User's Guide
MSPM0 Universal FOC Tuning

ABSTRACT

This tuning guide provides step-by-step guidance to set up MSPM0 MCU and supported driver hardware board to tune and spin a 3-phase brushless DC motor using Universal FOC Motor Control Library.

Universal FOC motor control library is an open source FOC library that supports wide range of rotor position estimation algorithms. This initial version of Universal FOC supports Enhanced Sliding Mode Observer and Finite BEMF estimation methods for rotor position estimation in sensorless FOC.

Note

This Tuning guide is in reference to the Universal FOC v1.01.00 from SDK Version **2.04.00.00** and above.

Table of Contents

List of Figures

List of Tables

Trademarks

LaunchPad™ is a trademark of Texas Instruments.
Arm® and Cortex® are registered trademarks of Arm Limited.
All trademarks are the property of their respective owners.

1 Introduction

The MSPM0Gxxx family of 80MHz Arm®-Cortex® M0+ MCUs can commutate a 3-phase brushless DC (BLDC) motor based on various sensorless and Sensored FOC control.

The BLDC motor is driven by a three-phase brushless DC (BLDC) MOSFET gate driver or integrated MOSFET motor driver at nominal DC rails or battery-pack voltages. The driver typically integrates three current-sense amplifiers (CSAs) for sensing the three-phase currents of BLDC motors to achieve optimum FOC control.

Figure 1-1 shows a simplified schematic of an MSPM0Gxxx MCU and BLDC motor driver.

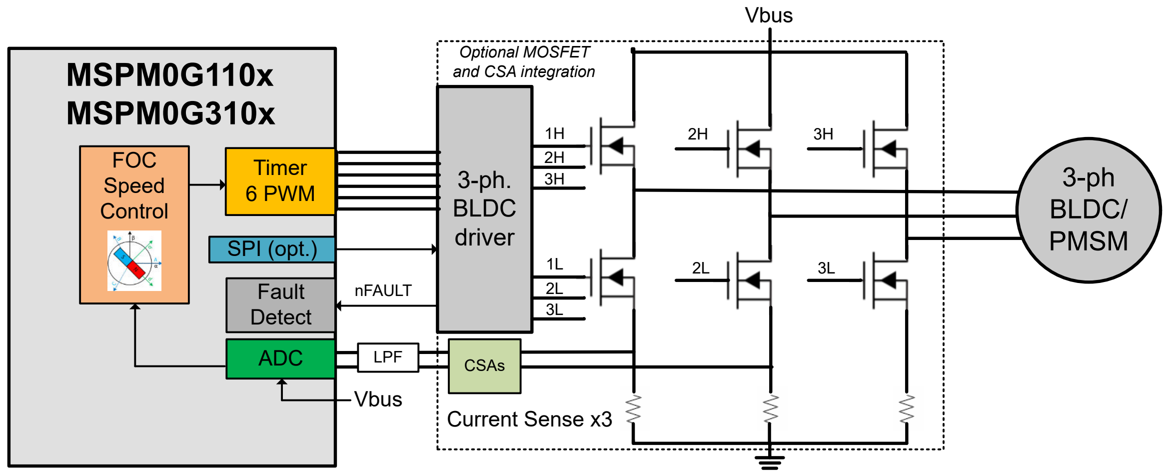


Figure 1-1. Simplified Schematic of MSPM0Gxxx + BLDC Motor Driver

This tuning guide provides the steps to tune a 3-phase BLDC motor using an MSPM0Gxxxx MCU. The tuning process is classified into four sections: **Hardware Setup, Software Setup, Basic Tuning** and **Advanced Tuning**.

- ***Hardware setup*** *:* Steps to set up TI-provided hardware or use a custom PCB for the tuning process.
- ***Software setup:*** Steps to set up TI-provided software for spinning and tuning a BLDC motor.
- ***GUI setup (optional)****:* Steps to use a graphical user interface (GUI) for spinning & tuning a BLDC motor.
- ***Basic tuning:*** Tuning steps to successfully spin the motor in closed loop.
- ***Advanced tuning:*** Tuning steps to conform to use-case and explore features in the device.

2 Hardware Setup

The following items are required to use this tuning guide:

- LP-MSPM0G3507/LP-MSPM0G3519 board
- Supported DRV83xx motor driver evaluation module (EVM)
  - BOOSTXL-DRV8323RS
  - DRV8316REVM
  - DRV8329 EVM
- Jumper wires for pin table connections
- A computer with the MSPM0 FOC software installed
- A BLDC motor to be tuned using this process. The motor data sheet is helpful but not mandatory.
- A DC power supply rated for the motor
- Basic lab equipment such as a digital multimeter (DMM), oscilloscope, current probe, and voltage probe

Figure 2-1 shows the block diagram connections for a sensorless FOC motor system. The system can be built using:

- TI-provided hardware (LP-MSPM0G3507/LP-MSPM0G3519 and DRV83xx EVM)
- Custom PCB hardware with an onboard MSPM0Gxxx MCU and a BLDC motor driver

The following sections describe how to configure the pins for each portion of the sensorless FOC block diagram.

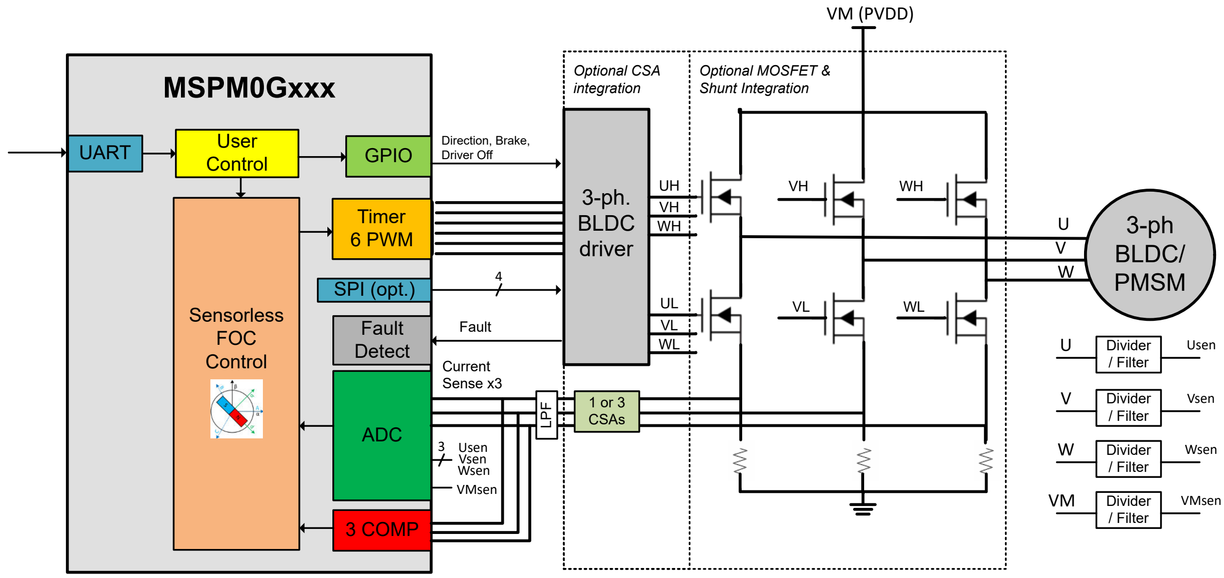


Figure 2-1. MSPM0Gxxx + BLDC Motor Driver - Sensorless FOC Block Diagram

The System Configuration tool (SysConfig) helps to configure the pins in a motor control system. The default pin configurations are provided for the EVM hardware setup to spin a motor, but pins can be remapped to other pins visually inside SysConfig. This is useful for reconfiguring different pins (such as PWM, ADC, or other control signals) on a custom PCB or for scaling to different packages across MSPM0 devices.

2.1 EVM Hardware Setup

TI provides LaunchPad™ development kits to evaluate MSPM0 Arm Cortex-M0+ microcontrollers and evaluation modules (EVMs) to evaluate the DRV83xx family of brushless-DC motor drivers. These evaluation boards are available on ti.com and can be used as a system evaluation platform for Universal FOC motor control.

For supported evaluation boards, refer to Section 2.1.1.

Note

The provided defaults have pre-configured pins that are intended to support hardware evaluation boards. If a custom PCB is used, refer to the following *Pin Configurations* sections to assign the supported pins for the 3-phase motor driver.

2.1.1 EVM Hardware Support

Table 2-1 shows the supported MSPM0 LaunchPad kits and EVMs and the connection guides for 3-phase Universal FOC motor control.

Table 2-1. Supported Hardware for Universal FOC Using MSPM0

MSPM0Gxxx LaunchPad™ Kit	Motor Driver Hardware	Hardware User's Guide	Current Sense Amplifiers	SPI Driver Support	Recommended Motor Voltage Range	Recommended Motor Power
LP-MSPM0G3507 LP-MSPM0G3519	BOOSTXL-DRV8323RS	BOOSTXL-DRV8323Rx EVM User's Guide	3	Yes	6V to 60V	< 1000W
LP-MSPM0G3507 LP-MSPM0G3519	DRV8316REVM	DRV8316REVM User's Guide	3	Yes	4.5V to 35V	< 80W
LP-MSPM0G3507 LP-MSPM0G3519	DRV8329EVM	DRV8329AEVM User's Guide	1	No	4.5V to 60V	<1000W
MSPM0G1507	TIDA-010250 Reference Design	TIDA-010250 Design Guide	1, 2, 3	No	265V maximum AC supply	<1000W

Note

Make sure that the jumper configurations for the LaunchPad kit and EVM are correct. For more information, see the user's guides for the LaunchPad kit and EVM.

2.2 Peripheral Configurations for IPD Usage

Initial Position detection algorithm utilizes the ADC current sensing path to identify the phase current rise times to detect the rotor position. The window comparators of ADC continuously monitor the phase current against a pre-set IPD threshold current limit to generate a current Pulse. A Timer is used to capture these various pulse rise times across different sectors and compared against to detect the rotor position. The Timer and ADC configurations are updated during the IPD initialization based on the Sysconfig. The algorithm configures the required WCOMP settings based on the current sense selected and Section 8.3.1.

Note

During IPD pulse time, the rest of the algorithm interrupts are halted to continuously monitor the ADC current at very high sampling rates. The normal interrupt operation resumes once the IPD operation is complete.

2.3 Pin Configurations for PWM Outputs

The default pin configurations for PWM outputs are shown in Table 2-2. The required connections are six PWM output signals that send the commutation patterns for universal FOC motor control. TIMA includes features for motor control, such as complimentary PWM outputs with deadband, fault handling with <40ns response time, and repeat counters for configuring FOC loop rates.

TIMA0 is the preferred timer for motor control because this timer provides three complimentary pairs of PWM outputs from the same timer counter (such as TIMA0_C1 and TIMA0_C1N), but any TIMA0 or TIMA1 output pair can be used and cross-triggered to provide the six PWM output signals.

Table 2-2. Pin Configurations for PWM Outputs

MSPM0 Pin	Function	DRV Connection	DRV Function
TIMA0_C0	TIMA0 channel 0 output pin	INHA	Phase A high side PWM input
TIMA0_C0N	TIMA0 channel 0 complimentary output pin	INLA	Phase A low side PWM input
TIMA0_C1	TIMA0 channel 1 output pin	INHB	Phase B high side PWM input
TIMA0_C1N	TIMA0 channel 1 complimentary output pin	INLB	Phase B low side PWM input
TIMA0_C2	TIMA0 channel 2 output pin	INHC	Phase C high side PWM input
TIMA0_C2N	TIMA0 channel 2 complimentary output pin	INLC	Phase C low side PWM input

2.4 Pin Configurations for ADC Currents

ADC configuration for three phase current sensing: The default pin configurations for ADC currents for three phase current sensing are shown in Table 2-3 and Table 2-5, depending on the DRV device used. The required connections are three ADC inputs connected to the three CSA outputs from the motor driver or external CSAs.

ADC0 and ADC1 are two simultaneous-sampling 4Msps analog-to-digital converters that are used to measure phase currents and voltages. ADC0 and ADC1 measure phase currents simultaneously and bus voltage sequentially depending on the rotor angle under normal motor run conditions.

An optional low-pass RC filter can be placed in series from the CSA outputs to the ADC inputs to filter out any high-frequency noise from the switching output signals for proper ADC sampling as shown in Figure 2-2.

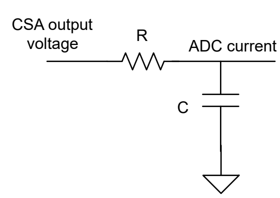


Figure 2-2. CSA Output Filter

Choose a filtering frequency f_c which is at least 10 times the PWM switching frequency (f_{PWM}). Use Equation 1 to calculate f_c based on the RC filter design.

$$f_c = \frac{1}{2\pi RC} \tag{1}$$

Table 2-3. Pin Configurations for ADC Currents With Simultaneous Sampling in DRV8316 - LP MSPM0G3507

MSPM0 Pin	Function	DRV Connection	DRV Function
A0_3	ADC0, channel 3 input	SOA	Phase A current sense output
A0_2	ADC0, channel 2 input	SOB	Phase B current sense output
A1_2	ADC1, channel 2 input	SOB	Phase B current sense output
A1_1	ADC1, channel 1 input	SOC	Phase C current sense output

Table 2-4. Pin Configurations for ADC Currents With Simultaneous Sampling in DRV8316 - LP MSPM0G3519

MSPM0 Pin	Function	DRV Connection	DRV Function
A1_13	ADC1, channel 13 input	SOA	Phase A current sense output
A1_14	ADC1, channel 14 input	SOB	Phase B current sense output
A0_12	ADC0, channel 12 input	SOB	Phase B current sense output
A0_2	ADC0, channel 2 input	SOC	Phase C current sense output

Table 2-5. Pin Configurations for ADC Currents Without Simultaneous Sampling in DRV8323 - LP MSPM0G3507

MSPM0 Pin	Function	DRV Connection	DRV Function
A1_2	ADC1, channel 2 input	SOA	Phase A current sense output
A0_3	ADC0, channel 2 input	SOB	Phase B current sense output
A1_3	ADC1, channel 3 input	SOC	Phase C current sense output

Table 2-6. Pin Configurations for ADC Currents Without Simultaneous Sampling in DRV8323 - LP MSPM0G3519

MSPM0 Pin	Function	DRV Connection	DRV Function
A1_14	ADC1, channel 14 input	SOA	Phase A current sense output
A1_13	ADC1, channel 13 input	SOB	Phase B current sense output
A0_4	ADC0, channel 4 input	SOC	Phase C current sense output

ADC configuration for single shunt current sensing: The ADC pin configurations for single shunt current sensing in DRV8329 is shown in ADC Pin Configuration for Single Shunt Current Sensing in DRV8329 - LP_MSPM0G3507.

In single shunt current sensing either of ADC0/ADC1 is used to sample the same shunt current at two different instances in a single PWM cycle to estimate the three phase currents. User need to configure appropriate ADC to sample the Currest sense output from the Memory '0' index, Memory '1' Index for FOC operation. Refer to the Universal FOC User Guide for details on ADC cufigurations for Single shunt current sensing.

Table 2-7. ADC Pin Configuration for Single Shunt Current Sensing in DRV8329 - LP_MSPM0G3507

MSPM0 Pin	Function	DRV Connection	DRV Function
A1_2	ADC 1, Channel 2 Input	SOX	DC bus current sense

Table 2-8. ADC Pin Configuration for Single Shunt Current Sensing in DRV8329 - LP_MSPM0G3519

MSPM0 Pin	Function	DRV Connection	DRV Function
A1_14	ADC 1, Channel 14 Input	SOX	DC bus current sense

2.5 Pin Configurations for ADC Voltages

The default pin configurations for ADC voltages are shown in the following tables. The required connections are four ADC inputs:

- Three ADC inputs connected to the three sensed phase voltages from the motor (VSENA,VSENB, VSENC)
- One ADC input connected to the sensed VM motor voltage (VSENVM)

The sensed voltages are realized using a resistor divider with an optional bypass filtering cap as shown in Figure 2-3. Size the resistors so any motor voltage transients do not exceed the maximum voltage of the ADC inputs. For more information on the resistor divider ratio, see Section 6.1.6.

Note

Phase voltage sensing is needed only if initial speed detection feature is required.

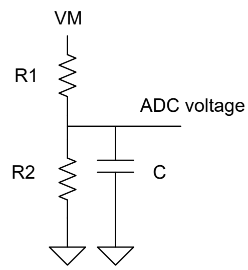


Figure 2-3. ADC Voltage Divider

Table 2-9. Pin Configurations for ADC Phase Voltages for DRV8316/DRV8329 /DRV8323- LP MSPM0G3507

MSPM0 Pin	Function	DRV Connection	DRV Function
A1_6	ADC1, channel 6 input	VSENA	Phase A sensed voltage output
A0_7	ADC0, channel 7 input	VSENB	Phase B sensed voltage output
A1_5	ADC1, channel 5 input	VSENC	Phase C sensed voltage output

Table 2-10. Pin Configurations for ADC Phase Voltages for DRV8316/DRV8329/DRV8323 - LP MSPM0G3519

MSPM0 Pin	Function	DRV Connection	DRV Function
A0_0	ADC0, channel 0 input	VSENA	Phase A sensed voltage output
A0_1	ADC0, channel 1 input	VSENB	Phase B sensed voltage output
A0_5	ADC0, channel 5 input	VSENC	Phase C sensed voltage output

Table 2-11. Pin Configurations for ADC DC Bus Voltage Sensing for DRV8316/DRV8329 - MSPM0G3507

MSPM0 Pin	Function	DRV Connection	DRV Function
A1_3	ADC1, channel 3 input	VSEN -Vm	DC bus voltage output

Table 2-12. Pin Configurations for ADC DC Bus Voltage Sensing for DRV8316/DRV8329 - MSPM0G3519

MSPM0 Pin	Function	DRV Connection	DRV Function
A0_4	ADC1, channel 3 input	VSEN -Vm	DC bus voltage output

Table 2-13. Pin Configurations for ADC DC Bus Voltage Sensing for DRV8323 - LP MSPM0G3507

MSPM0 Pin	Function	DRV Connection	DRV Function
A0_2	ADC0, channel 2 input	VSEN -Vm	DC bus voltage output

Table 2-14. Pin Configurations for ADC DC Bus Voltage Sensing for DRV8323 - LP- MSPM0G3519

MSPM0 Pin	Function	DRV Connection	DRV Function
A0_12	ADC0, channel 2 input	VSEN -Vm	DC bus voltage output

2.6 Pin Configurations for Faults

Faults can be detected in hardware by the motor driver or MCU.

Typically, a motor driver drives an active-low open-drain fault pin (nFAULT) when there is a detected fault in the system. Examples are MOSFET overcurrent, gate drive, or power supply-related faults connections in the driver.

MSPM0 MCUs can detect fault inputs with dedicated hardware paths to provide low latency and response times as fast as 40ns. This is faster than using a conventional GPIO interrupt with software latency. The fault input paths can be configured for fault handling using TIMA fault handler, such as shutting off the PWMs during an overcurrent condition. Examples of TIMA inputs include an external fault pin (such as TIMA_FLT0) and low-side overcurrent using comparators (such as COMP0_IN0+).

2.7 Pin Configurations for GPIO Output Functions

Many GPIO output functions from the MSPM0 can be used for motor driver specific functions controlled by logic-level pins. Examples of motor driver functions are:

- Enable pin (ENABLE) / active-low sleep mode control (nSLEEP)
- Active high gate driver shutoff (DRVOFF)
- Active-high CSA Calibration (CAL)
- Active-high brake (BRAKE) / active-low brake (nBRAKE)
- Direction pin (DIR)

Note

See the motor driver data sheet and the user guide for GPIO configurable pins.

2.8 Pin Configurations for SPI Communication

The default pin configurations for SPI connections are shown in Section 2.8. Some motor drivers include an optional SPI that is used for configuring control registers and reading status registers for fault diagnosis. Some examples of SPI registers are:

- Configuring gate drive source/sink current strength
- Configuring CSA output behavior
- Running diagnostics
- Reading fault bits when the fault pin has been detected as active low
- Clearing fault status bits once the fault condition is removed
- Clearing watchdog timers

Note

If a SPI or hardware interface is used to configure system settings, see the motor driver device-specific data sheet.

Table 2-15. Pin Configurations for SPI Connections

MSPM0 Pin	Function	DRV Connection	DRV Function
SPIx_CSy	SPI chip select (y = 0,1,2,3)	nSCS	SPI chip select
SPIx_SCK	SPI clock	SCLK	SPI clock
SPIx_POCI	SPI peripheral out controller in	SDO	SPI data out
SPIx_PICO	SPI peripheral in controller out	SDI	SPI data in

Note

To determine if the SDO pin is open-drain and requires a pullup resistor, see the motor driver device-specific data sheet.

2.9 Pin Configurations for UART Communication

UART can be used to receive commands to configure, spin, and control the motor. The commands are sent from a host MCU or GUI and can optionally be used for advanced protocols such as LIN communication.

Note

Use UART instance 0 (UART0_RX, UART0_TX) to configure the UART interface when used along with DMA and LIN interface.

Note

Use UART instance 3 (UART3_RX, UART3_TX) to configure the UART interface for GUI communication when used along with DMA.

Table 2-16. Pin Configurations for UART Connections

MSPM0 Pin	Function
UARTx_RX	UART receive
UARTx_TX	UART transmit

2.10 External Connections for Evaluation Boards

Follow the steps below when connecting an MSPM0 LaunchPad to a DRV83xx EVM:

1. Connect the three motor phase terminals to the driver board (phases A, B, and C). If the motor has a center tap connection or wires for Hall-effect sensors, leave these wires unconnected.
2. Make the inter-device connections from the MSPM0 LaunchPad kit to the DRV83xx EVM by mating the EVM to the LaunchPad kit or using jumper wires as shown in Figure 2-4. For hardware user guide connection details, see Section 2.1.1.

Note

If using the GUI to communicate to the MSPM0 device using USB to backchannel UART, connect the backchannel UART connections to UART3_TX and UART3_RX as shown in Figure 2-5/ Figure 2-6

3. Connect a micro-USB cable from the MSPM0 LaunchPad kit to the PC.
   a. Remove GND and 3V3 isolation jumpers on the bridge if desired to isolate the PC from the motor system. If this step is done, 3V3 must be provided externally or from the DRV83xx EVM board, if available.
4. Supply a voltage compliant with the Power Supply Voltage (VM) range. For the recommended voltage range, see the board-specific user's guide or DRV-specific data sheet.
5.

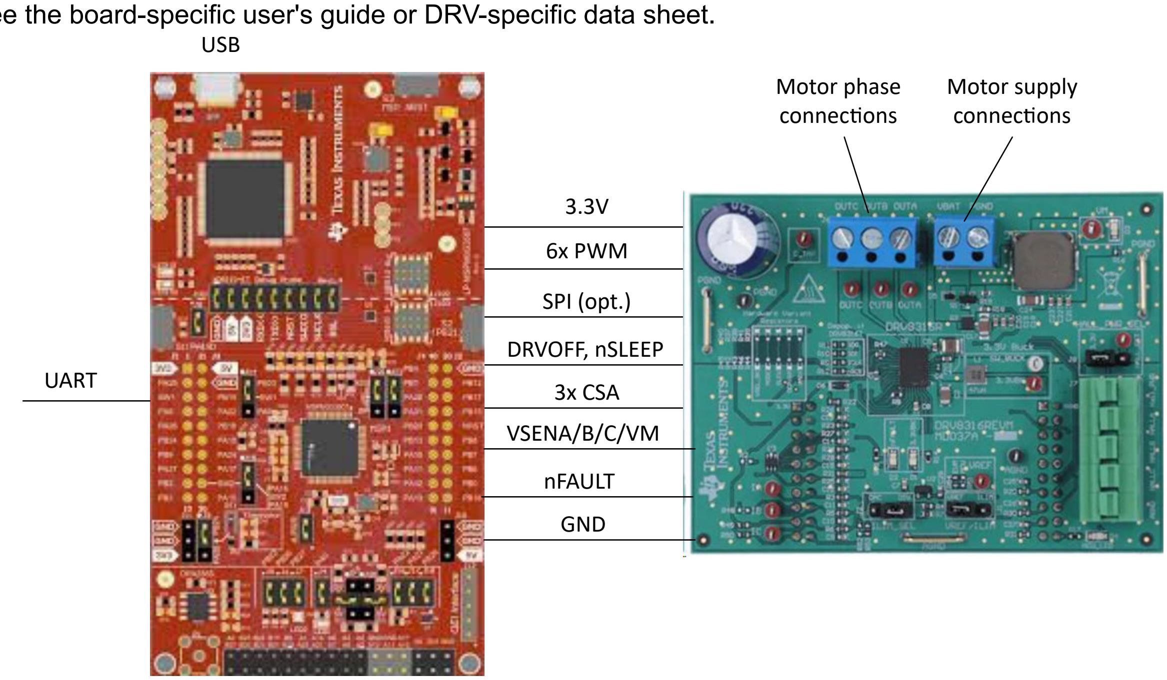


Figure 2-4. MSPM0 LaunchPad Kit and DRV83xx EVM External Configuration

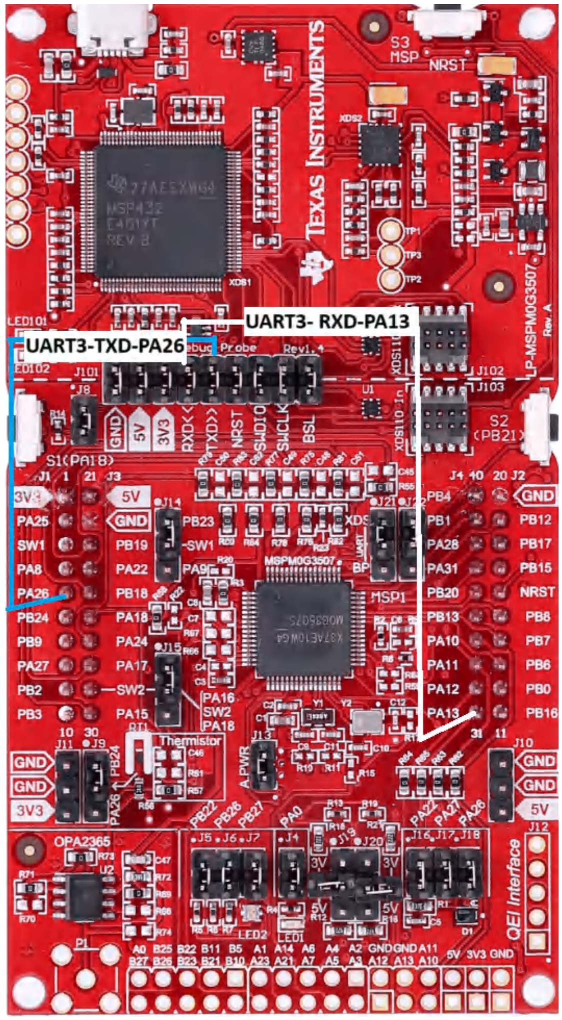


Figure 2-5. LP-MSPM0G3507 Backchannel Connection to UART3

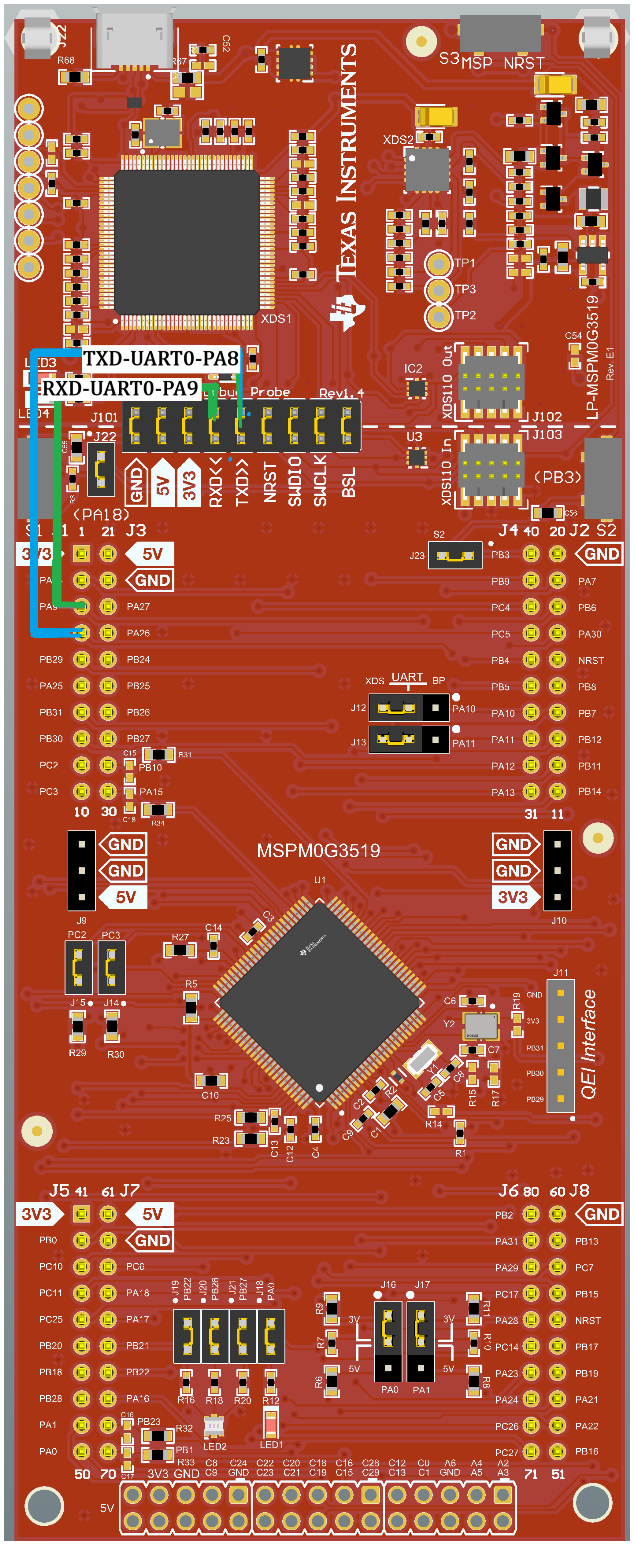


Figure 2-6. LP-MSPM0G3519 Backchannel Connection to UART0

Note

If a different UARTn port is used to connect the serial pins, configure appropriate UART instance pins in the SYSCONFIG file of the CCS project.

---

3 Software Setup

Universal FOC Motor control library examples for MSPM0 MCU's are provided as part of MSPM0-SDK and is available for evaluation with Code Composer Studio IDE.

Table 3-1 shows the software and documentation supported for Universal FOC control in TI Resource Explorer.

Table 3-1. Software Support for FOC Control

Universal FOC User's Guide[(1)]	Code Examples	GUI
Universal FOC User's Guide	Universal FOC Examples	MSPM0G Universal FOC GUI

(1) Includes library overview, software setup, hardware setup, and more.

4 GUI Setup

The user can optionally use the MSPM0 Universal FOC GUI as a host to send commands to the MSPM0 MCU at the target to control the motor using serial to UART interface.

The GUI contains a USB-to-UART codec that can send UART commands as a host to the MSPM0 LaunchPad kit. The application software includes a configurable UART register map and data format that translates the UART data into simplified motor control commands.

Table 4-1. GUI Connection Types

Connection	Interface	Hardware Connections
GUI to target MSPM0 MCU	UART	UART3_TX, UART3_RX

To launch the GUI, go to the MSPM0 Universal FOC GUI. page.

4.1 Serial Port Configuration

Configure the serial port based on the connected port to the PC and configure the Baud rate as 115200.

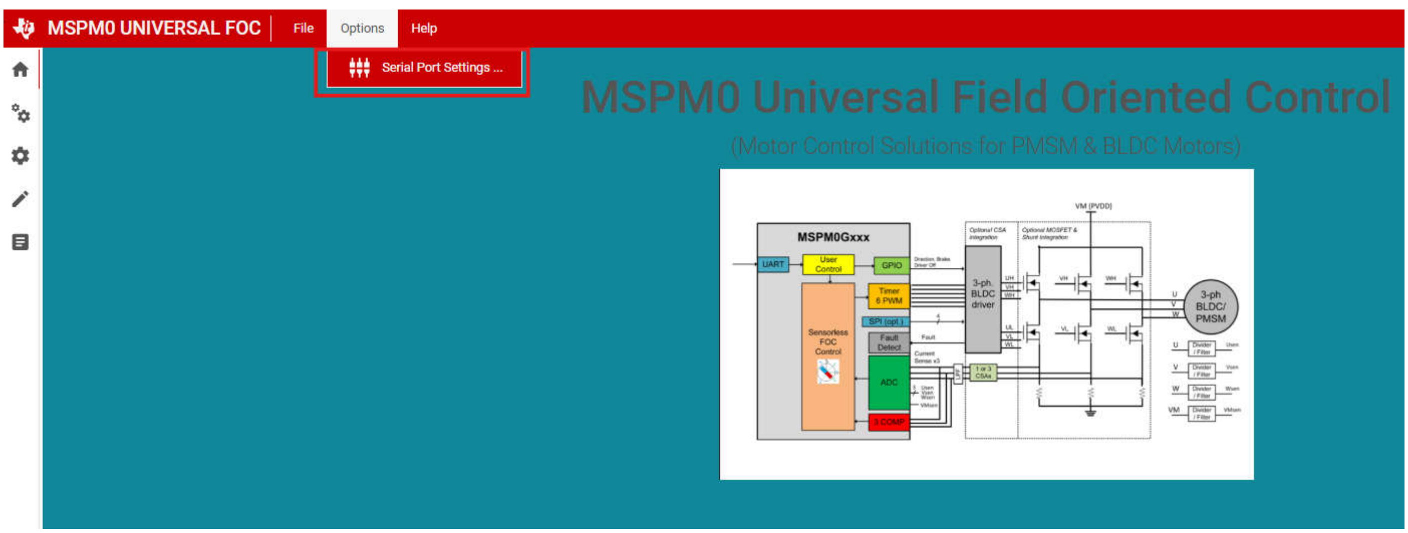


Figure 4-1. Option to Select Serial Port Settings

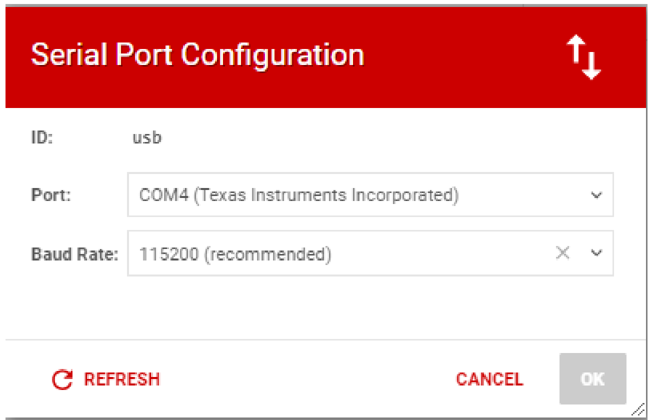


Figure 4-2. Serial Port Configurations

4.2 GUI Home Page

Below is the GUI home page from which user can navigate to various windows for specific configurations.

4.3 System Configurations

User can set the basic configurations of Motor and EVM system parameters from the system configuration page

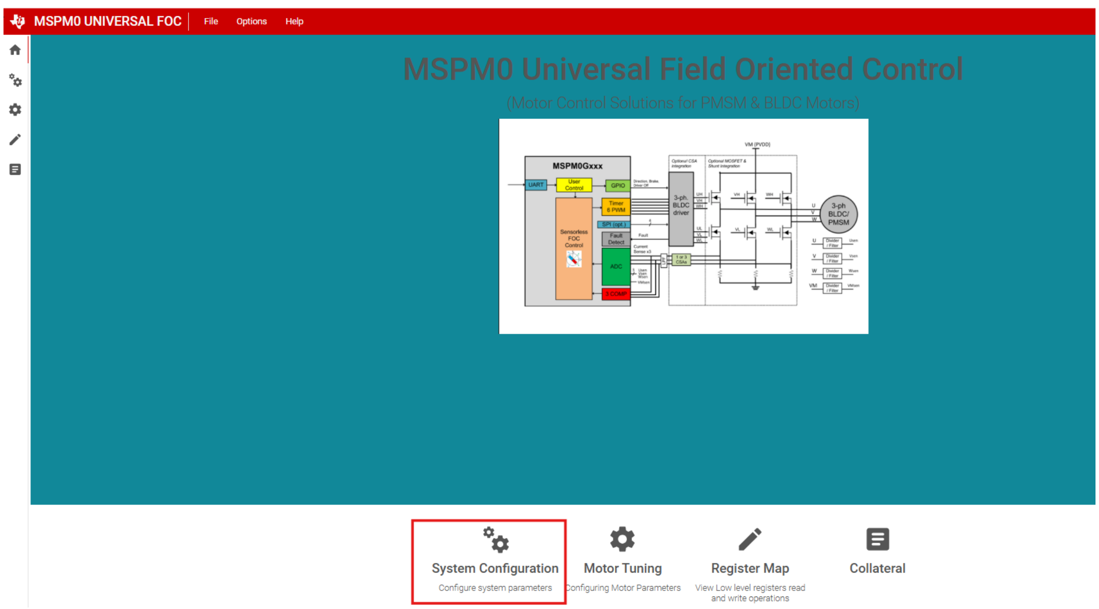


Figure 4-3. Option to Select System Configurations

4.4 Register Map

Register Map page contains the configurations for all the available Motor Tuning parameters that can be configured before starting the Motor. Register map page also contains the Status variables, which can be continuously monitored.

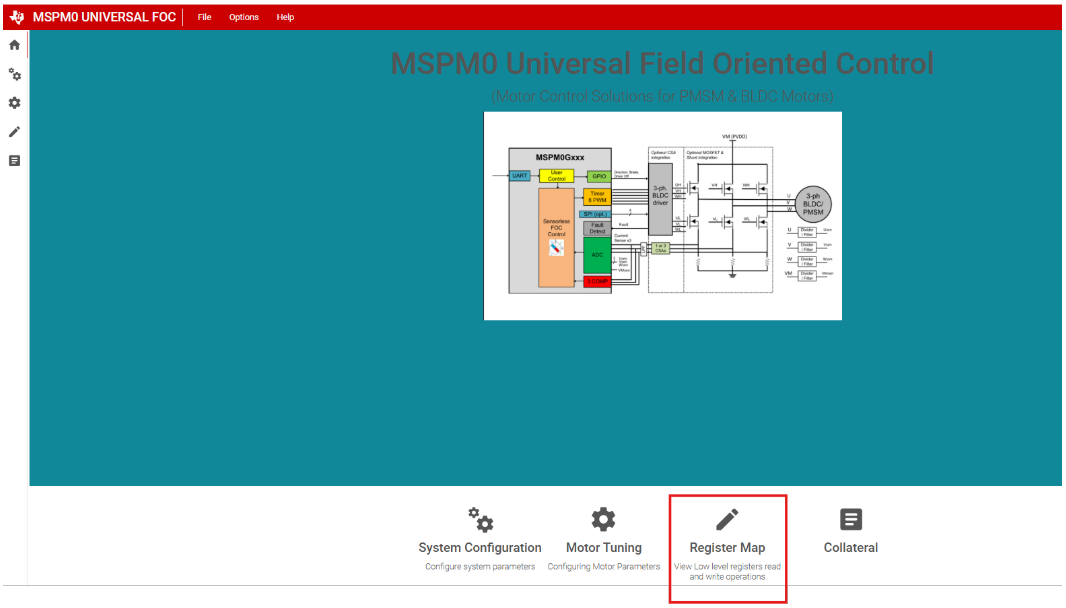


Figure 4-4. Option to Select Register Map page

4.5 Motor Tuning Page

User can set the speed command and monitor the motor status and fault variables from this window.

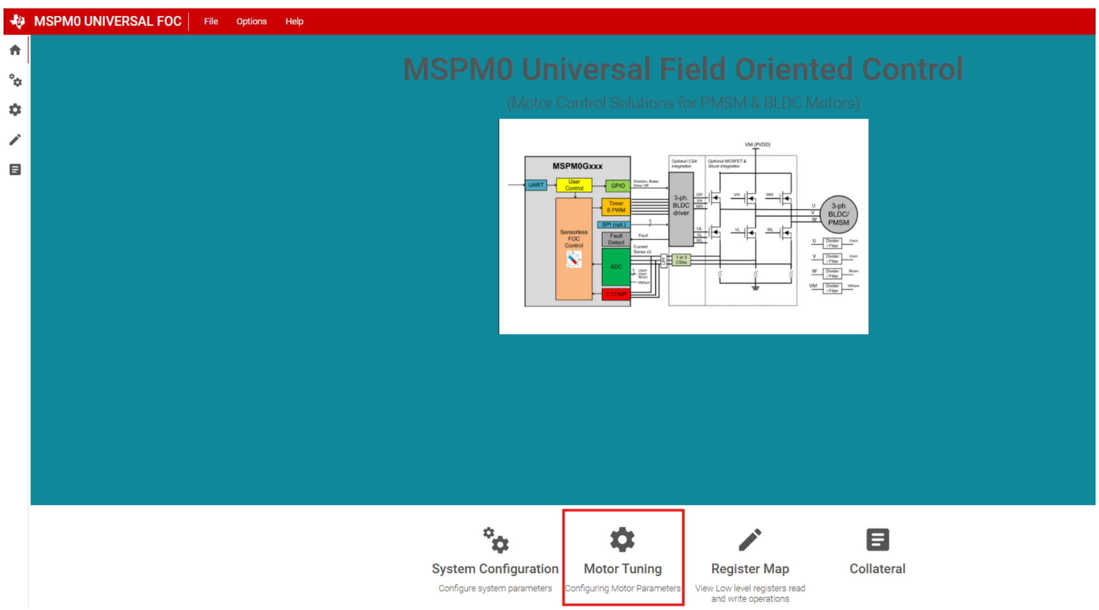


Figure 4-5. Option to Select Motor Tuning Page

4.6 Collateral Page

This page holds the links to various user guides to set up the software and migrate to different platforms.

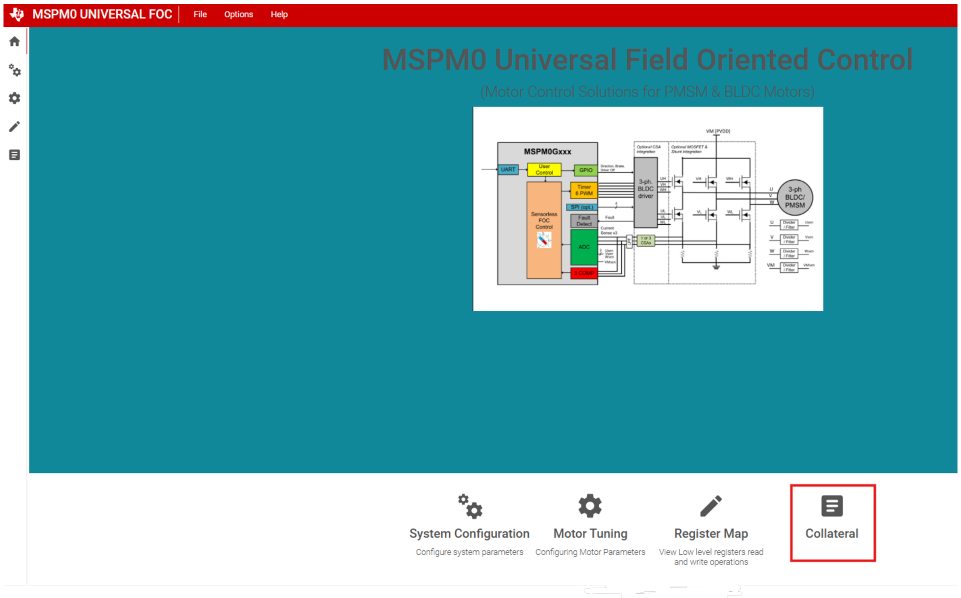


Figure 4-6. Option to Select Collaterals page

4.7 Loading and Saving Register Configurations

Once the desired tuning for a given motor is completed, the tuning configurations can be saved for future reference. In the top menu, click on the File → Save Register option as detailed below to save the configurations to downloads.

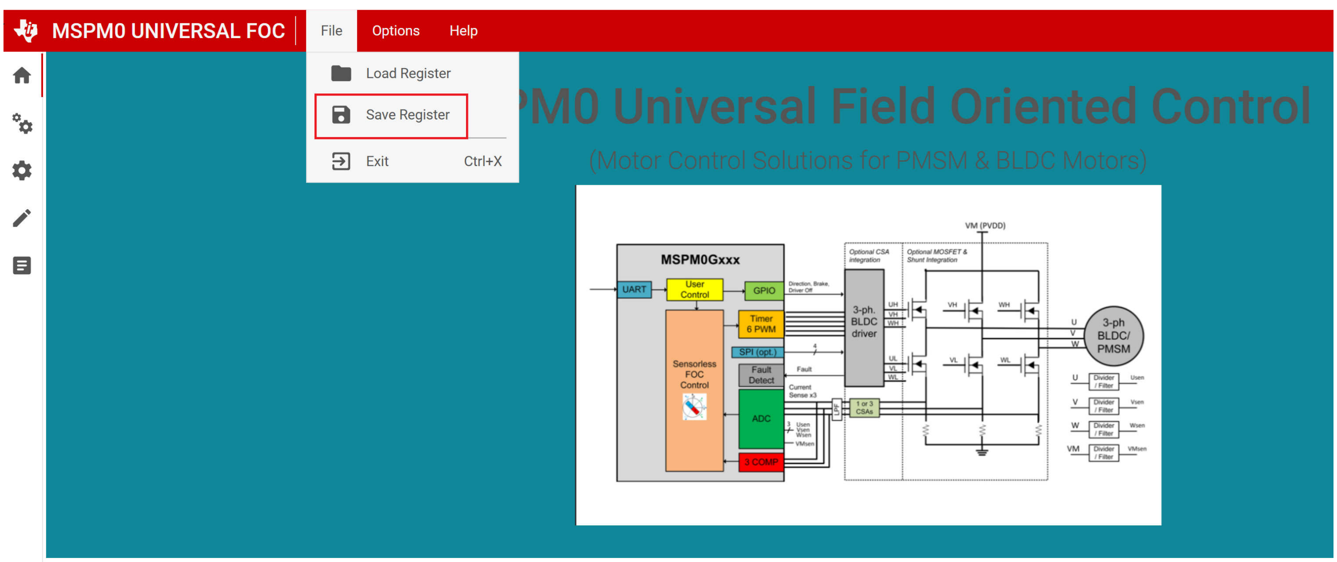


Figure 4-7. Option to Save Configuration Registers

Similarly, to load the tuned configurations which are saved previously, click in the menu options File → load Register to populate the configurations into the register space.

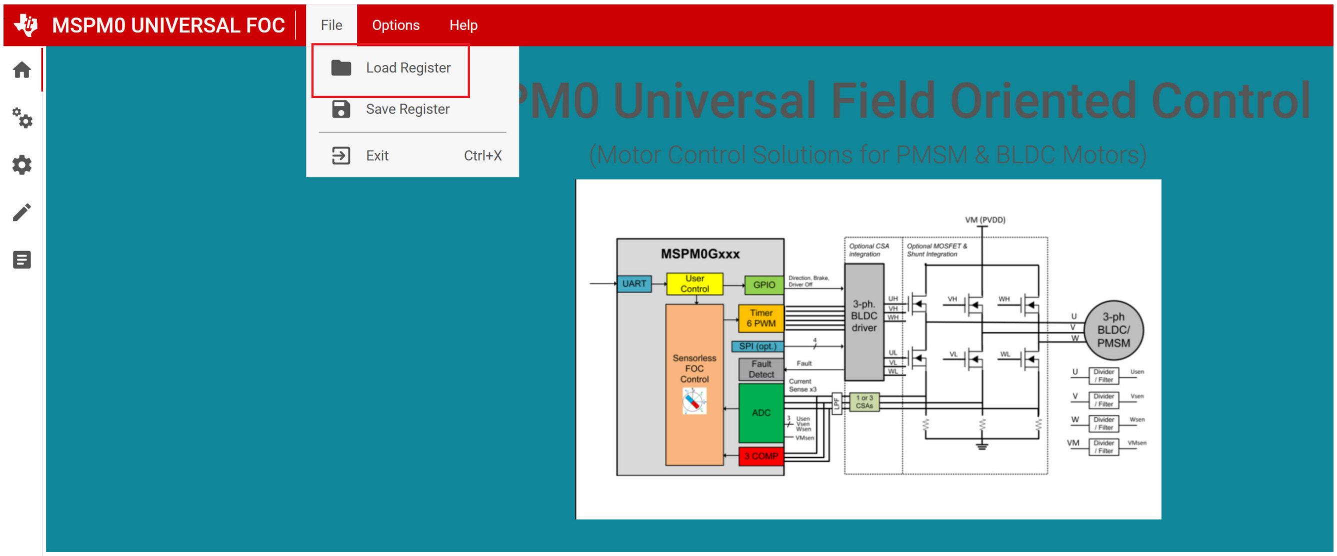


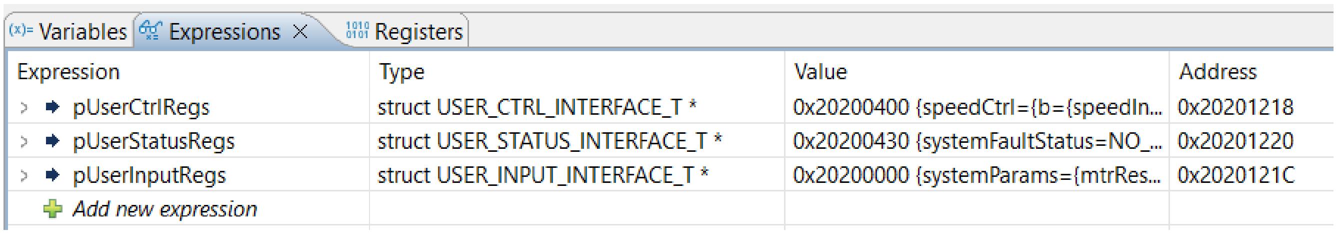
Figure 4-8. Option to Load Register Configurations

5 Register Map

Register map contains set of three register structures for Setting the Motor Control Tuning Parameters, Monitoring the Motor Status variables and Setting the Real-Time Control parameters using User Input registers, User Status registers and User Control Registers, respectively.

Real-time control of the FOC registers can be performed in three ways.

1. Import the structures into the expression window of CCS during the code debug as shown in Figure 5-1.



Expression	Type	Value	Address
pUserCtrlRegs	struct USER_CTRL_INTERFACE_T *	0x20200400 {speedCtrl={b={speedIn...	0x20201218
pUserStatusRegs	struct USER_STATUS_INTERFACE_T *	0x20200430 {systemFaultStatus=NO_...	0x20201220
pUserInputRegs	struct USER_INPUT_INTERFACE_T *	0x20200000 {systemParams={mtrRes...	0x2020121C
Add new expression			

Figure 5-1. Expressions for Input, Control and Status Registers in CCS Debug Mode

2. Read/Write the parameters over UART as described in UART_COMUNICATION_GUIDE
3. Control and monitor the variables using Universal FOC GUI.

The following sections describe registers and the variables associated with these structures.

5.1 Register Map Page in GUI

The above register variables can also be configured using GUI. Figure 5-2 details all the available user configurable registers available in the Universal FOC application. After connecting the GUI with the controller, click ***Read all*** option in the register page to reflect the default programmed parameters.

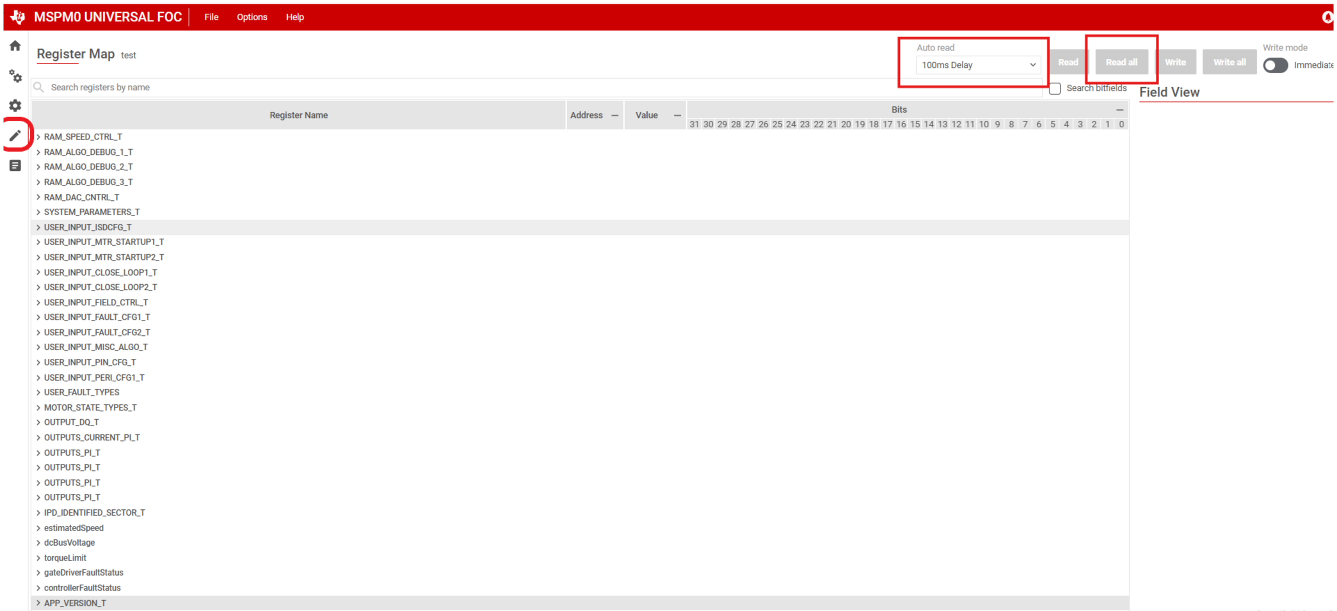


Figure 5-2. Register Map Page in GUI

5.2 User Control Registers (Base Address = 0x20200400h)

User Control Registers are set of user configurable parameters to control the Motor in real time.

These set of registers can be modified in the application code using pointer variable ***pUserCtrlRegs*** . Figure 5-3 shows the set of user Control registers as imported in CSS expression window.

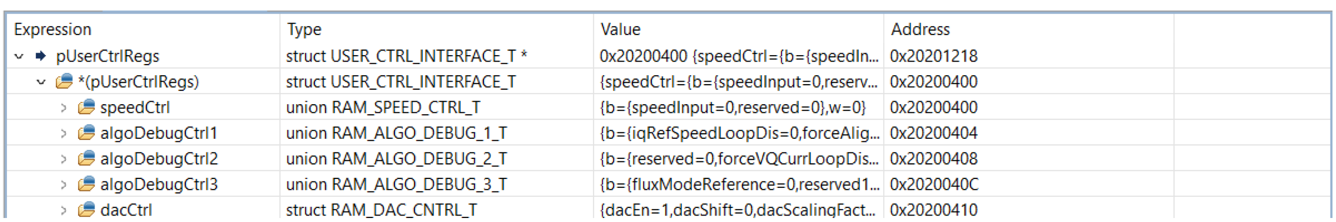


Figure 5-3. User Control Registers in CCS Debug Mode

Table 5-1. User Control registers

Offset	Acronym	Register Name	Section
0h	SPEED_CTRL	Speed Control Register	Section 5.2.1
4h	ALGO_DEBUG_CTRL1	Algorithm Debug Control 1 register	Section 5.2.2
8h	ALGO_DEBUG_CTRL2	Algorithm Debug Control 2 register	Section 5.2.3
Ch	ALGO_DEBUG_CTRL3	Algorithm Debug Control 3 register	Section 5.2.4
10h	DAC_CTRL	DAC Configuration and Control register	Section 5.2.5

Complex bit access types are encoded to fit into small table cells as shown in Table 5-2.

Table 5-2. Register Configuration Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

5.2.1 Speed Control Register (Offset = 0h) [Reset = 00000000h]

Table 5-3 shows the register to control Motor Speed.

Table 5-3. SPEED_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31 - 15	RESERVED	R	0h	Reserved
14-0	SPEED_CTRL	W	000000000000000b	Target Motor Speed/Torque value % of speed or Torque command × 32768

5.2.2 Algo Debug Control 1 Register (Offset = 4h) [Reset = 00000000h]

Table 5-4 shows the register to control Algorithm debug functions.

Table 5-4. Algorithm Debug Control 1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	CLEAR_FAULT	W	0b	Bit to clear set controller and Gate Driver Faults. Bit is automatically reset. 1h = Clear Fault Command
30-22	FORCED_ALIGN_ANGLE	W	000000000b	9-bit value (in °) used during forced align state (FORCE_ALIGN_EN = 1) Angle applied (°) = FORCED_ALIGN_ANGLE % 360°
21-16	RESERVED	R	0h	
15	CLOSED_LOOP_DIS	W	0b	Use to disable closed loop 0h = Enable closed loop 1h = Disable closed loop, motor commutation in open loop
14	FORCE_ALIGN_EN	W	0b	Force align state enable 0h = Disable force align state, device comes out of align state if MTR_STARTUP is selected as ALIGN or DOUBLE ALIGN 1h = Enable force align state, device stays in align state if MTR_STARTUP is selected as ALIGN or DOUBLE ALIGN
13	FORCE_SLOW_FIRST_CYCLE_EN	W	0b	Force slow first cycle enable 0h = Disable force slow first cycle state, device comes out of slow first cycle state if MTR_STARTUP is selected as SLOW FIRST CYCLE 1h = Enable force slow first cycle state, device stays in slow first cycle state if MTR_STARTUP is selected as SLOW FIRST CYCLE
12	FORCE_IPD_EN	W	0b	Force IPD enable 0h = Disable Force IPD state, device comes out of IPD state if MTR_STARTUP is selected as IPD 1h = Enable Force IPD state, device stays in IPD state if MTR_STARTUP is selected as IPD

Table 5-4. Algorithm Debug Control 1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
11	FORCE_ISD_EN	W	0b	Force ISD enable 0h = Disable Force ISD state, device comes out of ISD state if ISD_EN is set 1h = Enable Force ISD state, device stays in ISD state if ISD_EN is set
10	FORCE_ALIGN_ANGLE_SRC_SEL	W	0b	Force align angle state source select 0h = Force Align Angle defined by ALIGN_ANGLE 1h = Force Align Angle defined by FORCED_ALIGN_ANGLE
9-0	Reserved	R	0b	Reserved

5.2.3 Algo Debug Control 2 Register (Offset = 8h) [Reset = 00000000h]

Table 5-5 shows the register to control Algorithm Debug functions.

Table 5-5. Algorithm Debug Control 2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31 - 30	RESERVED	R	0h	Reserved
29	UPDATE_SYS_ PARAMETERS	W	1h	Dynamically updates System parameters every 200mS like PI gains of Speed/Torque loops to tune for required performance 0b = Dynamic System updates Disabled 1b = Dynamic System updates Enabled
28	UPDATE_CONFIGS	R	0b	This bit gives user the status of configuration updates. This bit is reset every 200mS when the tuning configurations are updated by algorithm and motor is not spinning. To make sure that User configurations are reflected in algorithm, user can set this bit after required tuning configurations are made and wait for this status bit to reset before giving the speed command.
27	STATUS_UPDATE_ENAB LE	W	0b	This bit enables the continuous update of user Status variables in real time.
26	CURRENT_LOOP_DIS	W	0b	Use to control the FORCE_VD_CURRENT_LOOP_DIS and FORCE_VQ_CURRENT_LOOP_DIS. If CURRENT_LOOP_DIS = 1b, current loop and speed loop are disabled 0h = Enable Current Loop 1h = Disable Current Loop
25-16	FORCE_VD_CURRENT_ LOOP_DIS	W-IQ(9)	0h	Sets Vd_ref in IQ(9) PU when current loop and speed loop are disabled If CURRENT_LOOP_DIS = 1b, then Vd is controlled using FORCE_VD_CURRENT_LOOP_DIS Vd_ref = (FORCE_VD_CURRENT_LOOP_DIS / 500) if FORCE_VD_CURRENT_LOOP_DIS < 500 - (FORCE_VD_CURRENT_LOOP_DIS - 512) / 500 if FORCE_VD_CURRENT_LOOP_DIS > 512 Valid values: 0 to 500 and 512 to 1000
15-6	FORCE_VQ_CURRENT_ LOOP_DIS	W-IQ(9)	0h	Sets Vq_ref in IQ(9) PU when current loop speed loop are disabled If CURRENT_LOOP_DIS = 1b, then Vq is controlled using FORCE_VQ_CURRENT_LOOP_DIS Vq_ref = (FORCE_VQ_CURRENT_LOOP_DIS / 500) if FORCE_VQ_CURRENT_LOOP_DIS < 500 - (FORCE_VQ_CURRENT_LOOP_DIS - 512) / 500 if FORCE_VQ_CURRENT_LOOP_DIS > 512 Valid values: 0 to 500 and 512 to 1000
5-0	RESERVED	R	0h	Reserved

5.2.4 Algo Debug Control 3 Register (Offset = Ch) [Reset = 00000000h]

Table 5-6 shows the register to control Algorithm Debug 3 functions.

Table 5-6. Algorithm Debug Control 3 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R	0h	
9-0	FLUX_MODE_REF	W-IQ(9)	0h	Sets Id_ref in IQ(9) PU when flux of the motor along D-axis is to be controlled Positive Id Control: (FLUX_MODE_REF/ 511) , if FLUX_MODE_REF < 512 Negative Id Control : -(FLUX_MODE_REF - 512) / 511 if FLUX_MODE_REF > 512 Valid values are 0 to 511 and 512 to 1000

5.2.5 DAC Configuration Register (Offset = 10h) [Reset = 00000000h]

DAC control registers defines configurations for monitoring the Real Time Algorithm and Hardware Register data on scope using the 12 bit DAC available on MSPM0G. For a detailed example on how to monitor an algorithm variable using DAC, see Table 5-7.

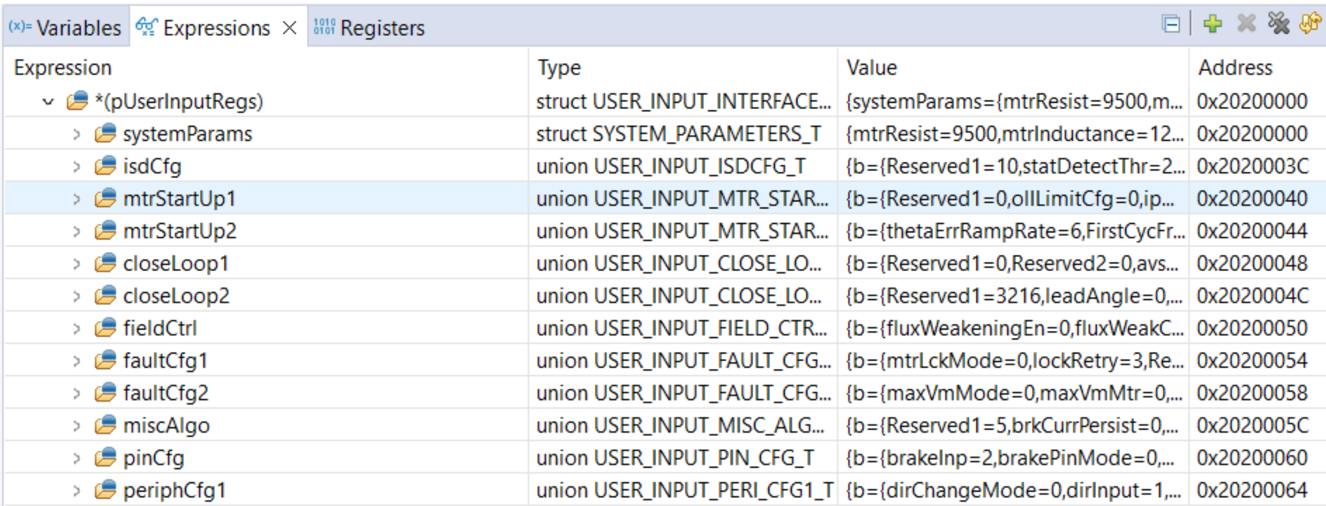
Table 5-7. DAC Configuration Registers

Variables	Type	Reset	Description
DAC_EN	Unsigned Short (RW)	0h	0h = Disable DAC 1h = Enable DAC
DAC_SHIFT	short (RW)	0h	+ve value specifies the number of left bit shifts before loading the value to 12-bit DAC register. -ve value specifies the number of right bit shifts before loading the value to 12-bit DAC register. DAC Shift is used for monitoring unsigned integer values and Registers
DAC_SCALING_FACTOR	int(RW)	0x00000000h	Non zero scaling factor is used for numbers represented in IQ format to be monitored in DAC. To monitor the Global IQ(27) format variables DAC scaling factor of _IQ(1.0) is used. To represent other IQx format variables, set DAC scaling factor to IQx/IQGlobal.
DACOUT_ADDRESS	unsigned int(RW)	0x00000000h	Defines the 32-bit aligned address of 32-bit variable that is to be monitored through DAC.

5.3 User Input Registers (Base Address = 0x20200000h)

User input registers are set of configurable registers to tune the motor performance in real time for various motor control features and save them in flash once required performance tuning is achieved.

Below are the set of Input Registers that can be imported in the CCS expression window using structure pointer ***pUserInputRegs.*** .



Variables | Expressions | Registers

Expression	Type	Value	Address
*(pUserInputRegs)	struct USER_INPUT_INTERFACE...	{systemParams={mtrResist=9500,m...	0x20200000
systemParams	struct SYSTEM_PARAMETERS_T	{mtrResist=9500,mtrInductance=12...	0x20200000
isdCfg	union USER_INPUT_ISDCFG_T	{b={Reserved1=10,statDetectThr=2...	0x2020003C
mtrStartUp1	union USER_INPUT_MTR_STAR...	{b={Reserved1=0,olILimitCfg=0,ip...	0x20200040
mtrStartUp2	union USER_INPUT_MTR_STAR...	{b={thetaErrRampRate=6,FirstCycFr...	0x20200044
closeLoop1	union USER_INPUT_CLOSE_LO...	{b={Reserved1=0,Reserved2=0,avs...	0x20200048
closeLoop2	union USER_INPUT_CLOSE_LO...	{b={Reserved1=3216,leadAngle=0,...	0x2020004C
fieldCtrl	union USER_INPUT_FIELD_CTR...	{b={fluxWeakeningEn=0,fluxWeakC...	0x20200050
faultCfg1	union USER_INPUT_FAULT_CFG...	{b={mtrLckMode=0,lockRetry=3,Re...	0x20200054
faultCfg2	union USER_INPUT_FAULT_CFG...	{b={maxVmMode=0,maxVmMtr=0,...	0x20200058
miscAlgo	union USER_INPUT_MISC_ALG...	{b={Reserved1=5,brkCurrPersist=0,...	0x2020005C
pinCfg	union USER_INPUT_PIN_CFG_T	{b={brakeInp=2,brakePinMode=0,...	0x20200060
periphCfg1	union USER_INPUT_PERI_CFG1_T	{b={dirChangeMode=0,dirInput=1,...	0x20200064

Figure 5-4. User Input Registers in CCS Debug Mode

Table 5-8. User Input Registers

Offset	Acronym	Register Name	Section
0h	SYSTEM_PARAMETERS	System Parameters	Section 5.3.1
3Ch	ISD_CFG	Initial Speed Detection Configuration	Section 5.3.2
40h	MOTOR_STARTUP1	Motor Startup 1 Configuration	Section 5.3.3
44h	MOTOR_STARTUP2	Motor Startup 2 Configuration	Section 5.3.4
48h	CLOSELOOP1	Close Loop1 Configuration	Section 5.3.5
4Ch	CLOSELOOP2	Close Loop2 Configuration	Section 5.3.6
50h	FLIED_CTRL	Flux Control Configuration	Section 5.3.7
54h	FAULT_CONFIG1	Fault Configuration 1	Section 5.3.8
58h	FAULT_CONFIG2	Fault Configuration 2	Section 5.3.9
5Ch	MISC_ALGO_CONFIG	Miscellaneous Algorithm Configuration	Section 5.3.10
60h	PIN_CONFIGURATION	Pin Configuration	Section 5.3.11
64h	PERI_CONFIG	Peripheral Configuration	Section 5.3.12

Complex bit access types are encoded to fit into small table cells as shown in Table 5-9.

Table 5-9. Register Configuration Access Type Codes

Access Type	Code	Description
Read Type		
R	R	Read
Write Type		
W	W	Write
Reset or Default Value		
-n		Value after reset or the default value

5.3.1 SYSTEM_PARAMETERS (Offset = 0h)

Set of basic system configuration parameters essential for motor control system functionality.

Table 5-10. Motor Resistance Configuration Registers (Offset = 0h)

Bit	Field	Type	Reset	Description
31-0	MTR_RESISTANCE	R/W	0000h	Motor Resistance in milliohms

Table 5-11. Motor Inductance Configuration (Offset = 4h)

Bit	Field	Type	Reset	Description
31-0	MTR_INDUCTANCE	R/W	0000h	Motor Inductance in microhenry. For Salient pole motors (Lq + Ld)/2

Table 5-12. Motor Saliency Configuration (Offset = 8h)

Bit	Field	Type	Reset	Description
31-0	MTR_SALIENCY	R/W	0.0(Float)	Saliency of Motor (Lq-Ld)/(Lq+Ld) in float.

Table 5-13. Motor BEMF Constant Configuration (Offset = Ch)

Bit	Field	Type	Reset	Description
31-0	MTR_BEMF_CONSTANT	R/W	0000h	Motor BEMF constant in mV/Hz × 10.

Table 5-14. Base Voltage Configuration (Offset = 10h)

Bit	Field	Type	Reset	Description
31-0	VOLTAGE_BASE	R/W	0.0(Float)	Base voltage of the board calculatedas the maximum measurable voltage detailed inEquation 7 MAX_DC_VOLTAGE/sqrt(3)

Table 5-15. Base Current Configuration (Offset = 14h)

Bit	Field	Type	Reset	Description
31-0	CURRENT_BASE	R/W	0.0(Float)	Base current of the board calculated based on the CSA gain in as (1.65V - ADC OffsetVoltage/ CSA Gain in volts/amp) in amps. 1.65V is the reference mid point voltage of the ADC for bidirectional current sensing. 0.4125 is the offset Voltage in DRV8329 If the CSA gain is in V/V , multiply with current sense resistor value in ohms to compute CSA gain in volts/amp

Table 5-16. Motor Max Speed Configuration (Offset = 18h)

Bit	Field	Type	Reset	Description
31-0	MOTOR_MAX_SPEED	R/W	0.0(Float)	Rated motor speed in Hz from the data sheet

Table 5-17. Motor Max Power Configuration (Offset = 1Ch)

Bit	Field	Type	Reset	Description
31-0	MOTOR_MAX_POWER	R/W	0.0(Float)	Rated motor power in Hz from the data sheet

Table 5-18. Speed Loop Proportional Gain (Offset = 20h)

Bit	Field	Type	Reset	Description
31-0	SPEED_POWER_LOOP_KP	R/W	0.0(Float)	Proportional gain for the closed loop speed control /Power Loop Control in float

Table 5-19. Speed Loop Integral Gain (Offset = 24h)

Bit	Field	Type	Reset	Description
31-0	SPEED_POWER_LOOP_KI	R/W	0.0(Float)	Integral gain for the closed loop speed control /Power Loop Control in float

Table 5-20. Torque Loop Proportional Gain (Offset = 28h)

Bit	Field	Type	Reset	Description
31-0	CURR_LOOP_KP	R/W	0.0(Float)	Proportional gain for the closed loop torque control in float

Table 5-21. Torque Loop Integral Gain (Offset = 2Ch)

Bit	Field	Type	Reset	Description
31-0	CURR_LOOP_KI	R/W	0.0(Float)	Integral gain for the closed loop torque control in float

Table 5-22. Flux Weakening Controller Proportional Gain (Offset = 30h)

Bit	Field	Type	Reset	Description
31-0	FLUX_WEAK_KI	R/W	0.0(Float)	Proportional gain for the Flux weakening control in float

Table 5-23. Flux Weakening Controller Integral Gain (Offset = 34h)

Bit	Field	Type	Reset	Description
31-0	FLUX_WEAK_KP	R/W	0.0(Float)	Integral gain for the Flux weakening control in float

Table 5-24. Sliding Control Gain for ESMO Observer(Offset = 38h)

Bit	Field	Type	Reset	Description
31-0	KSLIDE	R/W	0.0(Float)	Sliding Gain for the ESMO ESTIMATOR

5.3.2 ISD_CONFIG Register (Offset = 3Ch) [Reset = 00000000h]

Table 5-25 shows the register to configure Initial Speed Detection.

Table 5-25. ISD_CONFIG Register

Bit	Field	Type	Reset	Description
31-30	Reserved	R	00b	Reserved
29	ISD_EN	R/W	0b	ISD Enable 0h = Disable 1h = Enable
28	BRAKE_EN	R/W	0b	Brake enable 0h = Disable 1h = Enable
27	HIZ_EN	R/W	0b	Hi-Z enable 0h = Disable 1h = Enable
26	RVS_DR_EN	R/W	0b	Reverse drive enable 0h = Disable 1h = Enable
25	RESYNC_EN	R/W	0b	Resynchronization Enable 0h = Disable 1h = Enable

Table 5-25. ISD_CONFIG Register (continued)

Bit	Field	Type	Reset	Description
24-21	FW_DRV_RESYN_THR	R/W	0h	Minimum Speed threshold to resynchronize to close loop (% of MAX_SPEED) 0h = 5% 1h = 10% 2h = 15% 3h = 20% 4h = 25% 5h = 30% 6h = 35% 7h = 40% 8h = 45% 9h = 50% Ah = 55% Bh = 60% Ch = 70% Dh = 80% Eh = 90% Fh = 100%
20	BRK_CONFIG	R/W	0b	Brake configuration 0h = Brake time is used to come out of brake state 1h = Brake current threshold is used to come out of brake state
16-19	BRK_TIME	R/W	0b	Brake time 0h = 10ms 1h = 50ms 2h = 100ms 3h = 200ms 4h = 300ms 5h = 400ms 6h = 500ms 7h = 750ms 8h = 1s 9h = 2s Ah = 3s Bh = 4s Ch = 5s Dh = 7.5s Eh = 10s Fh = 15s

Table 5-25. ISD_CONFIG Register (continued)

Bit	Field	Type	Reset	Description
15-12	HIZ_TIME	R/W	0b	Hi-Z time 0h = 10ms 1h = 50ms 2h = 100ms 3h = 200ms 4h = 300ms 5h = 400ms 6h = 500ms 7h = 750ms 8h = 1s 9h = 2s Ah = 3s Bh = 4s Ch = 5s Dh = 7.5s Eh = 10s Fh = 15s
11-9	STAT_DETECT_THR	R/W	000b	BEMF threshold to detect if motor is stationary 0h = 50mV 1h = 75mV 2h = 100mV 3h = 250mV 4h = 500mV 5h = 750mV 6h = 1000mV 7h = 1500mV
0-9	Reserved	R	0b	Reserved

5.3.3 MOTOR_STARTUP1 Register (Offset = 40h) [Reset = 00000000h]

Table 5-26 shows the register to configure motor startup settings1.

Table 5-26. MOTOR_STARTUP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	MTR_STARTUP_OPTION	R/W	00b	Motor start-up method 0h = Align 1h = Double align 2h = IPD 3h = Slow first cycle
29-26	ALIGN_SLOW_RAMP_RATE	R/W	0h	Align, slow first cycle and open loop current ramp rate 0h = 0.1A/s 1h = 1A/s 2h = 5A/s 3h = 10A/s 4h = 15A/s 5h = 25A/s 6h = 50A/s 7h = 100A/s 8h = 150A/s 9h = 200A/s Ah = 250A/s Bh = 500A/s Ch = 1000A/s Dh = 2000A/s Eh = 5000A/s Fh = No Limit A/s
25-22	ALIGN_TIME	R/W	0h	Align time 0h = 10ms 1h = 50ms 2h = 100ms 3h = 200ms 4h = 300ms 5h = 400ms 6h = 500ms 7h = 750ms 8h = 1s 9h = 1.5s Ah = 2s Bh = 3s Ch = 4s Dh = 5s Eh = 7.5s Fh = 10s

Table 5-26. MOTOR_STARTUP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-17	ALIGN_OR_SLOW_CURRENT_ILIMIT	R/W	00h	Current limit during Align/Slow First Cycle in % of CURRENT_BASE 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
16-14	IPD_CLK_FREQ	R/W	000b	IPD clock frequency 0h = 50Hz 1h = 100Hz 2h = 250Hz 3h = 500Hz 4h = 1000Hz 5h = 2000Hz 6h = 5000Hz 7h = 10000Hz
13-7	IPD_CURR_THR	R/W	0h	7 bit value for IPD current limit × CURRENT_BASE / 2^7
6	Reserved	R	0b	Reserved
5-4	IPD_ADV_ANGLE	R/W	00b	IPD advance angle 0h = 0° 1h = 30° 2h = 60° 3h = 90°

Table 5-26. MOTOR_STARTUP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-2	IPD_REPEAT	R/W	00b	Number of times IPD is executed 0h = 1 time 1h = average of 2 times 2h = average of 3 times 3h = average of 4 times
1	OL_ILIMIT_CONFIG	R/W	0b	Open loop current limit configuration 0h = Open loop current limit defined by OL_ILIMIT 1h = Open loop current limit defined by ILIMIT
0	Reserved	R	0b	Reserved

5.3.4 MOTOR_STARTUP2 Register (Offset = 44h) [Reset = 00000000h]

Table 5-27 shows the register to configure motor startup settings2.

Table 5-27. MOTOR_STARTUP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	OL_ILIMIT	R/W	0h	Open loop current limit in % of CURRENT_BASE 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%

Table 5-27. MOTOR_STARTUP2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26-23	OL_ACC_A1	R/W	0h	Open loop acceleration coefficient A1 0h = 0.01Hz/s 1h = 0.05Hz/s 2h = 1Hz/s 3h = 2.5Hz/s 4h = 5Hz/s 5h = 10Hz/s 6h = 25Hz/s 7h = 50Hz/s 8h = 75Hz/s 9h = 100Hz/s Ah = 250Hz/s Bh = 500Hz/s Ch = 750Hz/s Dh = 1000Hz/s Eh = 5000Hz/s Fh = 10000Hz/s
22-19	OL_ACC_A2	R/W	0h	Open loop acceleration coefficient A2 0h = 0.0Hz/s2 1h = 0.05Hz/s2 2h = 1Hz/s2 3h = 2.5Hz/s2 4h = 5Hz/s2 5h = 10Hz/s2 6h = 25Hz/s2 7h = 50Hz/s2 8h = 75Hz/s2 9h = 100Hz/s2 Ah = 250Hz/s2 Bh = 500Hz/s2 Ch = 750Hz/s2 Dh = 1000Hz/s2 Eh = 5000Hz/s2 Fh = 10000Hz/s2
18	RESERVED	R	0h	Reserved

Table 5-27. MOTOR_STARTUP2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-13	OPN_CL_HANDOFF_THR	R/W	0h	Open to close loop handoff threshold (% of MAX_SPEED) 0h = 1% 1h = 2% 2h = 3% 3h = 4% 4h = 5% 5h = 6% 6h = 7% 7h = 8% 8h = 9% 9h = 10% Ah = 11% Bh = 12% Ch = 13% Dh = 14% Eh = 15% Fh = 16% 10h = 17% 11h = 18% 12h = 19% 13h = 20% 14h = 22.5% 15h = 25% 16h = 27.5% 17h = 30% 18h = 32.5% 19h = 35% 1Ah = 37.5% 1Bh = 40% 1Ch = 42.5% 1Dh = 45% 1Eh = 47.5% 1Fh = 50%

Table 5-27. MOTOR_STARTUP2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12-8	ALIGN_ANGLE	R/W	0h	Align angle 0h = 0° 1h = 10° 2h = 20° 3h = 30° 4h = 45° 5h = 60° 6h = 70° 7h = 80° 8h = 90° 9h = 110° Ah = 120° Bh = 135° Ch = 150° Dh = 160° Eh = 170° Fh = 180° 10h = 190° 11h = 210° 12h = 225° 13h = 240° 14h = 250° 15h = 260° 16h = 270° 17h = 280° 18h = 290° 19h = 315° 1Ah = 330° 1Bh = 340° 1Ch = 350° 1Dh = N/A 1Eh = N/A 1Fh = N/A
7-4	SLOW_FIRST_CYC_FREQ	R/W	0h	Frequency of first cycle in close loop startup (% of MAX_SPEED) 0h = 1% 1h = 2% 2h = 3% 3h = 5% 4h = 7.5% 5h = 10% 6h = 12.5% 7h = 15% 8h = 17.5% 9h = 20% Ah = 25% Bh = 30% Ch = 35% Dh = 40% Eh = 45% Fh = 50%
3	FIRST_CYCLE_FREQ_SEL	R/W	0h	First cycle frequency in open loop for align, double align and IPD startup options 0h = Defined by SLOW_FIRST_CYC_FREQ 1h = 0Hz

Table 5-27. MOTOR_STARTUP2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
2-0	THETA_ERROR_RAMP_ RATE	R/W	0h	Ramp rate for reducing difference between estimated theta and open loop theta 0h = 0.01 deg/ms 1h = 0.05 deg/ms 2h = 0.1 deg/ms 3h = 0.15 deg/ms 4h = 0.2 deg/ms 5h = 0.5 deg/ms 6h = 1 deg/ms 7h = 2 deg/ms

5.3.5 CLOSED_LOOP1 Register (Offset = 48h) [Reset = 00000000h]

Table 5-28 shows the register to configure close loop settings1.

Table 5-28. CLOSED_LOOP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R/W	0h	Reserved
29-28	CONTROL_MODE	R/W	0h	FOC Closed loop Mode of operation 0h = Closed Loop Speed Control 1h = Closed Loop Power Control 2h = Closed Loop Torque Control 3h = Voltage Control mode.
27	HIGH_FREQ_FOC_EN	R/W	0b	Enable /Disable High FOC Sampling rate. Higher the Sampling rate, lower the CPU bandwidth available for other tasks. 0h = High Frequency FOC Enable.(Max FOC Frequency 16KHz) 1h = High Frequency FOC Disable(Max FOC Frequency 8KHz)

Table 5-28. CLOSED_LOOP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
26-22	ILIMIT	R/W	0h	Current limit in Closed loop Torque Mode and Closed loop Speed control in % of CURRENT_BASE 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
21-20	MTR_STOP	R/W	00b	Motor stop method 0h = Hi-z 1h = Active spin down 2h = Braking 3h = Reserved
19	OVERMODULATION_ ENABLE	R/W	0b	Overmodulation enable 0h = Disable Over Modulation 1h = Enable Over Modulation

Table 5-28. CLOSED_LOOP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
18-14	CL_ACC	R/W	0h	Closed loop acceleration 0h = 0.5Hz/s 1h = 1Hz/s 2h = 2.5Hz/s 3h = 5Hz/s 4h = 7.5Hz/s 5h = 10Hz/s 6h = 20Hz/s 7h = 40Hz/s 8h = 60Hz/s 9h = 80Hz/s Ah = 100Hz/s Bh = 200Hz/s Ch = 300Hz/s Dh = 400Hz/s Eh = 500Hz/s Fh = 600Hz/s 10h = 700Hz/s 11h = 800Hz/s 12h = 900Hz/s 13h = 1000Hz/s 14h = 2000Hz/s 15h = 4000Hz/s 16h = 6000Hz/s 17h = 8000Hz/s 18h = 10000Hz/s 19h = 20000Hz/s 1Ah = 30000Hz/s 1Bh = 40000Hz/s 1Ch = 50000Hz/s 1Dh = 60000Hz/s 1Eh = 70000Hz/s 1Fh = No limit
13	CL_DEC_CONFIG	R/W	0h	Closed loop deceleration configuration 0h = Closed loop deceleration defined by CL_DEC 1h = Closed loop deceleration defined by CL_ACC

Table 5-28. CLOSED_LOOP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
12-8	CL_DEC	R/W	0h	Closed loop deceleration. This register is used only if AVS is disabled and CL_DEC_CONFIG is set to '0' 0h = 0.5Hz/s 1h = 1Hz/s 2h = 2.5Hz/s 3h = 5Hz/s 4h = 7.5Hz/s 5h = 10Hz/s 6h = 20Hz/s 7h = 40Hz/s 8h = 60Hz/s 9h = 80Hz/s Ah = 100Hz/s Bh = 200Hz/s Ch = 300Hz/s Dh = 400Hz/s Eh = 500Hz/s Fh = 600Hz/s 10h = 700Hz/s 11h = 800Hz/s 12h = 900Hz/s 13h = 1000Hz/s 14h = 2000Hz/s 15h = 4000Hz/s 16h = 6000Hz/s 17h = 8000Hz/s 18h = 10000Hz/s 19h = 20000Hz/s 1Ah = 30000Hz/s 1Bh = 40000Hz/s 1Ch = 50000Hz/s 1Dh = 60000Hz/s 1Eh = 70000Hz/s 1Fh = No limit
7-8	PWM_FREQ_OUT	R/W	0h	Output PWM switching frequency 0h = 5kHz 1h = 10kHz 2h = 16kHz 3h = 20kHz 4h = 25kHz 5h = 32kHz 6h = 40kHz 7h = 48kHz 8h = 50kHz 9h = 64kHz Ah = 80kHz Bh = N/A Ch = N/A Dh = N/A Eh = N/A Fh = N/A
14	PWM_MODE	R/W	0b	PWM modulation 0h = Continuous Space Vector Modulation 1h = Discontinuous Space Vector Modulation

Table 5-28. CLOSED_LOOP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	AVS_EN	R/W	0b	AVS enable 0h = Disable 1h = Enable
2	RESERVED	R	0b	Reserved
1	SPEED_LOOP_DIS	R/W	0b	Speed loop disable 0h = Enable 1h = Disable

5.3.6 CLOSED_LOOP2 Register (Offset = 4Ch) [Reset = 00000000h]

Table 5-29 shows the register to configure close loop settings2.

Table 5-29. CLOSED_LOOP2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-28	ACT_SPIN_THR	R/W	0h	Speed threshold for active spin down (% of MAX_SPEED) 0h = 100% 1h = 90% 2h = 80% 3h = 70% 4h = 60% 5h = 50% 6h = 45% 7h = 40% 8h = 35% 9h = 30% Ah = 25% Bh = 20% Ch = 15% Dh = 10% Eh = 5% Fh = 2.5%
27-24.	BRAKE_SPEED_ THRESHOLD	R/W	0h	Speed threshold for BRAKE pin and motor stop options (Low Side Braking or align braking) (% of MAX_SPEED) 0h = 100% 1h = 90% 2h = 80% 3h = 70% 4h = 60% 5h = 50% 6h = 45% 7h = 40% 8h = 35% 9h = 30% Ah = 25% Bh = 20% Ch = 15% Dh = 10% Eh = 5% Fh = 2.5%

Table 5-29. CLOSED_LOOP2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
23-19	BRK_CURR_THR	R/W	0h	Brake current limit in % of CURRENT_BASE 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
18-14	LEAD_ANGLE	R/W	0h	Lead Angle in degrees applied in Voltage Control Mode 0 - 15 = 1 * Bit Value 15 - 31 = 2 *(Bit Value -15) + 15
13-0	RESERVED	R/W	0h	Reserved

5.3.7 FIELD_CTRL Register (Offset = 50h) [Reset = 00000000h]

Table 5-30 shows the register to configure Flux Control settings.

Table 5-30. FIELD_CTRL Register Bit Descriptions

Bit	Field	Type	Reset	Description
31-7	Reserved	R	0h	Reserved
6	MTPA_EN	R/W	0b	Enable/Disable Maximum Torque Per Ampere Control (MTPA) 0h = Disable MTPA 1h = Enable MTPA
5-4	FLUX_WEAK_REF	R/W	00b	Modulation Index Reference to be tracked in Flux Weakening mode 0h = 70% 1h = 80% 2h = 90% 3h = 95%
3-1	FLUX_WEAK_CURR_RATIO	R/W	000b	Max value of Flux Weakening Current Reference as % of ILIMIT 0h = Only Circular Limit in Place 1h = 80% 2h = 70% 3h = 60% 4h = 50% 5h = 40% 6h = 30% 7h = 20%
0	FLUX_WEAK_EN	R/W	0b	Enable/Disable Flux Weakening Control (MTPA) 0h = Disable Flux Weakening 1h = Enable Flux Weakening

5.3.8 FAULT_CONFIG1 Register (Offset = 54h) [Reset = 00000000h]

Table 5-31 shows the register to configure fault settings1.

Table 5-31. FAULT_CONFIG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R/W	0h	Reserved
5-2	LCK_RETRY	R/W	0h	Lock detection retry time 0h = 100ms 1h = 500ms 2h = 1s 3h = 2s 4h = 3s 5h = 4s 6h = 5s 7h = 6s 8h = 7s 9h = 8s Ah = 9s Bh = 10s Ch = 11s Dh = 12s Eh = 13s Fh = 14s

Table 5-31. FAULT_CONFIG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
1-0	MTR_LCK_MODE	R/W	00b	Motor Lock Mode 0h = Motor lock detection causes latched fault; nFAULT active; 1h = Fault automatically cleared after LCK_RETRY time. 2h = Motor lock in report only mode. 3h = Motor lock detection is disabled

5.3.9 FAULT_CONFIG2 Register (Offset = 58h) [Reset = 00000000h]

Table 5-32 shows the register to configure fault settings2.

Table 5-32. FAULT_CONFIG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R/W	0h	Reserved
26	LOCK1_EN	R/W	0b	Lock 1 : Abnormal speed enable 0h = Disable 1h = Enable
25	LOCK2_EN	R/W	0b	Lock 2 : Abnormal BEMF enable 0h = Disable 1h = Enable
24	LOCK3_EN	R/W	0b	Lock 3 : No motor enable 0h = Disable 1h = Enable
23-21	LOCK_ABN_SPEED	R/W	000b	Abnormal speed lock threshold (% of MAX_SPEED) 0h = 130% 1h = 140% 2h = 150% 3h = 160% 4h = 170% 5h = 180% 6h = 190% 7h = 200%
20-18	ABNORMAL_BEMF_THR	R/W	000b	Abnormal BEMF lock threshold (% of estimated BEMF w.r.t Vdc below which the Abnormal BEMF fault is triggered) 0h = 1% 1h = 2% 2h = 3% 3h = 5% 4h = 8% 5h = 10% 6h = 12% 7h = 15%

Table 5-32. FAULT_CONFIG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-13	NO_MTR_THR	R/W	00000b	No Motor current limit in % of CURRENT_BASE 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
12-8	RESERVED	R/W	0h	Reserved.
7-5	MIN_VM_MOTOR	R/W	000b	Minimum voltage for running motor in % of BASE_VOLTAGE 0h = No Limit 1h = 5% 2h = 10% 3h = 12% 4h = 15% 5h = 18% 6h = 20% 7h = 25%
4	MIN_VM_MODE	R/W	0b	Undervoltage fault mode 0h = Latch on Undervoltage 1h = Automatic clear if voltage in bounds

Table 5-32. FAULT_CONFIG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-1	MAX_VM_MOTOR	R/W	000b	Maximum voltage for running motor in % of BASE_VOLTAGE 0h = 60% 1h = 65% 2h = 70% 3h = 75% 4h = 80% 5h = 85% 6h = 90% 7h = Max Voltage
0	MAX_VM_MODE	R/W	0b	Overvoltage fault mode 0h = Latch on Overvoltage 1h = Automatic clear if voltage in bounds

5.3.10 MISC_ALGO Register (Offset = 5Ch) [Reset = 00000000h]

Table 5-33 shows the register to multiple miscellaneous Algorithm Configuration.

Table 5-33. MISC_ALGO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-22	RESERVED	R/W	0h	Reserved
21-20	IPD_MAX_OVERFLOW	R/W	00b	Maximum Number of Timer Overflows before hitting the IPD Time Out Fault with 16 bit Timer running at 80Mhz 0b = 5 Overflows (4mS) 1b = 10 Overflows (8mS) 10b = 20 Overflows (16mS) 11b = 40 Overflows (32mS)
19-16	CL_SLOW_ACC	R/W	0h	Close loop acceleration when estimator is not yet fully aligned 0h = 0.1Hz/s 1h = 1Hz/s 2h = 2Hz/s 3h = 3Hz/s 4h = 5Hz/s 5h = 10Hz/s 6h = 20Hz/s 7h = 30Hz/s 8h = 40Hz/s 9h = 50Hz/s Ah = 100Hz/s Bh = 200Hz/s Ch = 500Hz/s Dh = 750Hz/s Eh = 1000Hz/s Fh = 2000Hz/s
15-14	RESERVED	R	0b	Reserved
13-12	ISD_STOP_TIME	R/W	00b	Persistence time for declaring motor has stopped 0h = 1ms 1h = 5ms 2h = 50ms 3h = 100ms
11-10	ISD_RUN_TIME	R/W	00b	Persistence time for declaring motor is running 0h = 1ms 1h = 5ms 2h = 50ms 3h = 100ms

Table 5-33. MISC_ALGO Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
9-5	RESERVED	R	0b	Reserved
4-3	BRAKE_CURRENT_ PERSIST	R/W	00b	Persistence time for current below threshold during brake 0h = 50ms 1h = 100ms 2h = 250ms 3h = 500ms
2-0	RESERVED	R	0b	Reserved

5.3.11 PIN_CONFIG Register (Offset = 60h) [Reset = 00000000h]

Table 5-34 shows the register to configure hardware pins.

Table 5-34. PIN_CONFIG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	RESERVED	R/W	0h	Reserved
19	VDC_FILT_DIS	R/W	0b	Vdc Filter Disable 0h = Enabled 1h = Disabled
18-3	RESERVED	R/W	0h	Reserved
2	BRAKE_PIN_MODE	R/W	0b	Brake pin mode 0h = Low side Brake 1h = Align Brake
1-0	BRAKE_INPUT	R/W	00b	Brake pin override 0h = Hardware Pin BRAKE 1h = Override pin and brake / align according to BRAKE_PIN_MODE 2h = Override pin and do not brake / align 3h = Hardware Pin BRAKE

5.3.12 PERI_CONFIG Register (Offset = 64h) [Reset = 00000000h]

Table 5-35 shows the register to peripheral.

Table 5-35. PERI_CONFIG1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-15	RESERVED	R	0h	Reserved
14-9	MCU_DEAD_TIME	R/W	0h	Dead time applied between the High Side and Low side switches = 50ns × MCU_DEAD_TIME
8-4	BUS_CURRENT_LIMIT	R/W	00000b	Bus Current Limit in % of CURRENT_BASE 0h = 7.5% 1h = 8.0% 2h = 8.5% 3h = 9.0% 4h = 9.5% 5h = 10% 6h = 11% 7h = 12% 8h = 13% 9h = 14% Ah = 15% Bh = 16% Ch = 17% Dh = 18% Eh = 20% Fh = 22.5% 10h = 25% 11h = 27.5% 12h = 30% 13h = 35% 14h = 40% 15h = 45% 16h = 50% 17h = 55% 18h = 60% 19h = 70% 1Ah = 75% 1Bh = 80% 1Ch = 85% 1Dh = 90% 1Eh = 95% 1Fh = 100%
3	BUS_CURRENT_LIMIT_ ENABLE	R/W	0b	Bus current limit enable 0h = Disable 1h = Enable
2-1	DIR_INPUT	R/W	00b	DIR pin override 0h = Hardware Pin DIR 1h = Override DIR pin with clockwise rotation OUTA-OUTB-OUTC 2h = Override DIR pin with counter clockwise rotation OUTA-OUTC-OUTB 3h = Hardware Pin DIR

Table 5-35. PERI_CONFIG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	DIR_CHANGE_MODE	R/W	0b	Response to change of DIR pin status 0h = Follow motor stop options and ISD routine on detecting DIR change 1h = Change the direction through Reverse Drive while continuously driving the motor

5.4 User Status Registers (Base Address = 0x20200430h)

User Status Registers are set of consolidated variables available for user to read the Motor status and analyze the control performance.

Figure 5-5 shows the set of Status Registers that can be imported in the CCS expression window using structure pointer ***pUserStatusRegs***.

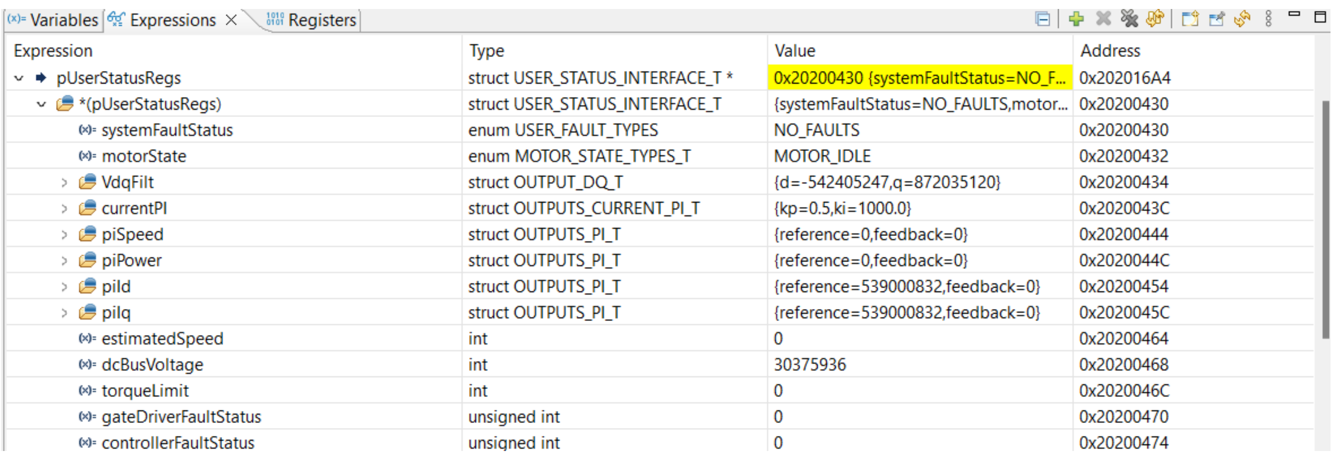


Figure 5-5. User Status Registers in CCS Debug Mode

Table 5-36 lists the definitions of variables available for monitoring.

Table 5-36. User Status Registers

Variables	Type	Reset Value	Description
SYSTEM_FAULT_STATUS	USER_FAULT_TYPES	NO_FAULT	Defines the status of motor faults. **MOTOR_STALL** : Indicates motor lock faults - abnormal BEMF, no motor, abnormal speed **VOLTAGE_OUT_OF_BOUNDS:**Indicates undervoltage or overvoltage. **LOAD_STALL:**Indicates IPD fault. **HARDWARE_OVER_CURRENT:**Indicates DC bus current limit fault **HV_DIE:** Indicates gate driver fault if applicable.

Table 5-36. User Status Registers (continued)

Variables	Type	Reset Value	Description
MOTOR_STATE	MOTOR_STATE_TYPE	MOTOR_IDLE	Defines the state of Current Motor Running Status ***MOTOR_IDLE*** : Motor Idle State ***MOTOR_ISD*** : Motor in Initial Speed Detection state ***MOTOR_TRISTATE*** : Motor in Tristate or Hi-Z mode. ***MOTOR_BRAKE_ON_START*** : Motor Brake during Start up. ***MOTOR_IPD*** : Motor in Initial position Detection ***MOTOR_SLOW_FIRST_CYCLE*** : Motor in Slow First Cycle Startup Method. ***MOTOR_ALIGN*** : Motor in Align Start State ***MOTOR_OPEN_LOOP*** : Motor in openloop ramp up state. ***MOTOR_CLOSE_LOOP_UNALIGNED*** : Motor in Closed loop run State with Angle unaligned ***MOTOR_CLOSE_LOOP_ALIGNED*** : Motor in closed loop run state aligned angle. ***MOTOR_SOFT_STOP*** : Motor in Stop state ***MOTOR_BRAKE_ON_STOP*** Motor in Brake stop state ***MOTOR_FAULT*** Motor in Motor Fault State
V_DQ_FILT	IQ GLOBAL 27	IQ27(0)	Indicates the filtered Vd and Vq applied to the motor. Output of current PI controllers.
I_DQ_PI	IQ GLOBAL 27	IQ27(0)	Indicates the Kp and Ki values of current PI controllers.
PI_SPEED	IQ GLOBAL 27	IQ27(0)	Indicates the reference and feedback values of speed PI controller set by FOC algorithm in PU.
PI_POWER	IQ GLOBAL 27	IQ27(0)	Indicates the reference and feedback values of Power PI controller set by FOC algorithm in PU.
PI_ID	IQ GLOBAL 27	IQ27(0)	Indicates the reference and feedback values of direct current PI controller set by FOC algorithm in PU.
PI_IQ	IQ GLOBAL 27	IQ27(0)	Indicates the reference and feedback values of quadrature current PI controller set by FOC algorithm in PU.
IPD_IDENTIFIED_SECTOR	COMMUTATION_STATE	0b	Indicates the IPD identified nearest rotor state.
ESTIMATED_SPEED	IQ GLOBAL 27	IQ27(0)	Indicates the motor speed in PU estimated by the FOC observer algorithm.
DC_BUS_VOLTAGE	IQ GLOBAL 27	IQ27(0)	Indicates the DC bus voltage value in PU
TORQUE_LIMIT	IQ GLOBAL 27	IQ27(0)	Indicates the quadrature current controller saturation limit set by FOC. This value is based on the limit set in ClosedLoop1 configuration.
GATE_DRIVER_FAULT_STATUS	Unsigned Int	0x00000000h	Defines the Index of Gate Driver Specific faults as defined in gateDriverLib.
CONTROLLER_FAULT_STATUS	Unsigned Int	0x00000000h	Defines the Index of FOC Control Algorithm Specific Faults as defined in main.h.
APP_VERSION	unsigned hex	0x00000000h	Defines the Version number of Application Firmware

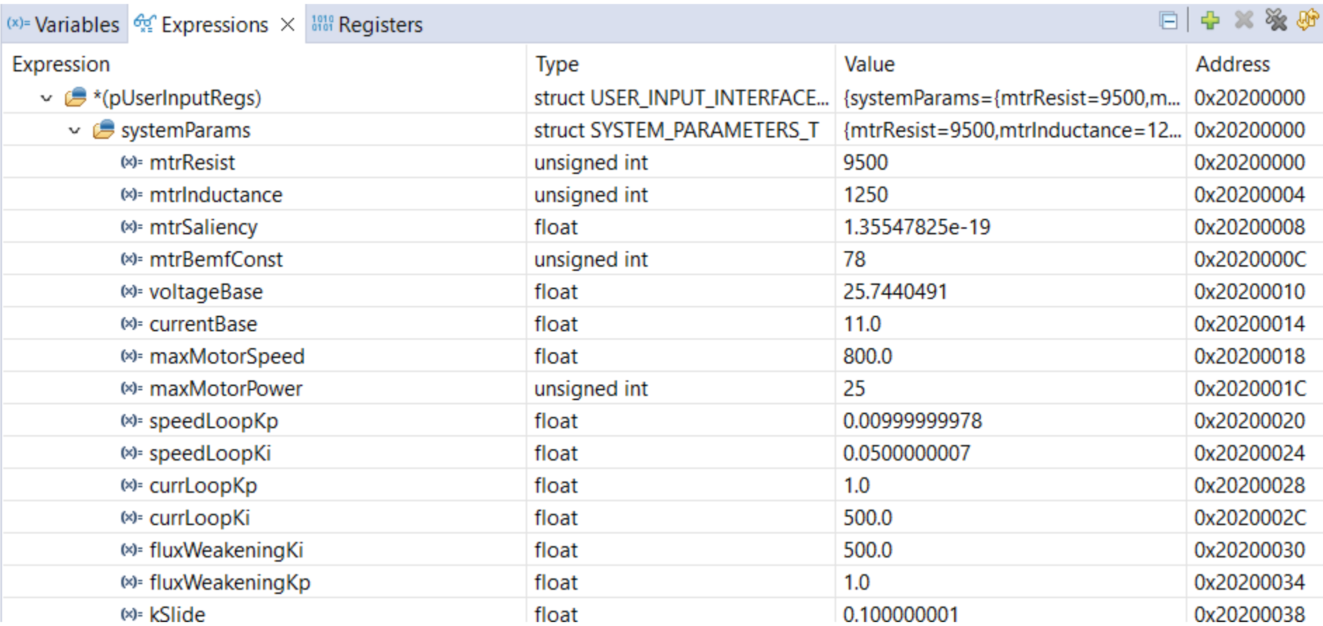
6 Basic Tuning

The goal of this section is to help spin user motors successfully in closed loop with minimal configurations. This section provides standardized mandatory steps to tune parameters for successful Motor spin-up in closed loop. "**Closed loop"** is defined as sensorless closed loop Field-oriented control where the motor spins at the commanded Speed/Torque reference.

6.1 System Configuration Parameters

The system configuration defines the primary parameters associated with the motor control system to start the motor spinning in closed loop torque/speed control modes.

Figure 6-1 shows the set of System parameters that are to be specified for accurate sensorless FOC operation. These variables can be added in the expression window using the pUserInputRegs.



Expression	Type	Value	Address
*(pUserInputRegs)	struct USER_INPUT_INTERFACE...	{systemParams={mtrResist=9500,m...	0x20200000
systemParams	struct SYSTEM_PARAMETERS_T	{mtrResist=9500,mtrInductance=12...	0x20200000
mtrResist	unsigned int	9500	0x20200000
mtrInductance	unsigned int	1250	0x20200004
mtrSaliency	float	1.35547825e-19	0x20200008
mtrBemfConst	unsigned int	78	0x2020000C
voltageBase	float	25.7440491	0x20200010
currentBase	float	11.0	0x20200014
maxMotorSpeed	float	800.0	0x20200018
maxMotorPower	unsigned int	25	0x2020001C
speedLoopKp	float	0.00999999978	0x20200020
speedLoopKi	float	0.0500000007	0x20200024
currLoopKp	float	1.0	0x20200028
currLoopKi	float	500.0	0x2020002C
fluxWeakeningKi	float	500.0	0x20200030
fluxWeakeningKp	float	1.0	0x20200034
kSlide	float	0.100000001	0x20200038

Figure 6-1. System Configurations Registers in CCS Debug Mode

6.1.1 Configuring System Parameters From GUI

Configure the system parameters using the **System Configuration** page in the GUI as shown in Figure 6-2. If the parameters are already programmed in the firmware for a given system, the GUI page displays the default programmed values upon pressing READ ALL REGS. These parameters to be updated accordingly from the below steps.

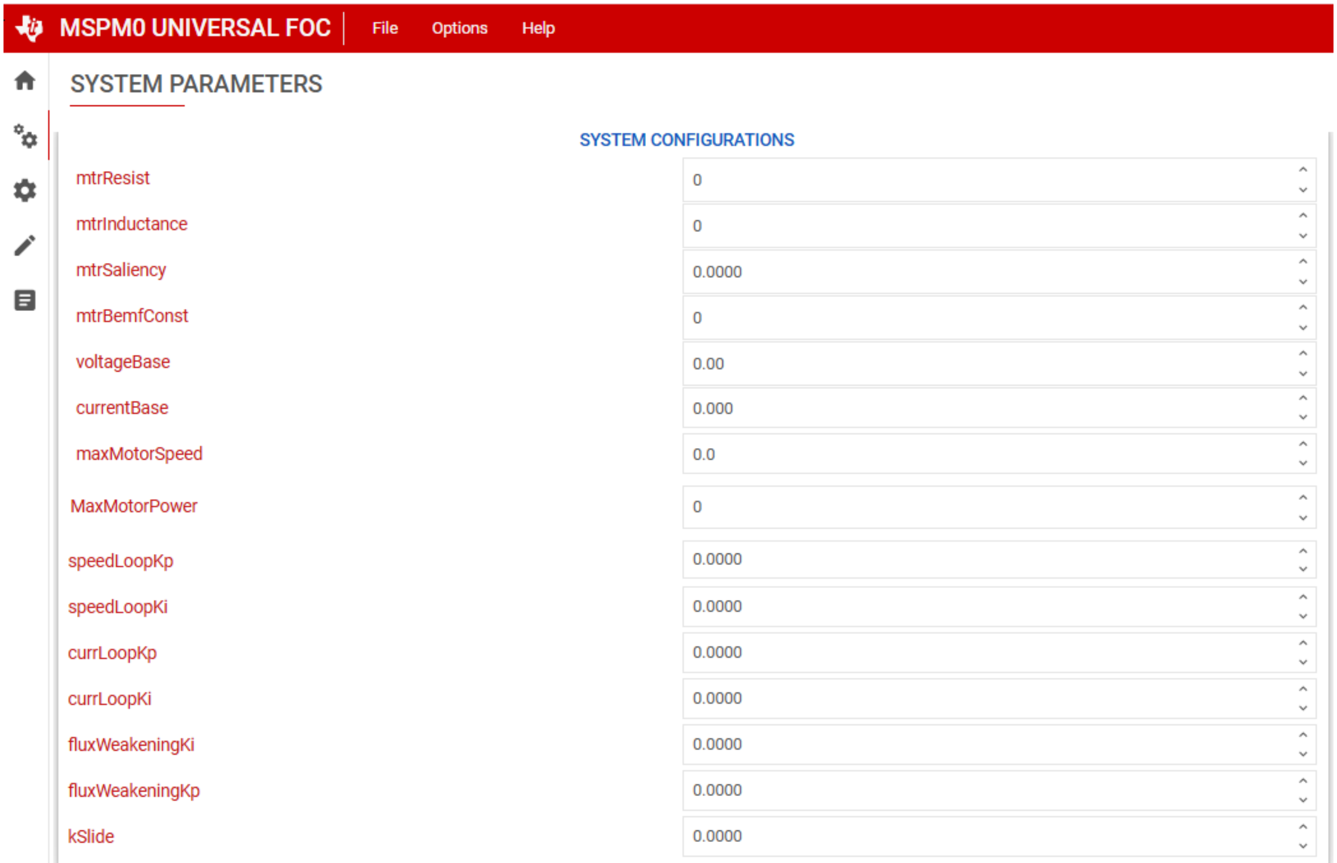


Figure 6-2. GUI System Parameter Configuration

6.1.2 Motor Resistance in Milliohms (mΩ)

Using the motor data sheet, the user can input the motor phase resistance in milliohms (mΩ) using the *mtrResist* parameter in the **System Configuration** page. If the motor does not have a data sheet, then measure the phase-to-phase resistance across any two phases using a digital multimeter and calculate the phase resistance by dividing the phase-to-phase resistance by 2 as shown in Resistance Measurement:

$$\text{Phase resistance} = \text{Measured Phase to Phase Resistance} \times (0.5) \tag{2}$$

The motor phase resistance refers to the equivalent phase to center tap resistance, R_{PH}, as shown in Figure 6-3. This measurement is valid for both star wound and delta wound motors.

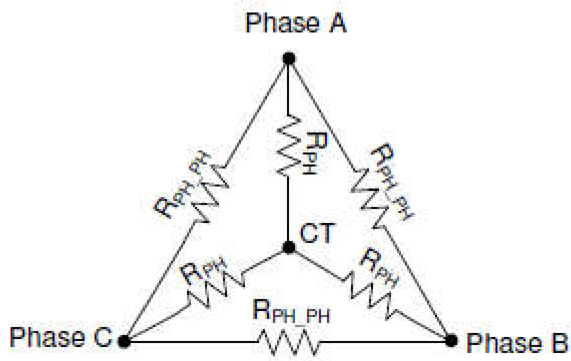


Figure 6-3. Resistance Measurement

6.1.3 Motor Inductance in Microhenries (µH)

From the motor data sheet, input the motor phase inductance in microhenry (µH) using the *mtrInductance* parameter in the **System Configuration** page. To know the motor inductance, measure the phase-to-phase inductance at 1kHz across any two phases using an LCR meter. Calculate the phase inductance by dividing the phase to phase inductance by 2 as shown in Figure 6-4.

$$\text{Phase Inductance} = \text{Measured Phase to Phase Inductance} \times (0.5) \tag{3}$$

Motor phase inductance refers to the inductance from the phase output to the center tap, L_{PH} , as shown in Figure 6-4. For motors with different phase to phase inductances(Salient pole motors), measure the L_d and L_q values as defined in below section to compute the average value $(L_d + L_q)/2$ and use this value as the phase to phase inductance. This measurement is valid for both star wound and delta wound motors.

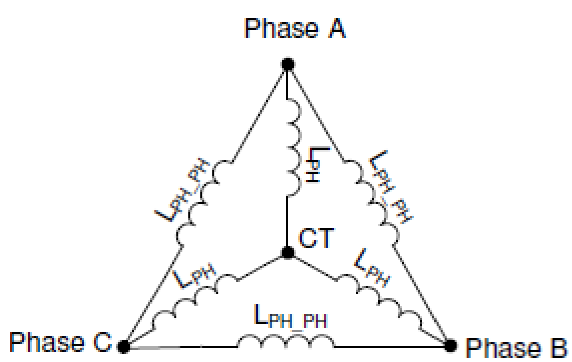


Figure 6-4. Inductance Measurement

6.1.4 Saliency of IPMSM Motor

Saliency of an IPMSM motor is a measure of variation in Inductance between the quadrature axis and direct rotor axis. For FOC algorithm this value is to be given as (Lq – Ld) / (Ld + Lq) in float variable.

The simplest method to deduce the Ld and Lq values is by measuring the inductance across any two phases and vary the rotor position slowly for one full rotation. The maximum measured inductance value can be noted as Lq, and the minimum measured inductance value can be noted as Ld.

6.1.5 Motor BEMF Constant

Using the motor’s data sheet, the user can input the motor's BEMF constant Ke in mV/Hz and program *mtrBEMFConst* in the **System Configuration** Page as Ke × 10.

Equation 4 and Equation 5 can be used to convert Ke in mV/rpm, mV*sec/rad and torque constant Kt to Ke in mV/Hz.

$$\text{BEMF Constant}\left[\frac{\text{mV}}{\text{Hz}}\right] = \frac{\text{Ke}\left[\frac{\text{mV}}{\text{RPM}}\right]*60}{\#\ \textit{pole pairs}} \quad (4)$$

$$\text{BEMF Constant}\left[\frac{\text{mV}}{\text{Hz}}\right] = \frac{\text{Kt}\left[\frac{\text{mN}\cdot\text{m}}{\text{A}}\right]*2\pi}{\#\ \textit{pole pairs}} \quad (5)$$

If the motor does not have a data sheet, measure the voltage across any two phases of the motor using an oscilloscope by manually spinning the motor. A sinusoidal or trapezoidal voltage appears on the oscilloscope. Measure the peak voltage Ep in milli-volts and time period Tp in seconds. Calculate BEMF constant Ke as shown in Equation 6.

$$\text{Bemf Constant Ke} = \text{Ep} * \text{Tp} / \sqrt{3} \quad (6)$$

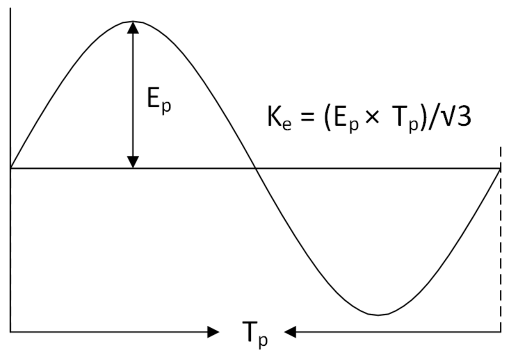


Figure 6-5. BEMF Constant Measurement

6.1.6 Base Voltage (V)

Base voltage represents the maximum measurable bus voltage and phase voltages in the motor control system. Input the system base voltage (in volts) in the *voltageBase* parameter of the **System Configuration** page in GUI. The user can compute the system base voltage based on the Voltage scaling resistor divider bridge values R1 and R2 and the Full Scale ADC voltage (FSV) of 3.3V as shown in Equation 7. For hardware configuration of the voltage divider scaling ratio, see Figure 2-3.

$$\text{Base}_{\text{Voltage}} = \frac{\text{ADC Full Scale Value}}{\text{Voltage Divider Scaling Ratio}} = \frac{3.3V}{\frac{R1}{R1+R2}} \quad (7)$$

For example, in a system with resistor divider scaling ratio of 1/20 from DC supply voltage to ADC input, the base voltage or maximum measurable system voltage by the ADC is 3.3V / (1/20) = **66V**.

6.1.7 Base Current (A)

Base current represents the maximum measurable motor phase current in the motor control system. The user inputs the system base current (in Amps) in the *currentBase* parameter of the **System Configuration** page in GUI. The user can compute the system base current based on the current sense amplifier gain (CSAGAIN) in volts/amp and the full-scale ADC voltage (FSV) of 3.3V and Zero Current Offset voltage from the CSA as shown in Equation 8:

$$\text{Base Current} = \frac{3.3\text{v} - \text{ADC\_Offset\_volts}}{\text{CSA\_GAIN}\left(\frac{\text{V}}{\text{A}}\right)} \quad (8)$$

In case of bidirectional current sensing the zero-current offset is often selected as 1.65V , half of ADC Full Scale Voltage 3.3V.

In case of Unidirectional Current Sensing, 80-90% of ADC Full Scale Voltage is typically used for +ve Current Sensing. In general, Unidirectional current sensing is used in single shunt DC Bus current sensing architecture.

If the system uses a current sense resistor (R_{SENSE}) with CSAGAIN units mentioned in volts/volt (V/V), the CSA gain in volts/amp can be computed using Equation 9.

$$\text{CSA GAIN}\left(\frac{\text{volts}}{\text{Amp}}\right) = R_{\text{sense}} \times \text{CSA GAIN}\left(\frac{\text{volts}}{\text{volts}}\right) \quad (9)$$

For example, in a system with CSAGAIN = 0.15V/A and ADC Offset Voltage of 1.65V , the base current or maximum measurable system current by the ADC is (3.3V -1.65V)/ (0.15V/A) = **11A**.

In system with Rsense of 10milli Ohms, CSA gain of 10V/V , ADC Offset Voltage of 0.4125V, the base current is derived as (3.3-0.4125)/(10m * 10) = 28.875A

Note

In some driver devices, CSAGAIN can be set as a register over I2C or SPI or by hardware using a resistor value. For how to configure the driver CSAGAIN setting, see the device-specific driver data sheet.

6.1.8 Maximum Motor Electrical Speed (Hz)

Using the motor's data sheet, the user can input the maximum motor electrical speed in Hz using the *maxMotorSpeed* parameter in the **System Configuration** Page. If this data is not available, the user can input the number of pole pairs and motor mechanical speed in RPM. The user can convert the motor mechanical speed in RPM to motor electrical speed in Hz using Equation 10.

$$f_{Electrical} = \frac{n_{PolePairs} \cdot \omega_{Mechanical}}{60} \quad (10)$$

Where:

- $\omega_{\textbf{Mechanical}}$ is the mechanical speed in units revolutions per minute (RPM)
- $f_{\textbf{Electrical}}$ is the electrical speed in units of hertz (Hz)
- $n_{\textbf{PolePairs}}$ is the number of motor pole pairs

Note

To determine the number of motor poles without a motor data sheet:

1. Use a lab power supply and make sure the current limit is set to less than the motor rated current. Do not turn on the supply.
2. Connect V+ of the supply to phase A and V- of the supply to phase B of the motor. Any 2 of the 3 phases can be chosen at random if the phases not labeled.
3. Turn on supply, The rotor settles at one position by injecting current.
4. Manually rotate the rotor until rotor snaps to another settle position. Rotor settle down at various positions around one mechanical cycle.
5. Count the number of settle-down positions for one fully mechanical cycle, which is the number of pole pairs. Multiplying by two calculates the number of poles.

Be careful of gearing systems within a motor. The gearing ratio determines how many rotor revolutions correlate to the shaft mechanical revolution.

6.1.9 Maximum Motor Power(W)

User need to input the Maximum Power Rating of the motor when Closed loop Power Control is required. To determine the Maximum Rated Power of the motor, see the Motor data sheet and compute the product of Rated Motor Voltage in Volts and Rated Motor Current in Amps and feed the value to MOTOR_MAX_POWER in the System parameters.

6.2 Control Configurations for Basic Motor Spinning

After configuring the system parameters in the GUI, the user can go to the **Register Map** page and configure the Register Map Tuning parameters as shown in Figure 6-6. By default, the firmware has the recommended settings, which can be read into the GUI by pressing the "READ ALL REG" button.

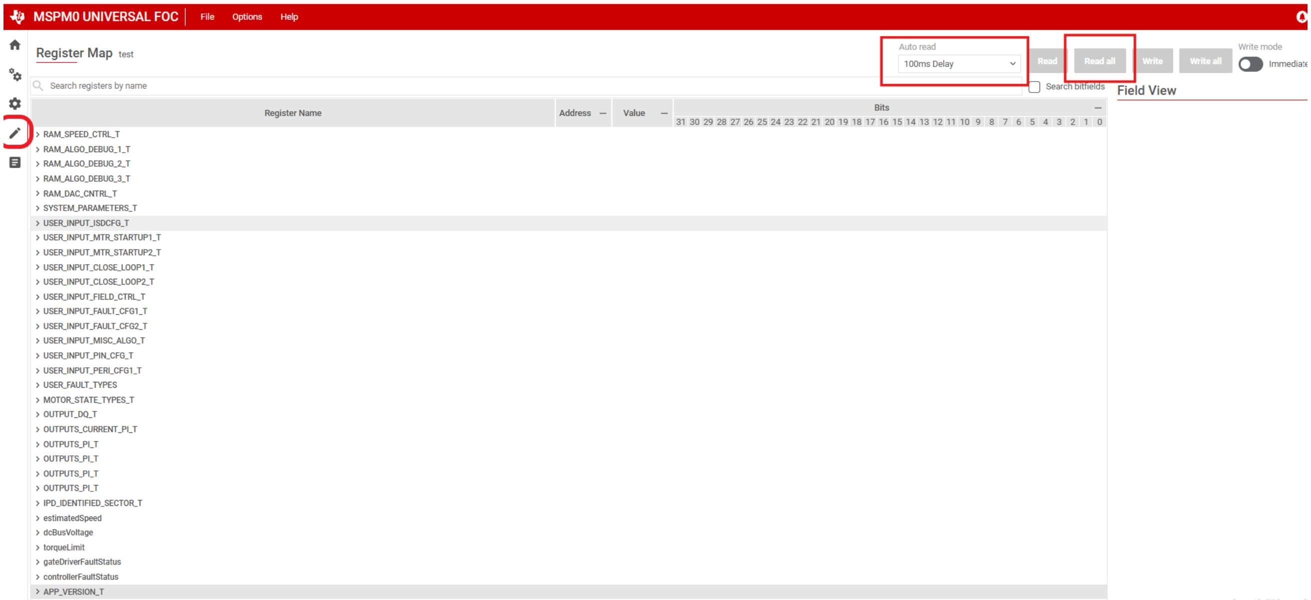


Figure 6-6. Register Map GUI Page

6.2.1 Basic Motor Startup

Sensorless FOC relies on the position detection estimated from BEMF to accurately drive the motor. At startup, since the motor can be at standstill or the motor can be spinning with unknown speeds, the rotor position is unknown.

The FOC algorithm has various startup algorithms to reliably start the motor and ramp the motor with sufficient speeds or estimate the position of an already spinning motor before switching to the estimator for continuous rotor position tracking. The following sections describe the basic configuration needed to start and ramp the motor from standstill until open-loop spin up. If any motor faults are observed during basic open-loop spin up, see Section 6.3.

6.2.1.1 Disable ISD

Initial Speed Detection (ISD) is a feature to enter the motor automatically when the motor is already spinning. This is also called Headwind/Tailwind startup, or Catch-on-the-fly startup. For basic spinning for motor, the ISD feature is disabled by default by setting *isdEn* = 0 on the **Register Map** page. For basic motor tuning, the motor is expected to be at standstill before giving the speed command. To tune the motor for ISD, see Section 7.

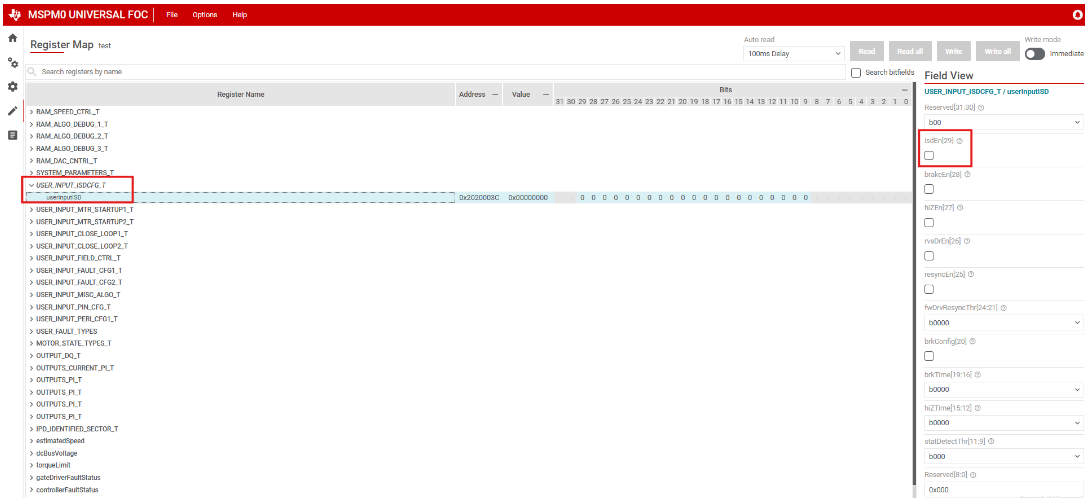


Figure 6-7. Disabling ISD in GUI

6.2.1.2 Motor Start Option - Align

When the motor is ramping up from standstill, the Motor Align startup algorithm forces the rotor to align to a fixed ALIGN_ANGLE with a defined current limit acting as a torque reference for a predefined ALIGN_TIME. By default, the motor startup option is set as Align (*mtrStartUpOption* = 0b) in the MTR_STARTUP of MOTOR_STARTUP1 configuration. For basic spinning, use the default parameters which works for most of the motors.

If the motor fails to align for the given load setup, increase the *alignOrSlowCurrentLimit* parameter on the **Register Map** page. For fine tuning the Motor Align configuration, see Section 7.

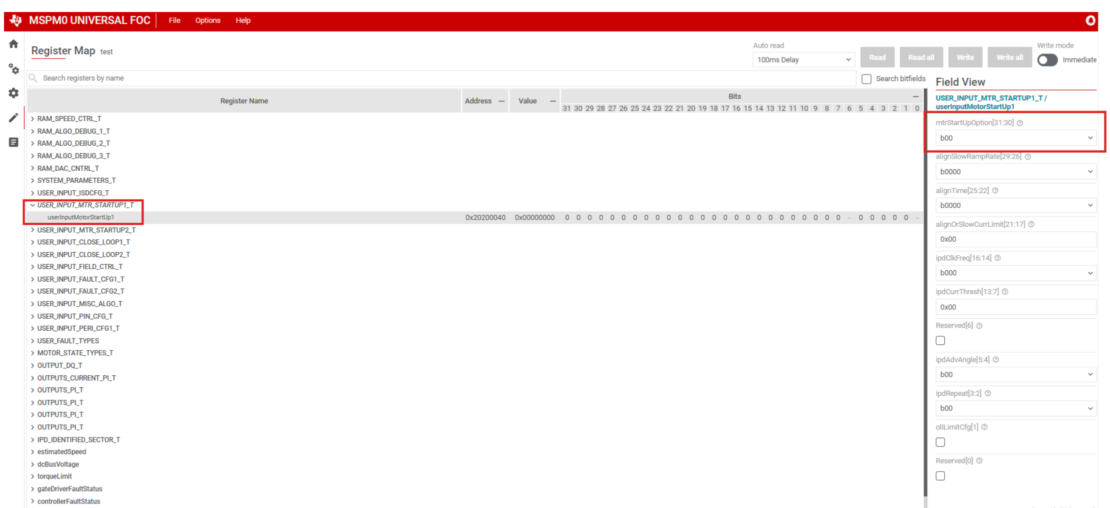


Figure 6-8. Motor Startup in GUI

6.2.1.3 Motor Open Loop Ramp

To estimate the rotor position accurately, the motor needs to build sufficient BEMF before switching to closed loop. During startup, the FOC algorithm accelerates the motor with a second order open loop ramp profile to increase the speed until sufficient BEMF is built. By default, for basic spin up of motor, the open loop ramp up parameters are configured with a linear first order configuration with sluggish acceleration, which works for most of the motors. Disable switching to closed loop control to verify the appropriate functionality of open loop , using *closedloopDis* = 1b in ALGO_DEBUG_CTRL on the **Motor Tuning** page. To further optimize the startup time, see Section 7 to fine tune the startup performance.

Depending on the load of the motor, tune the OL_ILIMIT to the lowest possible value for smooth handoff once the closed loop is enabled (*closedloopDis* = 0b).

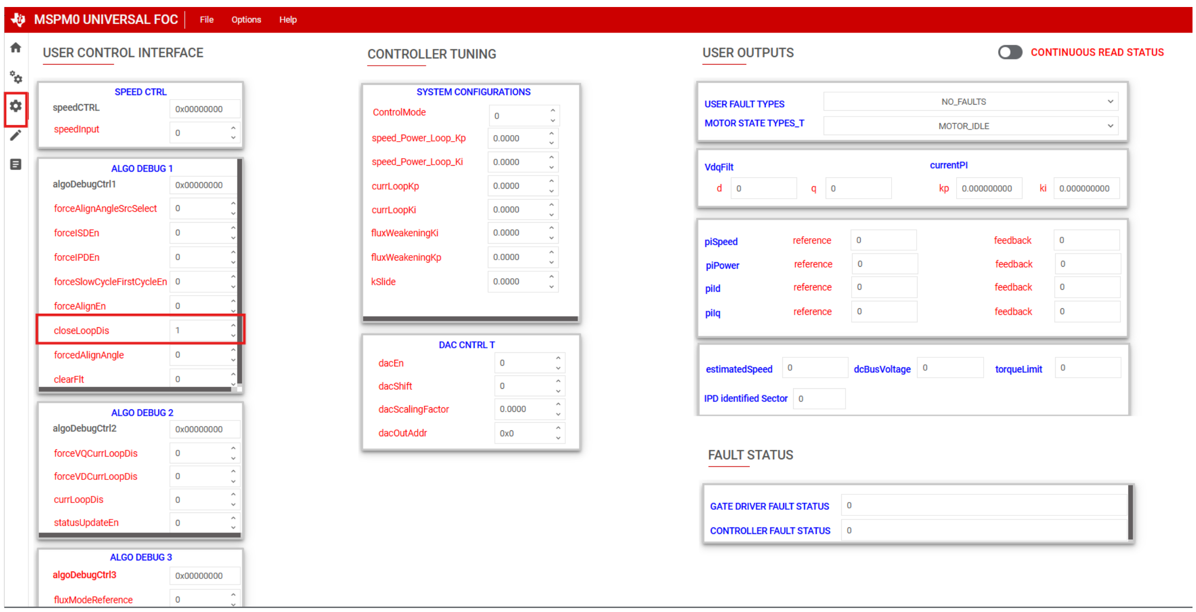


Figure 6-9. Setting ClosedLoop Disable in GUI

6.2.1.4 Motor Open Loop Debug

If motor fails to spin in open loop continuously or if motor current oscillates without spinning the motor, the user can fine tune the open loop configurations in the RAM ALGO DEBUG 2 parameters.

To verify the signal path or check the motor parameters accuracy, the user can disable the current loop by setting the *currLoopDis* bit and setting the *forceVQCurrLoopDis* and *forceVDCurrLoopDis* on the **Register Map** page.

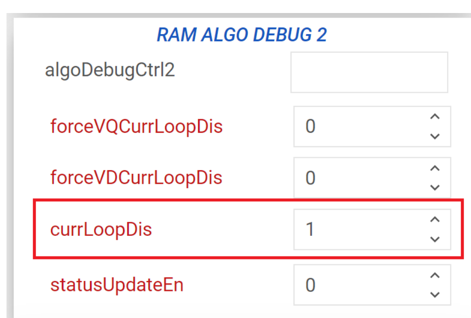


Figure 6-10. Disabling Current Loop in GUI

Adjust Vd and Vq values to spin the motor. Once the motor is spinning at a constant speed, observe the Id and Iq outputs in the User Outputs section of the **Motor Tuning** page to check the stability of the current loop.

6.2.2 Controller Configuration for Spinning the Motor in Closed Loop

After tuning the open loop with the motor spinning continuously in the open loop state, the user can switch to closed loop by clearing *closedloopDis* = 0b in ALGO_DEBUG_CTRL on the **Motor Tuning** page. Follow the steps below to achieve closed loop speed control.

6.2.2.1 BEMF estimation for Sensorless Rotor Position detection

In applications similar to home appliances, mechanical sensor adds to cost, reliability and maintenance. In general, Sensorless based rotor position estimation methods are employed to efficiently drive the motor in applications where ultralow speed operation is not a requirement. To detect the rotor position in sensorless methods, BEMF of the motor is estimated through various methods and there by the rotor speed and angle are approximated. In the Universal Motor Control Application code, user has option to select either of Sliding Mode Observer or the Finite difference BEMF Estimation methods.The sliding mode observer is commonly utlized due to reliability and robustness against system parameter variations. The finite BEMF estimation is simple equation based BEMF estimation without sliding mode controller and filter for BEMF, this eliminates the Kslide tuning and filter tuing but BEMF is prone to noise and can create stability issues . Estimated BEMF's from both of the methods are used for rotor position tracking using a PLL as detailed below.

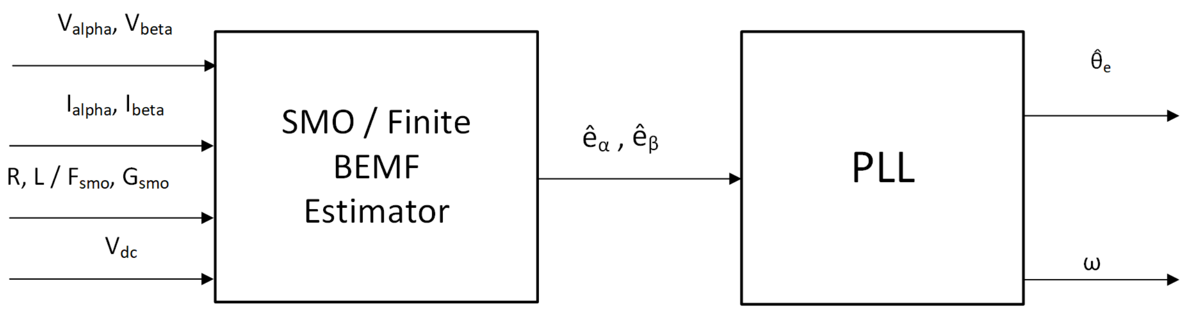


Figure 6-11. Sensorless PMSM Rotor Position Estimation Block Diagram

SMO based BEMF estimation is chosen as default by keeping the predefined symbol " ***ESMO_ESTIMATOR*** . To select the Finite difference equation based BEMF estimator, user can modify this predefined symbol to ***ESMO_ESTIMATOR_N*** from the CCS project settings as below.

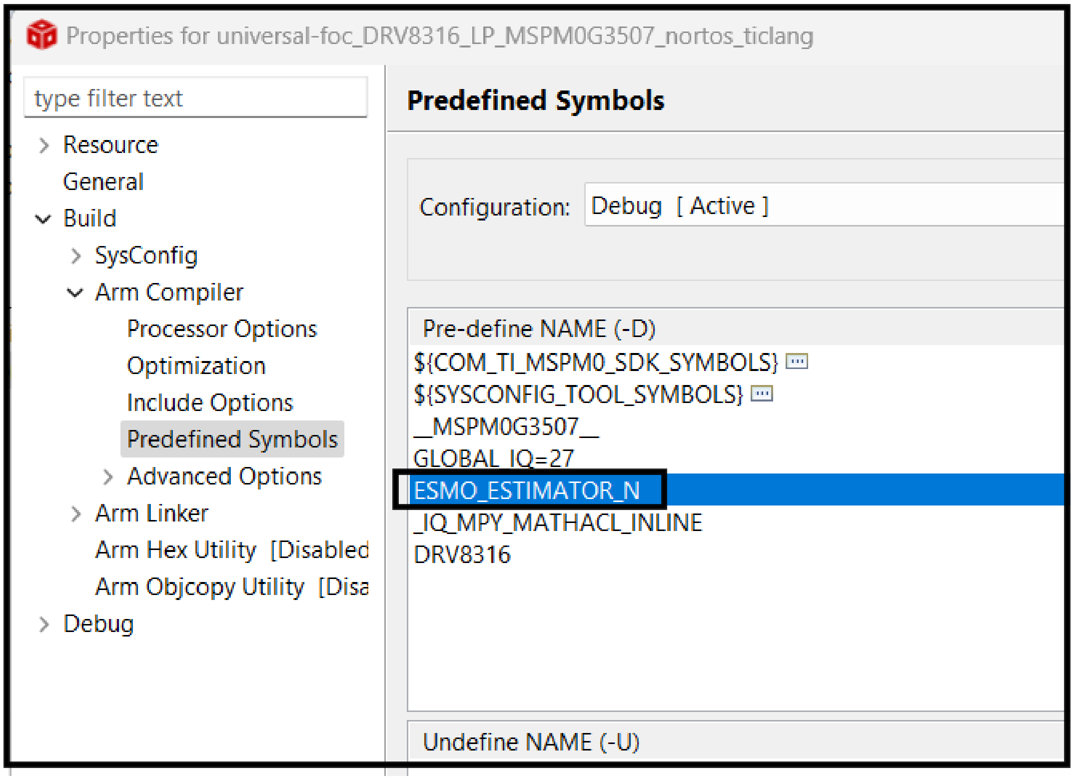


Figure 6-12. Configuration for Estimator Selection

6.2.2.1.1 Enhanced Sliding Mode Observer

Model-based BEMF estimation methods are used to achieve position sensorless control of the IPMSM drive system when the motor runs at middle or high speeds. The model methods estimates the rotor position by the back-EMF or the flux linkage model. The sliding mode observer is an observer-design method based on sliding mode control. The structure of the system is not fixed but purposefully changed according to the current state of the system, forcing the system to move according to the predetermined sliding mode trajectory. Advantages include fast response, strong robustness, and insensitivity to both parameter changes and disturbances.

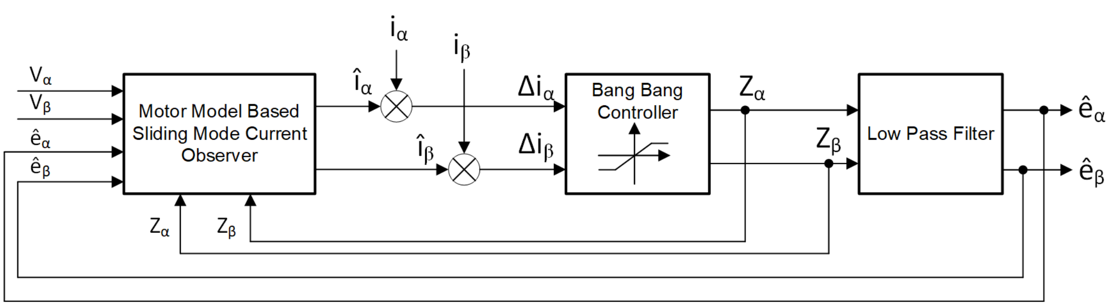


Figure 6-13. Sliding Mode Observer based BEMF Estimation

In a digital control application, a time discrete equation of the SMO is needed. The Euler method is the appropriate way to transform to a time discrete observer. The time discrete system matrix of motor model is given in stationary reference frame is given as in Equation 11.

$$\begin{bmatrix} i_{\hat{\alpha}}(n+1) \\ i_{\hat{\beta}}(n+1) \end{bmatrix} = F_{smo}\begin{bmatrix} i_{\hat{\alpha}}(n) \\ i_{\hat{\beta}}(n) \end{bmatrix} + G_{smo}\begin{bmatrix} V_\alpha(n) - e_{\hat{\alpha}}(n) - z_\alpha(n) \\ V_\beta(n) - e_{\hat{\beta}}(n) - z_\beta(n) \end{bmatrix} \quad (11)$$

where F_{smo} and G_{smo} are constants defined based on the Motor parameters as shown below.

$$F_{smo} = e^{-\frac{R}{L}},\ G_{smo} = \frac{1}{R}\left(1 - e^{-\frac{R}{L}}\right) \tag{12}$$

z_α and z_β are sliding mode components and are defined as:

$$\begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} = K_{slide}\begin{bmatrix} \mathrm{sign}(\hat{i}_\alpha - i_\alpha) \\ \mathrm{sign}(\hat{i}_\beta - i_\beta) \end{bmatrix} \tag{13}$$

where K_{slide} is the constant sliding mode gain designed by Lyapunov stability analysis. The Kslide value can be tuned using the System Parameters. Having higher Kslide tracks the sliding surface current faster but having very high Kslide leads to errors in BEMF estimation due to switching noise.

Below equation represents the time discrete form of filtered BEMF. Low pass filter removes the high frequency sliding mode output, where the cutoff frequency f_c is usually chosen as the fundamental frequency of stator current. This introduces a phase shift of 45 degrees which is compensated from the estimated rotor position.

$$\begin{bmatrix} \hat{e}_\alpha(n+1) \\ \hat{e}_\beta(n+1) \end{bmatrix} = \begin{bmatrix} \hat{e}_\alpha(n) \\ \hat{e}_\beta(n) \end{bmatrix} + 2\pi f_c \begin{bmatrix} z_\alpha(n) - \hat{e}_\alpha(n) \\ z_\beta(n) - \hat{e}_\beta(n) \end{bmatrix} \tag{14}$$

6.2.2.1.2 Finite BEMF Estimation Based on Motor model

User can select finite BEMF estimator in cases where Kslide tuning is not desired or the noise disturbances in the system are minimal. In finite BEMF estimation method, the BEMF is derived based on stator reference frame voltage equations as shown in Equation 15.

$$\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = \begin{bmatrix} \hat{v}_\alpha \\ \hat{v}_\beta \end{bmatrix} - \begin{bmatrix} r_s & 0 \\ 0 & r_s \end{bmatrix}\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} - \frac{d}{dt}\left(\begin{bmatrix} L_0 - L_1\cos(2\theta) & -L_1\sin(2\theta) \\ -L_1\sin(2\theta) & L_0 + L_1\cos(2\theta) \end{bmatrix}\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix}\right) \tag{15}$$

where , $L_0 = \left(\frac{L_{ds} + L_{qs}}{2}\right)$, $L_1 = \left(\frac{L_{ds} - L_{qs}}{2}\right)$;

If the motor doesn't have any saliency, like in Surface mounted PMSM motor, the L1 value is set to zero. By default the application code assumes the motor is non salient motor.

6.2.2.2 Rotor Position and Speed Estimation With PLL

In conventional SMO based rotor position estimators, the rotor flux angle is determined based on the arc tangent of estimated stationary co-ordinate BEMF values as in Equation 16:

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) \tag{16}$$

With this method, the accuracy of the position and velocity estimations are affected due to the existence of noise and harmonic components. To eliminate this issue, the PLL model can be used for velocity and position estimations in the sensorless control structure of the PMSM. The estimated BEMF values $\hat{e}_\alpha$ and $\hat{e}_\beta$ can be used with a PLL model to converge motor angular velocity and compute the rotor position as shown in Figure 6-14.

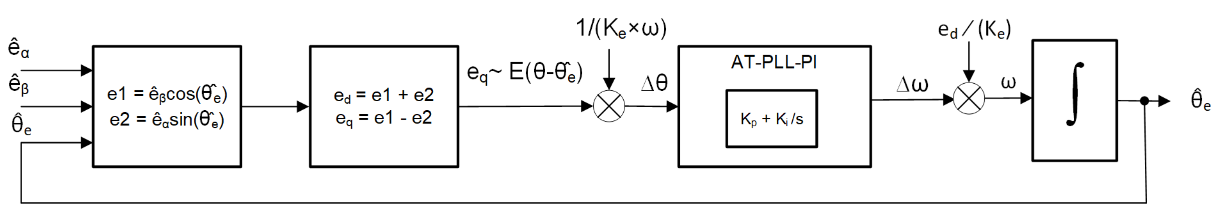


Figure 6-14. PLL-Based Rotor Speed and Position Estimation

Since $e_{\hat{\alpha}} = E\cos(\theta)$, $e_{\hat{\beta}} = E\sin(\theta)$ and $E = K_e \times \omega$; the error in angle $\Delta\theta$ between the actual rotor position θ and the estimated rotor angle $\theta_{\hat{e}}$ can be computed as

$$e_q = E\cos(\theta)\sin(\theta_{\hat{e}}) - E\sin(\theta)\cos(\theta_{\hat{e}}), \quad e_q = E\sin(\theta - \theta_{\hat{e}}); \tag{17}$$

$$\text{as } \Delta\theta \text{ approaches near } 0,\ e_q \approx E(\theta - \theta_{\hat{e}}),\ \text{with normalization } e_n = (\theta - \theta_{\hat{e}}); \tag{18}$$

The closed loop transfer function of above plant can be treated as second order transfer function including PI controller and an integrator for position estimator defined by Equation 19:

$$\frac{\theta_{\hat{e}}}{\theta} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{19}$$

here the PI gains are defined by $k_p = 2\zeta\omega_n$, $k_i = \omega_n^2$ where ζ is the damping factor of the response and ω_n is the natural frequency of the second order response.

As the feedforward term for estimated speed is added , the PI controller estimates only the error in speed thus Kp and Ki can be independent of speeds and can be fixed values unless noise or sudden disturbances in load are expected. By default the K_p and K_i values are defined in "angleTrackingPLL.c" under modules/algoLib/libraries/semiCloseLoopEstim/source.

6.2.2.3 PI Controller Tuning for Closed Loop Speed Control

6.2.2.3.1 Current Loop PI Tuning

The FOC Algorithm uses two current PI controllers: one each for Id and Iq to control flux and torque separately. Kp and Ki coefficients are the same for both PI controllers and are configured through *currLoopKp* and *currLoopKi* in the **Motor Tuning** page.

For the basic tuning, configure *currLoopKp* and *currLoopKi* parameters as "0" so that these values are auto-computed based on the motor parameters and reflected in the User Outputs section of the **Motor Tuning** page in the GUI. These values can be further updated for fine tuning the performance and controlling the dynamics of the system.Typically the default generated Kp and Ki values can be further scaled down 3 to 10 times to meet the desired overdamped response.

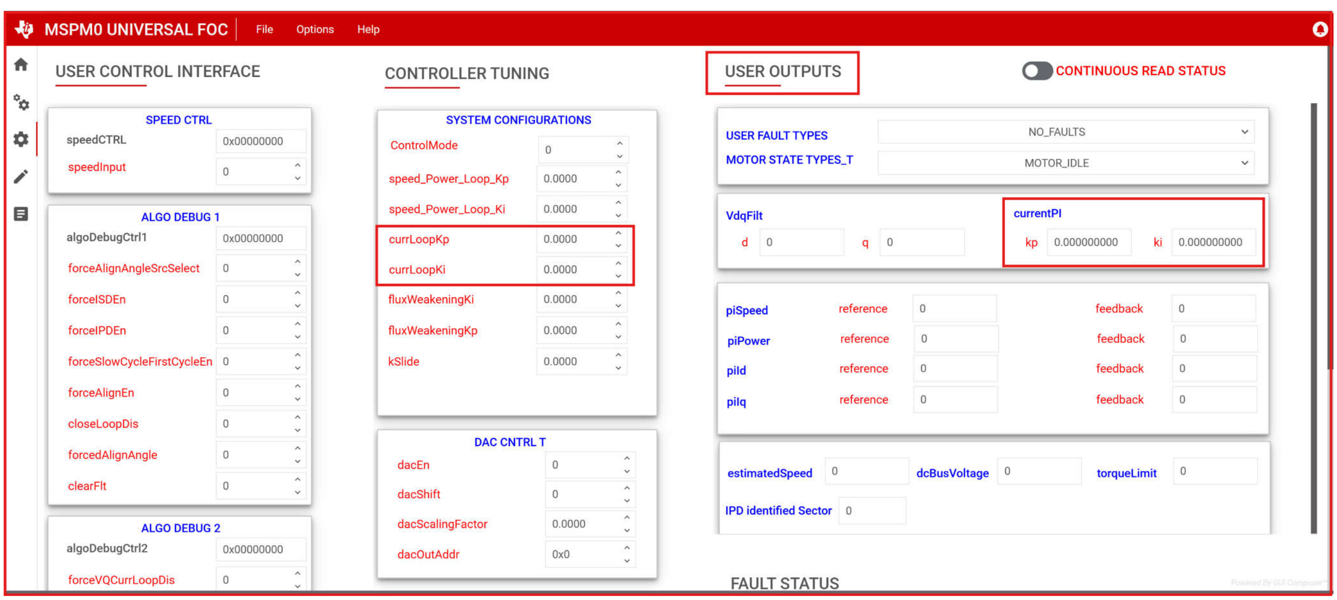


Figure 6-15. PI Loop Tuning in GUI Motor Tuning page

6.2.2.3.2 Speed Controller Tuning

The FOC Algorithm uses an integrated speed control loop /Power Control Loop that helps maintain a constant speed/ Constant power over varying operating conditions. The Kp and Ki coefficients are configured through *speedLoopKp* and *speedLoopKi* in the "System Configurations" section on the **Motor Tuning** page. The output of the speed loop / power Loop is used to generate the current reference for torque control. The output of the speed loop /power Loop is limited by configuring *iLIMIT* in the closedLoop1 configuration in the Register Map page of GUI. When output of the speed loop / Power Loop saturates, the integrator is disabled to prevent integral wind-up.

To tune the Kp and Ki values for speed loop:

1. Configure the motor to spin continuously in open loop by setting *closedloopDis* to 1b. Disable the automatic handoff by setting *autoHandOffEn* to *0b*.
2. Set the closed loop hand off threshold to around 50% of maximum speed using *olClHandOffThr*.
3. Set the *iqRampEn* bit to *1b* in the userInputMotorStartUp1 register.
4. The current reference gradually decreases and settles down to the lowest possible Iqref to run at the given threshold speed.
5. Speed loop Kp [SPD_LOOP_KP] is calculated using this equation: SpeedLoop K_p = Current Reference at olClHandOffThr in Amps / olClHandOffThr in Hz
6. Speed loop Ki [SPD_LOOP_KI] is calculated using this equation: Speed Loop K_i = Speed Loop K_p × 0.1
7. Enable closed loop by clearing the closedloopDis to (0b) in the configuration in the **Register Map** page of the GUI.

Note

The tuning of speed loop Kp and Ki is experimental. If the above recommendation does not work, manually tune speed loop Kp and Ki until the desired results are achieved.

The following table shows general variations in the dynamics upon increase in the controller gains.

Parameter	Rise Time	Overshoot	Settling Time	Stead State Error	Stability
Kp	Decreases	Increases	Small Change	Decreases	Degrades
Ki	Decreases	Increases	Increases	Eliminates	Degrades

6.2.2.4 Testing for Successful Startup Into Closed Loop

1. **Apply a nonzero speed command.**

   Change the "Speed Input Command" to a nonzero value. When the speed command is issued, the device starts to commutate and the motor spins at a speed that is proportional to Speed Command × MAXIMUM MOTOR SPEED / 32767.

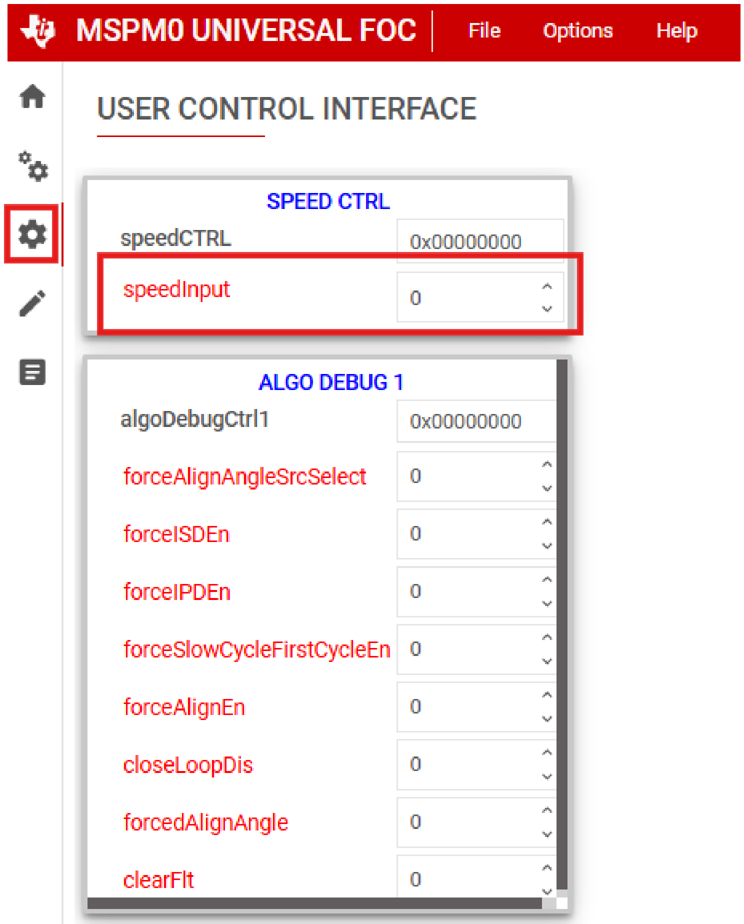


Figure 6-16. Setting Speed Input From GUI

2. **Check if motor spins in closed loop at commanded speed.**

   Enable the "Continuous Read status" toggle button towards the top right corner of the GUI and monitor the Fault Status register. If no faults is triggered, move to the Section 7.

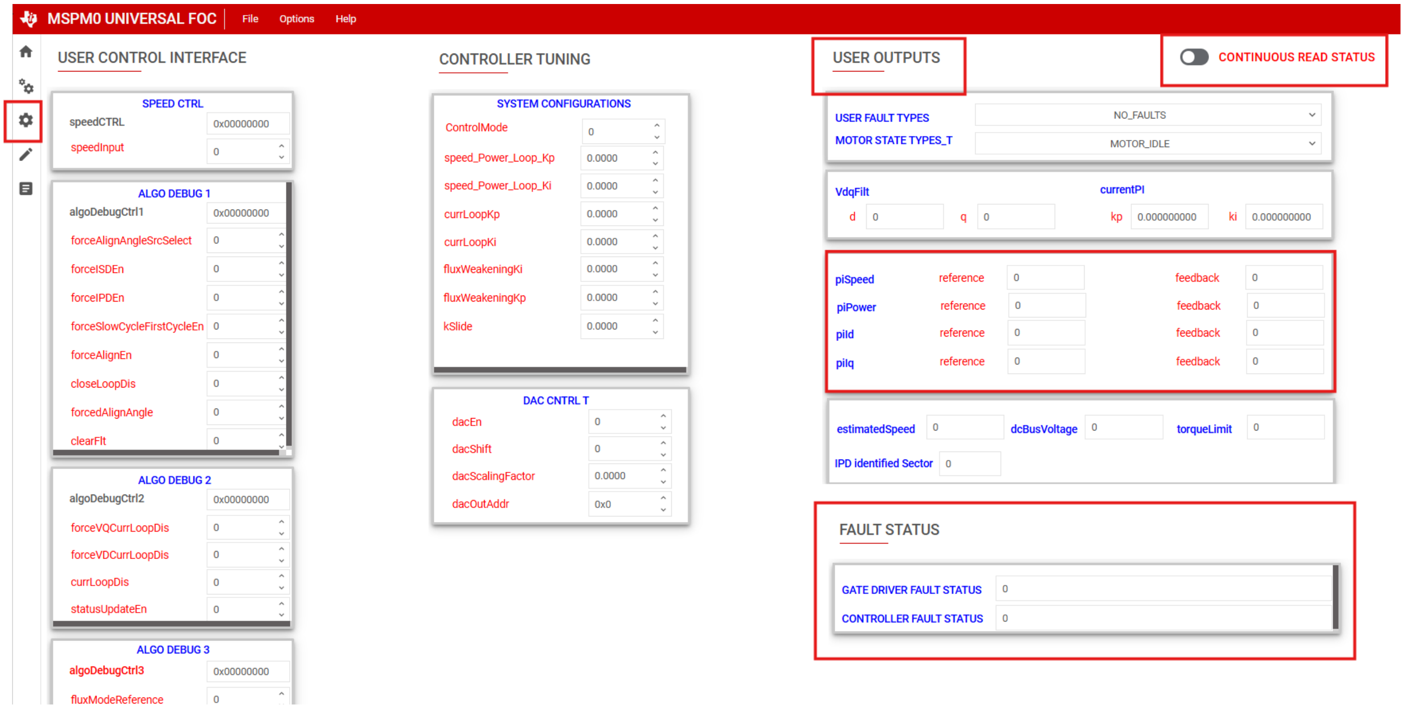


Figure 6-17. Reading Fault Status From GUI

3. If any fault is triggered, tune the configuration for fault handling using these steps:
   a. Set zero speed command by setting the Speed Input command to 0.
   b. Clear the fault status registers by setting the clear fault bit (*ClearFlt)* bit in the ALGO DEBUG CTRL1 register.
   c. Check Section 6.3 for steps to debug faults.

6.3 Fault Handling

The following sections describe faults that can be triggered based on the default register configuration.

6.3.1 Abnormal BEMF Fault [ABN_BEMF]

This fault is triggered when the estimated BEMF voltage drops below the programmed Abnormal BEMF threshold percentage [ABNNORMAL_BEMF_THR]. For example, if the estimated or measured Ke is 5mV/Hz and the programmed Abnormal BEMF threshold is 40%, this fault is triggered when the estimated Ke drops below 2mV/Hz. This fault can also be triggered when the programmed Ke is inaccurate.

There are two cases for Abnormal BEMF threshold:

Case 1: Estimated BEMF voltage drops when the motor speed drops. Motor speed can drop due to load dynamics (sudden change in load). For applications with load dynamics, the speed is expected to drop and recover back. Because the speed drops, the BEMF voltage also drop and can trigger this fault. For such applications, the recommended value of 10% is to be set for the Abnormal BEMF threshold to avoid triggering this fault.

Case 2: This fault can be triggered if the programmed Ke is inaccurate. Follow steps recommended in section MOTOR_BEMF_CONSTANT to obtain accurate Ke.

6.3.2 Monitoring Power Supply Voltage Fluctuations for Voltage Out of Bound Faults

In applications where the power supply fluctuates, user needs to specify the minimum and maximum power supply voltage range. During an undervoltage condition, the motor operates in overmodulation region to achieve the target speed leading to current distortion, inefficiency or noise. During an overvoltage condition, the MOSFETs and motor are stressed with continued operation in high voltage.

Tuning Under Voltage Limit 1: Keep decreasing the supply voltage until there is a drop in the speed. Measure the bus voltage at which the speed drops and set MIN_VM_MOTOR to that value. Range of minimum bus voltage that can be configured is between 0 to 25% of Maximum BASE_VOLTAGE.

Tuning Over Voltage Limit: Keep increasing the bus voltage to a point where the motor phase voltage reaches the maximum rated voltage of the motor. MAX_VM_MOTOR is the bus voltage at which motor phase voltage reaches the maximum rated voltage of the motor. Range of maximum bus voltage that can be configured is between 60% to Maximum BASE_VOLTAGE.

Note

The FOC Algorithm provides an undervoltage recovery mode [MIN_VM_MODE] and an overvoltage recovery mode [MAX_VM_MODE]. Undervoltage recovery mode can be configured to either automatically clear Undervoltage fault [MTR_UNDER_VOLTAGE] or latch on Undervoltage fault. Overvoltage recovery mode can be configured to either automatically clear Overvoltage fault [MTR_OVER_VOLTAGE] or latch on Overvoltage fault.

6.3.3 No Motor Fault [NO_MTR]

This fault is triggered when the phase current is below the No motor lock threshold % of base current.

Step 1: Make sure the motor phases are connected to the terminals as shown in the Hardware User Guide.

Step 2: If the fault persists, decrease the No-Motor lock current threshold [NO_MTR_THR].

7 Advanced Tuning

This section helps you spin motors successfully in closed loop with minimal configurations. This section provides standardized mandatory steps to tune parameters for successful motor spin-up in closed loop. Closed loop is defined as the sensorless closed loop where motor spins at the commanded speed and torque reference.

7.1 Control Configurations Tuning

7.1.1 Control Mode of Operation

FOC Application can be controlled in below four modes using the variable CONTROL_MODE in CLOSED_LOOP1 Register. The reference input for Speed/Power/Torque/Voltage is configured through the SPEED_CTRL register.

7.1.1.1 Closed Loop Speed Control Mode

This mode can be selected by setting the CONTROL_MODE in CLOSED_LOOP1 Register as 0h. In speed control mode, the speed of the motor (Electrical Hz) is controlled using a closed loop PI control according to the input reference set as SPEED_CTRL value in Speed Control Register (P.U value in IQ15 format). The P.U speed is computed as the ACTUAL_MOTOR_SPEED / MOTOR_MAX_SPEED value configured in SYSTEM_PARAMETERS.

Example: For MOTOR_MAX_SPEED set to 100Hz , Setting reference Input in SPEED_CTRL to 0x3FFFh (0.5 P.U in IQ15) sets the Motor Speed to 50Hz.

7.1.1.2 Closed Loop Power Control Mode

This mode can be selected by setting the CONTROL_MODE in CLOSED_LOOP1 Register as 1h. In power control mode, the input electrical Power of the motor (Watts) is controlled using a closed loop PI control according to the input reference set as SPEED_CTRL value in Speed Control Register (P.U value in IQ15 format). The P.U power is computed as the ACTUAL_MOTOR_POWER / MOTOR_MAX_POWER value configured in SYSTEM_PARAMETERS.

Example: For MOTOR_MAX_POWER set to 100Watts , Setting reference Input in SPEED_CTRL to 0x3FFFh (0.5 P.U in IQ15) operates the Motor at a constant power of 50W.

7.1.1.3 Closed Loop Torque Control Mode

This mode can be selected by setting the CONTROL_MODE in CLOSED_LOOP1 Register as 2h. In Torque control mode, the Torque Component current Iq of the motor (in Amps) is controlled using a closed loop PI control according to the input reference set as SPEED_CTRL value in Speed Control Register (P.U value in IQ15 format). The P.U Torque Component of Current is computed as the TORQUE_CURRENT_COMPONENT / CURRENT_BASE value configured in SYSTEM_PARAMETERS.

Example: For CURRENT_BASE set to 10 Amps , Setting reference Input in SPEED_CTRL to 0x3FFFh (0.5 P.U in IQ15) operates the Motor at a constant Iq current of 5A (Appropriate load to be connected to the motor).

7.1.1.4 Voltage Control Mode

This mode can be selected by setting the CONTROL_MODE in CLOSED_LOOP1 Register as 3h. In Voltage control mode, the modulation index of the motor is controlled according to the input reference set as SPEED_CTRL value in Speed Control Register (P.U value in IQ15 format).

Example: Setting reference Input in SPEED_CTRL to 0x3FFFh (0.5 P.U in IQ15) operates the Motor with a constant modulation Index of 0.5.

Lead Angle Control : In Voltage control mode , Lead Angle can be adjusted to derive the best efficiency from the motor for a given speed. LEAD_ANGLE configuration in CLOSED_LOOP2 register can be used to set the Lead Angle.

For a given LeadAngle θ, Appled Voltages Vq and Vd are defined as

Vq = MODULATION_INDEX * Cosθ

Vd = MODULATION_INDEX * Sinθ

Figure 7-1 can be used to set the above Control modes through GUI.

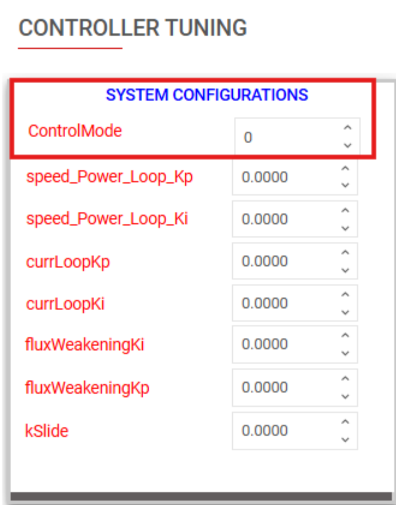


Figure 7-1. Control Mode Configuration

7.1.2 Initial Speed Detection of the Motor for Reliable Motor Resynchronization

The Initial Speed Detection (ISD) function is used to identify the initial condition of the motor. It is important to know the initial condition of the motor for reliable resynchronization. Motor resynchronization failures can occur when the device attempts to start the motor while the motor is coasting or spinning in the direction opposite to the intended direction of spin. Motors can coast in applications that require frequent motor starts and stops, if the motor is being forced externally, or if there is a power interruption. Motors can spin in the direction opposite to the intended direction of spin if motor phase wires are connected to OUTA, OUTB, and OUTC in wrong sequence or when the wrong direction command is issued. Motors with higher inertia coast for a longer period of time. Enable ISD in applications that require frequent motor starts and stops, and use higher inertia motors.

For example, ceiling fan motors have higher inertia due to the fan blades and can coast for a long time before stopping.

Step 1: Enable ISD [ISD_EN]

Step 2: Enable Motor ISD Resynchronize [RESYNC_EN]

Note

If the motor fails to start:

1. Increase the Motor Stationary BEMF Threshold [STAT_DETECT_THR].
2. Increase the Motor Stationary Persistence Time [ISD_STOP_TIME].
3. Increase the Motor Run Persistence Time [ISD_RUN_TIME].
4. Increase the minimum speed threshold to resynchronize to closed loop.

7.1.3 Unidirectional Motor Drive Detecting Backward Spin

For applications that require spinning the motor in a specific direction, it is important to know if the motor is coasting or spinning in the direction opposite to the intended direction of spin. The MSPM0 FOC Algorithm reverse drive function acts to reverse decelerate the motor through zero speed and to accelerate after changing direction until it transitions into closed loop as shown in Figure 7-2.

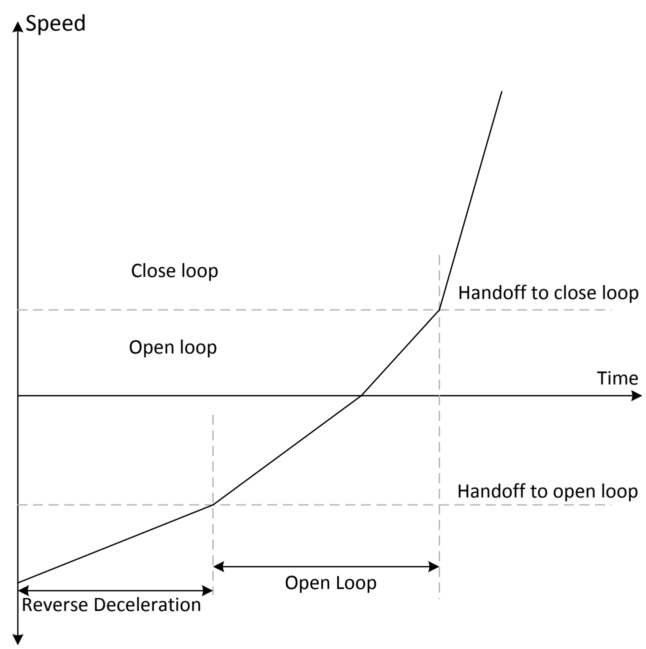


Figure 7-2. Reverse Drive Function

The MSPM0 FOC algorithm provides an option to apply brakes and stop the motor while the motor is coasting or spinning in reverse direction and then accelerate into closed loop after changing the direction.

In applications such as ceiling fans and pumps, it is required to spin the motor in a specific direction for desired results. For such applications, follow these recommendations:

Step 1: Enable ISD [ISD_EN]

Step 2: Enable Motor ISD Reverse drive [RVS_DR_EN]

Step 3: Enable reverse resynchronization [RESYNC_EN]

Note

Follow these recommendations if the motor fails to resynchronize in reverse direction:

1. Increase the reverse deceleration speed threshold to transition to open loop
2. Enable Open loop reverse drive configuration [REV_DRV_CONFIG]
3. Increase the Reverse Drive Open Loop Current Reference [REV_DRV_OPEN_LOOP_CURRENT]
4. Decrease open loop acceleration coefficient A1 and A2 during reverse drive

---

7.1.4 Preventing Back Spin of Rotor During Startup

For applications where a reverse spin is not acceptable, the Initial Position Detection algorithm (IPD) function is an alternative way to start up the motor. With the proper IPD setting, the motor startup can be faster than using align. While this function is suitable for motors with high inertia, such as heavy blades (for example: a ceiling or appliance fan), it is not suitable for motors with low inertia, such as small blades (for example: a computer fan), because the current injection can cause the motor to shake, resulting in the IPD not being accurate.

In applications where the acoustic noise ("chirp") generated by IPD is not acceptable during startup, select "Slow first cycle" as the startup method.

7.1.4.1 Option 1: IPD

Step 1: If IPD is chosen as startup method, select IPD in the Motor startup option [MTR_STARTUP] in "USER INPUT MTR_STARTUP1_T configuration of REGISTER MAP" tab in the GUI.

Step 2: Select the IPD Current threshold [IPD_CURR_THR]. IPD current threshold is selected based on the inductance saturation point of the motor. A higher current has better chance to accurately detect the initial position. However, higher current can result in rotor movement, vibration, and noise. Start with 50% of the rated current of the motor. If the motor startup is unsuccessful, increase the threshold until the motor starts successfully. Do not set the IPD current threshold higher than the rated current of the motor.

Step 3: Select IPD clock value [IPD_CLK_FREQ]. IPD clock defines how fast the IPD pulses are applied. Higher inductance motors and higher current thresholds need a longer time to settle the current, so set the clock at a slower time. However, a slower clock makes the IPD noise louder and last longer, so set the clock as fast as possible as long as IPD current is able to settle completely.

Note

The device triggers the IPD timeout fault IPD_FAULT_CLOCK_TIMEOUT for motors with very high inductance, or if the motor is not connected. If this fault is triggered, make sure that the motor is connected to the device. If the fault still persists, set the IPD release mode [IPD_RLS_MODE] to Tri-state if any overshoot in DC bus voltage is acceptable.

The device triggers the IPD Frequency fault IPD_FAULT_DECAY_TIME if the IPD clock frequency is set too high. If this fault is triggered, decrease the IPD Clock value [IPD_CLK_FREQ].

Step 4: Select IPD Advance Angle [IPD_ADV_ANGLE]. Start with 90º for maximum startup torque. If there is sudden jerk observed during startup, reduce the angle to 60º or 30º for a smoother startup.

7.1.4.2 Option 2: Slow First Cycle

Follow these steps if Slow first cycle is chosen as the startup method:

Step 1: Select Slow first cycle in the Motor startup option [MTR_STARTUP] in "Control Configuration – Motor Startup Stationary" tab in the GUI.

Step 2: Select align or slow first cycle current reference [ALIGN_OR_SLOW_CURRENT_ILIMIT]. Lower current reference may lose synchronization of motor. Higher current may lead to sustained oscillations for high inertia motors, or sudden jerky motion for low inertia motors. It is recommended to start with 50% of the rated current of the motor. In applications where the startup torque is high, the motor might lose synchronization. In such applications, increase the current reference. In applications where sustained oscillations or sudden jerks are observed, decrease the current reference.

Step 3: Select align or slow first cycle current ramp rate [ALIGN_SLOW_RAMP_RATE]. Current reference is ramped to avoid reverse rotation of the motor. Lower current ramp rate may lose synchronization of motor. A higher current ramp rate may lead to sustained oscillations for high inertia motors, or sudden jerking motion for low inertia motors. Start with setting the ramp time to 0.5 seconds to ramp to rated current of the motor. In applications where the startup torque is high, the motor can lose synchronization. In such applications, increase the current ramp rate. In applications where sustained oscillations or sudden jerks are observed, decrease the current ramp rate.

Step 4: Select the frequency of the first cycle [SLOW_FIRST_CYC_FREQ]. Lower frequency can give a jerky motion at startup. Higher frequency might not be able to synchronize the motor. Start with 20% of the maximum speed of the motor. In applications where the startup torque is high, the motor might lose synchronization. In such applications, decrease the frequency. In applications where jerky motions are observed, increase the frequency.

7.1.5 Gradual and Smooth Start up Motion

For applications that require slow and gradual startup with lower speed overshoots during handoff, follow the below recommendations:

Step 1: Decrease Open loop acceleration coefficient A1 [OL_ACC_A1] and Open loop acceleration coefficient A2 [OL_ACC_A2].

Step 2: Enable Iq ramp down after transition to closed loop [IQ_RAMP_EN]

If there is speed overshoots, decrease ramp rate for reducing difference between estimated theta and open loop theta [THETA_ERROR_RAMP_RATE].

7.1.6 Faster Startup Timing

Startup time is the time taken for the motor to reach the target speed from zero speed. For applications that require quick startup time, we recommend choosing either Initial Position Detection (IPD) or Slow first cycle as the startup method.

7.1.6.1 Option 1: Initial Position Detection (IPD)

Step 1: Select IPD [MTR_STARTUP] as the motor startup method.

Step 2: Increase IPD current threshold [IPD_CURR_THR] to rated current of the motor.

Step 3: Increase IPD clock value [IPD_CLK_FREQ] to higher frequency up to a value where the device does not trigger IPD frequency fault. Check Section 7.1.4 (Step 3) for more details.

Step 4: Select IPD repeating times [IPD_REPEAT] to 1 time.

Step 5: Select Open loop current limit [OL_ILIMIT] to be the same as Current limit for Torque PI Loop [ILIMIT].

Note

Configuring current Limits to a value higher than motor stall current overheats or damages the motor.

Step 6: Increase Open loop acceleration coefficient A1 [OL_ACC_A1] and Open loop acceleration coefficient A2 [OL_ACC_A2].

Step 7: Select Minimum BEMF for handoff [AUTO_HANDOFF_MIN_BEMF] to 0mV.

If the device triggers Abnormal BEMF [ABN_BEMF] fault, then recommended to increase the [AUTO_HANDOFF_MIN_BEMF].

Step 8: Keep increasing ramp rate for reducing difference between estimated theta and open loop theta to 2 deg/ms.

Step 9: Increase Closed loop acceleration rate [CL_ACC]

7.1.6.2 Option 2: Slow First Cycle

Step 1: Select Slow first cycle as the motor startup method in [MTR_STARTUP].

Step 2: Select Align or slow first cycle current limit [ALIGN_OR_SLOW_CURRENT_ILIMIT] to be the same as Current limit for Torque PI loop [ILIMIT].

Step 3: Keep increasing Align or slow first cycle current ramp rate [ALIGN_SLOW_RAMP_RATE] until the open loop current reaches 100% of the rated current of the motor.

Step 4: Follow Step 5 to Step 9 in Option 1.

7.1.7 Stopping Motor Quickly

For applications that require stopping the motor quickly, configure Motor stop options [MTR_STOP] to either Low side braking:

Step 1: Configure Motor stop options [MTR_STOP] to Low side braking.

Step 2: Select Speed threshold for BRAKE pin and Motor STOP options. Setting speed threshold to higher speed can result in FETs carrying large current. Setting speed threshold to lower speed results in increase in the stop time of the motor. Recommended to start with 50% of the maximum speed, If the motor phase current exceeds the FET maximum current rating, then decrease the threshold. If the stop time is too high, then recommended to increase the threshold without hitting the maximum current limit.

7.1.8 Flux Weakening: Operating Motor at Speeds Higher than Rated Speed

The FOC algorithm provides control for adjusting the rotor flux by changing the flux current component Id. Reducing the rotor flux enables motor to enter the field weakening zone through which motor speed can go beyond rated speed.

Note

During flux weakening operation, the motor cannot deliver the rated torque. The torque limit Iq is automatically adjusted based on the circular motor current limit defined by $I_{LIMIT} = Id^2 + Iq^2$.

Steps to enable the flux weakening:

1. Set the FLUX_WEAK_EN bit in FieldCtrl register as 1b.
2. Adjust FLUX_WEAK_CURR_RATIO to limit the maximum flux component of current to torque component current ratio. This value limits the flux component current Id and maintain the torque component current Iq based on the circular limit ILIMIT as $Id^2 + Iq^2$.
3. Maximum modulation index beyond which the field weakening is enabled can be tuned using FLUX_WEAKE_REF configuration. This register field values sets the square of modulation index value above which the Id is regulated to weaken the flux.

Note

Entering field weakening is not efficient below rated speeds. Field weakening is recommended to be activated only when the modulation index limit is reached and no longer be able to meet the desired speed requirement with the sine modulation.

7.1.9 Maximum Torque Per Ampere : Improve Efficiency of IPMSM Motors

The FOC algorithm enables users to achieve maximum efficiency for motors with saliency (IPM motors). User can configure the saliency of the motor as nonzero value as detailed in saliency parameter description.

Users can enable this feature by configuring the MTPA_EN in the Section 5.3.7.

7.1.10 Preventing Supply Voltage Overshoot During Motor Stop.

For applications that require preventing supply voltage overshoots during motor stop, select active spin down as Motor stop options. Active spin down can be used as a motor stop option in applications where fast stop is not required but some amount of inductive energy going back to power supply is acceptable

Step 1: Configure Motor stop options [MTR_STOP] to Active spin down

Step 2: Configure active spin down speed threshold [ACT_SPIN_THR]. It is recommended to set the ACT_SPIN_THR to 50% of the maximum speed. If there is voltage overshoot seen on the power supply, decrease the ACT_SPIN_THR till the voltage overshoot reaches acceptable limit.

7.1.11 Protecting the Power Supply

Protecting the power supply from drawing higher current or potential voltage overshoots is important in battery operated applications or applications that do not have an internal overcurrent or overvoltage protection built into the power supply.

Step 1: When the load on the motor increases, the device draws higher current from the power supply. To limit the current drawn from the power supply, enable bus current limit [BUS_CURRENT_LIMIT_ENABLE] and configure the bus current limit [BUS_CURRENT_LIMIT] to protect the power supply from drawing higher current.

For example, limiting the current drawn from power supplies such as batteries is required because the battery life depends on the charge and discharge cycles. Enabling the bus current limit limits the power supply current by limiting the speed of the motor.

Step 2: When a command is issued for the motor to decelerate, based on the deceleration rate, energy from the motor can be pumped back to the power supply, increasing the supply voltage to levels that are possibly unsafe for electronics. Enable the antivoltage surge [AVS] to protect the power supply from voltage overshoots that override any deceleration limit set by any other register and automatically apply a safe deceleration rate.

Figure 7-3 shows overshoot in power supply voltage when AVS is disabled. Motor decelerates from 100% duty cycle to 10% duty cycle at a deceleration rate of 70000Hz/s. Figure 7-4 shows no overshoot in power supply voltage when AVS is enabled.

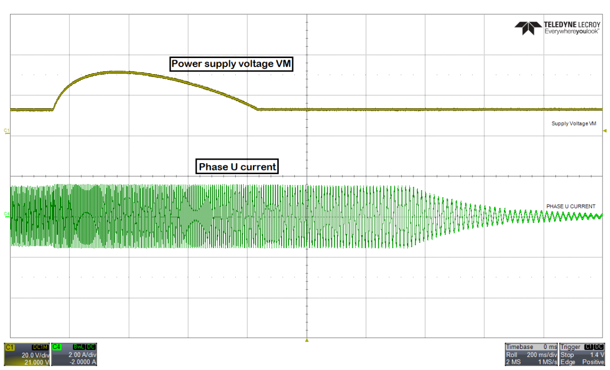


Figure 7-3. Power Supply Voltage and Phase Current Waveform When AVS is Disabled

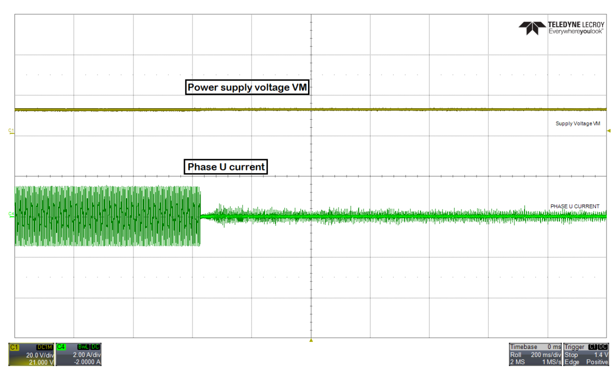


Figure 7-4. Power Supply Voltage and Phase Current Waveform When AVS is Enabled

7.1.12 FOC Bandwidth Selection

The FOC algorithm is periodically executed in the interrupt routine to update the rotor angle to extract the maximum efficiency from the motor. This FOC rate can be configured by user based on the application bandwidth requirements.

HIGH_FREQ_FOC_EN bit configuration in the Section 5.3.5 can be set to 1b to get the maximum FOC execution rate of 10kHz. Setting this bit to 0b reduces the maximum FOC execution rate by 2.

Note

FOC routine can only be executed at a multiple of PWM frequency, Hence, the maximum achievable FOC rate of 10kHz is applicable for PWM frequencies with multiple of 10 (for example, 10kHz, 20kHz, 30kHz). For PWM frequencies of 15kHz, 45kHz, and so on, the maximum FOC rate is 7.5kHz (15kHz/2, 45kHz/6, and so on).

---

8 Hardware Configurations

8.1 Direction Configuration

The FOC algorithm lets you set the direction of the motor using a register-based direction configuration:

- **Register-based direction configuration**

The direction of motor spin can be set based on register setting as shows below:

- **DIR_INPUT 01b**: apply phase sequence OUT A → OUT B →OUT C.
- **DIR_INPUT 10b**: apply phase sequence OUT A → OUT C → OUT B.

8.2 Brake Configuration

FOC Algorithm enables user to brake the motor under various scenarios. Brake state can be configured to either low side braking (Low-Side Braking) or align brake (Align Braking) through *BRAKE_PIN_MODE*. FOC Algorithm decreases output speed to value defined by *BRAKE_SPEED_THRESHOLD* before entering brake state. As long as BRAKE is driven 'High', motor stays in brake state. Brake functionality can be achieved the following way:

- **Register based Brake configuration**.

  User can configure the *BRAKE_INPUT* in *PIN_CONFIG* register to apply the brakes using register settings as below.

  – *BRAKE_INPUT - 1b*: Override pin and brake / align according to *BRAKE_PIN_MODE*
  – *BRAKE_INPUT - 10b*: Override pin and do not brake / align

8.3 Main.h Definitions

8.3.1 Sense Amplifier Configuration

Sense amplifier configuration defines the direction of CSA output. Decreasing CSA Output for a Positive current coming out of phase indicates the Sense Amplifier is inverting. Increasing CSA output for a Positive current indicates Non Inverting Current sense amplifier.

For Inverting amplifier configuration, **#define _INVERT_ISEN** has to be included in the main.h.

For Non Inverting amplifier configuration, **#define _NONINVERT_ISEN** has to be included in the main.h file

8.3.2 Driver Propagation Delay

Driver propagation delay defines the Time delay in ns between the Input PWM logic edge fed to the gate driver and actual Gate Driver output. This delay impacts the Current Sense sampling instance on the actual gate driver output and has to be fed to the algorithm for accurate Current Sensing.

This value in ns has to be defined using **#define DRIVER_PROPAGATION_DELAY_nS** macro in the main,h file.

8.3.3 Driver Min On Time

Driver Min on time defines the combined rise time and settling time of the current sense amplified output. This value has to be captured independently for a full scale change in voltage across the current shunt. For accurate current sense reading, the current sense amplifier output to be settled before the current signal is captured.

This CSA Settling + Rise time is to be set using **#define DRIVER_MIN_ON_TIME_nS** macro in the main,h file.

8.3.4 Current Shunt Configuration Selection

The SDK FOC example can be configured for various shunt configuration options such as Single Shunt, Dual Shunt and Three Shunt. Based on the HW design the appropriate shunt configuration has to be selected for proper operation of algorithm.

FOC Application supports simultaneously sampling the two phases at a given instance to optimize the current sampling time. By default in all shunt configurations, both the ADC instances are used to utilize the simultaneous sampling feature. User needed to route at least one current sense onto each of the two ADC instance channels and appropriate shunt configuration has to be defined in the main.h configuration as below.

8.3.4.1 Three Shunt Configurations

#define CURRENT_THREE_SHUNT_AB_C : Select this configuration if A and B phases are sensed through ADC0 and C phase is sensed through ADC1.

#define __CURRENT_THREE_SHUNT_A_BC : Select this configuration if A phase is sensed through ADC0 and B, C phases are sensed through ADC1.

User can also route one of the Phases say 'B' to both the ADC 0 and 1 instance and the other two phases to two different ADC instances. For example say 'phase A' is routed to ADC0 and Phase 'C' is routed to ADC1, Phase B is routed to both ADC0 and ADC1 instances, In this example, algorithm can dynamically switch to the two samples which gives the best current sampling time based on the given sector.

In this three shunt configuration, application supports shifting the current sensing estimation dynamically to the two phases for maximizing the modulation index. As in a balanced three phase Motor, any one of the phase current can be estimated with the other two phase currents as $i_a = -(i_b + i_c)$. Based on the operational sector the two phases with lowest modulation index are selected for current measurement and third phase with highest modulation index is estimated using the other two phase currents. This method helps in extending the modulation index to higher limits with continuous SVM operation.

To select this configuration user can include **#define __CURRENT_THREE_SHUNT_DYNAMIC** macro in the main.h file. Along with this user need to enable the dynamic shunt selection by setting the macro **#define DYNAMIC_CURRENT_SHUNT_CONFIG_EN** to **TRUE**.

8.3.4.2 Dual Shunt Configuration

#define __CURRENT_TWO_SHUNT_A_B : Select this configuration if only two shunt sense across phase A and B are available current sampling where A is channeled to ADC 0 and B to ADC1.

#define __CURRENT_TWO_SHUNT_B_C : Select this configuration if only two shunt sense across phase B and C are available current sampling where B is channeled to ADC 0 and C to ADC1.

#define __CURRENT_TWO_SHUNT_C_A : Select this configuration if only two shunt sense across phase A and B are available current sampling and A is channeled to ADC 0 and C to ADC1.

Note

If user has a different combination of phases routed to the ADC 0 and 1 instances than the default connections in SDK. Appropriate changes can be done in following file: *projectroot/modules/hal/gateDriverInterface/source/<driverSpecific>_focHalInterface.c*

8.3.4.3 Single Shunt Configuration

If a single shunt is used in the HW for current sensing, below definitions are to be included for proper motor operation.

#define __CURRENT_SINGLE_SHUNT macro to be included in the main.h file.

8.3.5 CSA Offset Scaling Factor Selection

FOC application converts the sampled currents through ADC into PU system based on the maximum current that can be sensed through the ADC. This depends on the CSA offset introduced from the amplifier. Typically for bipolar current sense measurement, full scale value of ADC 3.3V / 2 = 1.65V is given as offset. For applications where the current sensing is always unipolar, offset values are set less than 0.5V to use the maximum full scale ADC output for +ve current measurement and small margin is left for -ve current measurement.

FOC application requires this scaling to be specified for appropriate functionality. As the ADC 12-bit value is converted to PU value, if the offset is set as 0 : Then the scaling factor to be set as _IQ(1).

If the CSA offset in HW is set as 1.65V (3.3V / 2) for bipolar current sense measurement the scaling factor to be set as _IQ(2).

For any arbitrary offset values, the scaling values to be specified as

_IQ(3.3v/(3.3v - CSA_OFFSET in volts)). This value to be added as macro definition in the *projectroot/modules/hal/gateDriverInterface/source/<driverSpecific>_focHalInterface.c*.

For example : Refer to example project DRV8329 - where #define DRV8329_CURRENT_SF_IQ is specified as _IQ(1.42857142) for a CSA offset of 0.4125V.

8.4 Real-Time Variable Tracking

32-bit algorithm variables can be output in real time from the MCU through the DAC. DAC output is enabled by setting DAC_EN = 1. The DAC in MSPM0 is 12 bit, thus a scaling needs to be applied before output. User has two ways to scale the variable before output.

- **For variables in global IQ format(IQ27):**

  $$\text{DAC\_OUTPUT\_VOLTAGE} = (\text{VARIABLE\_VALUE} \times \text{DAC\_SCALING\_FACTOR} + 1) \times 1.65\text{V} \quad (20)$$

  In the above equation setting the DAC_SCALING_FACTOR to 1 enables user to represent a data of IQ(1.0) to IQ(-1.0) in between 0V and 3.3V. To represent the data exceeding the value 1.0 use lower DAC_SCALING_FACTOR.

  For Example: To represent a data from -2.0 to +2.0 , set the DAC_SCALING_FACTOR to 0.5.
- **For variables in other IQ format:**

  For output of any other IQ, user can left shift or right shift the variable to bring the data in a 12-bit range before output. This mode is selected by setting DAC_SCALING_FACTOR to 0.

  If variable value is less than a 12-bit value, set DAC_SCALE to positive, the DAC output follows as shown in Equation 21:

  $$\text{DAC\_OUTPUT\_VOLTAGE} = (\text{VARIABLE\_VALUE} \ll \text{DAC\_SCALE}) \times 3.3\text{V} \quad (21)$$

  If variable value is greater than a 12-bit value, set DAC_SCALE to negative, the DAC output follows as shown in Equation 22:

  $$\text{DAC\_OUTPUT\_VOLTAGE} = (\text{VARIABLE\_VALUE} \gg \text{DAC\_SCALE}) \times 3.3\text{V} \quad (22)$$

Note

Settings DAC_EN = 1 feeds the variable output to the DAC registers, but user needs to enable the DAC peripheral in TI SysConfig for the DAC peripheral to function. Also make sure the DAC output pin is not loaded by any other peripheral.

Table 8-1. Address Table for DAC Monitoring

Variable	Address
A phase current	0x20200608
B phase current	0x2020060C
C phase current	0x20200610
A phase current raw ADC value	0x20200614
B phase current raw ADC value	0x20200618
C phase current raw ADC value	0x2020061C
A phase voltage	0x20200678
B phase voltage	0x2020067C
C phase voltage	0x20200680
A phase voltage raw ADC value	0x20200684
B phase voltage raw ADC value	0x20200688
C phase voltage raw ADC value	0x2020068C
D axis current	0x20200750
Q axis current	0x20200754
D axis voltage	0x20200758
Q axis voltage	0x2020075C
D axis Filtered BEMF	0x20200BD4
Q axis Filtered BEMF	0x20200BD8
Estimated motor velocity filtered	0x20200BF4
Estimated rotor angle	0x20200BFC
Power Feedback	0x20200930
SVM output duty A phase	0x20200720
SVM output duty B phase	0x20200724
SVM output duty C phase	0x20200728

9 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision * (January 2025) to Revision A (March 2025) **Page**

- Updated the Firmware version.......... 1
- Added support for MSPM0G3519 Devices to Section 2.1.1.......... 5
- Added support for MSPM0G3519 devices to Section 2.4.......... 6
- Added support for MSPM0G3519 devices to Section 2.5.......... 8
- Added support for MSPM0G3519 devices to Section 2.10.......... 10