User's Guide

MSPM0 Sensored FOC

ABSTRACT

This tuning guide provides step-by-step guidance to set up an MSPM0 MCU and supported DRV hardware board to tune and spin 3-phase brushless DC motor using Sensored FOC Algorithm.

Note

This Tuning guide is in reference to the Sensored FOC version **1.02.00** from SDK Version **2.04.00.00** and above.

Table of Contents

List of Figures

List of Tables

Trademarks

LaunchPad™ and Code Composer Studio™ are trademarks 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 Hall Sensor embedded 3-phase brushless DC (BLDC) motor with FOC control. The BLDC motor is driven by a three-phase brushless DC (BLDC) MOSFET gate driver or integrated MOSFET motor driver at 12V or 24V nominal DC rails or battery-powered applications. 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.

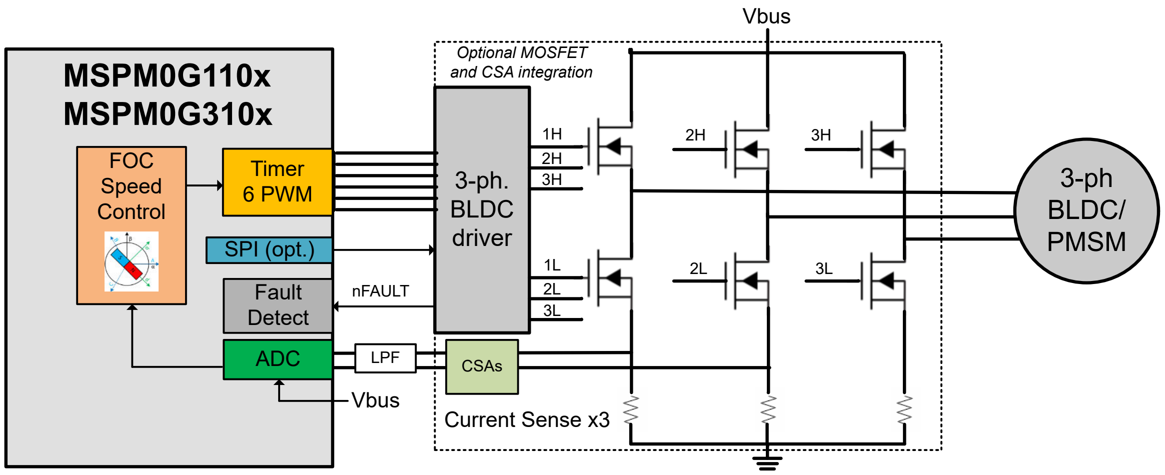


Figure 1-1. Simplified Schematic of MSPM0Gxxx + BLDC Motor Driver

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.
- ***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)
  - DRV8316REVM
- Jumper wires for pin table connections

- A computer with the MSPM0 Latest SDK software installed
- A BLDC motor with Hall Sensor 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 Sensored 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 Sensored FOC block diagram.

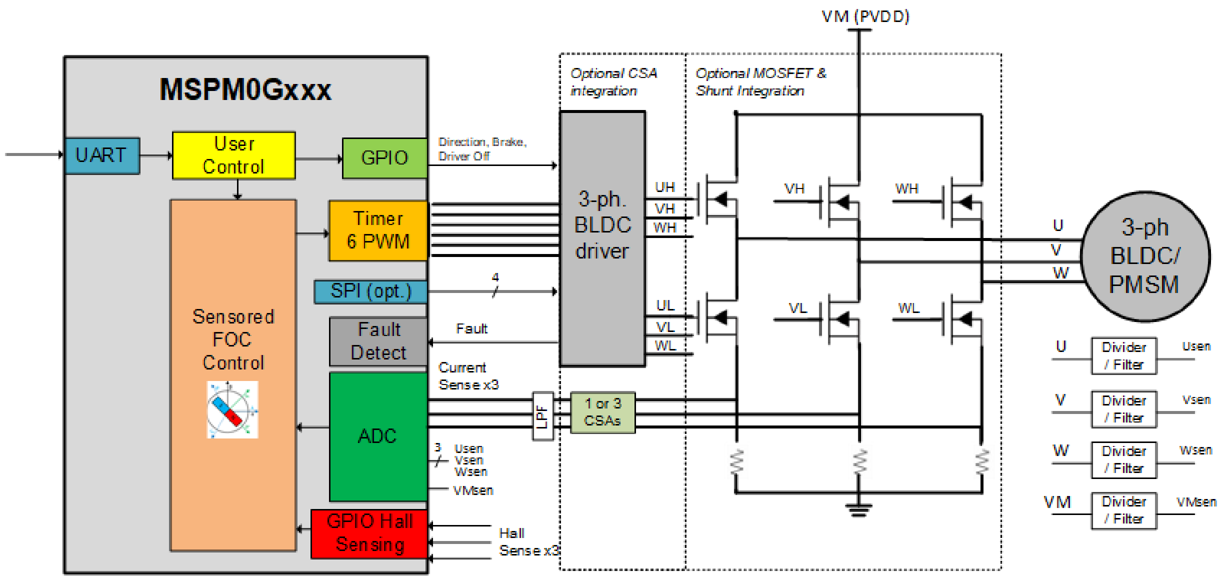


Figure 2-1. MSPM0Gxxx + BLDC Motor Driver - Sensored FOC Block Diagram

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 Sensored FOC motor control.

For supported evaluation boards, see 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, see 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 Sensored FOC motor control.

Table 2-1. Supported Hardware for Sensored 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	DRV8316REVM	DRV8316REVM User's Guide	3	Yes	4.5V to 35V	< 80W
MSPM0G1507	TIDA-010251 Reference Design	TIDA010251-Design-Guide	1	No	21V maximum DC supply	600W

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 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 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 can be configured to provide three complimentary pairs of PWM outputs (such as TIMA0_C1 and TIMA0_C1N) and is the preferred timer for motor control applications. 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

2.3 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. 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.

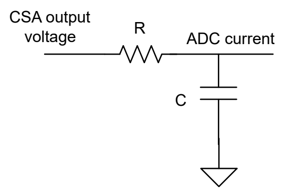


Figure 2-2. CSA Output Filter

Choose a filtering frequency f_c that it 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 input
A0_2	ADC0, channel 2 input	SOB	Phase B current sense input
A1_2	ADC1, channel 2 input	SOB	Phase B current sense input
A1_1	ADC1, channel 1 input	SOC	Phase C current sense input

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 input
A1_14	ADC1, channel 14 input	SOB	Phase B current sense input
A0_12	ADC0, channel 12 input	SOB	Phase B current sense input
A0_2	ADC0, channel 2 input	SOC	Phase C current sense input

ADC configuration for single shunt current sensing: The ADC pin configurations for single shunt current sensing with TIDA010251 is shown in ADC Pin Configuration for Single Shunt Current Sensing in TIDA010251.

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 needs to configure appropriate ADC to sample the Current sense output from the Memory '0' index, Memory '1' Index for FOC operation. for details on ADC configurations for Single shunt current sensing, see the Sensored FOC User Guide.

Table 2-5. ADC Pin Configuration for Single Shunt Current Sensing in TIDA010251

MSPM0 Pin	Function	Connection	Function
A1_13	ADC 1, Channel 13 Input	Internal Amplifier in Buffer Mode	DC bus current sense

2.4 Pin Configurations for ADC Voltages

The default pin configurations for ADC voltages are shown in the following tables. In Sensored FOC application, one ADC input is to be connected to the sensed VM motor voltage (VSENVM) for Fault sensing.

The sensed voltage is 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 6.1.2.5.

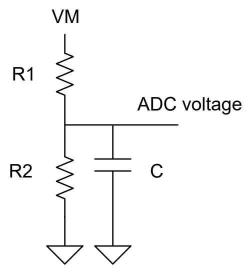


Figure 2-3. ADC Voltage Divider

Table 2-6. Pin Configurations for ADC DC Bus Voltage Sensing for DRV8316 + LP- MSPM0G3507

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

Table 2-7. Pin Configurations for ADC DC Bus Voltage Sensing for DRV8316 + LP- MSPM0G3519

MSPM0 Pin	Function	DRV Connection	DRV Function
A0_4	ADC0, channel 4 input	VSEN -Vm	DC bus voltage output

Table 2-8. Pin Configurations for ADC DC Bus Voltage Sensing for TIDA010251

MSPM0 Pin	Function	HW Connection	DRV Function
A0_1	ADC0, channel 1 input	VSEN -Vm	DC bus voltage output

2.5 Pin Configurations for Hall Sensor Inputs Through GPIO

Sensored FOC requires digital Hall Sensor data from BLDC / PMSM motor to operate the motor in Closed loop speed control and drive the motor efficiently. The three pin Hall Sensor data is to be fed through general-purpose input/output (GPIO) inputs and Events are generated based on the changes in GPIO levels. All the three GPIO pins are to be connected to the same port and a timer capture event is to be triggered based on any of the three GPIO input level changes. Table 2-9 summarizes the GPIO pin configurations with MSPM0 functionality on two HW boards DRV8316 and TIDA010251. Typically, the Hall signals need external pull up to drive the GPIO inputs.

Table 2-9. Pin Configurations for Hall Sensor GPIO Pins on DRV8316

MSPM0 Pin	MSPM0 Function
GPIO_IN_PA10	GPIO Port A Input Triggers TIMG12 Capture Event
GPIO_IN_PA11	GPIO Port A Input Triggers TIMG12 Capture Event
GPIO_IN_PA12	GPIO Port A Input Triggers TIMG12 Capture Event

Note

By default PA10 and PA11 on MSPM0G3507 LP are connected to backchannel UART pins through XDS. Shift the jumpers on J21 and J22 to pin positions 2-3 to enable the GPIO functionality on these pins through Boosterpack.

Table 2-10. Pin Configurations for Hall Sensor GPIO Pins on TIDA010251

MSPM0 Pin	MSPM0 Function
GPIO_IN_PA1	GPIO Port A Input Triggers TIMG12 Capture Event
GPIO_IN_PA2	GPIO Port A Input Triggers TIMG12 Capture Event
GPIO_IN_PA3	GPIO Port A Input Triggers TIMG12 Capture Event

2.6 Pin Configurations for Faults

The default pin configurations for faults are shown in Table 2-9. 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

For the Enable/DRVOFF functionality, see the motor driver device-specific data sheet.

2.8 Pin Configurations for SPI Communication

The default pin configurations for serial peripheral interface (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 2-11. 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 2-12. 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 DRV8316 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, leave these wires unconnected. Connect the Three Hall sensor Inputs to the header J7.
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 with DRV8316 using USB to backchannel UART, connect the backchannel UART connections to UART3_TX and UART3_RX as shown in Figure 2-5.

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 recommended voltage range, see the board-specific user's guide or DRV-specific data sheet.

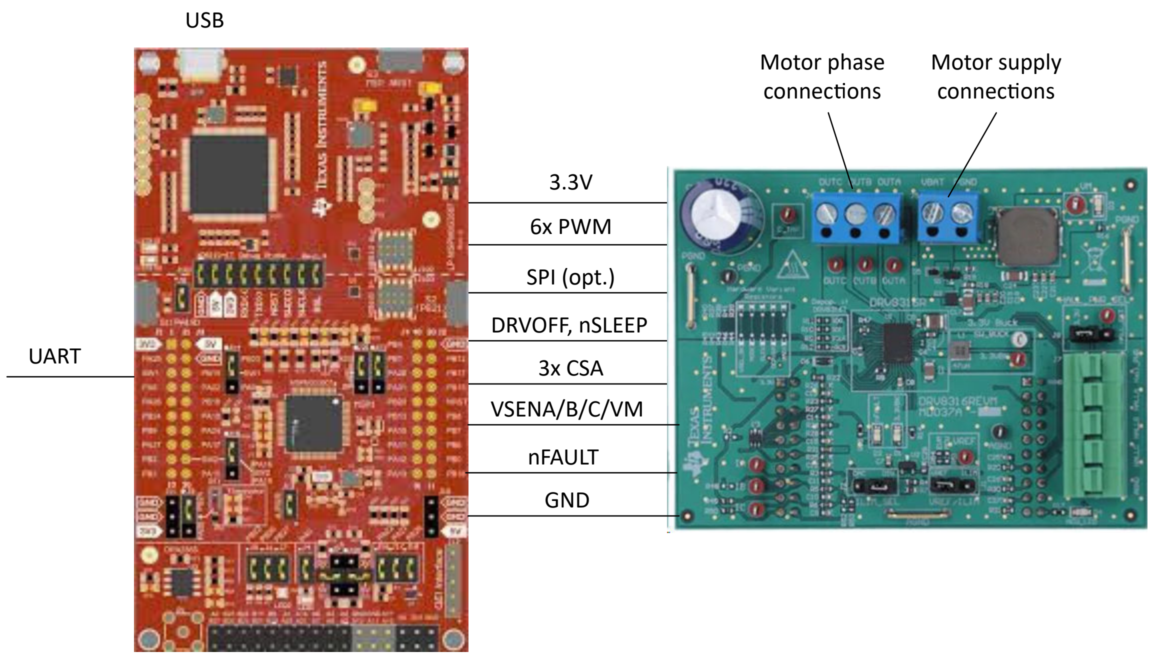


Figure 2-4. MSPM0 LaunchPad Kit and DRV8316 EVM External Configuration

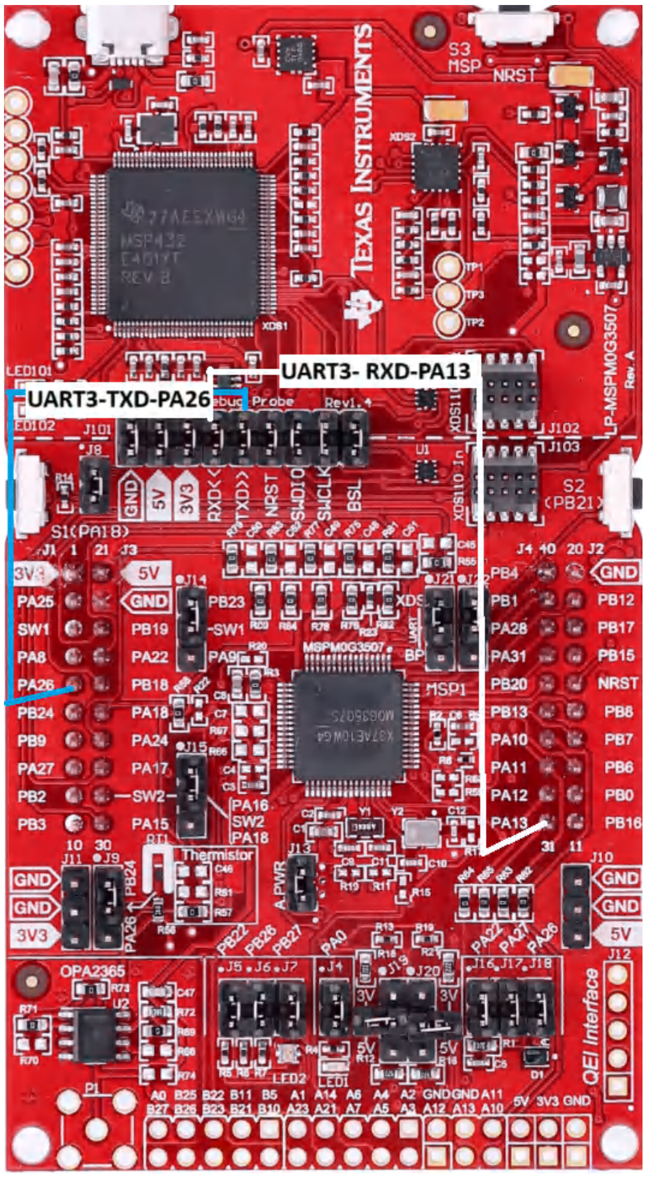


Figure 2-5. LP-MSPM0G3507 Backchannel Connection to UART3

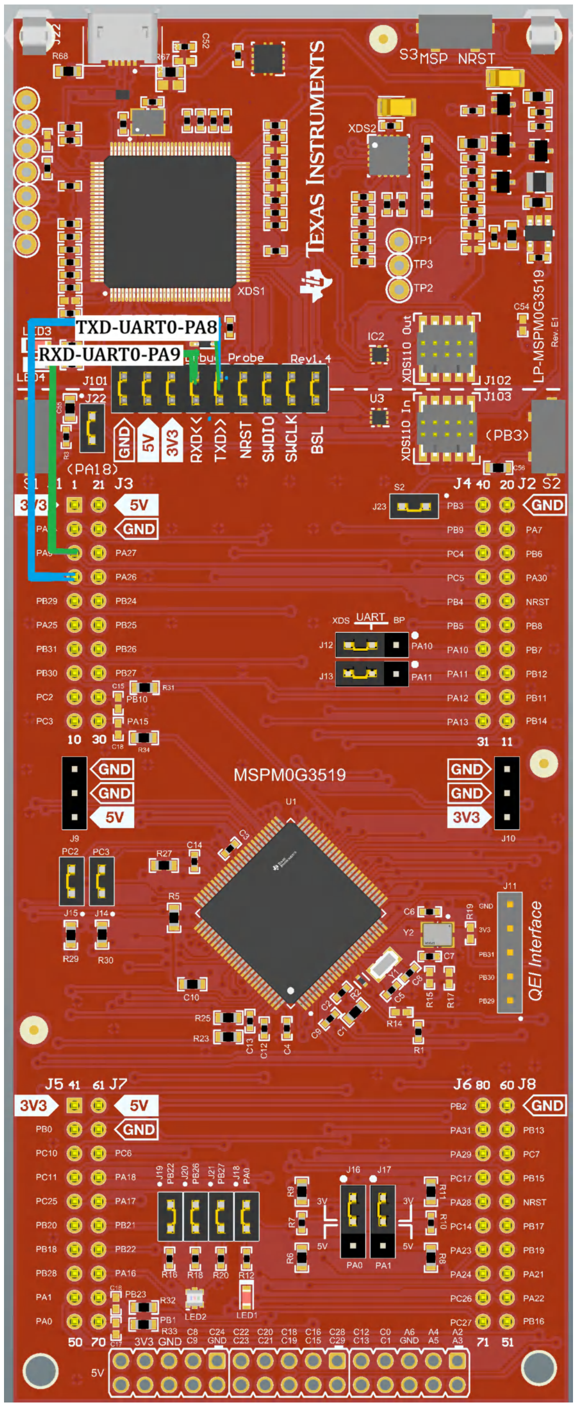


Figure 2-6. LP-MSPM0G3519 Backchannel Connection to UART0

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

Sensored FOC software for MSPM0 MCUs is provided inside MSPM0-SDK and example projects are available for evaluation with Code Composer Studio™ IDE.

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

Table 3-1. Software Support for FOC Control

Sensored FOC User's Guide[1]	Code Examples	GUI
Sensored FOC User's Guide	Sensored FOC Examples	MSPM0G Sensored FOC GUI

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

4 GUI Setup

The user can optionally use the MSPM0 Sensored 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.

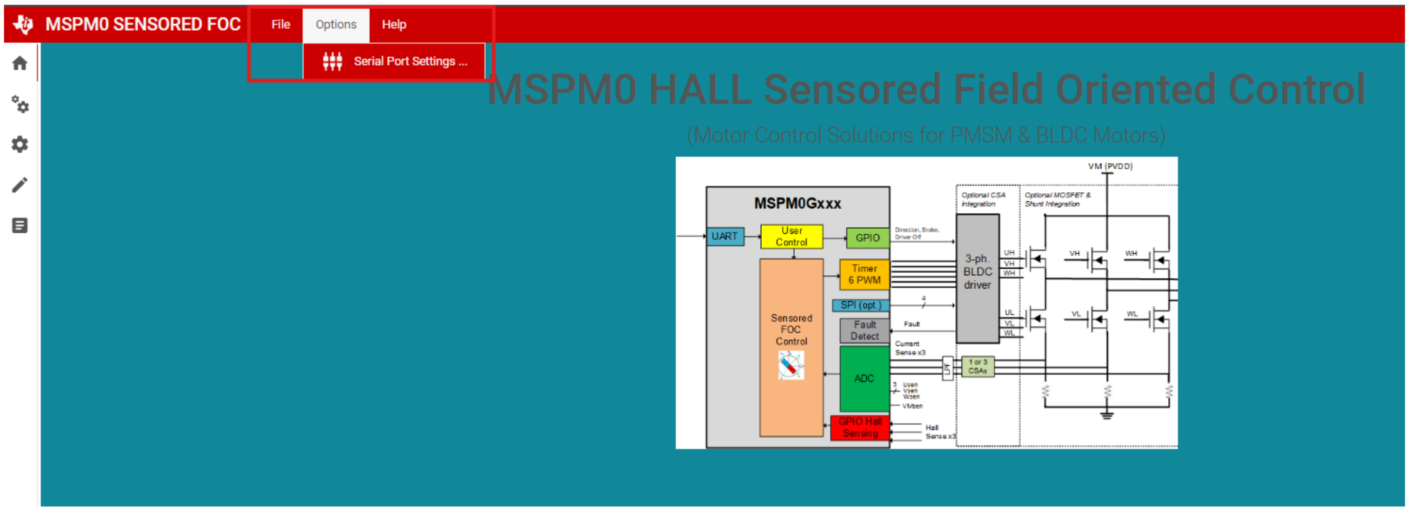
Table 4-1. GUI Connection Types

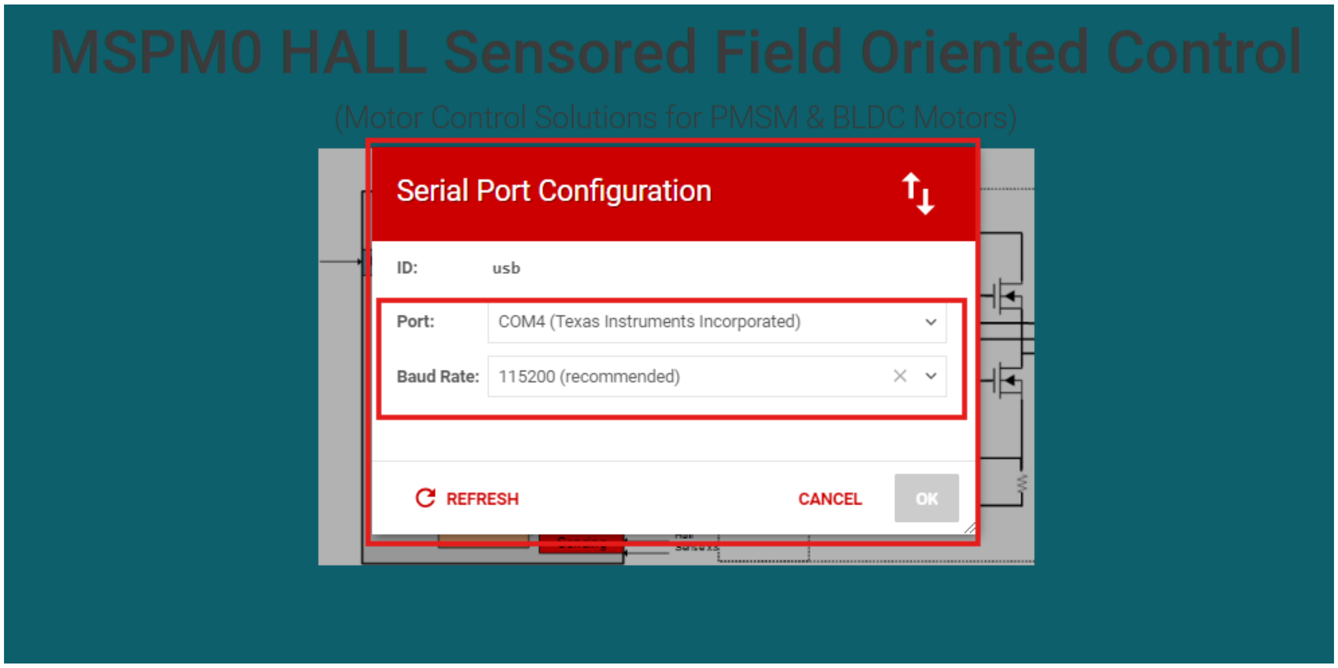
Connection	Interface	Hardware Connections
GUI to target MSPM0 MCU	UART	UARTn_TX, UARTn_RX (***n*** : available UART peripherals based on the device)

To launch the GUI, go to the MSPM0 Sensored 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.




MSPM0 HALL Sensored Field Oriented Control
(Motor Control Solutions for PMSM & BLDC Motors)
Serial Port Configuration
REFRESH
CANCEL
OK

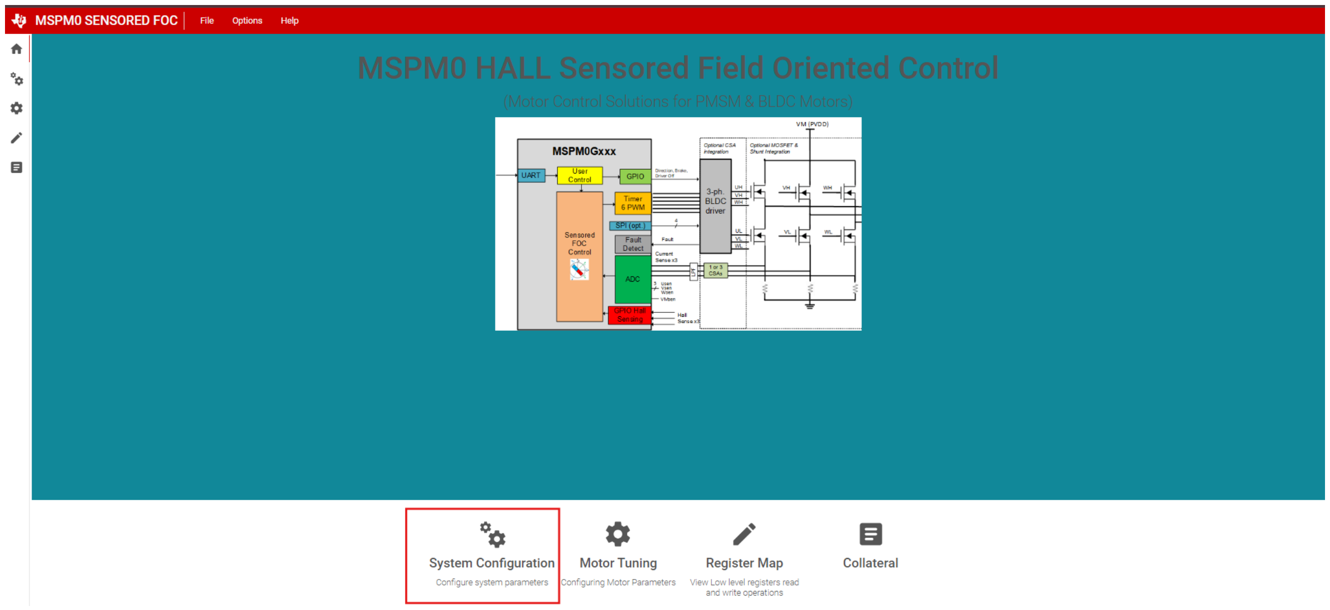
ID: usb
Port: COM4 (Texas Instruments Incorporated)
Baud Rate: 115200 (recommended)

4.2 GUI Home Page

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

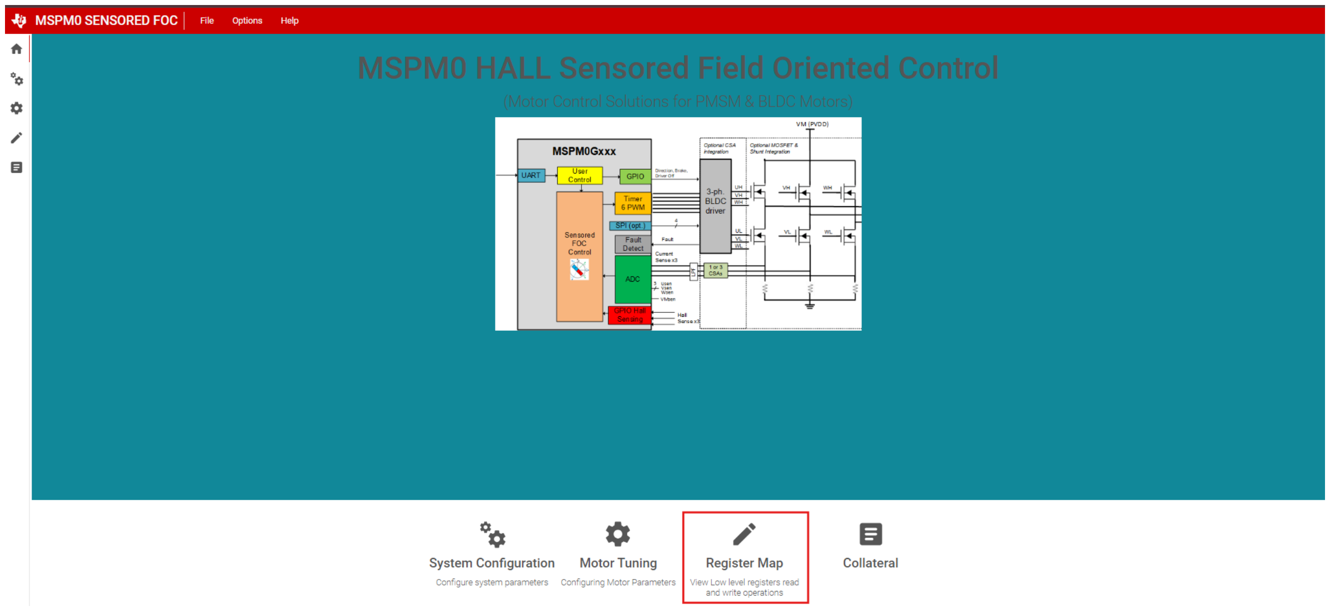
4.2.1 System Configurations

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



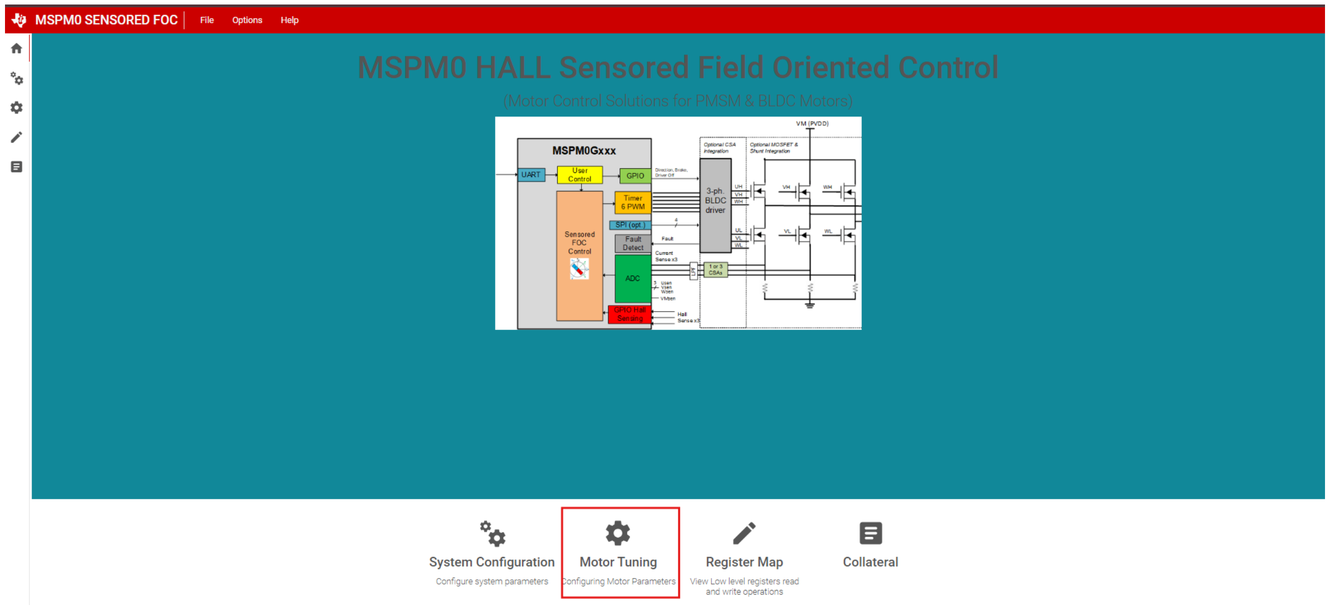
4.2.2 Register Map

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


MSPM0 SENSORED FOC
File
Options
Help
MSPM0 HALL Sensored Field Oriented Control
(Motor Control Solutions for PMSM & BLDC Motors)
System Configuration
Configure system parameters
Motor Tuning
Configuring Motor Parameters
Register Map
View Low level registers read and write operations
Collateral

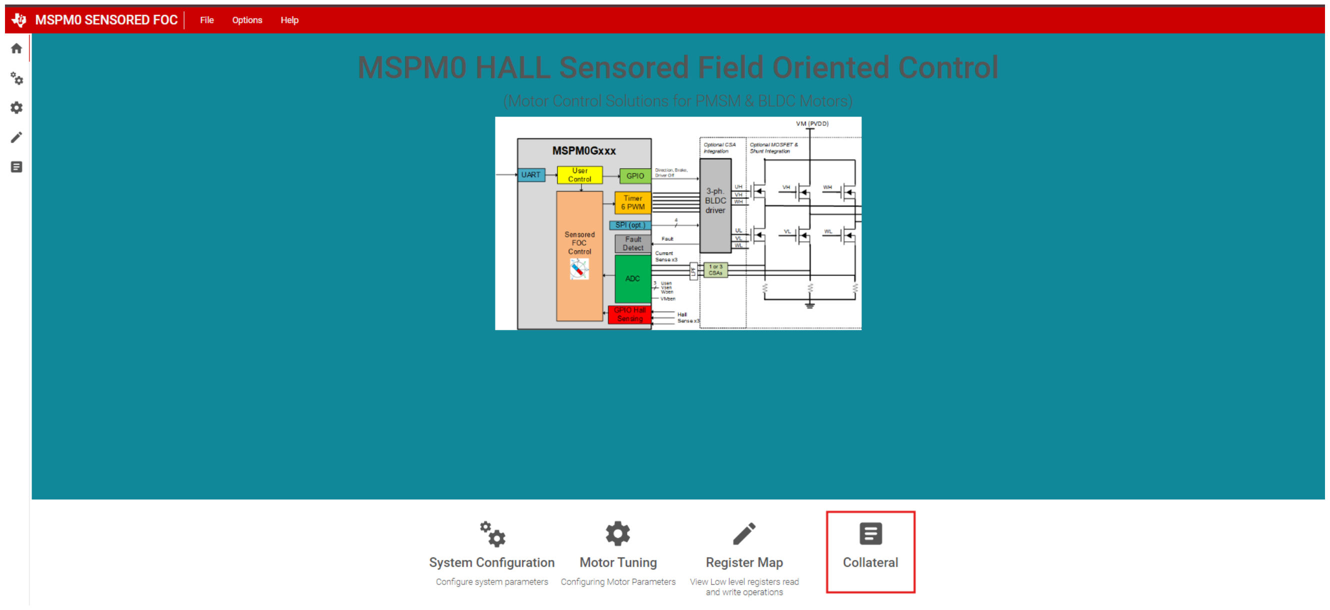
4.2.3 Motor Tuning Page

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


MSPM0 SENSORED FOC
File
Options
Help
MSPM0 HALL Sensored Field Oriented Control
(Motor Control Solutions for PMSM & BLDC Motors)
System Configuration
Configure system parameters
Motor Tuning
Configuring Motor Parameters
Register Map
View Low level registers read and write operations
Collateral

4.2.4 Collateral Page

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

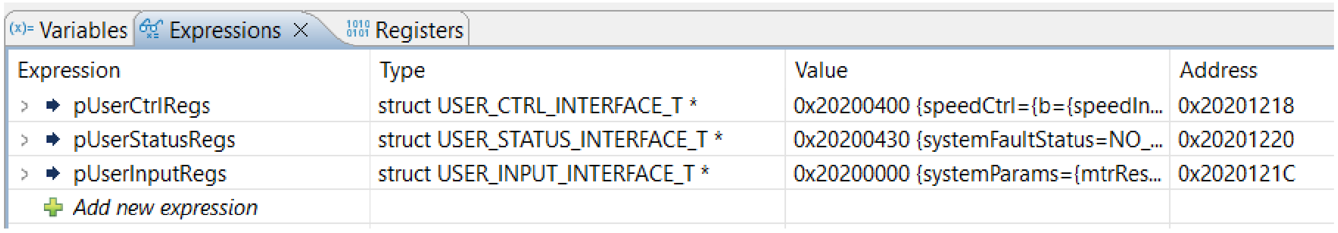

MSPM0 SENSORED FOC
File
Options
Help
MSPM0 HALL Sensored Field Oriented Control
(Motor Control Solutions for PMSM & BLDC Motors)
MSPM0Gxxx
System Configuration
Configure system parameters
Motor Tuning
Configuring Motor Parameters
Register Map
View Low level registers read and write operations
Collateral

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 two ways:

1. Import the structures into the expression window of CCS during the code debug as shown below.



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			

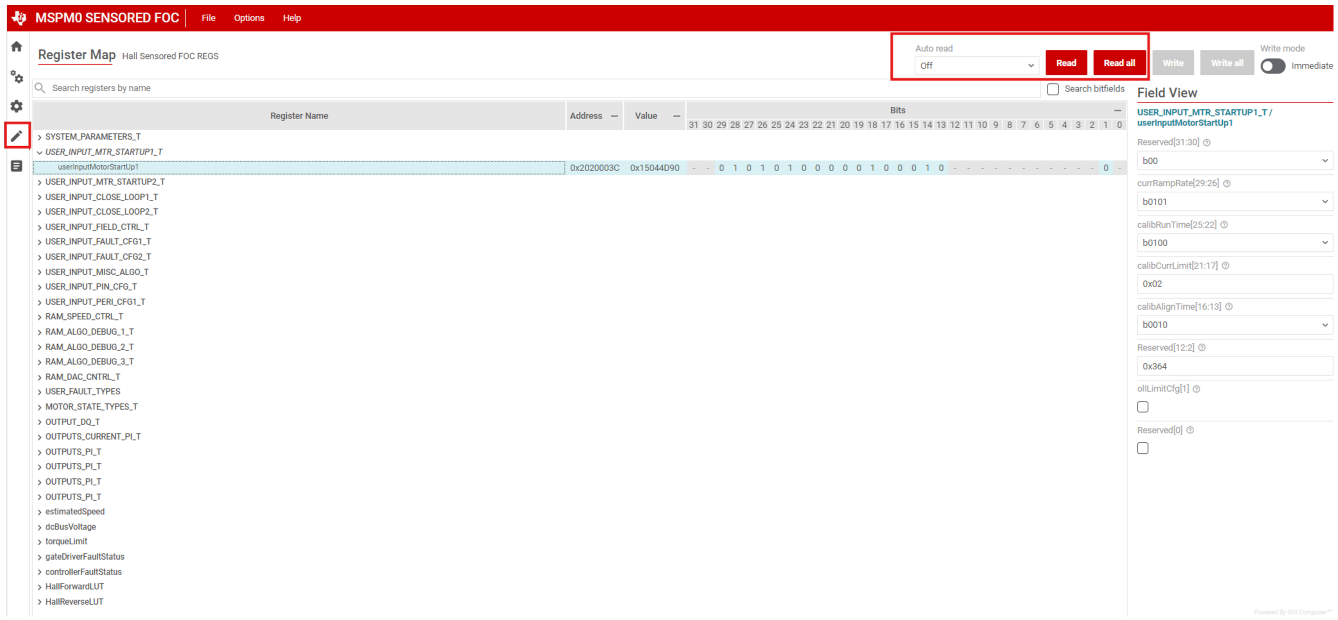
2. Read/Write the parameters over UART as described in UART_COMUNICATION_GUIDE.

Note

The default values in the tables for parameters can be updated in source code by setting appropriate values as per application needs in "configTables.c".

5.1 Register Map Page in GUI

The above register variables can also be configured using GUI. Figure 5-1 details all the available user configurable registers available in the Sensored FOC application. After connecting the GUI with the controller, click ***Read all*** option in the register page to reflect the default programmed parameters.



Register Name	Address	Value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
› SYSTEM_PARAMETERS_T																																		
⌄ *USER_INPUT_MTR_STARTUP1_T*																																		
userInputMotorStartUp1	0x2020003C	0x15044D90	-	-	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	-	-	-	-	-	-	-	-	-	-	-	0	-
› USER_INPUT_MTR_STARTUP2_T																																		
› USER_INPUT_CLOSE_LOOP1_T																																		
› USER_INPUT_CLOSE_LOOP2_T																																		
› USER_INPUT_FIELD_CTRL_T																																		
› USER_INPUT_FAULT_CFG1_T																																		
› USER_INPUT_FAULT_CFG2_T																																		
› USER_INPUT_MISC_ALGO_T																																		
› USER_INPUT_PIN_CFG_T																																		
› USER_INPUT_PERI_CFG1_T																																		
› RAM_SPEED_CTRL_T																																		
› RAM_ALGO_DEBUG_1_T																																		
› RAM_ALGO_DEBUG_2_T																																		
› RAM_ALGO_DEBUG_3_T																																		
› RAM_DAC_CNTRL_T																																		
› USER_FAULT_TYPES																																		
› MOTOR_STATE_TYPES_T																																		
› OUTPUT_DQ_T																																		
› OUTPUTS_CURRENT_PI_T																																		
› OUTPUTS_PI_T																																		
› OUTPUTS_PI_T																																		
› OUTPUTS_PI_T																																		
› OUTPUTS_PI_T																																		
› estimatedSpeed																																		
› dcBusVoltage																																		
› torqueLimit																																		
› gateDriverFaultStatus																																		
› controllerFaultStatus																																		
› HallForwardLUT																																		
› HallReverseLUT																																		

Field View

USER_INPUT_MTR_STARTUP1_T / userInputMotorStartUp1

Reserved[31:30]	b00
currRampRate[29:26]	b0101
calibRunTime[25:22]	b0100
calibCurrLimit[21:17]	0x02
calibAlignTime[16:13]	b0010
Reserved[12:2]	0x364
olILimitCfg[1]	☐
Reserved[0]	☐

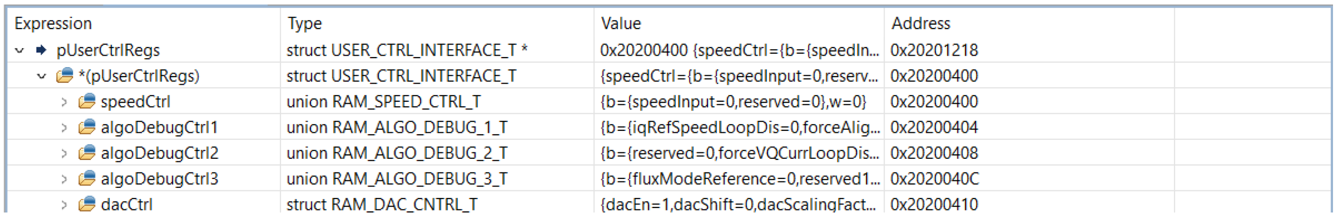
Figure 5-1. GUI Register Map Page

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

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*** . Table 5-1 shows the set of user Control registers as imported in CSS expression window.



Expression	Type	Value	Address
⌄ pUserCtrlRegs	struct USER_CTRL_INTERFACE_T *	0x20200400 {speedCtrl={b={speedIn...	0x20201218
⌄ *(pUserCtrlRegs)	struct USER_CTRL_INTERFACE_T	{speedCtrl={b={speedInput=0,reserv...	0x20200400
› speedCtrl	union RAM_SPEED_CTRL_T	{b={speedInput=0,reserved=0},w=0}	0x20200400
› algoDebugCtrl1	union RAM_ALGO_DEBUG_1_T	{b={iqRefSpeedLoopDis=0,forceAlig...	0x20200404
› algoDebugCtrl2	union RAM_ALGO_DEBUG_2_T	{b={reserved=0,forceVQCurrLoopDis...	0x20200408
› algoDebugCtrl3	union RAM_ALGO_DEBUG_3_T	{b={fluxModeReference=0,reserved1...	0x2020040C
› dacCtrl	struct RAM_DAC_CNTRL_T	{dacEn=1,dacShift=0,dacScalingFact...	0x20200410

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	Table 5-4
8h	ALGO_DEBUG_CTRL2	Algorithm Debug Control 2 register	Table 5-5
Ch	ALGO_DEBUG_CTRL3	Algorithm Debug Control 3 register	Table 5-6
10h	DAC_CTRL	DAC Configuration and Control register	Table 5-7

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	0000000000 00000b	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.
30-0	Reserved	R	000000 0000b	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	RESERVED	R	0h	Reserved
30	UPDATE_SYS_PARAMETERS	W	1h	Dynamically updates System parameters every 200mS like PI gains of Speed/Torque/Flux etc to tune for required performance 0b = Dynamic System updates Disabled 1b = Dynamic System updates Enabled
29	HALL_CALIB_ENABLE	W	0b	Use to enable Automatic Hall Calibration. This bit is reset automatically once calibration is completed. 0h = Hall Calibration Disabled/Completed 1h = Enable Hall Calibration
28	UPDATE_CONFIGS	R	0b	When the configurations are updated by the algorithm, this bit is reset. User can set this bit after giving the tuning command and wait for this bit to reset before starting the speed command.
27	STATUS_UPDATE_ENABLE	W	0b	This bit enables the continuous update of user Status variables in real time.

Table 5-5. Algorithm Debug Control 2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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 Section 8.4.

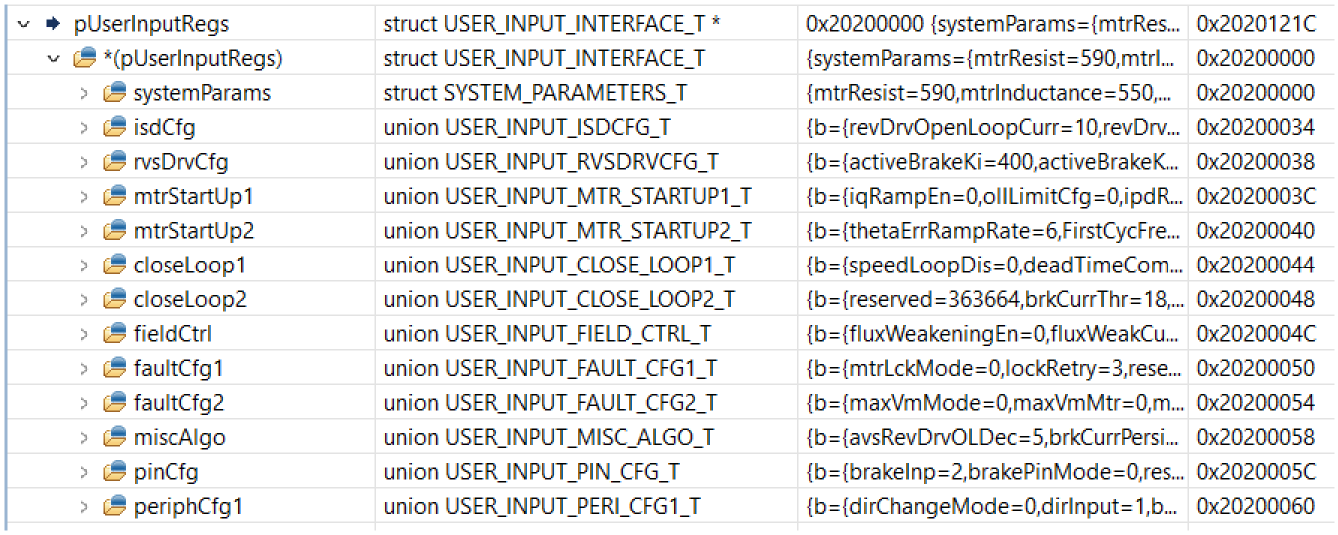
Table 5-7. DAC Configuration Registers

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 12bit DAC register. -ve value specifies the number of right bit shifts before loading the value to 12bit 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 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

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



pUserInputRegs	struct USER_INPUT_INTERFACE_T *	0x20200000 {systemParams={mtrRes...	0x2020121C
*(pUserInputRegs)	struct USER_INPUT_INTERFACE_T	{systemParams={mtrResist=590,mtrI...	0x20200000
systemParams	struct SYSTEM_PARAMETERS_T	{mtrResist=590,mtrInductance=550,...	0x20200000
isdCfg	union USER_INPUT_ISDCFG_T	{b={revDrvOpenLoopCurr=10,revDrv...	0x20200034
rvsDrvCfg	union USER_INPUT_RVSDRVCFG_T	{b={activeBrakeKi=400,activeBrakeK...	0x20200038
mtrStartUp1	union USER_INPUT_MTR_STARTUP1_T	{b={iqRampEn=0,olILimitCfg=0,ipdR...	0x2020003C
mtrStartUp2	union USER_INPUT_MTR_STARTUP2_T	{b={thetaErrRampRate=6,FirstCycFre...	0x20200040
closeLoop1	union USER_INPUT_CLOSE_LOOP1_T	{b={speedLoopDis=0,deadTimeCom...	0x20200044
closeLoop2	union USER_INPUT_CLOSE_LOOP2_T	{b={reserved=363664,brkCurrThr=18,...	0x20200048
fieldCtrl	union USER_INPUT_FIELD_CTRL_T	{b={fluxWeakeningEn=0,fluxWeakCu...	0x2020004C
faultCfg1	union USER_INPUT_FAULT_CFG1_T	{b={mtrLckMode=0,lockRetry=3,rese...	0x20200050
faultCfg2	union USER_INPUT_FAULT_CFG2_T	{b={maxVmMode=0,maxVmMtr=0,m...	0x20200054
miscAlgo	union USER_INPUT_MISC_ALGO_T	{b={avsRevDrvOLDec=5,brkCurrPersi...	0x20200058
pinCfg	union USER_INPUT_PIN_CFG_T	{b={brakeInp=2,brakePinMode=0,res...	0x2020005C
periphCfg1	union USER_INPUT_PERI_CFG1_T	{b={dirChangeMode=0,dirInput=1,b...	0x20200060

Table 5-8. User Input Registers

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

Table 5-9 shows that the complex bit access types are encoded to fit into small table cells.

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)

These tables show a 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 calculated based on the voltage divider as (3.3V × voltage divider ratio / √3) in volts. 3.3V is the full-scale value of the ADC.

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 / CSA Gain in volts/amp) in amps. 1.65V is the reference mid point voltage of the ADC for bidirectional current sensing. 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)	Maximum Power rating of the Motor. **Note** FOC Algorithm computes the PU power for controlling the Input Power in closed loop. **Note** The PU Power is defined as MOTOR_MAX_POWER/ √3 * VOLTAGE_BASE * CURRENT_BASE

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 / Closed Loop Power 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/ Closed Loop Power 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_KP	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_KI	R/W	0.0(Float)	Integral gain for the Flux weakening control in float

Table 5-24. Motor Pole Pairs (Offset = 38h)

Bit	Field	Type	Reset	Description
31-0	POLE_PAIRS	R/W	0(Int)	Total Pole pairs of Motor or Number of Poles / 2 .

5.3.2 MOTOR_STARTUP1 Register (Offset = 3Ch) [Reset = 00000000h]

Table 5-25 shows the register to configure motor startup settings1.

Table 5-25. MOTOR_STARTUP1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-30	RESERVED	R	00b	Reserved
29-26	CURR_RAMP_RATE	R/W	0h	Initial current ramp rate during Hall Calibration till the Maximum current is reached. 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	CALIB_RUN_TIME	R/W	0h	Time spent while calibrating Hall for each CALIBRATION_ANGLE_STEP(defined in hallCalib.h) 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 5-25. MOTOR_STARTUP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
21-17	CALIB_CURRENT_ILIMIT	R/W	00h	Current limit during Hall Calibration 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%
16-13	CALIB_ALIGN_TIME	R	000b	Time taken for initial rotor alignment to Zero Degrees before starting Hall Calibration. 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 Dh = 5s Eh = 7.5s Fh = 10s
12-2	RESERVED	R	0h	Reserved
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

Table 5-25. MOTOR_STARTUP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RESERVED	R	0b	Reserved

5.3.3 MOTOR_STARTUP2 Register (Offset = 40h) [Reset = 00000000h]

Table 5-26 shows the register to configure motor startup settings2.

Table 5-26. 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 5-26. 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/W	0h	Reserved

Table 5-26. MOTOR_STARTUP2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
17-13	OL_MAX_SPEED	R/W	0h	Maximum operational Electrical Frequency during Forced Commutation for First electrical Cycle (% 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%
12-8	Reserved	R	0h	Reserved
7-4	OL_FIRST_CYC_FREQ	R/W	0h	Frequency of first cycle in Open loop (% 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%

Table 5-26. MOTOR_STARTUP2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	FIRST_CYCLE_FREQ_SEL	R/W	0h	First cycle frequency in open loop 0h = Defined by OL_FIRST_CYC_FREQ 1h = 0Hz
2-0	RESERVED	R	0h	Reserved

5.3.4 CLOSED_LOOP1 Register (Offset = 44h) [Reset = 00000000h]

Table 5-27 shows the register to configure close loop settings1.

Table 5-27. 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)
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%

Table 5-27. CLOSED_LOOP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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 5-27. 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 5-27. CLOSED_LOOP1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3	AVS_EN	R/W	0b	AVS enable 0h = Disable 1h = Enable
2	DEADTIME_COMP_EN	R/W	0b	Dead-time compensation enable 0h = Disable 1h = Enable
1	RESERVED	R/W	0b	Reserved

5.3.5 CLOSED_LOOP2 Register (Offset = 48h) [Reset = 00000000h]

Table 5-28 shows the register to configure close loop settings2.

Table 5-28. 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 5-28. 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	0h	Reserved

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

Register to configure Flux Control settings

Table 5-29. 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%

Table 5-29. FIELD_CTRL register bit Descriptions (continued)

Bit	Field	Type	Reset	Description
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

5.3.7 FAULT_CONFIG1 Register (Offset = 50h) [Reset = 00000000h]

Register to configure fault settings1

Table 5-30. 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
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

5.3.8 FAULT_CONFIG2 Register (Offset = 54h) [Reset = 00000000h]

Table 5-31 shows the register to configure fault settings2.

Table 5-31. FAULT_CONFIG2 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-27	RESERVED	R/W	0h	Reserved
26	ABN_SPEED_LOCK_ EN	R/W	0b	Lock 1 : Abnormal speed Lock 0h = Disable 1h = Enable
25	HALL_INVALID_LOCK_ EN	R/W	0b	Lock 2 : Invalid Hall Input Lock 0h = Disable 1h = Enable

Table 5-31. FAULT_CONFIG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
24	NO_MOTOR_LOCK_EN	R/W	0b	Lock 3 : No motor Lock 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	RESERVED	R	0b	Reserved
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.

Table 5-31. FAULT_CONFIG2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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

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

Table 5-32 shows the register to multiple miscellaneous Algorithm Configuration.

Table 5-32. MISC_ALGO Register Field Descriptions

Bit	Field	Type	Reset	Description
31-10	RESERVED	R/W	0h	Reserved
9-6	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
5-2	RESERVED	R	0b	Reserved

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

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

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

Table 5-33 shows the register to configure hardware pins.

Table 5-33. 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

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

Table 5-34 show the register to peripheral.

Table 5-34. 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

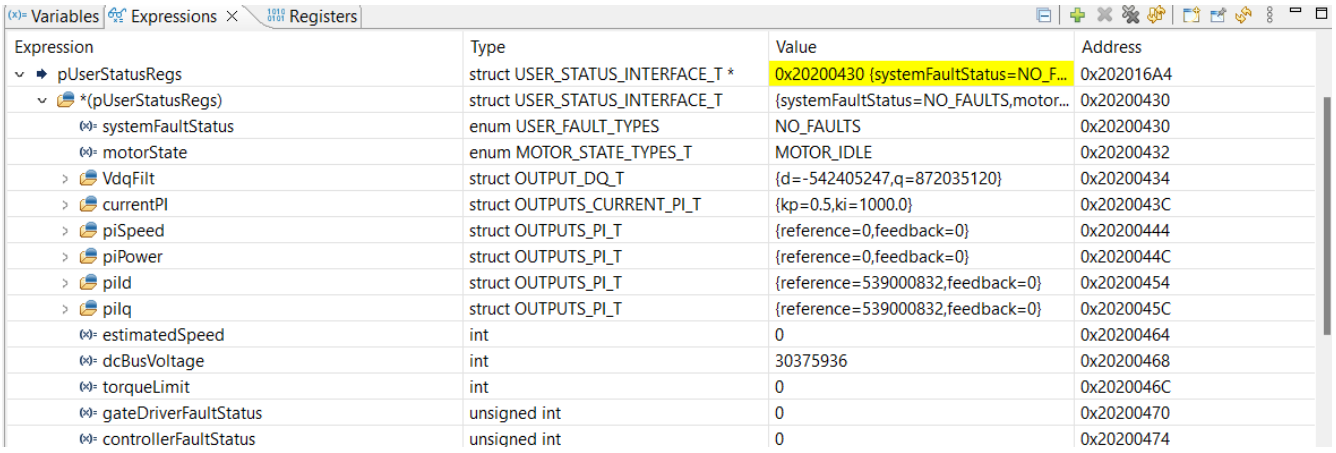
Table 5-34. PERI_CONFIG1 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
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
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.

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



Expression	Type	Value	Address
pUserStatusRegs	struct USER_STATUS_INTERFACE_T *	0x20200430 {systemFaultStatus=NO_F...	0x202016A4
*(pUserStatusRegs)	struct USER_STATUS_INTERFACE_T	{systemFaultStatus=NO_FAULTS,motor...	0x20200430
systemFaultStatus	enum USER_FAULT_TYPES	NO_FAULTS	0x20200430
motorState	enum MOTOR_STATE_TYPES_T	MOTOR_IDLE	0x20200432
VdqFilt	struct OUTPUT_DQ_T	{d=-542405247,q=872035120}	0x20200434
currentPI	struct OUTPUTS_CURRENT_PI_T	{kp=0.5,ki=1000.0}	0x2020043C
piSpeed	struct OUTPUTS_PI_T	{reference=0,feedback=0}	0x20200444
piPower	struct OUTPUTS_PI_T	{reference=0,feedback=0}	0x2020044C
piId	struct OUTPUTS_PI_T	{reference=539000832,feedback=0}	0x20200454
piIq	struct OUTPUTS_PI_T	{reference=539000832,feedback=0}	0x2020045C
estimatedSpeed	int	0	0x20200464
dcBusVoltage	int	30375936	0x20200468
torqueLimit	int	0	0x2020046C
gateDriverFaultStatus	unsigned int	0	0x20200470
controllerFaultStatus	unsigned int	0	0x20200474

The following table lists the definitions of variables available for monitoring.

Table 5-35. 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 -Hall Sensor Invalid, no motor, abnormal speed **VOLTAGE_OUT_OF_BOUNDS:**Indicates undervoltage or overvoltage. **HARDWARE_OVER_CURRENT:**Indicates DC bus current limit fault **HV_DIE:** Indicates gate driver fault if applicable.
MOTOR_STATE	MOTOR_STATE_TYPE	MOTOR_IDLE	Defines the state of Current Motor Running Status ***MOTOR_IDLE*** : Motor Idle 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.

Table 5-35. User Status Registers (continued)

Variables	Type	Reset Value	Description
PI_POWER	IQ GLOBAL 27	IQ27(0)	Indicates the Power reference and power feedback values of PI controller set by FOC algorithm in PU. **Note** FOC Algorithm computes the PU power for controlling the Input Power in closed loop. **Note** The PU Power is defined as MOTOR_MAX_POWER/ √3 * VOLTAGE_BASE * CURRENT_BASE
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.
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 closed loop Field-oriented control where the motor spins at the commanded Speed/Torque/Power/Voltage reference.

6.1 System Configuration Parameters

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

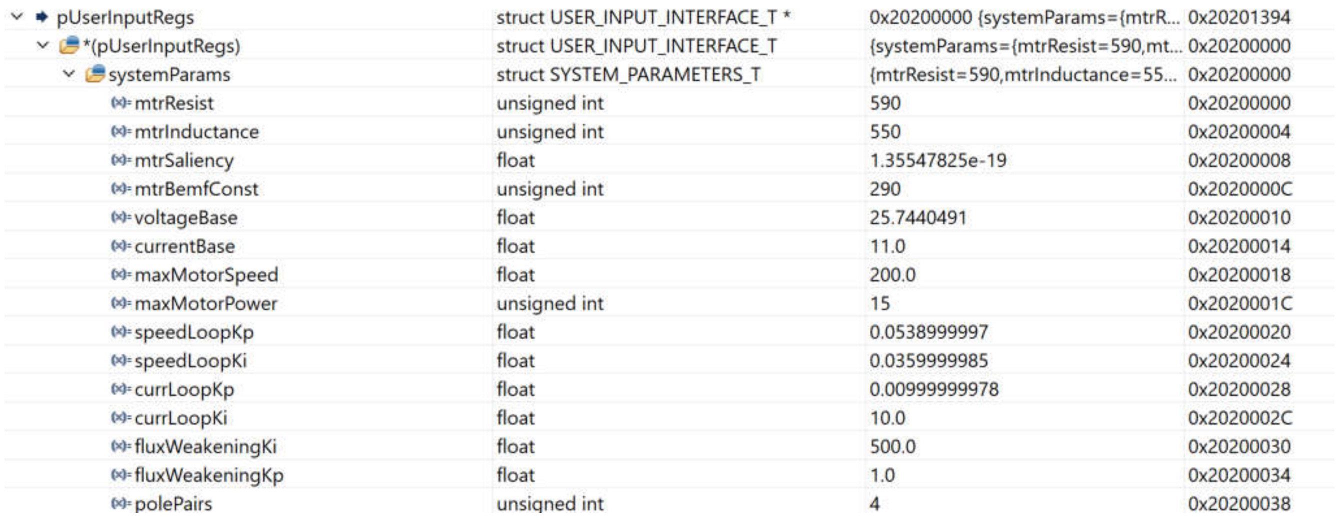
Below are the set of System parameters that are to be specified for full featured Sensored FOC operation. These variables can be added in the expression window using the pUserInputRegs.

Note

Motor resistance , Motor Inductance are used for the initial estimation of Current Loop Kp & Ki values.Ignore configuring these parameters if Current Loop Kp and Ki are tuned manually.

Note

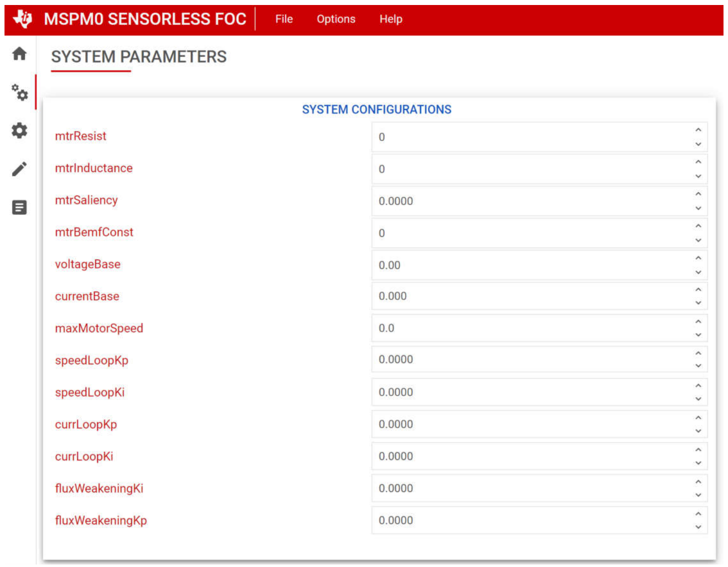
Motor Inductance, Saliency and Motor BEMF Constant of the Motor are critical for the operation of MTPA algorithm. If salient pole motor is used (Interior PMSM) and MTPA is enabled these parameters must be configured.



pUserInputRegs	struct USER_INPUT_INTERFACE_T *	0x20200000 {systemParams={mtrR...	0x20201394
*(pUserInputRegs)	struct USER_INPUT_INTERFACE_T	{systemParams={mtrResist=590,mt...	0x20200000
systemParams	struct SYSTEM_PARAMETERS_T	{mtrResist=590,mtrInductance=55...	0x20200000
mtrResist	unsigned int	590	0x20200000
mtrInductance	unsigned int	550	0x20200004
mtrSaliency	float	1.35547825e-19	0x20200008
mtrBemfConst	unsigned int	290	0x2020000C
voltageBase	float	25.7440491	0x20200010
currentBase	float	11.0	0x20200014
maxMotorSpeed	float	200.0	0x20200018
maxMotorPower	unsigned int	15	0x2020001C
speedLoopKp	float	0.0538999997	0x20200020
speedLoopKi	float	0.0359999985	0x20200024
currLoopKp	float	0.00999999978	0x20200028
currLoopKi	float	10.0	0x2020002C
fluxWeakeningKi	float	500.0	0x20200030
fluxWeakeningKp	float	1.0	0x20200034
polePairs	unsigned int	4	0x20200038

6.1.1 Configuring System Parameters From GUI

Configure the system parameters using the **System Configuration** page in the GUI as shown below. 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.

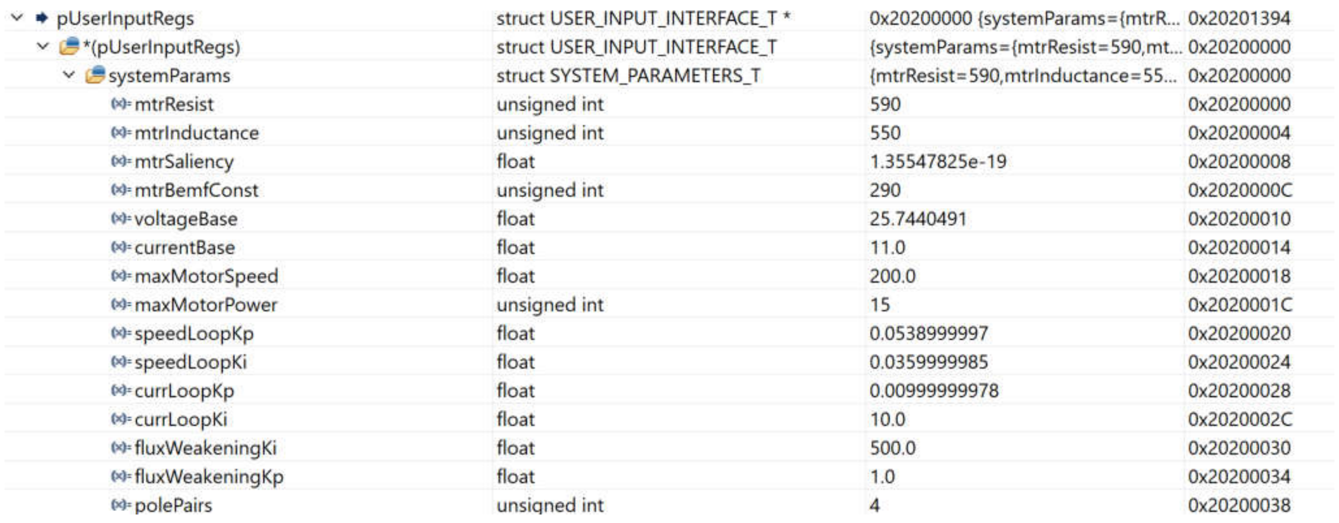


mtrResist	0
mtrInductance	0
mtrSaliency	0.0000
mtrBemfConst	0
voltageBase	0.00
currentBase	0.000
maxMotorSpeed	0.0
speedLoopKp	0.0000
speedLoopKi	0.0000
currLoopKp	0.0000
currLoopKi	0.0000
fluxWeakeningKi	0.0000
fluxWeakeningKp	0.0000

Figure 6-1. GUI System Parameter Configuration

6.1.2 Updating System Parameters Through CCS Debug Window

Alternately, user can update the System parameters from the CCS Debug window as shown in Figure 6-2.



pUserInputRegs	struct USER_INPUT_INTERFACE_T *	0x20200000 {systemParams={mtrR...	0x20201394
*(pUserInputRegs)	struct USER_INPUT_INTERFACE_T	{systemParams={mtrResist=590,mt...	0x20200000
systemParams	struct SYSTEM_PARAMETERS_T	{mtrResist=590,mtrInductance=55...	0x20200000
mtrResist	unsigned int	590	0x20200000
mtrInductance	unsigned int	550	0x20200004
mtrSaliency	float	1.35547825e-19	0x20200008
mtrBemfConst	unsigned int	290	0x2020000C
voltageBase	float	25.7440491	0x20200010
currentBase	float	11.0	0x20200014
maxMotorSpeed	float	200.0	0x20200018
maxMotorPower	unsigned int	15	0x2020001C
speedLoopKp	float	0.0538999997	0x20200020
speedLoopKi	float	0.0359999985	0x20200024
currLoopKp	float	0.00999999978	0x20200028
currLoopKi	float	10.0	0x2020002C
fluxWeakeningKi	float	500.0	0x20200030
fluxWeakeningKp	float	1.0	0x20200034
polePairs	unsigned int	4	0x20200038

Figure 6-2. CCS Debug Window

6.1.2.1 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

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

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.

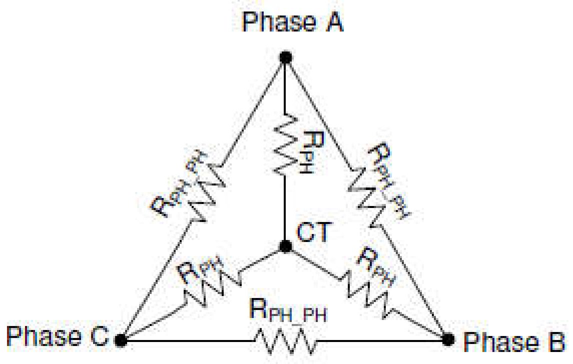


Figure 6-3. Resistance Measurement

6.1.2.2 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.

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

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 all three phase to phase inductances and calculate the average and use this value as the phase to phase inductance. This measurement is valid for both star wound and delta wound motors.

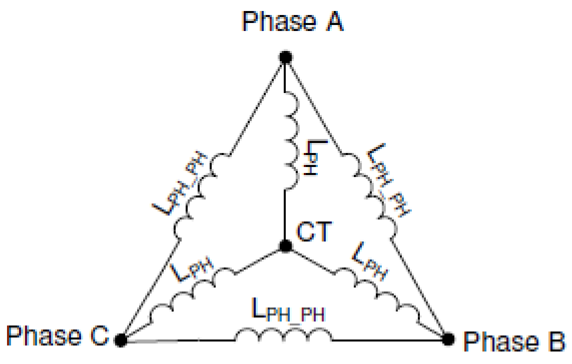


Figure 6-4. Inductance Measurement

6.1.2.3 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.

6.1.2.4 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.

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

$$\text{BEMF Constant in }\left(\frac{\text{mV}}{\text{Hz}}\right) = \frac{\text{Kt}\left[\frac{\text{mN} \times \text{m}}{\text{A}}\right] \times 2\pi}{\#\ \text{pole pairs}} \tag{5}$$

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.

$$\text{BEMf Constant Ke} = \text{EP} \times \text{TP}/\sqrt{3} \tag{6}$$

6.1.2.5 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.

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

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**. For hardware configuration of the voltage divider scaling ratio, see Section 2.3.

6.1.2.6 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 as shown in Equation 8. There is a factor of 2 considered to support bidirectional current sensing with 1.65V as the zero-current offset.

$$\mathrm{Base}_{\mathrm{Current}} = \frac{\mathrm{3.3V\ Full\ Scale\ Value}}{2 \times \mathrm{CSA\_GAIN}\left[\frac{\mathrm{V}}{\mathrm{A}}\right]} = \frac{\mathrm{3.3V}}{2 \times \mathrm{CSA\_GAIN}\left[\frac{\mathrm{V}}{\mathrm{A}}\right]} \tag{8}$$

For example, in a system with CSAGAIN = 0.15V/A , the base current or maximum measurable system current by the ADC is 3.3V / (2 × 0.15V/A) = **11A**.

Note

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

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.

$$\mathrm{CSA\_GAIn}\left[\frac{\mathrm{V}}{\mathrm{A}}\right] = \mathrm{R}_{\mathrm{SENSE}} \times \mathrm{CSA\_GAIN}\left[\frac{\mathrm{V}}{\mathrm{V}}\right] \tag{9}$$

6.1.2.7 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.

$$f_{\mathrm{Electrical}} = \frac{n_{\mathrm{PolePairs}} \times \omega_{\mathrm{Mechanical}}}{60} \tag{10}$$

Where:

- $\omega_{\mathbf{Mechanical}}$ is the mechanical speed in units revolutions per minute (RPM)
- $f_{\mathbf{Electrical}}$ is the electrical speed in units of hertz (Hz)
- $n_{\mathbf{PolePairs}}$ 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.

6.1.2.8 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.

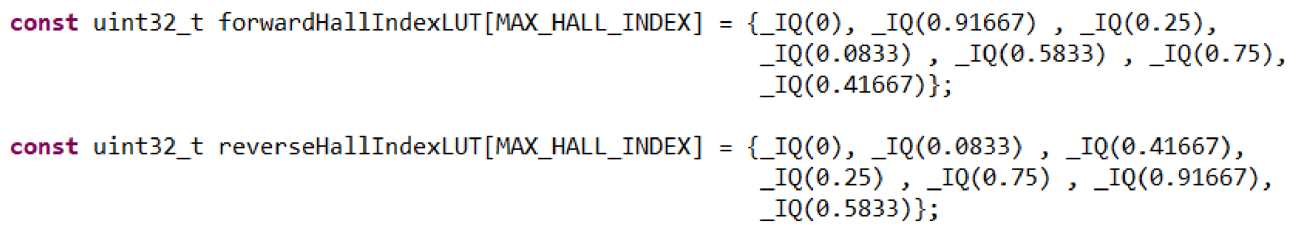
6.2 Control Configurations for Basic Motor Spinning

After configuring the system parameters in the Expressions window, User can update rest of the input parameters for tuning the motor to start spinning in closed loop. Below are some of the important configurations needed to start spinning the motor in closed loop.

6.2.1 Hall Sensor Auto Calibration

In Hall Sensor based FOC , Hall signals are used to detect the rotor position information and drive the motor efficiently. FOC application requires three digital Hall Inputs placed 60 degrees electrically connected to the GPIO's as inputs feeding the rotor position information.

User needs to populate the Hallangle table in IQ27 P.U appropriately for a given Hall Sequence with reference to Phase connections in the ISR.c file as shown in Figure 6-5.



```
const uint32_t forwardHallIndexLUT[MAX_HALL_INDEX] = {_IQ(0), _IQ(0.91667) , _IQ(0.25),
                                                       _IQ(0.0833) , _IQ(0.5833) , _IQ(0.75),
                                                       _IQ(0.41667)};

const uint32_t reverseHallIndexLUT[MAX_HALL_INDEX] = {_IQ(0), _IQ(0.0833) , _IQ(0.41667),
                                                       _IQ(0.25) , _IQ(0.75) , _IQ(0.91667),
                                                       _IQ(0.5833)};
```

Figure 6-5. Hall Angle Table Values

In general, the Hall placements can be erroneous and the electrical displacement can be less than or greater than 60 degrees. Any error in the hall placement results in torque ripple and non sinusoidal current. The Hall pin sequence is also significant with reference to the Motor Phase connection and the driving angle for a given phase depends on the sequence of connections. This information is typically available in data sheet.

Hall calibration eliminates need for the data sheet and corrects the hall placement error for improved current waveform in motors with erroneous hall placements.\

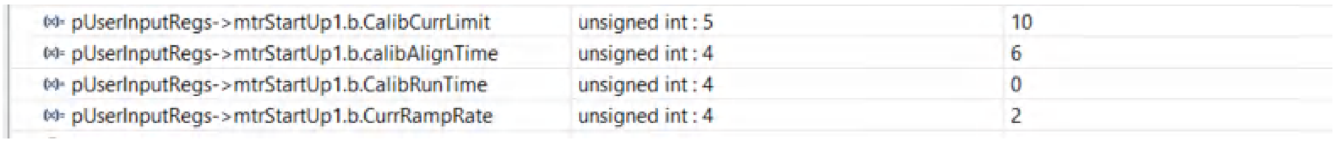
The above hall angle table values can be auto generated with Hall Calibration routine as described below:

Calibration Routine:

- **Motor_Align**: Aligning the motor to a known rotor angle is the first step in the calibration sequence. In this routine Motor aligns to *CALIBRATION_ALIGN_ANGLE* macro for a duration *calibAlignTime* set in the User Inputs Motor Startup1 parameters. Sufficient *calibAlignTime* is to be configured such that the motor aligns and stops movement before the calibration is initiated.
- **Motor_Calib_Run**: Once Align step is successfully completed, the rotor microsteps with an angle specified as in macro *CALIBRATION_ANGLE_STEP* for a duration of *calibRunTime* set in the User Inputs Motor Startup1 parameters. Motor rotates for one complete mechanical cycles (Pole_Pairs * Electrical cycles) in both forward and reverse direction to compute the average Hall angle.
- **Motor_Calib_Complete**: Once motor calibration is successfully completed , rotor angle for each Hall State transitions for both Forward and Reverse directions are generated in the Hall angle tables as *hallAngleTableForward and hallAngleTableReverse* .

Below is the sequence to follow for the Auto Hall Sensor Calibration:

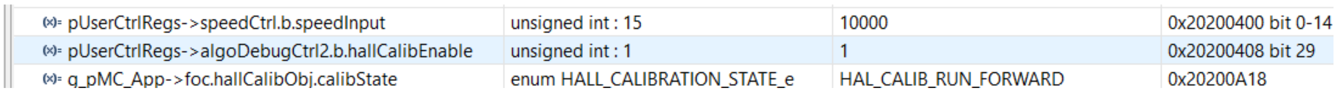
1. Configure and tune the below input parameters.
   - Configure the Current loop Kp & Ki Parameters in the System parameters.
   - Configure the Motor Startup 1 parameters as below to tune the Motor Align and Motor Calib Run States



pUserInputRegs->mtrStartUp1.b.CalibCurrLimit	unsigned int : 5	10
pUserInputRegs->mtrStartUp1.b.calibAlignTime	unsigned int : 4	6
pUserInputRegs->mtrStartUp1.b.CalibRunTime	unsigned int : 4	0
pUserInputRegs->mtrStartUp1.b.CurrRampRate	unsigned int : 4	2

Figure 6-6. Hall Calibration Configurations

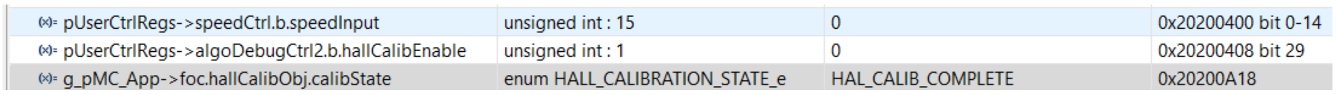
   – *CalibCurrLimit* : Id current forced during Calibration
   – *calibAlignTime* : Time in milli seconds applied for the CALIBRATION_ALIGN_ANGLE while aligning for the first time.
   – *calibRunTime*: Time in milli seconds applied while micro stepping and sweeping the mechanical cycle in steps of HALL_CALIBRATION_STEP.
   – *CurrRampRate*: Ramp rate of Id current during the first align to CALIBRATION_ALIGN_ANGLE.
2. Set the Hall Sensor Calibration Enable bit in Algo Debug Control 2 Register and set the Motor speed to non zero value.



pUserCtrlRegs->speedCtrl.b.speedInput	unsigned int : 15	10000	0x20200400 bit 0-14
pUserCtrlRegs->algoDebugCtrl2.b.hallCalibEnable	unsigned int : 1	1	0x20200408 bit 29
g_pMC_App->foc.hallCalibObj.calibState	enum HALL_CALIBRATION_STATE_e	HAL_CALIB_RUN_FORWARD	0x20200A18

Figure 6-7. Hall Calibration Enable

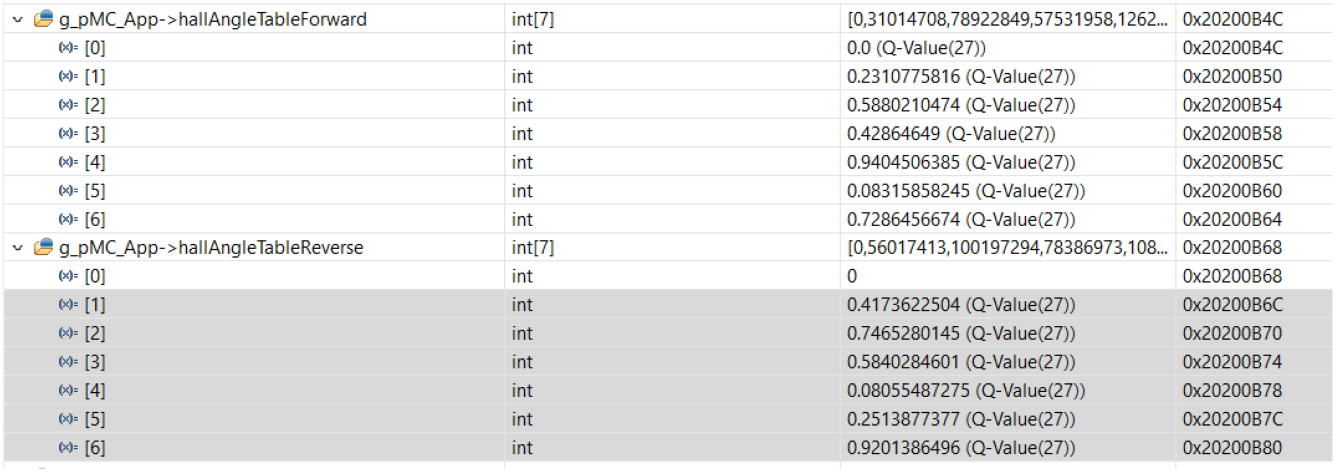
3. Once the Calibration sequence is completed, the CalibState changes to HALL_CALIB_ COMPLETE.



pUserCtrlRegs->speedCtrl.b.speedInput	unsigned int : 15	0	0x20200400 bit 0-14
pUserCtrlRegs->algoDebugCtrl2.b.hallCalibEnable	unsigned int : 1	0	0x20200408 bit 29
g_pMC_App->foc.hallCalibObj.calibState	enum HALL_CALIBRATION_STATE_e	HAL_CALIB_COMPLETE	0x20200A18

The Hall angle tables in the Motor Control application variables are updated as below. Angle tables are in Q27 format by default. _IQ27(1.0) corresponds to 360 degrees electrical cycle.

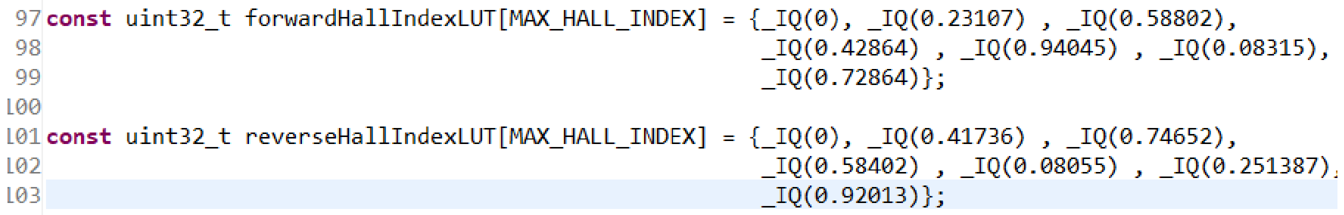
Figure 6-8. Hall Calib Complete



g_pMC_App->hallAngleTableForward	int[7]	[0,31014708,78922849,57531958,1262...	0x20200B4C
[0]	int	0.0 (Q-Value(27))	0x20200B4C
[1]	int	0.2310775816 (Q-Value(27))	0x20200B50
[2]	int	0.5880210474 (Q-Value(27))	0x20200B54
[3]	int	0.42864649 (Q-Value(27))	0x20200B58
[4]	int	0.9404506385 (Q-Value(27))	0x20200B5C
[5]	int	0.08315858245 (Q-Value(27))	0x20200B60
[6]	int	0.7286456674 (Q-Value(27))	0x20200B64
g_pMC_App->hallAngleTableReverse	int[7]	[0,56017413,100197294,78386973,108...	0x20200B68
[0]	int	0	0x20200B68
[1]	int	0.4173622504 (Q-Value(27))	0x20200B6C
[2]	int	0.7465280145 (Q-Value(27))	0x20200B70
[3]	int	0.5840284601 (Q-Value(27))	0x20200B74
[4]	int	0.08055487275 (Q-Value(27))	0x20200B78
[5]	int	0.2513877377 (Q-Value(27))	0x20200B7C
[6]	int	0.9201386496 (Q-Value(27))	0x20200B80

Figure 6-9. Generated Hall Angle Table

4. Once satisfactory Calibration routine is performed such that the Motor Spins smoothly in closed loop operation, user can store these calibrated angle values in the *forwardHallIndexLUT & reverseHallIndexLUT* defined in ISR.c. Upon MCU restart the values from these LUT's are used by default for spinning the motor.



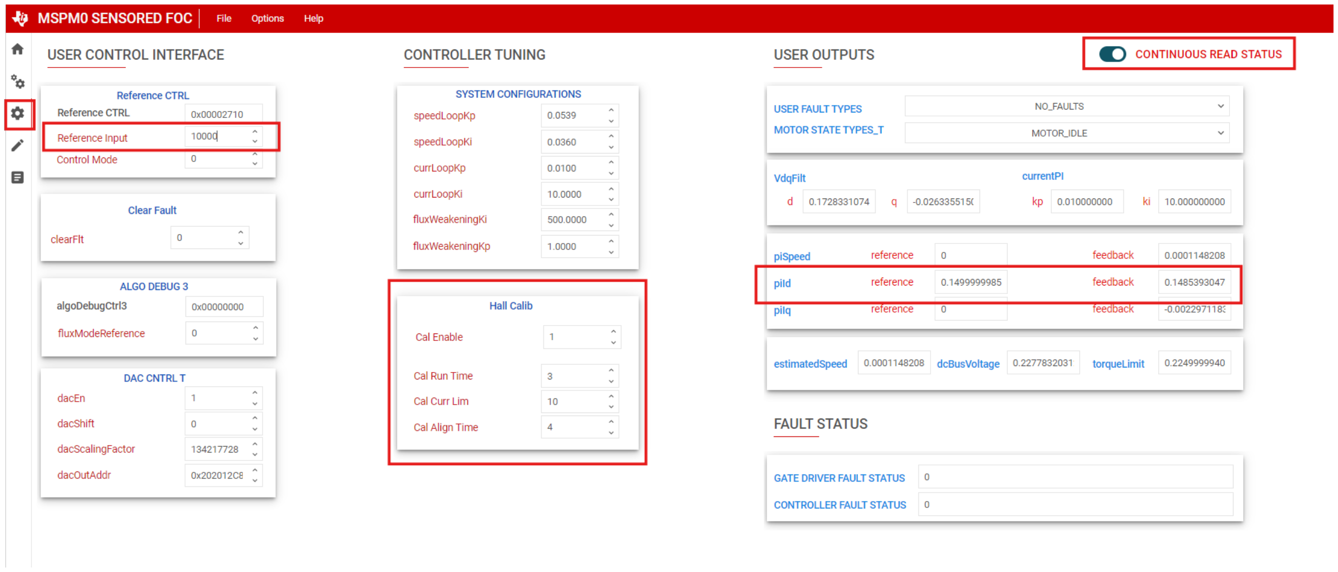
```
const uint32_t forwardHallIndexLUT[MAX_HALL_INDEX] = {_IQ(0), _IQ(0.23107) , _IQ(0.58802),
                                                      _IQ(0.42864) , _IQ(0.94045) , _IQ(0.08315),
                                                      _IQ(0.72864)};

const uint32_t reverseHallIndexLUT[MAX_HALL_INDEX] = {_IQ(0), _IQ(0.41736) , _IQ(0.74652),
                                                      _IQ(0.58402) , _IQ(0.08055) , _IQ(0.251387),
                                                      _IQ(0.92013)};
```

Figure 6-10. Updated Hall Angle Table

6.2.1.1 Hall Sensor Calibration Through GUI

Hall sensor calibration routine can also be performed from the GUI control Page using below widgets with the parameter tuning as mention in the previous section.



USER CONTROL INTERFACE

Reference CTRL

Reference CTRL	0x00002710
Reference Input	10000
Control Mode	0

Clear Fault

clearFlt	0

ALGO DEBUG 3

algoDebugCtrl3	0x00000000
fluxModeReference	0

DAC CNTRL T

dacEn	1
dacShift	0
dacScalingFactor	134217728
dacOutAddr	0x202012C8

CONTROLLER TUNING

SYSTEM CONFIGURATIONS

speedLoopKp	0.0539
speedLoopKi	0.0360
currLoopKp	0.0100
currLoopKi	10.0000
fluxWeakeningKi	500.0000
fluxWeakeningKp	1.0000

Hall Calib

Cal Enable	1
Cal Run Time	3
Cal Curr Lim	10
Cal Align Time	4

USER OUTPUTS

CONTINUOUS READ STATUS

USER FAULT TYPES	NO_FAULTS
MOTOR STATE TYPES_T	MOTOR_IDLE

VdqFilt: d 0.1728331074 q -0.026335515(
currentPI: kp 0.010000000 ki 10.000000000

piSpeed	reference	0	feedback	0.0001148208
piId	reference	0.1499999985	feedback	0.1485393047
piIq	reference	0	feedback	-0.002297118:

estimatedSpeed	0.0001148208	dcBusVoltage	0.2277832031	torqueLimit	0.2249999940

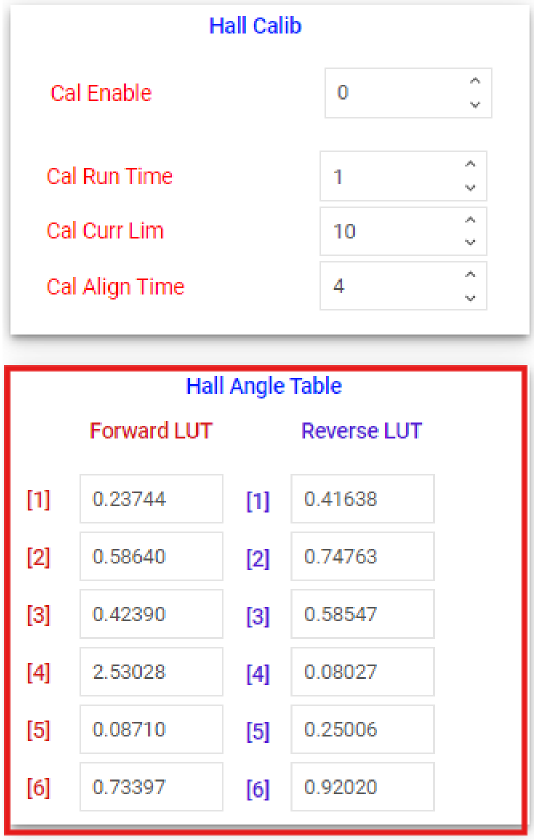
FAULT STATUS

GATE DRIVER FAULT STATUS	0
CONTROLLER FAULT STATUS	0

Figure 6-11. Hall Sensor Calibration From GUI

As described in the above section, Set the Hall Cal enable bit to 1, Configure the Hall Calibration Parameters, Set the Motor Speed as non zero value to start the Hall calibration. User can enable the continous read status button to monitor the Id Current reference and Feedback tracking and the updated Hall Angle Table values.

If there is lot of variation in the Id current feedback, tune the current loop Kp, Ki and increase the Cal Run time.



Hall Calib

Cal Enable	0
Cal Run Time	1
Cal Curr Lim	10
Cal Align Time	4

Hall Angle Table

	Forward LUT		Reverse LUT
[1]	0.23744	[1]	0.41638
[2]	0.58640	[2]	0.74763
[3]	0.42390	[3]	0.58547
[4]	2.53028	[4]	0.08027
[5]	0.08710	[5]	0.25006
[6]	0.73397	[6]	0.92020

Figure 6-12. HalAngleTable

Calibrated Hall angle tables are populared into the Hall angle table for both forward and reverse directions in IQ27 P.U format (_IQ(1.0) P.U corresponds to 30 degrees). Once satisfactory Calibration routine is performed i.e Motor Spins smoothly in closed loop operation, user can store these calibrated angle values in the *forwardHallIndexLUT* & *reverseHallIndexLUT* defined in ISR.c. Upon MCU restart the values from these LUT's are used by default for spinning the motor.

6.2.2 Motor Open Loop Ramp

In Hall sensor Based FOC applications, the speed is determined from the rate of change of Hall Signal timings, At startup for the first few commutation cycles, the speed of the motor varies and the speed readings are updated at 60 degree intervals and not used for angle interpolation as the motor continues to accelerate. So for the first electrical cycle, the motor is ramped up and angle is incremented based on the Open loop profile determined by the openloop parameters. The speed loop with speed feedback is enabled after the completion of first electrical cycle and the angle is generated based on the Hall sensor based speed.

During openloop startup, the FOC algorithm accelerates the motor with a second order open loop ramp profile to increase the speed and the angle is computed based on the Openloop profile speed. At every Hall event, the angle is reset based on the programmed angle based on the sector derived from Hall signals.

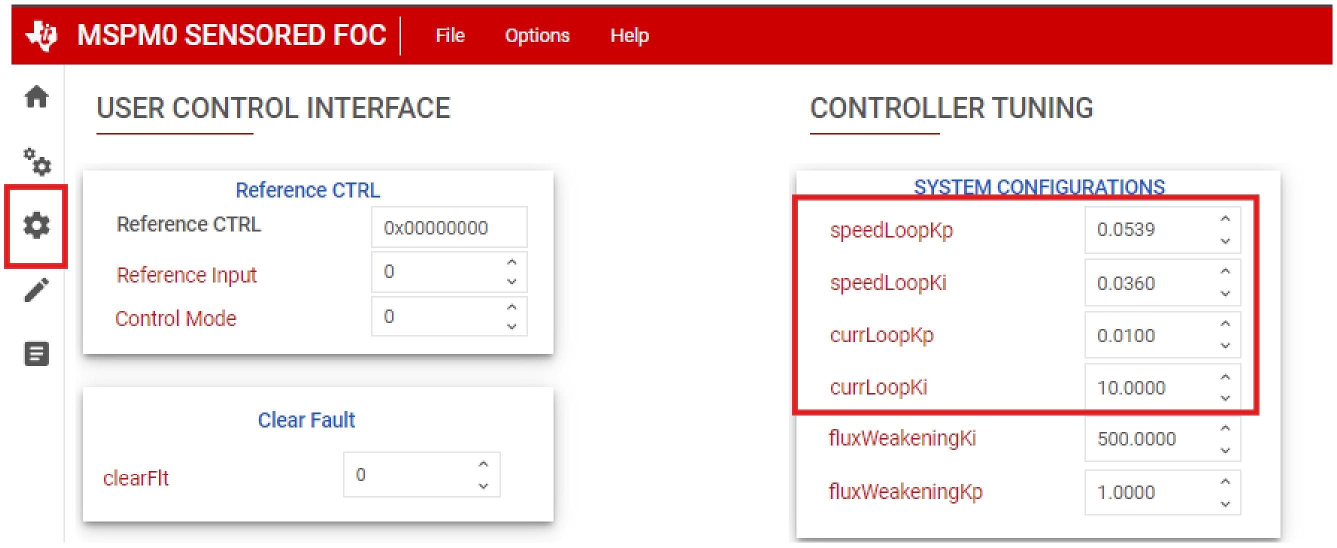
By default, the open loop ramp up parameters are configured with a linear first order configuration with sluggish acceleration, which works for most of the motors. Usr can tune the second order openloop parameters using the MotorStartup2 parameters.

6.2.3 PI Controller Tuning for Closed Loop Speed Control

6.2.3.1 Current Controller 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.



Reference CTRL	0x00000000
Reference Input	0
Control Mode	0
clearFlt	0

speedLoopKp	0.0539
speedLoopKi	0.0360
currLoopKp	0.0100
currLoopKi	10.0000
fluxWeakeningKi	500.0000
fluxWeakeningKp	1.0000

Figure 6-13. Speed Loop and Current Loop Tuning

6.2.3.2 Speed/Power Controller Tuning

The FOC Algorithm uses an integrated speed/Power control loop that helps maintain a constant speed/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/power loop is limited by configuring *iLIMIT* in the closedLoop1 configuration in the Register Map page of GUI. When output of the speed/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 Closed loop Torque Mode by setting controlMode in closedLoop1 to 10b.
2. Adjust the torque reference to spin the motor at 50% of rated Speeds.
3. The current feedback gradually settles down to the set Iqref to run at the target speed.
4. Speed loop Kp [SPD_LOOP_KP] is calculated using this equation: SpeedLoop K_p = Current Reference in Torque Mode / Operating Speed in Hz
5. Speed loop Ki [SPD_LOOP_KI] is calculated using this equation: Speed Loop K_i = Speed Loop K_p × 0.1
6. If the Speed continously fluctuates without settling, adjust the Current Loop Kp and Ki such that motor settles to constant speed for a given torque reference.

Note

Once satisfactory Kp and Ki are recorded switch to controlMode 0b to operate the motor in closed loop speed control. 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.

Table 6-1 shows general guidelines to change controller gains.

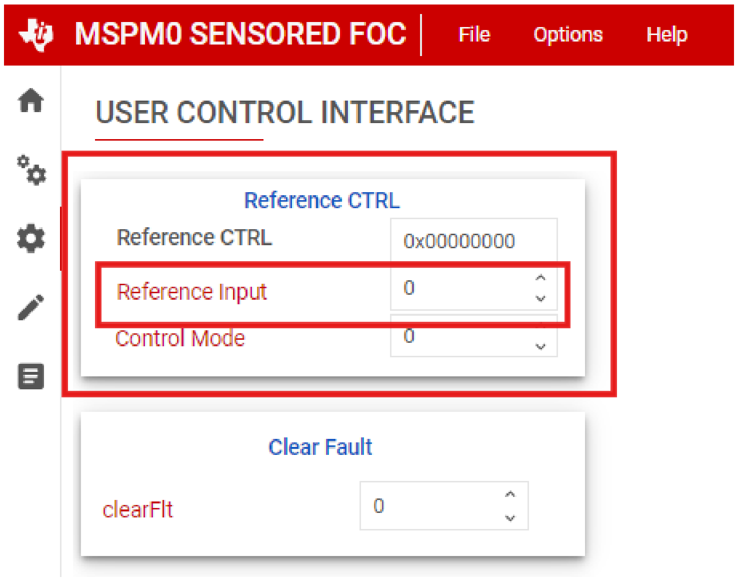
Table 6-1. Guidelines to Change Controller Gains

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

6.2.4 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.

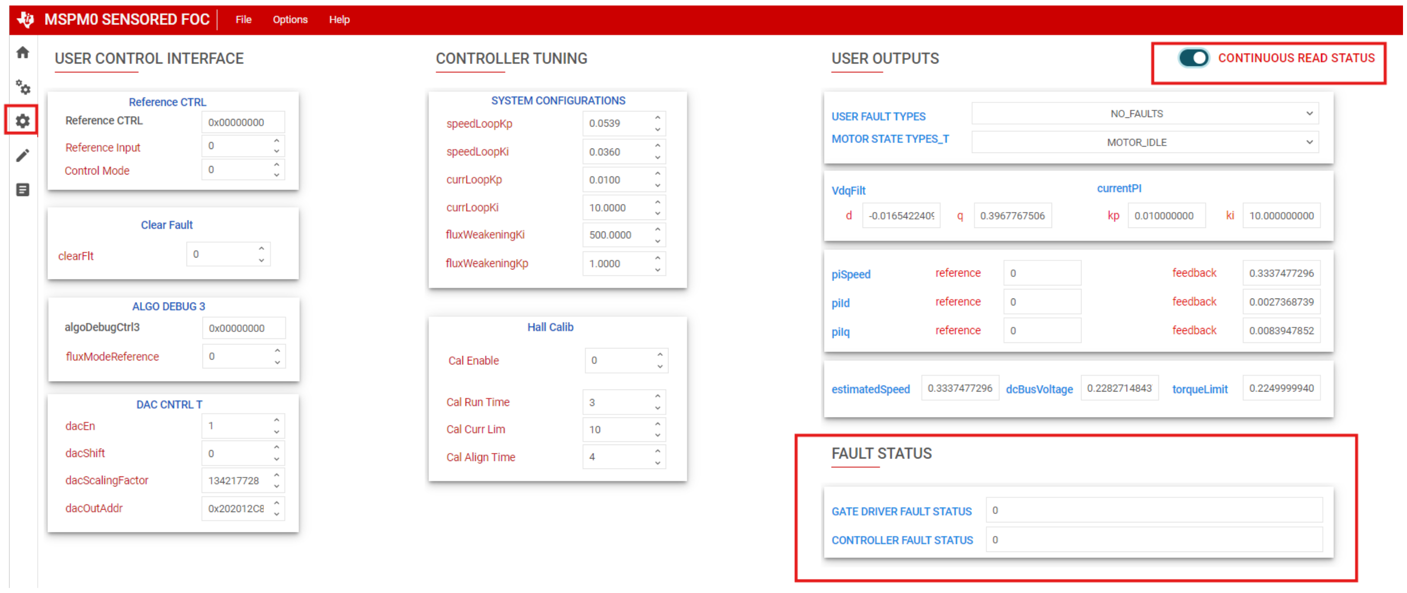


Reference CTRL: 0x00000000
Reference Input: 0
Control Mode: 0
clearFlt: 0

Figure 6-14. 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 bottom right corner of the GUI and monitor the Fault Status register. If no faults is triggered, move to the Section 7 section.



USER CONTROL INTERFACE

Reference CTRL

Reference CTRL	0x00000000
Reference Input	0
Control Mode	0

Clear Fault

clearFlt	0

ALGO DEBUG 3

algoDebugCtrl3	0x00000000
fluxModeReference	0

DAC CNTRL T

dacEn	1
dacShift	0
dacScalingFactor	134217728
dacOutAddr	0x202012C8

CONTROLLER TUNING

SYSTEM CONFIGURATIONS

speedLoopKp	0.0539
speedLoopKi	0.0360
currLoopKp	0.0100
currLoopKi	10.0000
fluxWeakeningKi	500.0000
fluxWeakeningKp	1.0000

Hall Calib

Cal Enable	0
Cal Run Time	3
Cal Curr Lim	10
Cal Align Time	4

USER OUTPUTS

CONTINUOUS READ STATUS

USER FAULT TYPES	NO_FAULTS
MOTOR STATE TYPES_T	MOTOR_IDLE

VdqFilt: d -0.016542240; q 0.3967767506

currentPI: kp 0.010000000; ki 10.000000000

piSpeed	reference	0	feedback	0.3337477296
piId	reference	0	feedback	0.0027368739
piIq	reference	0	feedback	0.0083947852

estimatedSpeed 0.3337477296; dcBusVoltage 0.2282714843; torqueLimit 0.2249999940

FAULT STATUS

GATE DRIVER FAULT STATUS	0
CONTROLLER FAULT STATUS	0

Figure 6-15. 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 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.

6.3.2 No Motor Fault [NO_MTR]

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

1. Make sure the motor phases are connected to the terminals as shown in the Hardware User's Guide.
2. If the fault persists, decrease the No-Motor lock current threshold [NO_MTR_THR].

6.3.3 Hall Invalid Fault

Hall Invalid fault is triggered if the FOC application detects an invalid Hall State such as ***0*** or ***7*** . This fault is reported through the Fault status Lock fault.

In case of Hall Invalid fault follow below steps to root cause the issue.

1. Make sure the Hall pin connections are pulled up at least to 3.3V and check the pin values are toggling as you rotate the motor shaft manually.
2. Check the GPIO's connected to the Hall pins are configured appropriately in the SYCONFIG.
3. Check the variation of the GPIO pin values in the register view in CCS debug window as you manually rotate the shaft.

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 closed loop Speed/Power/Torque loop where motor spins at the commanded speed, power 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 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).

7.1.1.2 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 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.

7.1.1.4 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.

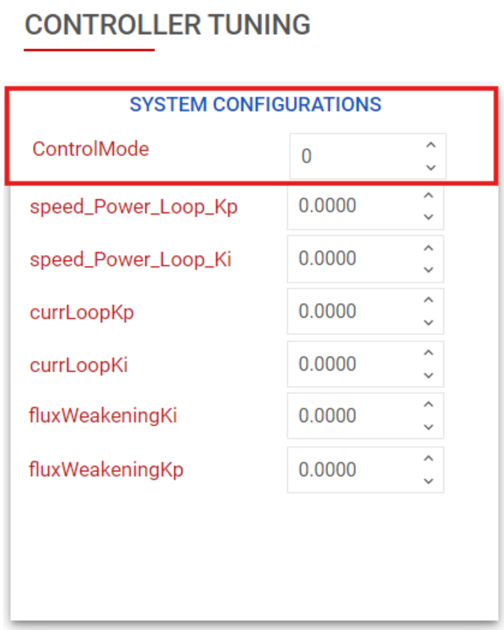
Lead Angle Control : In Voltage control mode , Lead Angle can be adjusted to derive the best efficiency from the motor for a given speed. LEAD_ANGLE configuration in CLOSED_LOOP2 register can be used to set the Lead Angle.

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.



ControlMode	0
speed_Power_Loop_Kp	0.0000
speed_Power_Loop_Ki	0.0000
currLoopKp	0.0000
currLoopKi	0.0000
fluxWeakeningKi	0.0000
fluxWeakeningKp	0.0000

Figure 7-1. Control Mode Configuration

7.1.2 Stopping Motor Quickly

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

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

7.1.3 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.

7.1.4 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 5.3.6.

7.1.5 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:

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

7.1.6 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.

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.
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 7-2 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 7-3 shows no overshoot in power supply voltage when AVS is enabled.

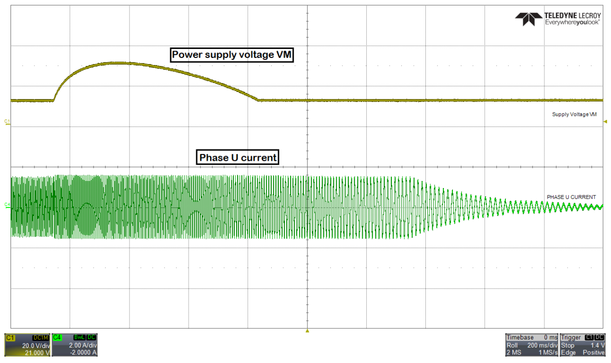


Figure 7-2. Power Supply Voltage and Phase Current Waveform When AVS is Disabled

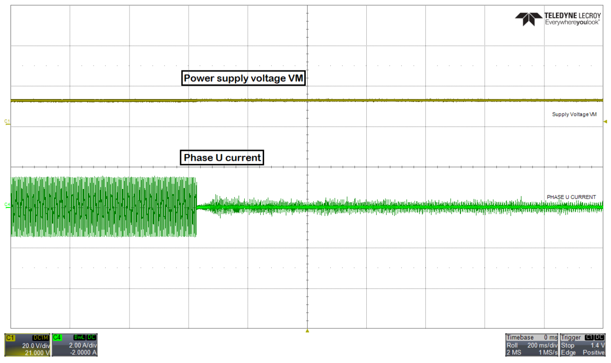


Figure 7-3. Power Supply Voltage and Phase Current Waveform When AVS is Enabled

7.1.7 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 5.3.4 can be set to 0b to get the maximum FOC execution rate of 16kHz. Setting this bit to 1b 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 16 (for example, 16kHz, 32kHz, 48kHz). For PWM frequencies of 10kHz, 20kHz, 30kHz and so on, the maximum FOC rate is 10kHz (10kHz/1, 20kHz/2, 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*

8.3.5 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):**

  DAC_OUTPUT_VOLTAGE = (VARIABLE_VALUE × DAC_SCALING_FACTOR + 1) × 1.65V

  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 higher 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 below:

  DAC_OUTPUT_VOLTAGE = (VARIABLE_VALUE << DAC_SCALE) × 3.3V

  If variable value is greater than a 12-bit value, set DAC_SCALE to negative, the DAC output follows as below

  DAC_OUTPUT_VOLTAGE = (VARIABLE_VALUE >> DAC_SCALE) × 3.3V

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	0x202005FC
B phase current	0x20200600
C phase current	0x20200604
A phase current raw ADC value	0x20200608
B phase current raw ADC value	0x2020060C
C phase current raw ADC value	0x20200614
A phase voltage	0x2020066C
B phase voltage	0x20200670
C phase voltage	0x20200674
A phase voltage raw ADC value	0x20200678
B phase voltage raw ADC value	0x2020067C
C phase voltage raw ADC value	0x20200680
D axis current	0x20200778
Q axis current	0x2020077C
D axis voltage	0x20200780
Q axis voltage	0x20200784
Estimated motor velocity filtered	0x20200A5C

Table 8-1. Address Table for DAC Monitoring (continued)

Variable	Address
Estimated rotor angle	0x20200A64
SVM output duty A phase	0x20200748
SVM output duty B phase	0x2020074C
SVM output duty C phase	0x20200750

9 Revision History

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

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

- Added support for MSPM0G3519 Devices in Section 2.1.1....................5
- Added support for MSPM0G3519 Devices in Section 2.3....................5
- Added support for MSPM0G3519 Devices in Section 2.4....................6
- Added support for MSPM0G3519 Devices in Section 2.10....................9
- Added Power Mode, Torque Mode and Voltage Control mode to Section 7.1.1.................... 50