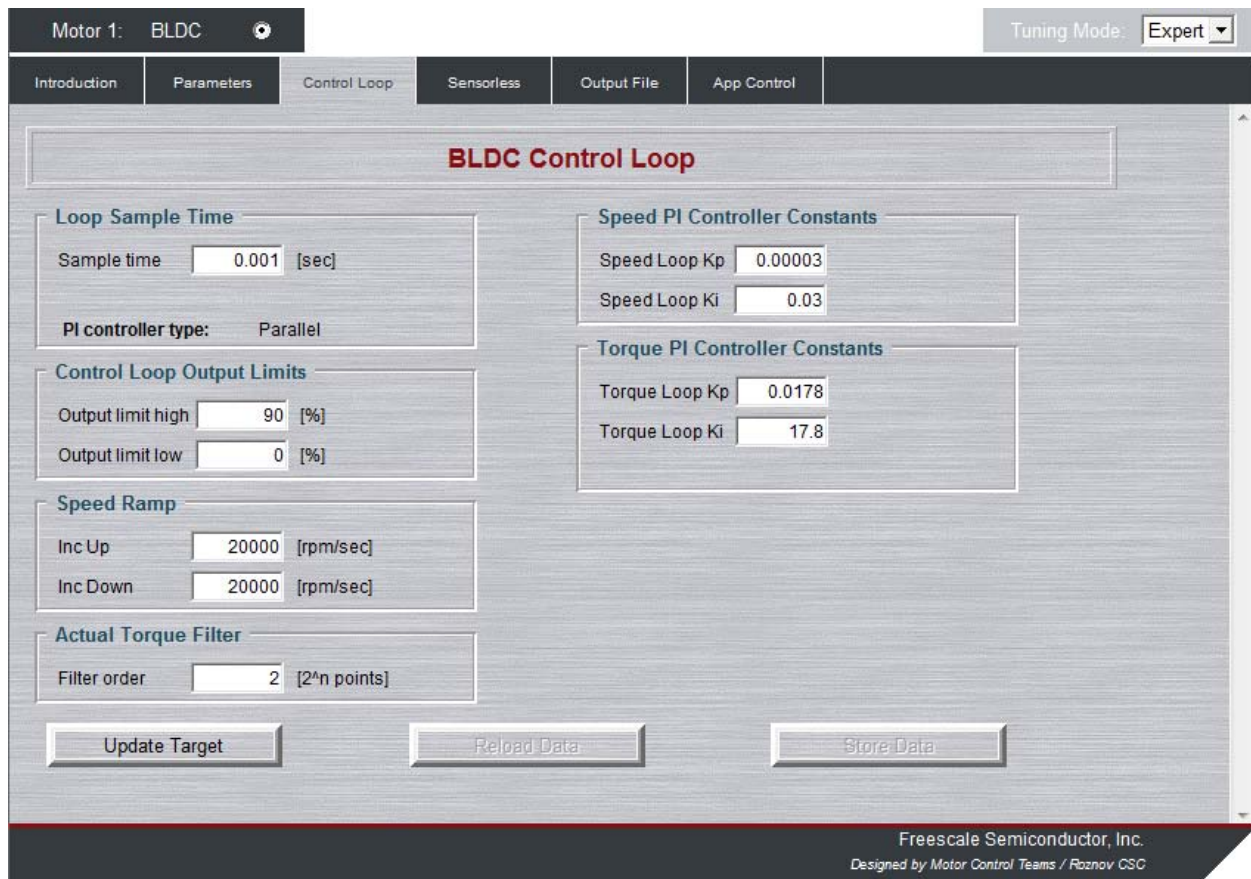


Figure 9. Control Loop tab

The [Table 3](#) shows the list of the speed loop input parameters with their physical units, brief description, typical range, and accessibility status in basic mode.

Table 3. Control Loop tab parameters

Parameter Name	Units	Description	Typical range	Basic mode accessibility
Sample time	[sec]	Control loop period	0.001 ~ 0.01	No
Output limit high	[%]	Control loop output higher limit	Output limit low ~ 100	No
Output limit low	[%]	Control loop output low limit	0 ~ Output limit high	No
Inc Up	[rpm/sec]	Speed ramp increment up	100 ~ 10,000	Yes
Inc Down	[rpm/sec]	Speed ramp increment down	100 ~ 10,000	Yes
Filter order	[2^n points]	Actual speed moving filter order	1 ~ 5	No
Speed Loop Kp	[-]	Proportional gain of speed PI controller in time domain	0.00001 ~ 0.1	No
Speed Loop Ki	[-]	Integration gain of speed PI controller in time domain	0.00001 ~ 0.1	No
Torque Loop Kp	[-]	Proportional gain of torque PI controller in time domain	0.00001 ~ 0.1	No
Torque Loop Ki	[-]	Integration gain of torque PI controller in time domain	0.00001 ~ 0.1	No

Clicking “Update Target” button effects update of control loop and speed ramp dedicated variables in target using actual inputs from the tab, see [Table 4](#).

Table 4. Updated Control Loop tab parameters

Variable name in FreeMASTER	Variable name in source code	Control algorithm part
Speed Loop Kp	gsM1_Drive.trSpeedPI.f16PropGain	Speed PI controller
Speed Loop Kp Shift	gsM1_Drive.trSpeedPI.w16PropGainShift	Speed PI controller
Speed Loop Ki	gsM1_Drive.trSpeedPI.f16IntegGain	Speed PI controller
Speed Loop Ki Shift	gsM1_Drive.trSpeedPI.w16IntegGainShift	Speed PI controller
Speed Loop Limit High	gsM1_Drive.trSpeedPI.f16UpperLimit	Speed PI controller
Speed Loop Limit Low	gsM1_Drive.trSpeedPI.f16LowerLimit	Speed PI controller
Torque Loop Kp	gsM1_Drive.trCurrentPI.f16PropGain	Torque PI controller
Torque Loop Kp Shift	gsM1_Drive.trCurrentPI.w16PropGainShift	Torque PI controller
Torque Loop Ki	gsM1_Drive.trCurrentPI.f16IntegGain	Torque PI controller
Torque Loop Ki Shift	gsM1_Drive.trCurrentPI.w16IntegGainShift	Torque PI controller
Torque Loop Limit High	gsM1_Drive.trCurrentPI.f16UpperLimit	Torque PI controller
Torque Loop Limit Low	gsM1_Drive.trCurrentPI.f16LowerLimit	Torque PI controller
Speed Ramp Up	gsM1_Drive.f32trSpeedRamp.f32RampUp	Speed ramp
Speed Ramp Down	gsM1_Drive.f32trSpeedRamp.f32RampDown	Speed ramp

5.4 Sensorless tab

The Sensorless tab enable parameter setting for BLDC sensorless control algorithm, described in DRM144. The tab is divided into two parts, the left side fields represent input parameters required for sensorless algorithm constant calculation and the right-side represents read-only calculated constants, as shown in [Figure 10](#).

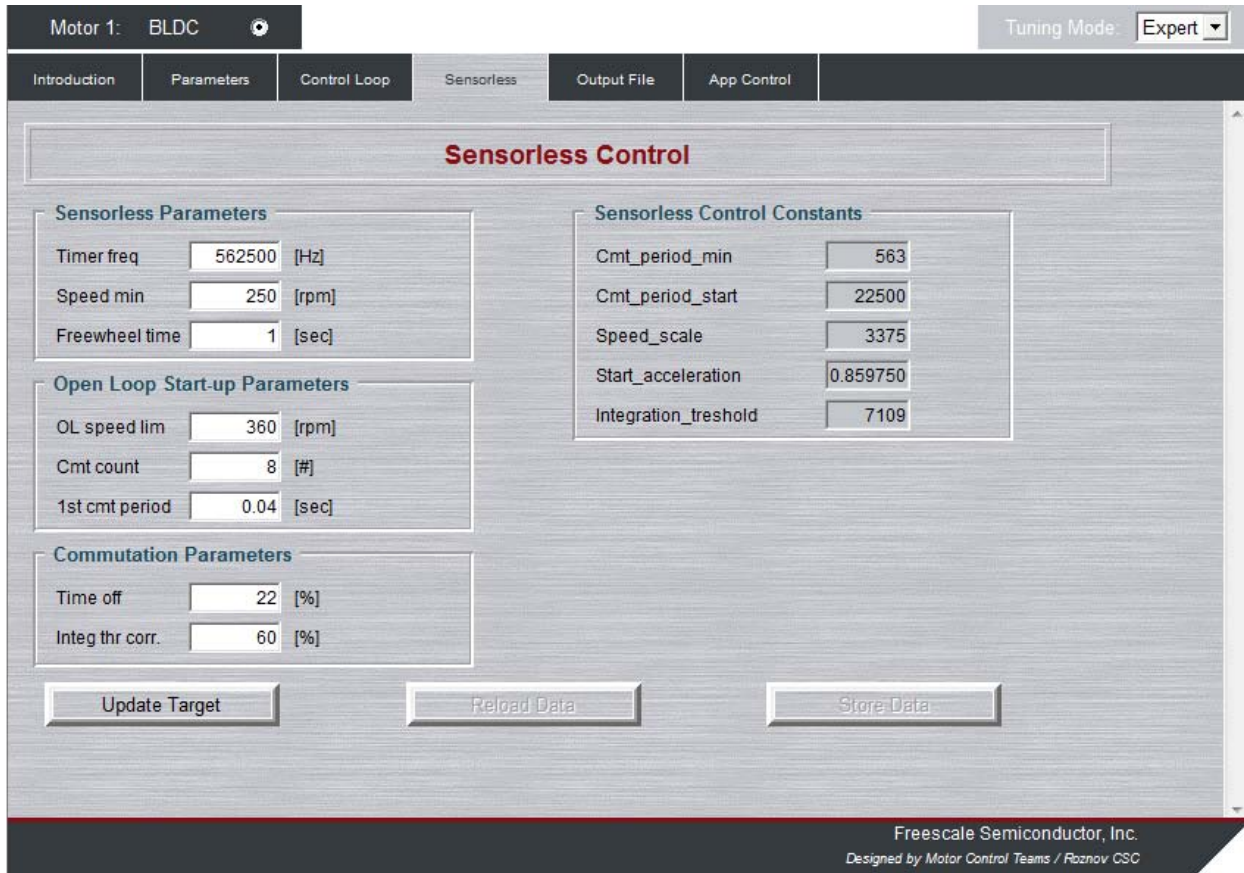


Figure 10. Sensorless tab

The [Table 5](#) shows the list of sensorless input parameters with their physical units, brief description, typical range, and accessibility status in basic mode.

Table 5. Sensorless tab parameters

Parameter name	Units	Description	Typical range	Basic mode accessibility
Timer freq	[Hz]	Frequency of timer used for commutation timing and period measurement	500 kHz to 1 MHz	No
Speed min	[rpm]	Minimal speed threshold for sensorless speed control	5-10% of N nom	No
Freewheel time	[sec]	Freewheel counter value	0.2 ~ 5	No
OL speed lim	[rpm]	Target open loop speed; threshold to switch to close loop operation	Speed min + 5% of N nom	No

Parameter name	Units	Description	Typical range	Basic mode accessibility
Cmt count	[#]	Commutation number for open-loop startup	5 ~ 15	No
1 st cmt period	[sec]	First commutation duration	0.01 ~ 0.0583	No
Time off	[%]	Current decay period in percentage of actual commutation period	20 ~ 40	No
Integ thr corr.	[%]	Back-EMF integration threshold correction factor	10 ~ 100	Yes

Clicking “Update Target” button effects update of control loop and speed ramp dedicated variables in target using actual inputs from the tab, see [Table 6](#).

Table 6. Updated Sensorless tab parameters

Variable name in FreeMASTER	Variable name in source code	Control algorithm part
Speed Minimal	gsM1_Drive.f16SpeedMinimal	Sensorless Control
Freewheel Period Long	gsM1_Drive.uw16FreewheelPeriodLongMs	Sensorless Control
Freewheel Period Short	gsM1_Drive.uw16FreewheelPeriodShortMs	Sensorless Control
Start Commutation Acceleration	gsM1_Drive.f16McatStartCmtAcceleration	Sensorless Control
Start Commutation Counter	gsM1_Drive.uw16McatStartCmtCounter	Sensorless Control
Next Commutation Period	gsM1_Drive.uw16McatPeriodCmtNext	Sensorless Control
Period Toff	gsM1_Drive.uw16McatPeriodToff	Sensorless Control
Back-EMF Integration Threshold	gsM1_Drive.f32IntegralBemfThreshold	Sensorless Control

5.5 Output File tab

The Output File tab serves a preview of the application constants corresponding to the tuned motor control application, as shown in [Figure 11](#). The constants are thematically divided into the groups according to selected control tabs as follows:

- Application scales
- Mechanical alignment
- BLDC control loop
- BLDC sensorless module
- FreeMASTER scale variables

Application tuning modes are not available for this tab.

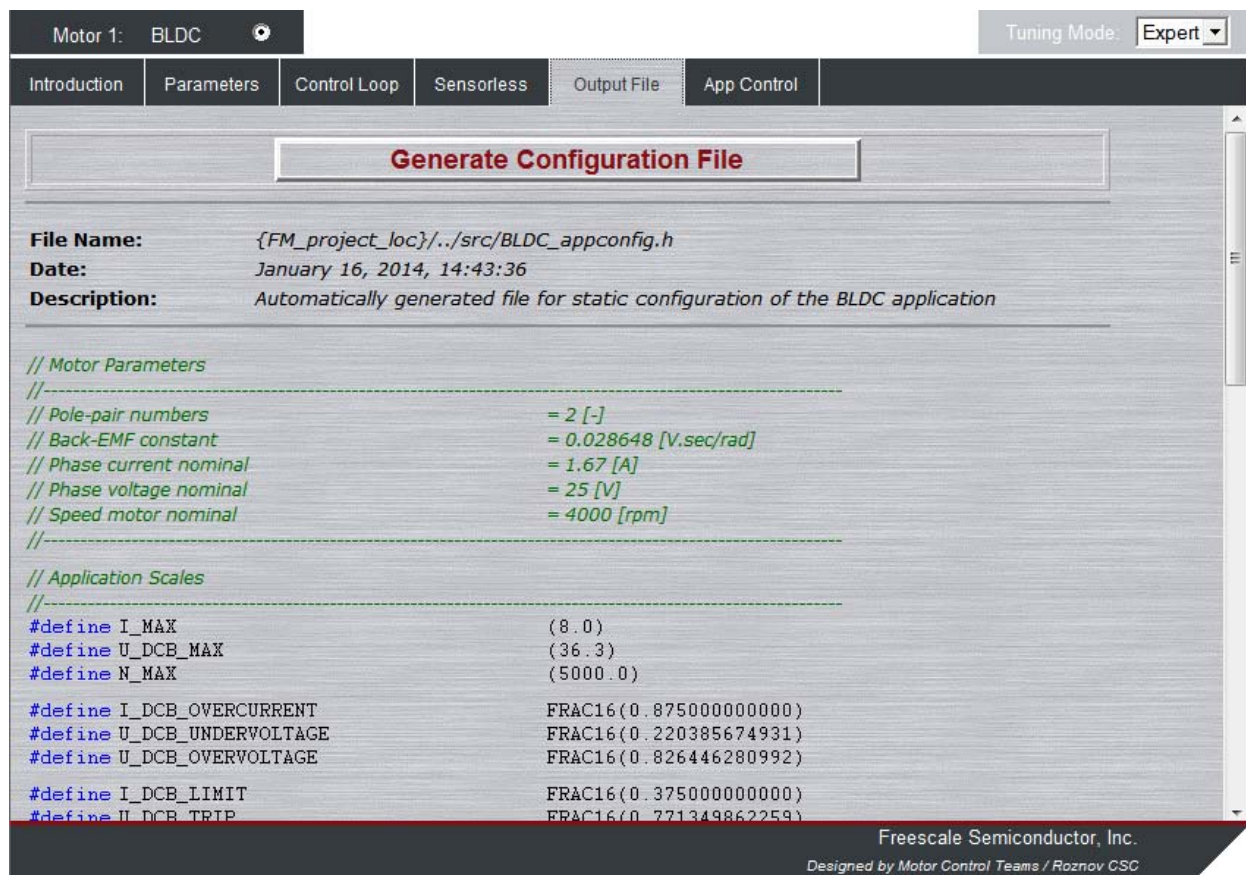


Figure 11. Output File tab

Click the Generate Configuration File button to generate the content of the Output File tab. The header file BLDC_appconfig.h is generated and will be saved to the default path.

5.6 Application Control tab

The last tab available from the menu is Application Control tab. The application control page is based on graphical components to provide user friendly control interface. Application control tab has been described in [Remote control using FreeMASTER, Figure 5](#).

5.7 Procedure to run a new motor

This section explains the procedure to run the motor in several parts. The first part describes MCAT parameter settings, the second part describes the motor's open-loop startup and closed-loop sensorless control parameter settings, and the last part provides the details to set the PI controllers.

5.7.1 MCAT parameters setting

MCAT in basic mode can be used for the default settings. Most of the parameters are calculated from basic application parameters. Expert mode can be selected for more precise application setting.

MCAT tool in Basic mode:

- Set motor parameters in Parameters tab:
 - o pp – number of pole-pairs
 - o I_{ph nom} - nominal motor current
 - o U_{ph nom} - nominal motor voltage
 - o N_{nom} - nominal motor speed
- Hardware scales need to be modify only if different power stage is used. No modifications are required when the default TWR-MC-LV3PH board is used.
- Set ramp increments in Control Loop tab to lower values, for example 500 rpm/s
- Go to Output File tab and click “Generate Configuration File” to replace original control constant reflecting modifications
- Compile application source code and load it to the target MCU

MCAT tool in Expert mode:

- Set motor parameters in Parameters tab:
 - o pp – number of pole-pairs
 - o I_{ph nom} - nominal motor current
 - o U_{ph nom} - nominal motor voltage
 - o N_{nom} - nominal motor speed
- Hardware scales need to be modify only if different power stage is used. In case the default TWR-MC-LV3PH board is used, no modifications required.
- Modify Fault Limits if required while consider following conditions:
 - o Typically I_{ph nom} < I_{DCB Over} < I_{max}
 - o Typically Supply voltage/2 < U_{DCB under} < Supply voltage
 - o Typically Supply voltage < U_{DCB over} < U_{DCB max}
- Modify Application Scales and Constants if required while consider following conditions:
 - o I_{DCB limit} is the limitation of torque PI controller. Typically I_{ph nom} < I_{DCB limit} < I_{DCB Over}.
 - o U_{DCB trip} is the threshold for turning the braking resistor on. The resistor needs to be connected to the TWR-MC-LV-3PH board. This option is used mostly for high-dynamic control or drive with high inertia. Typically U_{DCB trip} <= U_{DCB over}.
 - o N_{max} expresses speed scale, typically N_{max} >= 1.1 * N_{nom}.
 - o K_e is a motor Back-EMF constant that can is typically available in a motor datasheet, can be measured, or roughly calculated using equation:
 - o
$$k_e = \frac{U_{ph\ nom} * 60}{(2 * \pi * pp * N_{nom})}$$
 - o PWM freq corresponds to FlexTimer configuration (PWM frequency). It is highly recommended to not modify this parameter.

- Set rotor initial alignment parameters current and duration. Both parameters are proportional to motor load inertia. Final parameter setting can be done using online tuning:
 - o Align current = $I_{ph\ nom} \text{ to } 2.5 * I_{ph\ nom}$
 - o Align duration = 1 s
- Switch to Control Loop tab.
- Check sample time. The value corresponds to SLOW_TIMER_PERIOD defined in hwconfig.h file and it is highly recommended to not modify this parameter.
- Control Loop Output Limits can be set in range 0 ~ 100 % where Output low-limit is less than Output high-limit.
- Set initial ramp increments in Control Loop tab to lower values, for example 500 rpm/s. Final values can be set during on-line tuning.
- Set required value of Actual Torque filter order, typically in range 1 ~ 5.
- Set initial values of PI controllers. The Ki constants should be close to zero values. The final constant values are set during on-line tuning, see [PI controller tuning](#).
- Switch to Sensorless tab.
- Check Timer frequency value. The value corresponds to TIMER_FREQUENCY defined in hwconfig.h file and it is highly recommended to not modify this parameter.
- Set Speed minimal parameter that expresses the minimal value where sensorless algorithm is still able to run correctly. The initial value Speed min \geq 5% of Speed nom.
- Set Freewheel time that is proportional to motor inertia and expresses time the rotor decreases the speed below Speed min to zero rpm.
- Set Open loop (OL) speed min parameter where OL speed min $>$ Speed min. OL speed lim is target open loop startup speed where application switches to the sensorless motor control.
- Set Cmt count parameter. The typical initial range is 5 ~ 12.
- Check initialization values of Time off and Integration threshold correction to 50%.
- Go to Output File tab and click “Generate Configuration File” to replace original control constant reflecting modifications.
- Compile application source code and load it to the target MCU.

5.7.2 Open loop startup setting

After all motor parameters are entered the motor startup can be tuned. This application utilizes open loop startup. The rotor enters in an alignment state before entering in the open loop startup state. Rotor is aligned into a known position during the alignment state. Afterwards, the rotor starts to spin at roughly 10% of nominal speed during the open loop startup state.

None of the parameters are accessible in startup sequence of basic mode. Application utilizes default setting for startup. To test the startup, enter 20% of nominal speed in the Required Speed field and use application switch to start and stop the motor. Rotor shaft should make a few rotations. If the default startup sequence is not working then switch to expert mode and tune following parameters manually.

- For tune startup sequence

- OL speed limit – rotor speed which should be reached by open loop algorithm.
- cmt count – number of open loop commutation.
- 1st cmt period - time of first longest commutation period.

Note

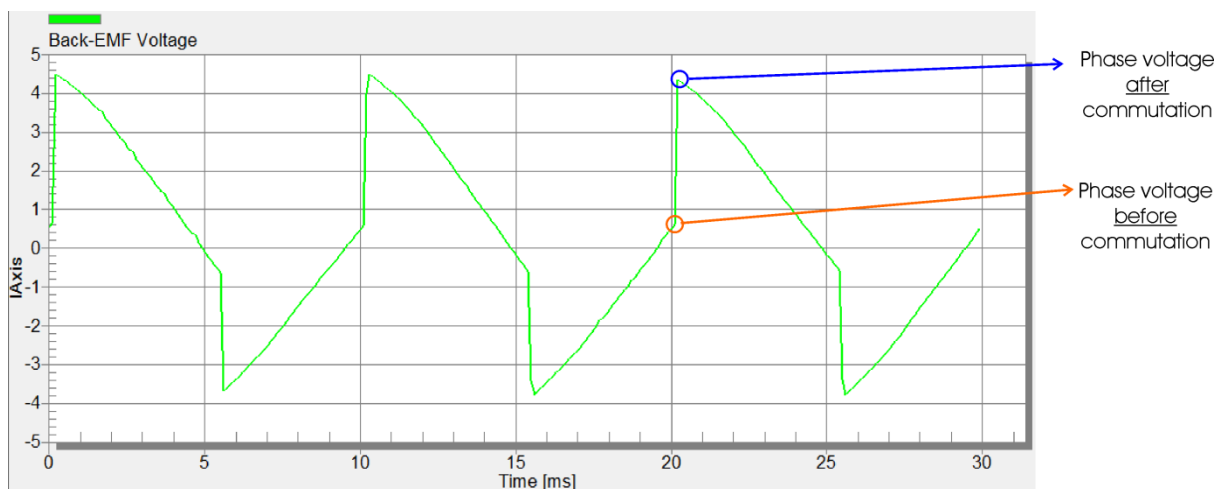
The alignment can be tuned only in expert mode.

- For tune alignment
 - Align current – Amplitude of current used during alignment and startup
 - Align duration – Time to make alignment of rotor
 - Sensorless algorithm setting

If the startup is working fine then the next step is tuning the sensorless part of algorithms. For tune commutation instance in basic mode only one parameter is needed. Time of next commutation is calculated based on Back-EMF voltage integration value. When this integrated value reaches the Back-EMF integration threshold, it is time to perform a commutation. Back-EMF integration threshold is calculated by MCAT tool from the motor and application scale parameters. To modify Back-EMF integration thresholds (and move time of commutations to prior or later) modify parameter 'Integ thr corr'. Value of 100% in 'Integ thr corr' parameter will apply 100% of calculated Back-EMF integration threshold value. The Back-EMF voltage signal will be observed while tuning. The shape of measured signal depends on the commutation threshold setup. Use the “Back-EMF Voltage” recorder to analyze the results. Target is to achieve same voltage before and after commutation at disconnected phases. Following situations can be observed during motor tuning.

Case 1: Commutation comes too early.

Behavior: Motor can achieve higher speed but with reduced torque. Voltage waveform is shown in [Figure 12](#).



Phase voltage after commutation >> Phase voltage before commutation



Figure 12. Early commutation timing

Solution: Increase the ‘Integ thr corr’ variable placed on control page ‘Sensorless’.

Case 2: Commutation comes precisely.

Behavior: Motor can deliver good torque and is silent. Voltage waveform is shown on [Figure 13](#). Voltages before and after the commutations are approximately equal. If the measured signal is asymmetric, then it is possible that the motor is not constructed precisely or the impedances of the phases are not equal.

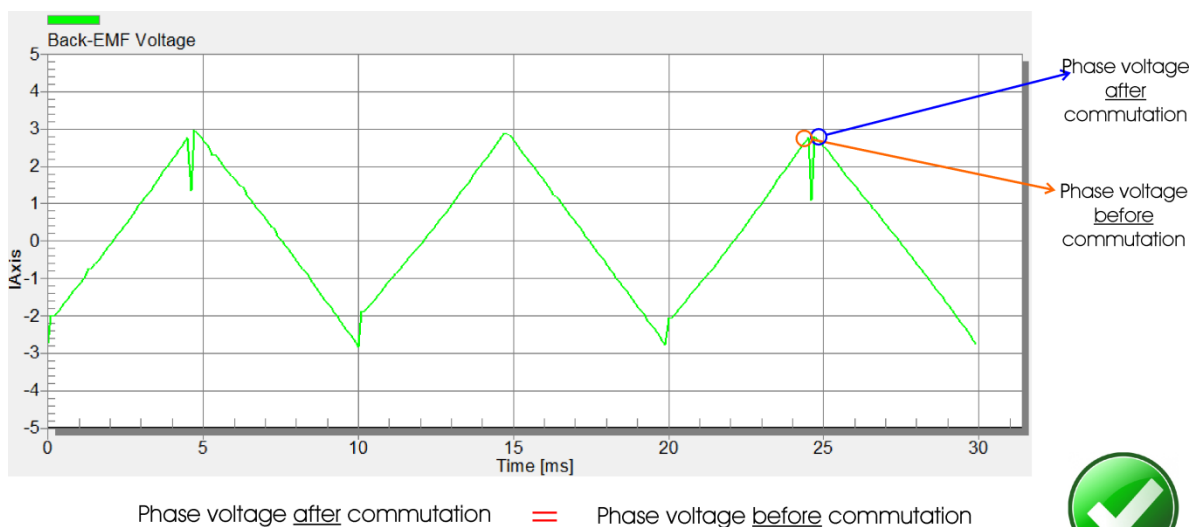


Figure 13. Precise commutation timing

Case 3: Commutation comes too late.

Behavior: The motor is significantly noisier. Also, the efficiency of the motor may be worse. Voltage waveform is shown in [Figure 14](#).

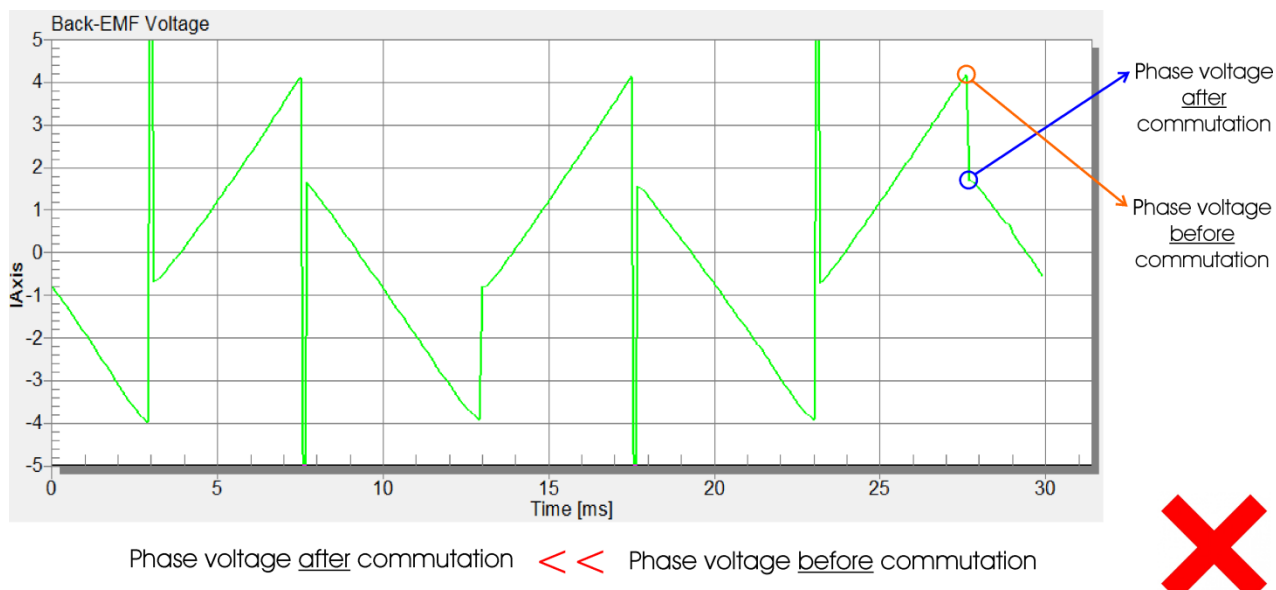


Figure 14. Late commutation timing

Solution: Decrease the “Integ thr corr” variable placed on control page “Sensorless”.

The voltage spikes during commutations are caused by current recirculation, interval and measuring BEMF in this interval is not required. Increase the Time off parameter to avoid influence of the current recirculation interval into Back-EMF voltage evaluation. This parameter is a percentage of previous commutation period. This setting should be done in respect to nominal motor load, because a current recirculation interval depends on the motor current (25% should fit requirements of most applications).

5.7.3 PI controller tuning

The motor mechanical dynamic is a first order function with mechanical time constant, which depends on motor inertia and friction. In practice, the parameters of motor and its load, inertia and friction is very often unknown and it is quite difficult to obtain them.

Therefore, the manual tuning of P and I portion of speed and torque controllers is used to achieve required speed response.

There are many approaches available to tune PI controller constants. The following steps provide one of the examples to set tune speed PI controller for BLDC motor.

1. Tuning of proportional gain
 - Set Speed loop Ki integral gain to zero
 - Set speed ramp to 1000 rpm/sec or higher
 - Run the motor at a convenient speed, about 30% of N nominal
 - Set a step in required speed to 40% of N nom

- Adjust proportional gain Speed Loop K_p until the system responds properly to required value without oscillations or excessive overshoot
 - If K_p is set low the system respond will be slow
 - If K_p is set high the system respond will be tighter
 - With $K_i = 0$, system will not probably achieve required speed
2. Tuning of integral gain
- Increase slowly K_i to force the difference between required and actual speed to zero
 - Adjust K_i in such way to not see any oscillation or big overshoot of actual speed value while a required speed step is applied.

6 References

Following references are available on freescale.com:

1. *K10 Sub-Family Reference Manual*, KV10P48M75RM, by Freescale Semiconductor, Inc., 2013.
2. *Three-Phase BLDC Sensorless Motor Control Application*, DRM144, by Freescale Semiconductor, Inc., 2014.
3. *Three-Phase BLDC Sensorless Control Using the MKV10x*, AN4862, by Freescale Semiconductor, Inc., 2014.

7 Revision history

Revision number	Date	Substantial changes
0	02/2014	Initial release



How to Reach Us:

Home Page:

freescale.com

Web Support:

freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, Kinetis are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Tower is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2014 Freescale Semiconductor, Inc.



Document Number: AN4870

Rev. 0

02/2014

