Design Guide: TIDA-010265

750-W Motor Inverter With C2000™ and MSPM0 Reference Design

Description

This reference design is a 750-W motor drive for a washing machine or similar application, which illustrates a method to implement sensorless FOC control for a 3-phase PMSM with a FAST™ software encoder or eSMO. With a modular design, this reference design supports both the C2000™ MCU and MSPM0 series microcontroller daughter-board on the same motherboard. The hardware and software available with this reference design are tested and ready-to-use to help accelerate development time to market. The hardware design details and test results are found in this design guide.

Resources

TIDA-010265	Design Folder
TMS320F2800137	Product Folder
MSPM0G1507	Product Folder
UCC28881, TPS54202, TLV9062	Product Folder
C2000WARE-MOTORCONTROL-SDK	Tool Folder

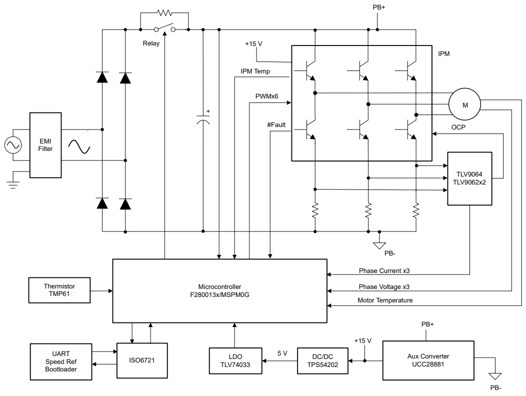
Ask our TI E2E™ support experts

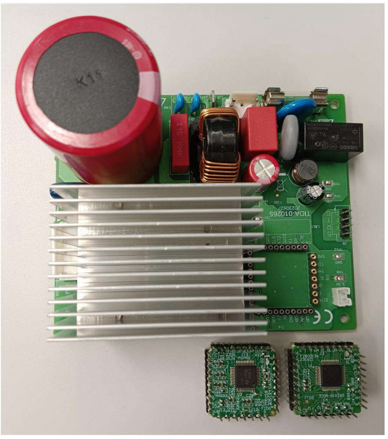
Features

- Wide operating voltage input range: 165 to 265 VAC, 50|60 Hz.
- Up to 750-W inverter stage, 15-kHz switching frequency, torque compensation, and automatic field weakening control
- Modular design with either C2000 or MSPM0 controller daughter-board on the same power motherboard
- Sensorless Field Oriented Control (FOC) motor control, supports both FAST and eSMO observer
- User-friendly graphical user interface (GUI) to control, identify, and monitor the motor

Applications

- Washer and dryer
- Air conditioner indoor unit
- Refrigerator and freezer
- Appliances: compressor





1 System Description

Motor control for major appliances or similar applications today must meet a growing list of demands on lower cost, smaller size, more power, and higher energy efficiency. Magnet Synchronous Motors (PMSM) are becoming increasingly popular in major appliance applications.

This reference design provides a single 750-W inverter motherboard with either the TMS320F2800137 and MSPM0G1507 daughter-card for control, making this reference design convenient for users to evaluate both the C2000 and MSPM0 series microcontroller on the same hardware platform.

Software supports both FAST and eSMO observer, so performance of the two devices can be compared. A user-friendly GUI also helps identify the motor, as well as tune the control parameters, thus accelerating the development time.

1.1 Terminology

SLYZ022	TI Glossary: This glossary lists and explains terms, acronyms, and definitions
PMSM	Permanent Magnet Synchronous Motor
BLDC	Brushless Direct Current
BEMF	Back Electromotive Force
PWM	Pulse Width Modulation
FET, MOSFET	Metal Oxide Semiconductor Field Effect Transistor
IGBT	Insulated Gate Bipolar Transistor
RMS	Root Mean Square
MTPA	Maximum Torque Per Ampere
FWC	Field Weakening Control
FOC	Field Oriented Control
HVAC	Heating, Ventilation, and Air Conditioning
ESMO	Enhanced Sliding-Mode Observer
PLL	Phase Locked Loop
FAST	Flux, Angle, Speed and Toque observer

1.2 Key System Specifications

The TIDA-010265 specifications are listed in Table 1-1.

Table 1-1. Key System Specifications

Parameters	TEST CONDITIONS	MIN	NOM	MAX	UNIT
	SYSTEM INPUT CHARACTERISTICS				
Input Voltage (V_{INAC})	–	165	230	265	VAC
Input Frequency (f_{LINE})	–	47	50	63	Hz
No Load Standby Power (P_{NL})	V_{INAC} = 230 V, I_{out} = 0 A	–	3.0	–	W
Input Current (I_{IN})	V_{INAC} = 230 V, I_{out} = I_{MAX}	–	8	–	A
	MOTOR INVERTER CHARACTERSITICS				
PWM switching frequency (f_{SW})	–	–	15	20	kHz
Rated output power (P_{OUT})	V_{INAC} = nom	–	500	750	W
Output current (I_{RMS})	V_{INAC} = nom	–	3	–	A
Inverter efficiency (Ƞ)	V_{INAC} = nom, P_{OUT} = nom	–	98	–	%
Motor electrical frequency (f)	V_{INAC} = min to max	20	200	400	Hz
Fault protections	Overcurrent, stall with recovery, undervoltage, overvoltage				
Drive control method and features	Sensorless-FOC with three or single shunt resistors for current sensing				
	SYSTEM CHARACTERISTICS				
Built-in auxiliary power supply	V_{INAC} = min to max	15 V ±10%, 200 mA , 3.3 V ±10%, 300 mA			
Operating ambient	Open frame	–10	25	55	°C
Board size	Length × width × height	105 mm × 85 mm × 60 mm			mm^3

WARNING

TI intends this reference design to be operated in a lab environment only and does not consider the device to be a finished product for general consumer use.

TI Intends this reference design to be used only by qualified engineers and technicians familiar with risks associated with handling high-voltage electrical and mechanical components, systems, and subsystems.

High voltage! There are accessible high voltages present on the board. The board operates at voltages and currents that can cause shock, fire, or injury if not properly handled or applied. Use the equipment with necessary caution and appropriate safeguards to avoid injuring yourself or damaging property.

Hot surface! Contact can cause burns. **Do not touch!** Some components can reach high temperatures > 55°C when the board is powered on. The user must not touch the board at any point during operation or immediately after operating, as high temperatures can be present.

CAUTION

Do not leave the design powered when unattended.

2 System Overview

2.1 Block Diagram

Figure 2-1 shows the block diagram of this reference design.

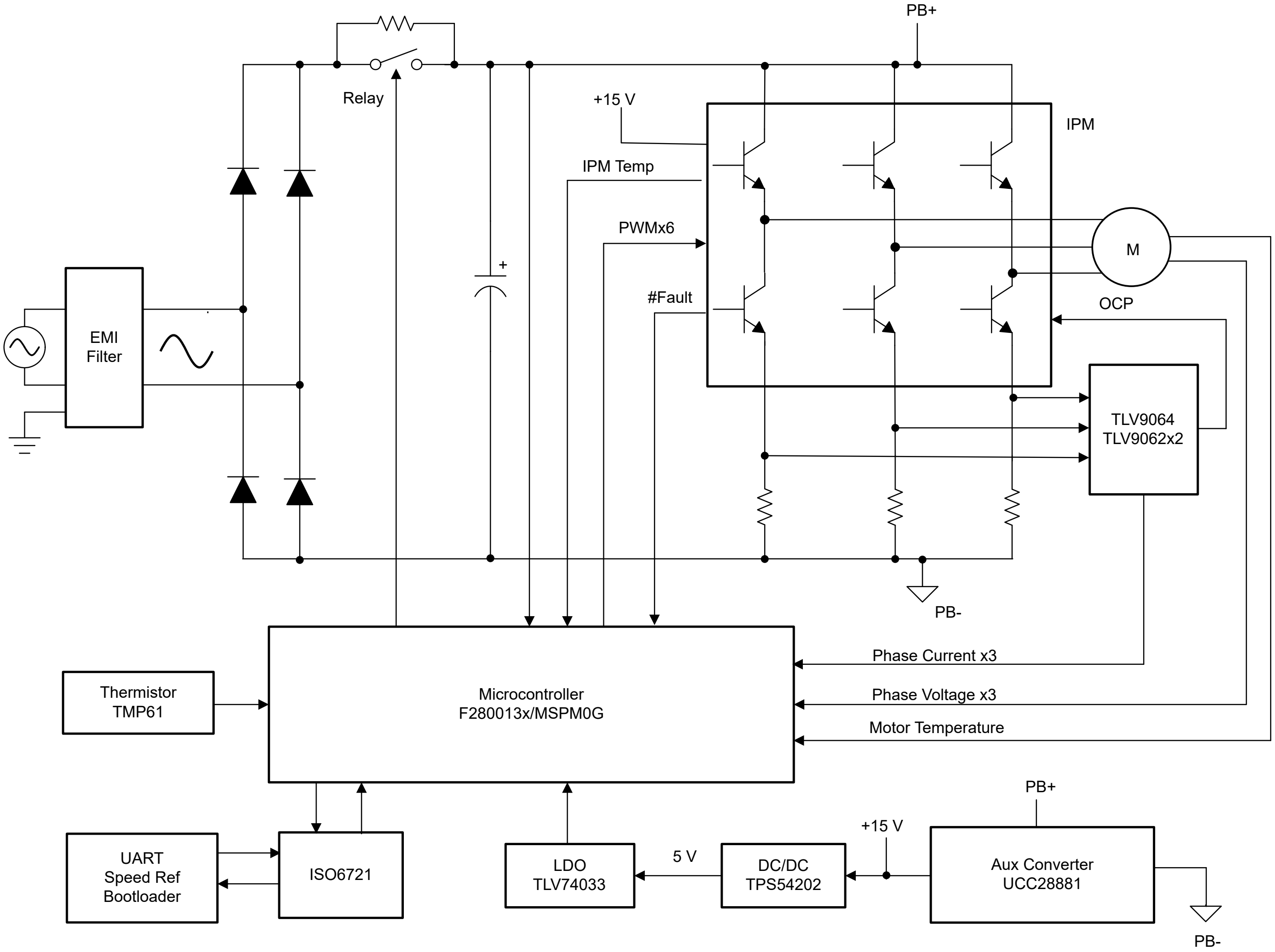


Figure 2-1. TIDA-010265 750-W Motor Inverter Block Diagram

The entire system is represented in seven blocks:

- EMI filter
- Bridge rectifier
- 3-phase inverter
- Auxiliary power supply
- TMS320F2800137 daughter-card
- MSPM0G1507 daughter card

2.2 Design Considerations

The design supports either the C2000 or MSPM0 controller for a single motor control. Highly noise-immune current and voltage sensing designs are necessary for precise motor drives. The following details the sensing and drive circuit that are used on this design. The hardware design files are available under the C2000Ware Motor Control SDK Install directory at `<install_location>\solutions\tida_010265_wminv\hardware`.

2.3 Highlighted Products

The following highlighted products are used in this reference design. Key features for selecting the devices for this reference design are revealed in the following sections. Find more details of the highlighted devices in respective product data sheets.

2.3.1 TMS320F2800137

The TMS320F280013x is a member of the C2000™ real-time microcontroller family of scalable, ultra-low latency devices designed for efficiency in power electronics applications. The real-time control subsystem is based on TI's 32-bit C28x digital-signal processor (DSP) core, which provides 120 MHz of signal-processing performance for floating- or fixed-point code running from either on-chip flash or SRAM. The C28x CPU is further boosted by the Trigonometric Math Unit (TMU) and Cyclical Redundancy Check (VCRC) extended instruction sets, speeding up common algorithms key to real-time control systems. High-performance analog blocks are integrated on the F280013x real-time microcontroller (MCU) and are closely coupled with the processing and PWM units to provide exceptional real-time signal chain performance. Fourteen PWM channels, all supporting frequency-independent resolution modes, enable control of various power stages from a 3-phase inverter to advanced multilevel power topologies. Interfacing is supported through various industry-standard communication ports (such as SPI, three SCI|URAT, I2C, and CAN) and offers multiple pin-MUXing options for excellent signal placement.

2.3.2 MSPM0G1507

The MSPM0G150x microcontrollers (MCUs) are part of the highly-integrated, ultra-low-power, 32-bit MCU family from mixed-signal processing (MSP) based on the enhanced Arm® Cortex®-M0+ 32-bit core platform operating at up to 80-MHz frequency. These cost-optimized MCUs offer high-performance analog peripheral integration, support extended temperature ranges from –40°C to 125°C, and operate with supply voltages ranging from 1.62 V to 3.6 V. The MSPM0G150x devices provide up to 128-KB embedded Flash program memory with built-in error correction code (ECC) and up to 32-KB SRAM with hardware parity option. The devices also incorporate a memory protection unit, seven-channel DMA, math accelerator, and a variety of high-performance analog peripherals such as two 12-bit 4-MSPS ADCs, configurable internal shared voltage reference, one 12-bit 1-MSPS DAC, three high-speed comparators with built-in reference digital-to-analog converters (DACs), two zero-drift, zero-crossover op amps with programmable gain, and one general-purpose amplifier. These devices also offer intelligent digital peripherals such as three 16-bit advanced control timers, three 16-bit general purpose timers, one 24-bit high-resolution timer, two windowed-watchdog timers, and one RTC with alarm and calendar mode. These devices provide data integrity and encryption peripherals and enhanced communication interfaces (four UARTs, two I2Cs, two serial-peripheral interfaces (SPIs)).

2.3.3 TMP6131

The TMP61x linear thermistor offers linearity and consistent sensitivity across temperature to enable simple and accurate methods for temperature conversion. The low power consumption and a small thermal mass of the device minimizes the impact of self-heating.

With built-in fail-safe behaviors at high temperatures and powerful immunity to environmental variation, these devices are designed for a long lifetime of high performance. The small size of the TMP6 series also allows for close placement to heat sources and quick response times.

2.3.4 UCC28881

The UCC28881 integrates the controller and a 14-Ω, 700-V power MOSFET into one monolithic device. The device also integrates a high-voltage current source, enabling start-up and operation directly from the rectified mains voltage. The UCC28881 is the same family device of the UCC28880, with higher current.

The low quiescent current of the device enables excellent efficiency. With the UCC28881, the most common converter topologies, such as buck, buck- boost, and flyback can be built using a minimum number of external components.

2.3.5 TPS54202

The TPS54202 is a 4.5-V to 28-V input voltage range, 2-A synchronous buck converter. The device includes two integrated switching FETs, internal loop compensation and 5-ms internal soft start to reduce component count.

Advanced Eco-mode implementation maximizes the light load efficiency and reduces the power loss.

Cycle-by-cycle current limit in both high-side MOSFETs protects the converter in an overload condition and is enhanced by a low-side MOSFET freewheeling current limit which prevents current runaway.

2.3.6 TLV9062

The TLV9062 is a dual-low-voltage (1.8 V to 5.5 V) operational amplifier (op amp) with rail-to-rail input- and output-swing capabilities. This device is a highly cost-effective design for applications where low-voltage operation, a small footprint, and high capacitive load drive are required. Although the capacitive load drive of the TLV906x is 100 pF, the resistive open-loop output impedance makes stabilizing with higher capacitive loads simpler. The TLV906xS devices include a shutdown mode that allow the amplifiers to switch into standby mode with typical current consumption less than 1 µA. The TLV906xS family helps simplify system design, because the family is unity-gain stable, integrates the RFI and EMI rejection filter, and provides no phase reversal in overdrive condition.

2.3.7 TLV74033

TheTLV740P low-dropout (LDO) linear regulator is a low quiescent current LDO with excellent line and load transient performance designed for power-sensitive applications. This device provides a typical accuracy of 1%.

The TLV740P also provides inrush current control during device power up and enabling. The TLV740P limits the input current to the defined current limit to avoid large currents from flowing from the input power source. This functionality is especially important in battery-operated devices.

2.4 System Design Theory

The main focus of this reference design is single motor control for washers or similar appliance applications.

2.4.1 Hardware Design

A typical motor control board has several blocks: auxiliary power supply, inverter, current and voltage sensing, protection circuit, and a microcontroller. These design concepts are explained in this section.

2.4.1.1 Modular Design

This reference design has two daughterboards and one motherboard for a modular design. One daughterboard has the MSPM0G1507 microcontroller and a two-phase current-amplifying circuit based on the two internal high-end op amps. Another daughterboard has the TMS320F2800137 microcontroller and two TLV9062 devices as phase-current amplifiers. The motherboard holds all the power devices including AC filters, rectifier, and the Intelligent Power Module (IPM). Figure 2-2 and Figure 2-3 show the motherboard and daughterboards.

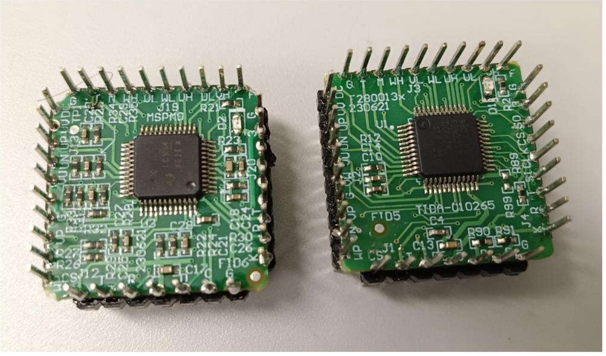


Figure 2-2. TIDA-010265 Daughterboards

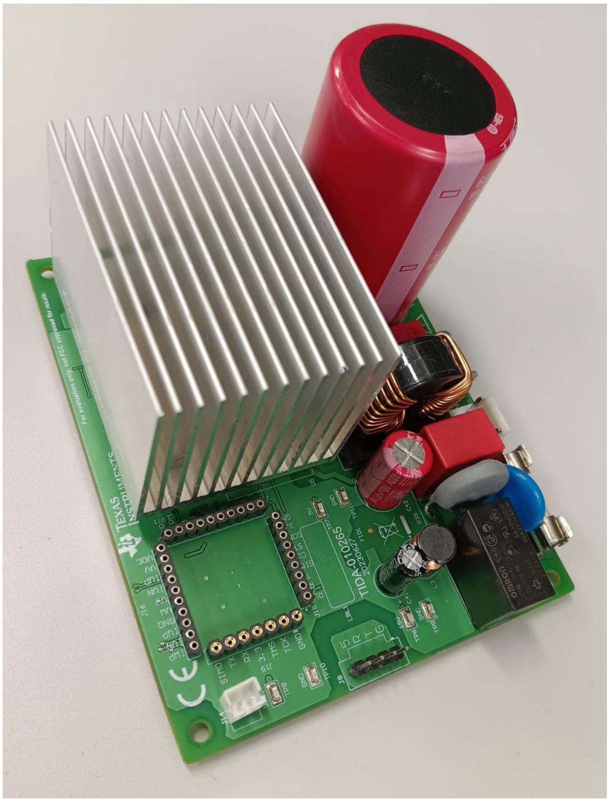


Figure 2-3. TIDA-010265 Motherboard

2.4.1.2 High-Voltage Buck Auxiliary Power Supply

A non-isolated UCC28881-based high-voltage buck supply provides auxiliary power supply for this reference design to delivery up to 200 mA for 15 VDC.Figure 2-4 shows the UCC28881 high-voltage buck supply circuit.

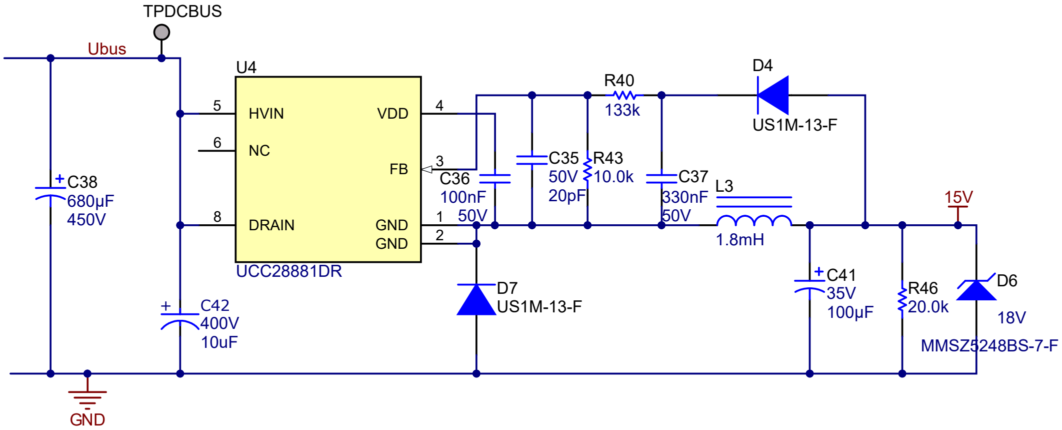


Figure 2-4. High-Voltage Buck Power Supply Circuit

2.4.1.3 DC Link Voltage Sensing

The DC voltage sensing circuit is used to convert the rectified voltage signal into a low-voltage signal implemented by a low-cost resistor network as shown in Figure 2-5. The DC bus voltage can also be used to estimate AC input voltage.

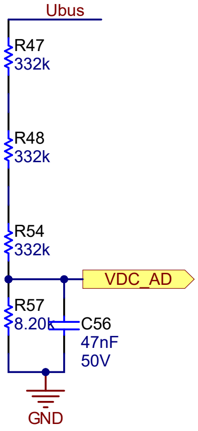


Figure 2-5. DC Bus Voltage Sensing Circuit

2.4.1.4 Motor Phase Voltage Sensing

C2000 software for the TIDA-010265 supports both enhanced Slide Mode Observer (eSMO) and Flux, Angle, Speed, and Torque (FAST™) observer. The FAST observer can improve low-speed performance and reduce speed tolerance; however, FAST needs 3 motor phase voltage sensing in addition to 3-phase current sensing. Section 2.4.2.4.2 has a detailed explanation of motor phase voltage sensing design.

2.4.1.5 Motor Phase Current Sensing

The MS320F2800137 daughterboard is designed to support 1-3 phase current sensing, while the MSPM0G1507 daughterboard supports 1-2 phase current sensing. Section 2.4.2.4.1 has the details for motor phase current sensing design.

2.4.1.6 External Overcurrent Protection

This reference design implements overcurrent protection (OCP) with both external comparator and internal comparator. Figure 2-6 shows OCP with the external comparator, this circuit summarizes three phases of current, then compares to a reference voltage at negative input of U10 to create a high voltage to IPM, then IPM creates overcurrent fault signal and report to microcontroller. The exact overcurrent protection current can be calculated by below equations.

$$V_{-} = \frac{3.3\ \mathrm{V}}{R_{20} + R_{108}} \times R_{108} = \frac{3.3\ \mathrm{V}}{20\ \mathrm{k} + 1\ \mathrm{k}} \times 1\mathrm{k} = 0.1571\ \mathrm{V} \tag{1}$$

$$V_{+} = \frac{I_{ocp} \times R_{80}}{R87 + (R_{104} + R_{105})/2)} \times \left(\left(R_{104} + R_{105}\right)/2\right) = \frac{0.05 \times I_{ocp}}{300 + 150} \times 150 = \frac{0.05 \times I_{ocp}}{3} \tag{2}$$

$$I_{ocp} = \frac{V_{+}}{R_{87}} \times 3 = \frac{0.1571}{0.05} \times 3 = 9.4286\ \mathrm{A} \tag{3}$$

Both MS320F2800137 and MSPM01507 daughterboards can be triggered by this external OCP circuit.

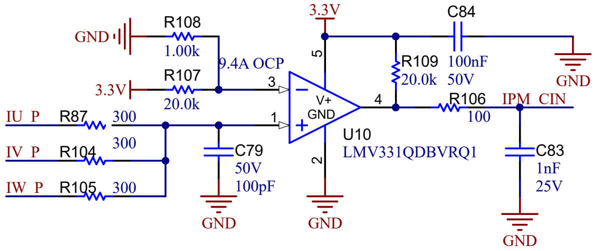


Figure 2-6. External Overcurrent Protection Circuit

2.4.1.7 Internal Overcurrent Protection for TMS320F2800F137

The TMS320F2800F137 has internal window comparators which can be configured to monitor three phases current, there is not any software or interrupt delay for internal comparators to trigger overcurrent to stop PWM, this quickly protects external IPM or power devices.

2.4.2 Three-Phase PMSM Drive

Permanent Magnet Synchronous motor (PMSM) has a wound stator, a permanent magnet rotor assembly, and internal or external devices to sense rotor position. The sensing devices provide position feedback for adjusting frequency and amplitude of stator voltage reference properly to maintain rotation of the magnet assembly. The combination of an inner permanent magnet rotor and outer windings offers the advantages of low rotor inertia, efficient heat dissipation, and reduction of the motor size.

- Synchronous motor construction: Permanent magnets are rigidly fixed to the rotating axis to create a constant rotor flux. This rotor flux usually has a constant magnitude. When energized, the stator windings create a rotating electromagnetic field. To control the rotating magnetic field, the stator currents must be controlled.
- The actual structure of the rotor varies depending on the power range and rated speed of the machine. Permanent magnets are an excellent choice for synchronous machines ranging up-to a few Kilowatts. For higher power ratings the rotor usually consists of windings in which a DC current circulates. The mechanical structure of the rotor is designed for number of poles desired, and the desired flux gradients desired.
- The interaction between the stator and rotor fluxes produces torque. Since the stator is firmly mounted to the frame, and the rotor is free to rotate, the rotor rotates, producing a useful mechanical output as shown in Figure 2-7.
- The angle between the rotor magnetic field and stator field must be carefully controlled to produce maximum torque and achieve high electromechanical conversion efficiency. For this purpose fine-tuning is needed after closing the speed loop using a sensorless algorithm to draw the minimum amount of current under the same speed and torque conditions.
- The rotating stator field must rotate at the same frequency as the rotor permanent magnetic field; otherwise, the rotor experiences rapidly alternating positive and negative torque. This results in less than excellent torque production, and excessive mechanical vibration, noise, and mechanical stresses on the machine parts. In addition, if the rotor inertia prevents the rotor from being able to respond to these oscillations, the rotor stops rotating at the synchronous frequency, and responds to the average torque as seen by the stationary rotor: Zero. This means that the machine experiences a phenomenon known as *pull-out*. This is also the reason why the synchronous machine is not self starting.
- The angle between the rotor field and the stator field must be equal to 90º to obtain the highest mutual torque production. This synchronization requires knowing the rotor position to generate the right stator field.
- The stator magnetic field can be made to have any direction and magnitude by combining the contribution of different stator phases to produce the resulting stator flux.

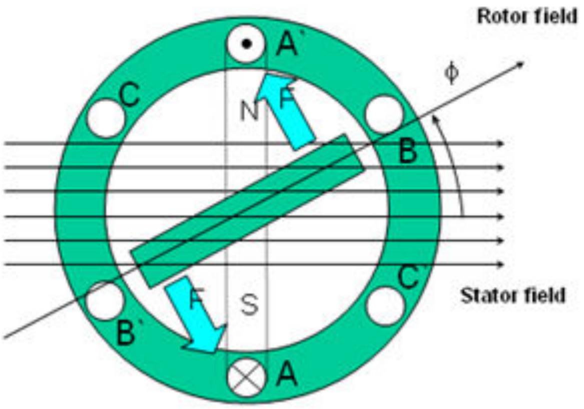


Figure 2-7. Interaction Between the Rotating Stator Flux and the Rotor Flux Produces Torque

2.4.2.1 Field-Oriented Control of PM Synchronous Motor

To achieve better dynamic performance, a more complex control scheme needs to be applied, to control the PM motor. With the mathematical processing power offered by the microcontrollers, advanced control strategies can be implemented, which use mathematical transformations to decouple the torque generation and the magnetization functions in PM motors. Such de-coupled torque and magnetization control is commonly called rotor flux oriented control, or simply Field-Oriented Control (FOC).

In a direct current (DC) motor, the excitation for the stator and rotor is independently controlled, the produced torque and the flux can be independently tuned as shown in Figure 2-8. The strength of the field excitation (for example, the magnitude of the field excitation current) sets the value of the flux. The current through the rotor windings determines how much torque is produced. The commutator on the rotor plays an interesting part in the torque production. The commutator is in contact with the brushes, and the mechanical construction is designed to switch into the circuit the windings that are mechanically aligned to produce the maximum torque. This arrangement then means that the torque production of the machine is fairly near exceptional all the time. The key point here is that the windings are managed to keep the flux produced by the rotor windings orthogonal to the stator field.

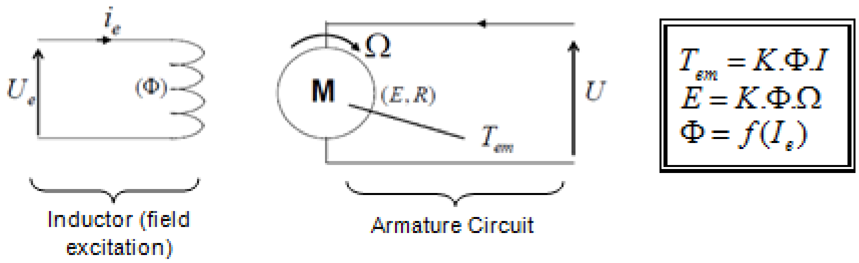


Figure 2-8. Flux and Torque are Independently Controlled in DC Motor Model

The goal of the FOC (also called vector control) on synchronous and asynchronous machines is to be able to separately control the torque-producing and magnetizing flux components. FOC control allows decoupling of the torque and of the magnetizing flux components of stator current. With decoupled control of the magnetization, the torque producing component of the stator flux can now be thought of as independent torque control. To decouple the torque and flux, it is necessary to engage several mathematical transforms, and this is where the microcontrollers add the most value. The processing capability provided by the microcontrollers enables these mathematical transformations to be carried out very quickly. This, in turn, implies that the entire algorithm controlling the motor can be executed at a fast rate, enabling higher dynamic performance. In addition to the decoupling, a dynamic model of the motor is now used for the computation of many quantities such as rotor flux angle and rotor speed. This means that the effect is accounted for, and the overall quality of control is better.

According to the electromagnetic laws, the torque produced in the synchronous machine is equal to the vector cross product of the two existing magnetic fields as in Equation 4 .

$$\tau_{\mathrm{em}} = \vec{\mathrm{B}}_{\mathrm{stator}} \times \vec{\mathrm{B}}_{\mathrm{rotor}} \tag{4}$$

This expression shows that the torque is maximum if stator and rotor magnetic fields are *orthogonal* meaning to maintain the load at 90 degrees. If this condition can be provided all the time and if the flux can be oriented correctly, the torque ripple is reduced and a better dynamic response is provided. However, the constraint is to know the rotor position: this can be achieved with a position sensor such as incremental encoder. For low-cost applications where the rotor is not accessible, different rotor position observer strategies are applied to get rid of position sensor.

In brief, the goal is to maintain the rotor and stator flux in quadrature: the goal is to align the stator flux with the q axis of the rotor flux, for example, orthogonal to the rotor flux. To do this, the stator current component in quadrature with the rotor flux is controlled to generate the commanded torque, and the direct component is set to zero. The direct component of the stator current can be used in some cases for field weakening, which has the effect of opposing the rotor flux, and reducing the back-emf, which allows for operation at higher speeds.

The FOC consists of controlling the stator currents represented by a vector. This control is based on projections which transform a three-phase time and speed dependent system into a two coordinate (d and q coordinates) time invariant system. These projections lead to a structure similar to that of a DC machine control. FOC machines need two constants as input references: the torque component (aligned with the q coordinate) and the flux component (aligned with d coordinate). As FOC is simply based on projections, the control structure handles instantaneous electrical quantities. This makes the control accurate in every working operation (steady state and transient) and independent of the limited bandwidth mathematical model. The FOC thus solves the classic scheme problems, in the following ways:

- The ease of reaching constant reference (torque component and flux component of the stator current)
- The ease of applying direct torque control because in the (d, q) reference frame the expression of the torque is defined in Equation 5.

$$\tau_{em} \propto \psi_R \times i_{sq} \tag{5}$$

By maintaining the amplitude of the rotor flux (ψ_R) at a fixed value, a linear relationship between torque and torque component (i_{Sq}) is obtained. Therefore, the torque can be controlled by controlling the torque component of the stator current vector.

2.4.2.1.1 Space Vector Definition and Projection

The 3-phase voltages, currents, and fluxes of AC motors can be analyzed in terms of complex space vectors. With regard to the currents, the space vector can be defined as follows. Assuming that i_a, i_b, i_c are the instantaneous currents in the stator phases, then the complex stator current vector is defined in Equation 6.

$$\bar{i}_s = i_a + \alpha i_b + \alpha^2 i_c \tag{6}$$

where

- $\alpha = e^{j\frac{2}{3}\pi}$ and $\alpha^2 = e^{j\frac{4}{3}\pi}$ represent the spatial operators

Figure 2-9 shows the stator current complex space vector.

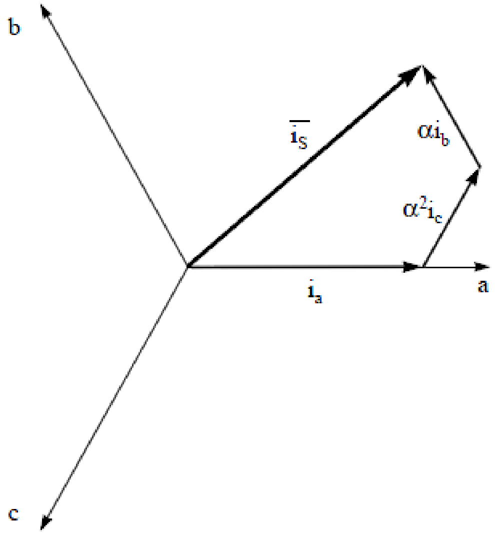


Figure 2-9. Stator Current Space Vector and Component in (a, b, c) Frame

where

- a, b, and c are the three-phase system axes

This current space vector depicts the three-phase sinusoidal system which still needs to be transformed into a two time invariant co-ordinate system. This transformation can be split into two steps:

- $(\mathrm{a},\ \mathrm{b}) \Rightarrow (\alpha, \beta)$ (Clarke transformation) which outputs a 2-coordinate time-variant system
- $(\alpha, \beta) \Rightarrow (\mathrm{d},\ \mathrm{q})$ (Park transformation) which outputs a 2-coordinate time-invariant system

2.4.2.1.1.1 $(\mathrm{a},\ \mathrm{b}) \Rightarrow (\alpha, \beta)$ ***Clarke Transformation***

The space vector can be reported in another reference frame with only two orthogonal axis called (α, β). Assuming that the axis a and the axis α*lpha* are in the same direction yields the vector diagram shown in Figure 2-10.

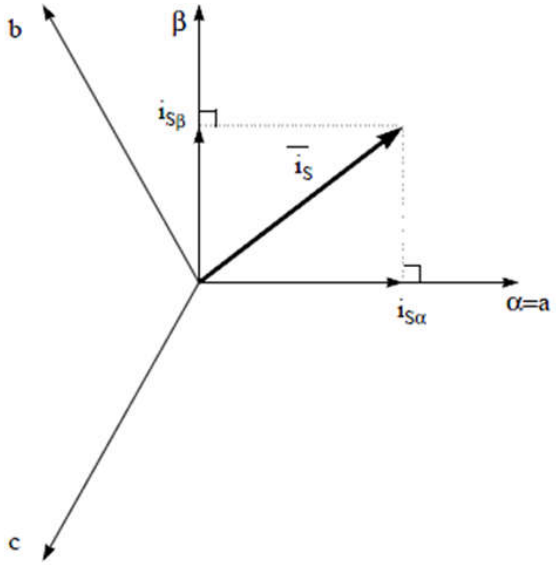


Figure 2-10. Stator Current Space Vector in the Stationary Reference Frame

The projection that modifies the 3-phase system into the (α, β) 2-dimension orthogonal system is presented in Equation 7.

$$\mathrm{i}_{s\alpha} = \mathrm{i}_\mathrm{a} \qquad (7)$$
$$\mathrm{i}_{s\beta} = \frac{1}{\sqrt{3}}\mathrm{i}_\mathrm{a} + \frac{2}{\sqrt{3}}\mathrm{i}_\mathrm{b}$$

The two phase (α, β) currents are still dependent on time and speed.

2.4.2.1.1.2 $(\alpha, \beta) \Rightarrow (\mathrm{d},\ \mathrm{q})$ ***Park Transformation***

This is the most important transformation in the FOC. In fact, this projection modifies a 2-phase orthogonal system (α, β) in the (d, q) rotating reference frame. Considering the d axis aligned with the rotor flux, Figure 2-11 shows the relationship for the current vector from the two reference frame.

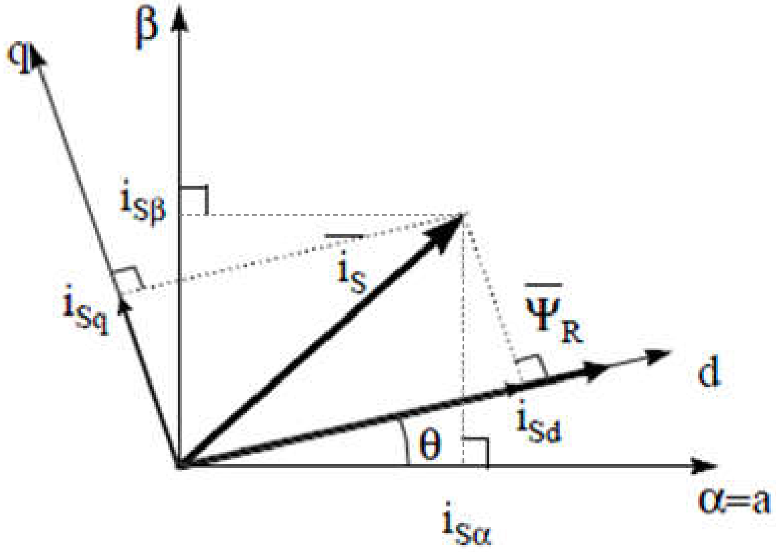


Figure 2-11. Stator Current Space Vector in the d,q Rotating Reference Frame

The flux and torque components of the current vector are determined by Equation 8.

$$\begin{aligned} i_{sd} &= i_{s\alpha}\cos(\theta) + i_{s\beta}\sin(\theta) \\ i_{sq} &= -i_{s\alpha}\sin(\theta) + i_{s\beta}\cos(\theta) \end{aligned} \tag{8}$$

where

- θ is the rotor flux position

These components depend on the current vector (α, β) components and on the rotor flux position; if the right rotor flux position is known then, by this projection, the d,q component becomes a constant. Two phase currents now turn into dc quantity (time-invariant). At this point the torque control becomes easier where constant i_{sd} (flux component) and i_{sq} (torque component) current components controlled independently.

2.4.2.1.2 Basic Scheme of FOC for AC Motor

Figure 2-12 summarizes the basic scheme of torque control with FOC.

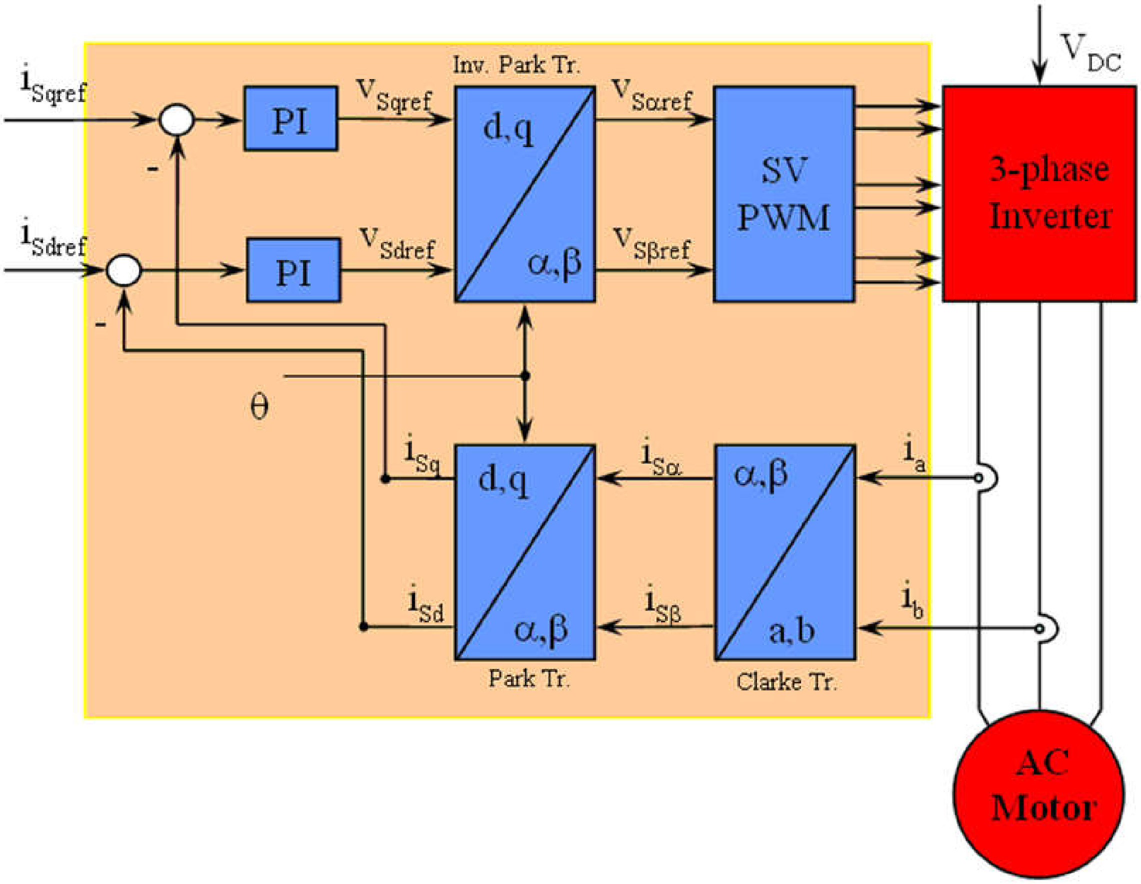


Figure 2-12. Basic Scheme of FOC for AC Motor

Two motor phase currents are measured. These measurements feed the Clarke transformation module. The outputs of this projection are designated $i_{s\alpha}$ and $i_{s\beta}$. These two components of the current are the inputs of the Park transformation that gives the current in the d,q rotating reference frame. The i_{sd} and i_{sq} components are compared to the references i_{sdref} (the flux reference component) and i_{sqref} (the torque reference component). At this point, this control structure shows an interesting advantage: the structure can be used to control either synchronous or induction machines by simply changing the flux reference and obtaining rotor flux position. As in synchronous permanent magnet a motor, the rotor flux is fixed determined by the magnets; there is no need to create one. Hence, when controlling a PMSM, set i_{sdref} to zero. As an AC induction motor needs a rotor flux creation to operate, the flux reference must not be zero. This conveniently solves one of the major drawbacks of the *classic* control structures: the portability from asynchronous to synchronous drives. The torque command i_{sqref} can be the output of the speed regulator when a speed FOC is used. The outputs of the current regulators are V_{sdref} and V_{sqref}; these outputs are applied to the inverse Park transformation. The outputs of this projection are $V_{s\alpha ref}$ and $V_{s\beta ref}$ which are the components of the stator vector voltage in the (α, β) stationary orthogonal reference frame. These are the inputs of the Space Vector PWM. The outputs of this block are the signals that drive the inverter. Note that both Park and inverse Park transformations need the rotor flux position. Obtaining this rotor flux position depends on the AC machine type (synchronous or asynchronous machine).

2.4.2.1.3 Rotor Flux Position

Knowledge of the rotor flux position is the core of the FOC. In fact if there is an error in this variable the rotor flux is not aligned with the d-axis and i_{sd} and i_{sq} are incorrect flux and torque components of the stator current. Figure 2-13 shows the (a, b, c), (α, β) and (*d, q*) reference frames, and the correct position of the rotor flux, the stator current and stator voltage space vector that rotates with d, q reference at synchronous speed.

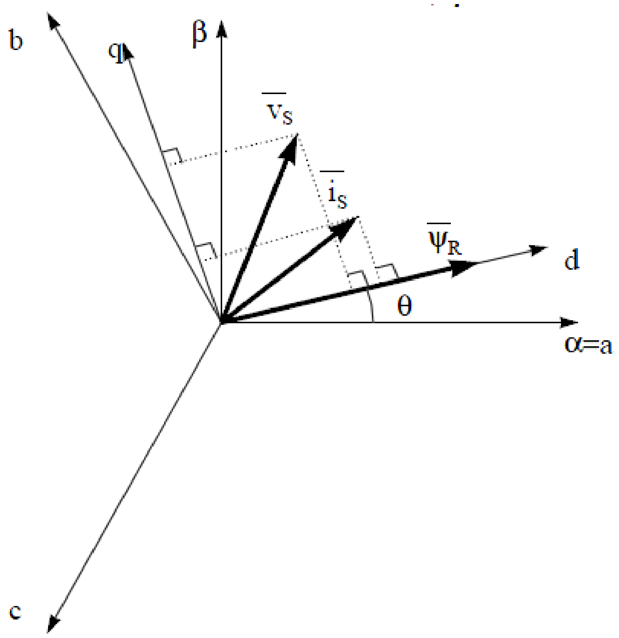


Figure 2-13. Current, Voltage and Rotor Flux Space Vectors in the (d, q) Rotating Reference Frame

The measure of the rotor flux position is different when considering the synchronous or asynchronous motor:

- In the synchronous machine the rotor speed is equal to the rotor flux speed. Then θ (rotor flux position) is directly measured by the position sensor or by integration of rotor speed.
- In the asynchronous machine, the rotor speed is not equal to the rotor flux speed (there is a slip speed), then a particular method is needed to calculate θ. The basic method is the use of the current model which needs two equations of the motor model in *d, q* reference frame.

Theoretically, the FOC for the PMSM drive allows the motor torque to be controlled independently with the flux like DC motor operation. In other words, the torque and flux are decoupled from each other. The rotor position is required for variable transformation from stationary reference frame to synchronously rotating reference frame. As a result of this transformation (so called Park transformation), q-axis current is controlling torque while d-axis current is forced to zero. Therefore, the key module of this system is the estimation of rotor position using enhance Sliding-Mode Observer (eSMO) or FAST estimator.

Figure 2-14 shows the overall block diagram of sensorless FOC of fan PMSM using eSMO with flying start in this reference design.

Figure 2-15 shows the overall block diagram of sensorless FOC of compressor PMSM using eSMO with field weakening control (FWC) and maximum torque per ampere (MTPA) in this reference design.

Figure 2-16 shows the overall block diagram of sensorless FOC of fan PMSM using FAST with flying start in this reference design.

Figure 2-17 shows the overall block diagram of sensorless FOC of compressor PMSM using FAST with field weakening control (FWC) and maximum torque per ampere (MTPA) in this reference design.

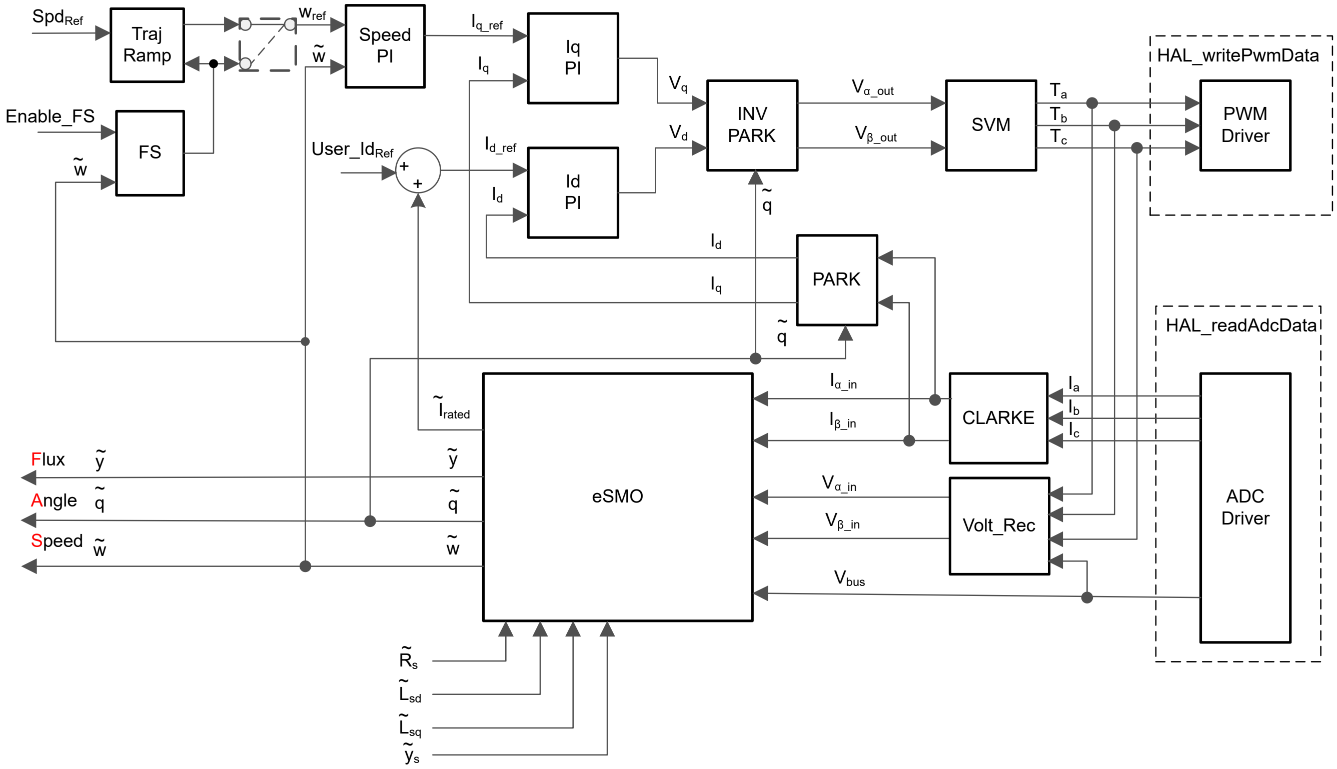


Figure 2-14. Sensorless FOC of PMSM Using eSMO With Flying Start (FS)

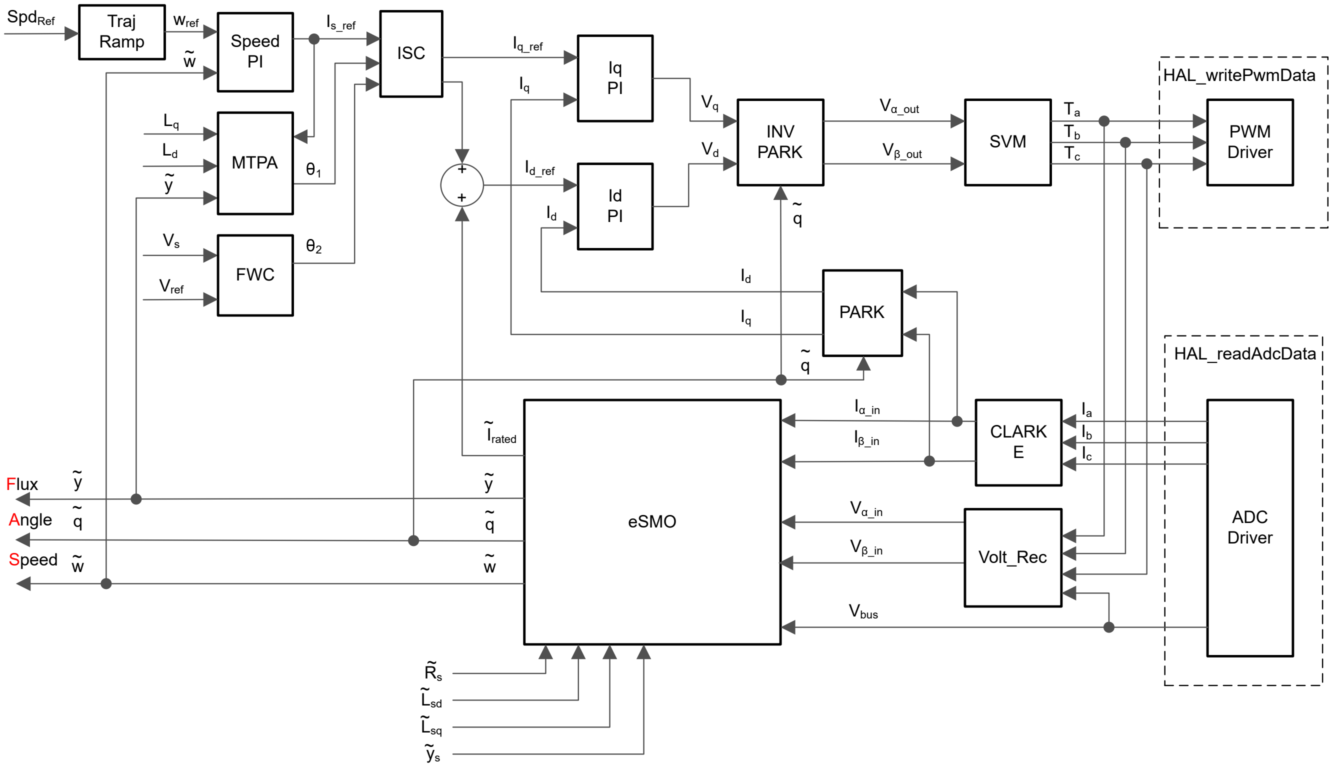


Figure 2-15. Sensorless FOC of PMSM Using eSMO With FWC and MTPA

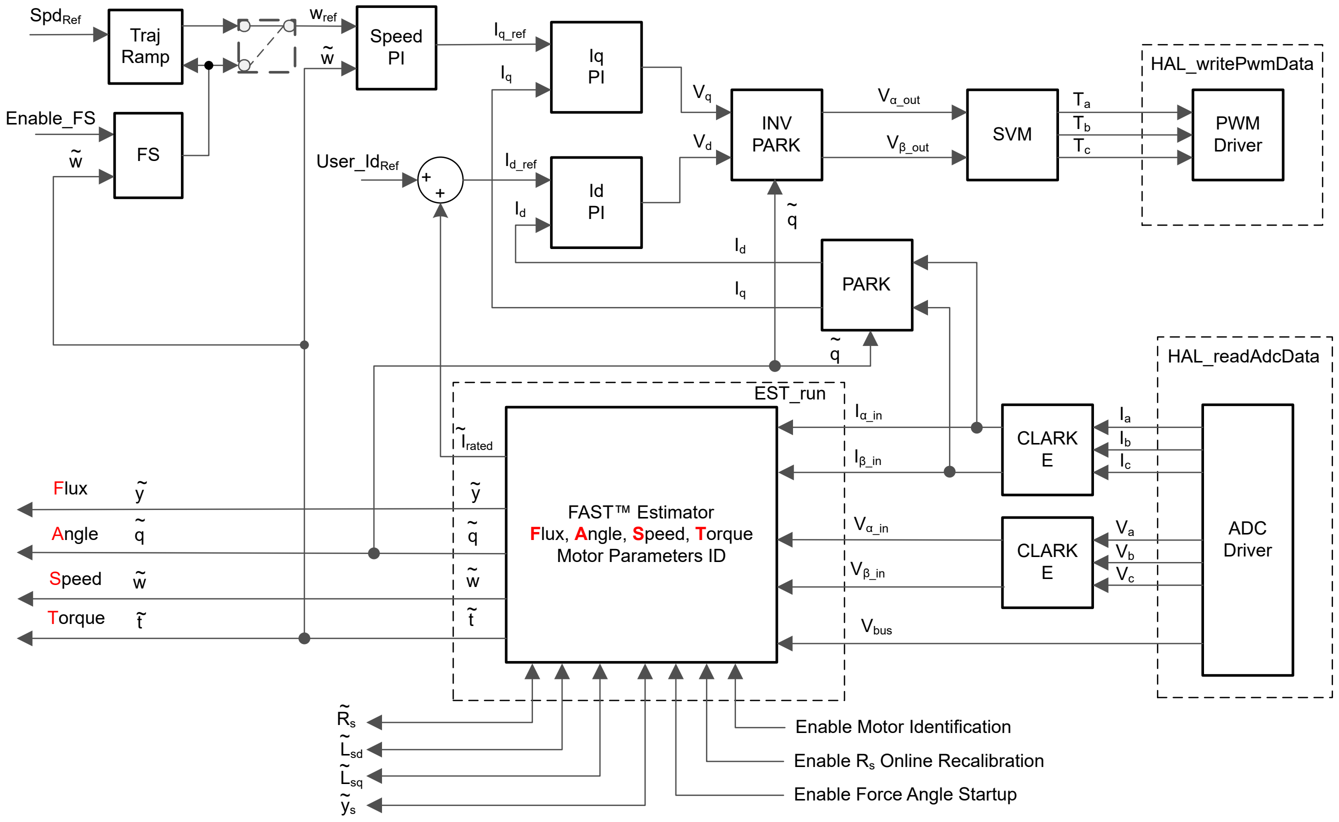


Figure 2-16. Sensorless FOC of PMSM Using FAST With Flying Start (FS)

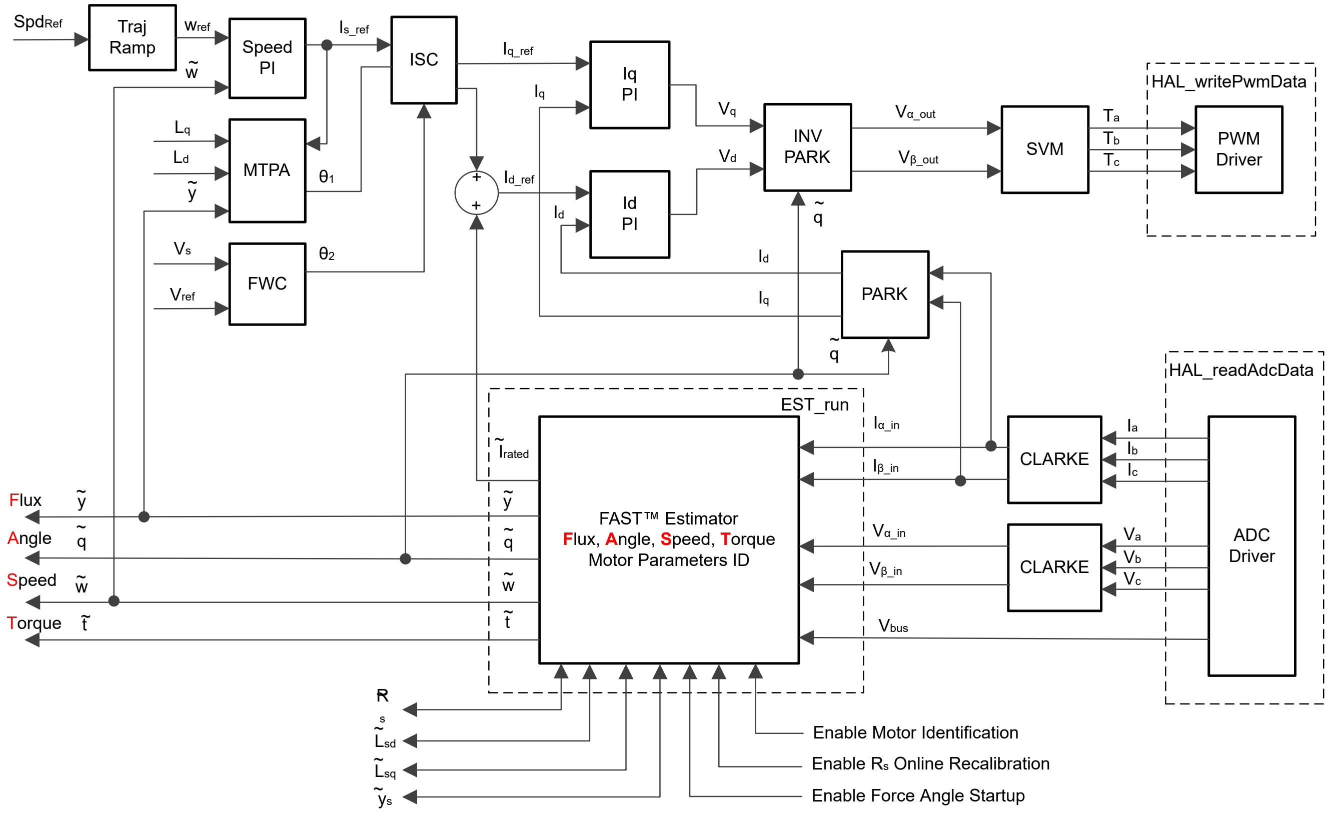


Figure 2-17. Sensorless FOC of PMSM Using FAST With FWC and MTPA

2.4.2.2 Sensorless Control of PM Synchronous Motor

In home appliance applications, using a mechanical sensor increases cost, size, and reliability problems. To overcome these problems, sensorless control methods are implemented. Several estimation methods are used to get the rotor speed and position information without mechanical position sensor. The sliding mode observer (SMO) is commonly utilized due to the various attractive features including reliability, desired performance, and robustness against system parameter variations.

2.4.2.2.1 Enhanced Sliding Mode Observer With Phase-Locked Loop

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

2.4.2.2.1.1 Mathematical Model and FOC Structure of an IPMSM

The sensorless FOC structure for an IPMSM is illustrated in Figure 2-18. In this system, the eSMO is used for achieving the sensorless control an IPMSM system, and the eSMO model is designed by utilizing the back EMF model together with a PLL model for estimating the rotor position and speed.

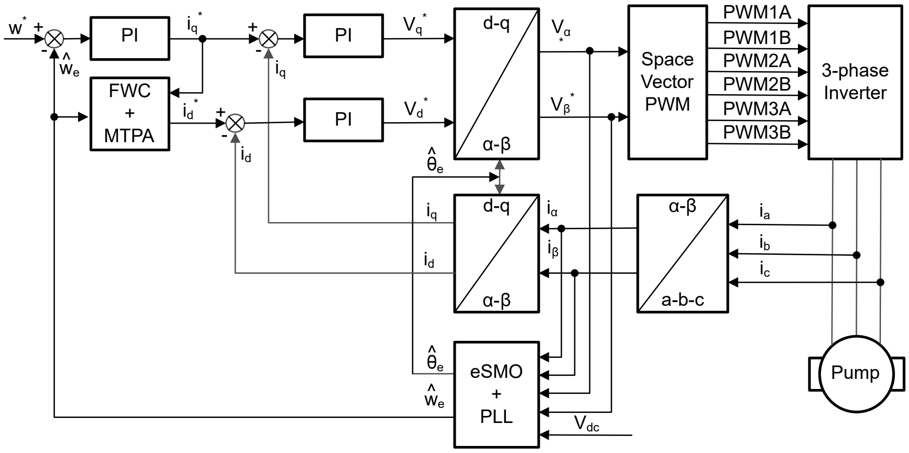


Figure 2-18. Sensorless FOC Structure of an IPMSM System

An IPMSM consists of a three-phase stator winding (a, b, c axes), and permanent magnets (PM) rotor for excitation. The motor is controlled by a standard three-phase inverter. An IPMSM can be modeled by using phase a-b-c quantities. Through proper coordinate transformations, the dynamic PMSM models in the d-q rotor reference frame and the α-β stationary reference frame can be obtained. The relationship among these reference frames are illustrated in Equation 9. The dynamic model of a generic PMSM can be written in the d-q rotor reference frame as:

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} R_s + pL_d & -\omega_e L_q \\ \omega_e L_d & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_e \lambda_{pm} \end{bmatrix} \quad (9)$$

where

- v_d and v_q are the q-axis and d-axis stator terminal voltages, respectively
- i_d and i_q are the d-axis and q-axis stator currents, respectively
- L_d and L_q are the q-axis and d-axis inductances, respectively
- *p* is the derivative operator, a short notation of $\frac{d}{dt}$
- λ_{pm} is the flux linkage generated by the permanent magnets
- R_s is the resistance of the stator windings
- ω_e is the electrical angular velocity of the rotor

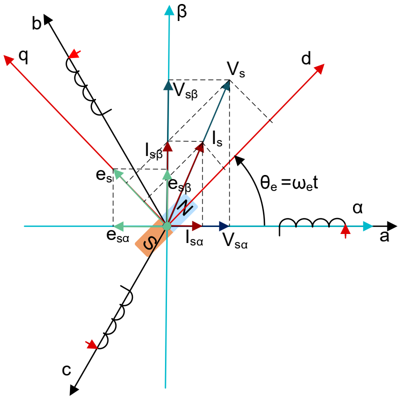


Figure 2-19. Definitions of Coordinate Reference Frames for PMSM Modeling

By using the inverse Park transformation as shown in Figure 2-19, the dynamics of the PMSM can be modeled in the α-β stationary reference frame as shown in Equation 10:

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} R_s + pL_d & \omega_e(L_d - L_q) \\ -\omega_e(L_d - L_q) & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} \tag{10}$$

where

- e_α and e_β are components of extended electromotive force (EEMF) in the α-β axis and can be defined as shown in Equation 11:

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \left(\lambda_{pm} + (L_d - L_q)i_d\right)\omega_e \begin{bmatrix} -\sin(\theta_e) \\ \cos(\theta_e) \end{bmatrix} \tag{11}$$

According to Equation 10 and Equation 11, the rotor position information can be decoupled from the inductance matrix by means of the equivalent transformation and the introduction of the EEMF concept, so that the EEMF is the only term that contains the rotor pole position information. And then the EEMF phase information can be directly used to realize the rotor position observation. Rewrite the IPMSM voltage equation Equation 10 as a state equation using the stator current as a state variable:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{1}{L_d} \begin{bmatrix} -R_s & -\omega_e(L_d - L_q) \\ \omega_e(L_d - L_q) & -R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} V_\alpha - e_\alpha \\ V_\beta - e_\beta \end{bmatrix} \tag{12}$$

Since the stator current is the only physical quantity that can be directly measured, the sliding surface is selected on the stator current path:

$$S(x) = \begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix} = \begin{bmatrix} \tilde{i}_\alpha \\ \tilde{i}_\beta \end{bmatrix} \tag{13}$$

where

- $\hat{i}_\alpha$ and $\hat{i}_\beta$ are the estimated currents
- the superscript **^** indicates the estimated value
- the superscript "~" indicates the variable error which refers to the difference between the observed value and the actual measurement value

2.4.2.2.1.2 Design of ESMO for the IPMSM

Figure 2-20 shows the conventional PLL integrated into the SMO.

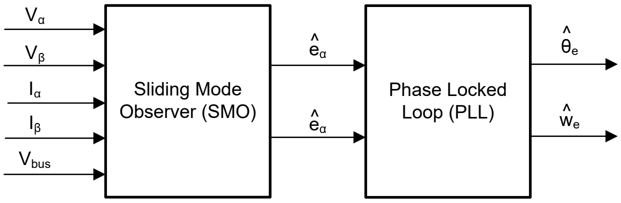


Figure 2-20. Block Diagram of eSMO With PLL for a PMSM

The traditional reduced-order sliding-mode observer is constructed, with the mathematical model shown in Equation 14 and the block diagram shown in Figure 2-21.

$$\begin{bmatrix} \dot{\hat{i}}_\alpha \\ \dot{\hat{i}}_\beta \end{bmatrix} = \frac{1}{L_d}\begin{bmatrix} -R_s & -\widehat{\omega}_e\left(L_d - L_q\right) \\ \widehat{\omega}_e\left(L_d - L_q\right) & -R_s \end{bmatrix}\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} + \frac{1}{L_d}\begin{bmatrix} V_\alpha - \hat{e}_\alpha + z_\alpha \\ V_\beta - \hat{e}_\beta + z_\beta \end{bmatrix} \tag{14}$$

where

- z_α and z_β are sliding-mode feedback components and are defined as shown in Equation 15:

$$\begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} = \begin{bmatrix} k_\alpha \mathrm{sign}\left(\hat{i}_\alpha - i_\alpha\right) \\ k_\beta \mathrm{sign}\left(\hat{i}_\beta - i_\beta\right) \end{bmatrix} \tag{15}$$

where

- k_α and k_β are the constant sliding-mode gain designed by Lyapunov stability analysis

If k_α and k_β are positive and significant enough to provide the stable operation of the SMO, then k_α and k_β are large enough to hold $k_\alpha > \max(|e_\alpha|)$ and $k_\beta > \max\left(|e_\beta|\right)$.

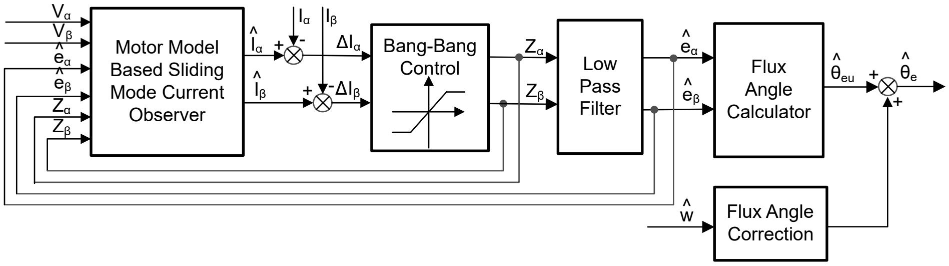


Figure 2-21. Block Diagram of Traditional Sliding-Mode Observer

The estimated value of EEMF in α-β axes ($\hat{e}_\alpha$, $\hat{e}_\beta$) can be obtained by low-pass filter from the discontinuous switching signals z_α and z_α :

$$\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = \frac{\omega_c}{s + \omega_c}\begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} \tag{16}$$

where

- $\omega_c = 2\pi f_c$ is the cutoff angular frequency of the LPF, which is usually selected according to the fundamental frequency of the stator current

Therefore, the rotor position can be directly calculated from arc-tangent the back EMF, as Equation 17 defines:

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) \tag{17}$$

Low-pass filters remove the high-frequency term of the sliding-mode function, which results in phase delay. The delay can be compensated by the relationship between the cut-off frequency ω_c and back EMF frequency ω_e , which is defined as shown in Equation 18:

$$\Delta\,\theta_e = -\tan^{-1}\left(\frac{\omega_e}{\omega_c}\right) \tag{18}$$

Then the estimated rotor position by using SMO method is found with Equation 19:

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) + \Delta\,\theta_e \tag{19}$$

In a digital control application, a time-discrete equation of the SMO is needed. The Euler method is the appropriate way to transform to a time-discrete observer. The time-discrete system matrix of Equation 14 in α-β coordinates is given by Equation 20 as:

$$\begin{bmatrix} \hat{i}_\alpha(n+1) \\ \hat{i}_\beta(n+1) \end{bmatrix} = \begin{bmatrix} F_\alpha \\ F_\beta \end{bmatrix} \begin{bmatrix} \hat{i}_\alpha(n) \\ \hat{i}_\beta(n) \end{bmatrix} + \begin{bmatrix} G_\alpha \\ G_\beta \end{bmatrix} \begin{bmatrix} V_\alpha^*(n) - \hat{e}_\alpha(n) + z_\alpha(n) \\ V_\beta^*(n) - \hat{e}_\beta(n) + z_\beta(n) \end{bmatrix} \tag{20}$$

where

- the matrix $[F]$ and $[G]$ are given by Equation 21 and Equation 22 as:

$$\begin{bmatrix} F_\alpha \\ F_\beta \end{bmatrix} = \begin{bmatrix} e^{-\frac{R_S}{L_d}} \\ e^{-\frac{R_S}{L_q}} \end{bmatrix} \tag{21}$$

$$\begin{bmatrix} G_\alpha \\ G_\beta \end{bmatrix} = \frac{1}{R_S} \begin{bmatrix} 1 - e^{-\frac{R_S}{L_d}} \\ 1 - e^{-\frac{R_S}{L_q}} \end{bmatrix} \tag{22}$$

The time-discrete form of Equation 16 is given by Equation 23 as:

$$\begin{bmatrix} \hat{e}_\alpha(n+1) \\ \hat{e}_\beta(n+1) \end{bmatrix} = \begin{bmatrix} \hat{e}_\alpha(n) \\ \hat{e}_\beta(n) \end{bmatrix} + 2\pi f_c \begin{bmatrix} z_\alpha(n) - \hat{e}_\alpha(n) \\ z_\beta(n) - \hat{e}_\beta(n) \end{bmatrix} \tag{23}$$

2.4.2.2.1.3 Rotor Position and Speed Estimation With PLL

With the arc tangent method, the accuracy of the position and velocity estimations are affected due to the existence of noise and harmonic components. To eliminate this issue, the PLL model can be used for velocity and position estimations in the sensorless control structure of the IPMSM. Section 2.4.2.2.1.2 illustrates the PLL structure used with SMO. The back-EMF estimations $\hat{e}_\alpha$ and $\hat{e}_\beta$ can be used with a PLL model to estimate the motor angular velocity and position as shown in Figure 2-22.

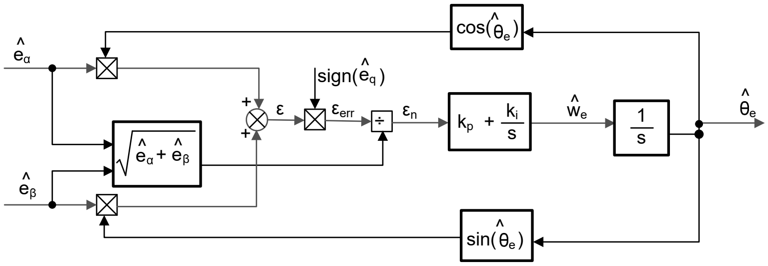


Figure 2-22. Block Diagram of Phase-Locked Loop Position Tracker

Since $e_\alpha = E\cos(\theta_e)$, $e_\beta = E\sin(\theta_e)$, and $E = \omega_e \lambda_{pm}$, the position error can be defined as Equation 24:

$$\varepsilon = \hat{e}_\beta \cos(\hat{\theta}_e) - \hat{e}_\alpha \sin(\hat{\theta}_e) = E\sin(\theta_e)\cos(\hat{\theta}_e) - E\cos(\theta_e)\sin(\hat{\theta}_e) = E\sin(\theta_e - \hat{\theta}_e) \tag{24}$$

where

- E is the magnitude of the EEMF, which is proportional to the motor speed ω_e

When $(\theta_e - \hat{\theta}_e) < \frac{\pi}{2}$, then Equation 24 can be simplified as Equation 25.

$$\varepsilon = E(\theta_e - \hat{\theta}_e) \tag{25}$$

Further, the position error after the normalization of the EEMF can be obtained (Equation 26):

$$\varepsilon_n = \theta_e - \hat{\theta}_e \tag{26}$$

According to the analysis, the simplified block diagram of the quadrature phase-locked loop position tracker can be obtained as shown in Figure 2-23. The closed-loop transfer functions of the PLL can be expressed as Equation 27:

$$\frac{\hat{\theta}_e}{\theta_e} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} = \frac{2\xi\omega_n s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{27}$$

where

- k_p and k_i are the proportional and the integral gains of the standard PI regulator

The natural frequency ω_n and the damping ratio ξ are given in Equation 28:

$$k_p = 2\xi\omega_n, \quad k_i = \omega_n^2 \tag{28}$$

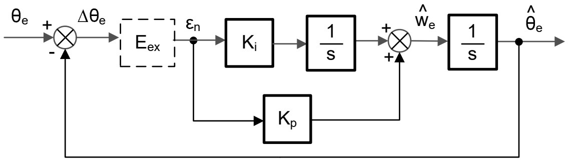


Figure 2-23. Simplified Block Diagram of Phase-Locked Loop Position Tracker

2.4.2.3 Field Weakening (FW) and Maximum Torque Per Ampere (MTPA) Control

Permanent magnet synchronous motor (PMSM) is widely used in home appliance applications due to the high power density, high efficiency, and wide speed range. The PMSM includes two major types: the surface-mounted PMSM (SPM), and the interior PMSM (IPM). SPM motors are easier to control due to the linear relationship between the torque and q-axis current. However, the IPMSM has electromagnetic and reluctance torques due to a large saliency ratio. The total torque is non-linear with respect to the rotor angle. As a result, the MTPA technique can be used for IPM motors to optimize torque generation in the constant torque region. The aim of the field-weakening control is to optimize to reach the highest power and efficiency of a PMSM drive. Field-weakening control can enable a motor operation over the base speed, expanding the operating limits to reach speeds higher than rated speed and allow exceptional control across the entire speed and voltage range.

The voltage equations of the mathematical model of an IPMSM can be described in d-q coordinates as shown in Equation 29 and Equation 30.

$$v_d = L_d \frac{di_d}{dt} + R_s i_d - p\omega_m L_q i_q \tag{29}$$

$$v_q = L_q \frac{di_q}{dt} + R_s i_q + p\omega_m L_d i_d + p\omega_m \psi_m \tag{30}$$

Figure 2-24 shows the dynamic equivalent circuit of an IPM synchronous motor.

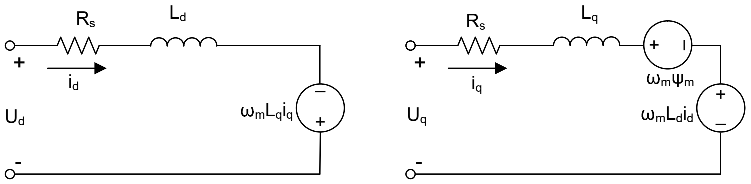


Figure 2-24. Equivalent Circuit of an IPM Synchronous Motor

The total electromagnetic torque generated by the IPMSM can be expressed as Equation 31 that the produced torque is composed of two distinct terms. The first term corresponds to the mutual reaction torque occurring between torque current i_q and the permanent magnet ψ_m , while the second term corresponds to the reluctance torque due to the differences in d-axis and q-axis inductance.

$$T_e = \frac{3}{2}p\left[\psi_m i_q + \left(L_d - L_q\right)i_d i_q\right] \quad (31)$$

In most applications, IPMSM drives have speed and torque constraints, mainly due to inverter or motor rating currents and available DC link voltage limitations respectively. These constraints can be expressed with the mathematical equations Equation 32 and Equation 33.

$$I_a = \sqrt{i_d^2 + i_q^2} \le I_{max} \quad (32)$$

$$V_a = \sqrt{v_d^2 + v_q^2} \le V_{max} \quad (33)$$

where

- V_{max} and I_{max} are the maximum allowable voltage and current of the inverter or motor

In a two-level three-phase Voltage Source Inverter (VSI) fed machine, the maximum achievable phase voltage is limited by the DC link voltage and the PWM strategy. The maximum voltage is limited to the value as shown in Equation 34 if Space Vector Modulation (SVPWM) is adopted.

$$\sqrt{v_d^2 + v_q^2} \le v_{max} = \frac{v_{dc}}{\sqrt{3}} \quad (34)$$

Usually the stator resistance R_s is negligible at high speed operation and the derivative of the currents is zero in steady state, thus Equation 35 is obtained as shown.

$$\sqrt{L_d^2\left(i_d + \frac{\psi_{pm}}{L_d}\right)^2 + L_q^2 i_q^2} \le \frac{V_{max}}{\omega_m} \quad (35)$$

The current limitation of Equation 32 produces a circle of radius I_{max} in the d-q plane, and the voltage limitation of Equation 34 produces an ellipse whose radius V_{max} decreases as speed increases. The resultant d-q plane current vector must be controlled to obey the current and voltage constraints simultaneously. According to these constraints, three operation regions for the IPMSM can be distinguished as shown in Figure 2-25.

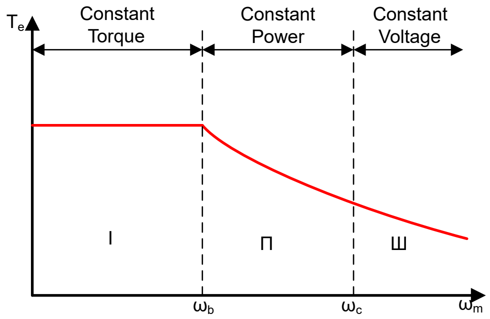


Figure 2-25. IPMSM Control Operation Regions

1. Constant Torque Region: MTPA can be implemented in this operation region to provide maximum torque generation.
2. Constant Power Region: Field-weakening control must be employed and the torque capacity is reduced as the current constraint is reached.
3. Constant Voltage Region: In this operation region, deep field-weakening control keeps a constant stator voltage to maximize the torque generation.

In the constant torque region, according to Equation 31, the total torque of an IPMSM includes the electromagnetic torque from the magnet flux linkage and the reluctance torque from the saliency between L_d and L_q . The electromagnetic torque is proportional to the q-axis current i_q , and the reluctance torque is proportional to the multiplication of the d-axis current i_d , the q-axis current i_q , and the difference between L_d and L_q .

Conventional vector control systems of SPM motors only utilizes electromagnetic torque by setting the commanded i_d to zero for non-field-weakening modes. But while the IPMSM utilizes the reluctance torque of the motor, the designer must also control the d-axis current. The aim of the MTPA control is to calculate the reference currents i_d and i_q to maximize the ratio between produced electromagnetic torque and reluctance torque. The relationship between i_d and i_q , and the vectorial sum of the stator current I_s is shown in the following equations.

$$I_s = \sqrt{i_d^2 + i_q^2} \tag{36}$$

$$I_d = I_s \cos\beta \tag{37}$$

$$I_q = I_s \sin\beta \tag{38}$$

where

- β is the stator current angle in the synchronous (d-q) reference frame

Equation 31 can be expressed as Equation 39 where I_s substituted for i_d and i_q .

Equation 39 shows that motor torque depends on the angle of the stator current vector:

$$T_e = \frac{3}{2} p I_s \sin\beta \left[\psi_m + \left(L_d - L_q\right) I_s \cos\beta \right] \tag{39}$$

This equation shows the maximum efficiency point can be calculated when the motor torque differential is equal to zero. The MTPA point can be found when this differential, $\frac{dT_e}{d\beta}$ is zero as given in Equation 40.

$$\frac{dT_e}{d\beta} = \frac{3}{2} p \left[\psi_m I_s \cos\beta + \left(L_d - L_q\right) I_s^2 \cos 2\beta \right] = 0 \tag{40}$$

Following this equation, the current angle of the MTPA control can be derived as in Equation 41.

$$\beta_{\mathrm{mtpa}} = \cos^{-1}\frac{-\psi_{\mathrm{m}} + \sqrt{\psi_{\mathrm{m}}^2 + 8 \times \left(\mathrm{L_d} - \mathrm{L_q}\right)^2 \times \mathrm{I_s^2}}}{4 \times \left(\mathrm{L_d} - \mathrm{L_q}\right) \times \mathrm{I_s}} \tag{41}$$

Thus, the effective d-axis and q-axis reference currents can be expressed by Equation 42 and Equation 43 using the current angle of the MTPA control.

$$\mathrm{I_d} = \mathrm{I_s} \times \cos\beta_{\mathrm{mtpa}} \tag{42}$$

$$\mathrm{I_q} = \mathrm{I_s} \times \sin\beta_{\mathrm{mtpa}} \tag{43}$$

However, as shown in Equation 41, the angle of the MTPA control, β_{mtpa} is related to d-axis and q-axis inductance. This means that the variation of inductance impedes the ability to find the exceptional MTPA point. To improve the efficiency of a motor drive, estimate the d-axis and q-axis inductance online, but the parameters $\mathrm{L_d}$ and $\mathrm{L_q}$ are not easily measured online and are influenced by saturation effects. A robust Look-Up Table (LUT) method provides controllability under electrical parameter variations. Usually, to simplify the mathematical model, the coupling effect between d-axis and q-axis inductance can be neglected. Thus, assume that $\mathrm{L_d}$ changes with $\mathrm{i_d}$ only, and $\mathrm{L_q}$ changes with $\mathrm{i_q}$ only. Consequently, d- and q-axis inductance can be modeled as a function of the d-q currents respectively, as shown in Equation 44 and Equation 45.

$$\mathrm{L_d} = f_1\left(\mathrm{i_d}, \mathrm{i_q}\right) = f_1\left(\mathrm{i_d}\right) \tag{44}$$

$$\mathrm{L_q} = f_2\left(\mathrm{i_q}, \mathrm{i_d}\right) = f_2\left(\mathrm{i_q}\right) \tag{45}$$

Reduce the ISR calculation burden by simplifying Equation 41. The motor-parameter-based constant, $\mathrm{K_{mtpa}}$ is expressed instead as Equation 47, where $\mathrm{K_{mtpa}}$ is computed in the background loop using the updated $\mathrm{L_d}$ and $\mathrm{L_q}$.

$$\mathrm{K_{mtpa}} = \frac{\psi_{\mathrm{m}}}{4 \times \left(\mathrm{L_q} - \mathrm{L_d}\right)} = 0.25 \times \frac{\psi_{\mathrm{m}}}{\left(\mathrm{L_q} - \mathrm{L_d}\right)} \tag{46}$$

$$\beta_{\mathrm{mtpa}} = \cos^{-1}\left(\mathrm{K_{mtpa}}/\mathrm{I_s} - \sqrt{\left(\mathrm{K_{mtpa}}/\mathrm{I_s}\right)^2 + 0.5}\right) \tag{47}$$

A second intermediate variable, $\mathrm{G_{mtpa}}$ described in Equation 48, is defined to further simplify the calculation. Using $\mathrm{G_{mtpa}}$, the angle of the MTPA control, β_{mtpa} can be calculated as Equation 49. These two calculations are performed in the ISR to achieve a real current angle β_{mtpa} .

$$\mathrm{G_{mtpa}} = \mathrm{K_{mtpa}}/\mathrm{I_s} \tag{48}$$

$$\beta_{\mathrm{mtpa}} = \cos^{-1}\left(\mathrm{G_{mtpa}} - \sqrt{\mathrm{G_{mtpa}^2} + 0.5}\right) \tag{49}$$

In all cases, the magnetic flux can be weakened to extend the achievable speed range by acting on the direct axis current $\mathrm{i_d}$. As a consequence of entering this constant power operating region, field-weakening control is chosen instead of the MTPA control used in constant power and voltage regions. Since the maximum inverter voltage is limited, PMSM motors cannot operate in such speed regions where the back-electromotive force, almost proportional to the permanent magnet field and motor speed, is higher than the maximum output voltage of the inverter. The direct control of magnet flux is not an option in PM motors. However, the air gap flux can be weakened by the demagnetizing effect due to the d-axis armature reaction by adding a negative $\mathrm{i_d}$. Considering the voltage and current constraints, the armature current and the terminal voltage are limited as Equation 32 and Equation 33. The inverter input voltage (DC-Link voltage) variation limits the maximum output of the motor. Furthermore, the maximum fundamental motor voltage also depends on the PWM method used. In Equation 35, the IPMSM has two factors: one is a permanent magnet value and the other is made by inductance and current of flux.

Figure 2-26 shows the typical control structure is used to implement field weakening. β_{fw} is the output of the field-weakening (FW) PI controller and generates the reference $\mathrm{i_d}$ and $\mathrm{i_q}$. Before the voltage magnitude reaches the limit, the input of the PI controller of FW is always positive and therefore the output is always saturated at 0.

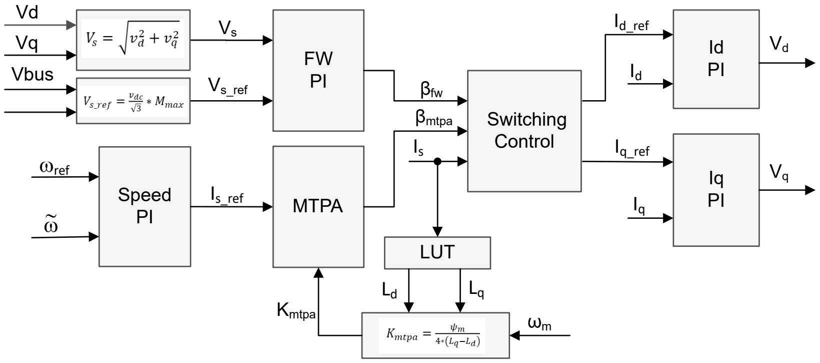


Figure 2-26. Block Diagram of Field-Weakening and Maximum Torque per Ampere Control

Figure 2-15 and Figure 2-17 show the implementation of FAST or eSMO based FOC block diagram. The block diagrams provide an overview of the functions and variables of the FOC system. There are two control modules in the motor drive FOC system: one is MTPA control and the other one is field-weakening control. These two modules generate current angle β_{mtpa} and β_{fw} , respectively, based on input parameters as shown in Figure 2-27.

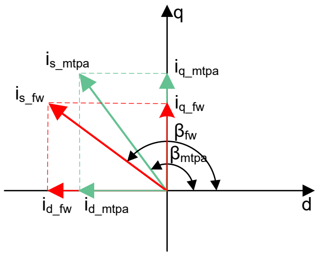


Figure 2-27. Current Phasor Diagram of an IPMSM During FW and MTPA

The switching control module is used to determine angle of application, and then calculate the reference $\mathrm{i_d}$ and $\mathrm{i_q}$ as shown in Equation 37 and Equation 38. The current angle is chosen as in the following: Equation 50 and Equation 51.

$$\beta = \beta_{\mathrm{fw}}\ \mathrm{if}\ \beta_{\mathrm{fw}} > \beta_{\mathrm{mtpa}} \tag{50}$$

$$\beta = \beta_{\mathrm{mpta}}\ \mathrm{if}\ \beta_{\mathrm{fw}} < \beta_{\mathrm{mtpa}} \tag{51}$$

The flow chart in Figure 2-28 shows the steps required to run InstaSPIN™-FOC with FW and MPTA in the main loop and interrupt.

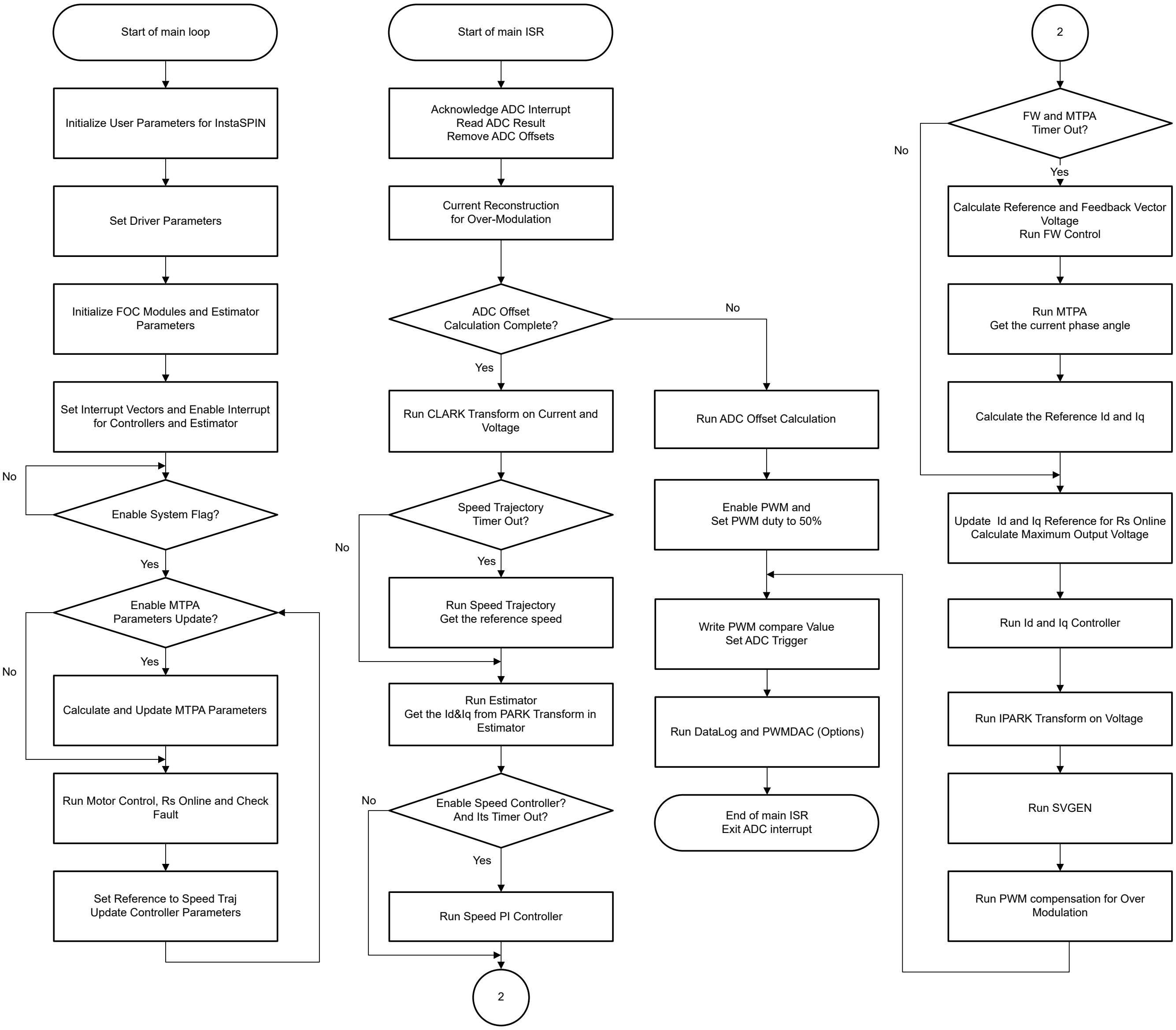

Start of main loop
Initialize User Parameters for InstaSPIN
Set Driver Parameters
Initialize FOC Modules and Estimator Parameters
Set Interrupt Vectors and Enable Interrupt for Controllers and Estimator
No
Enable System Flag?
Yes
Enable MTPA Parameters Update?
No
Yes
Calculate and Update MTPA Parameters
Run Motor Control, Rs Online and Check Fault
Set Reference to Speed Traj Update Controller Parameters
Start of main ISR
Acknowledge ADC Interrupt Read ADC Result Remove ADC Offsets
Current Reconstruction for Over-Modulation
ADC Offset Calculation Complete?
No
Yes
Run CLARK Transform on Current and Voltage
Speed Trajectory Timer Out?
No
Yes
Run Speed Trajectory Get the reference speed
Run Estimator Get the Id&Iq from PARK Transform in Estimator
No
Enable Speed Controller? And Its Timer Out?
Yes
Run Speed PI Controller
2
Run ADC Offset Calculation
Enable PWM and Set PWM duty to 50%
Write PWM compare Value Set ADC Trigger
Run DataLog and PWMDAC (Options)
End of main ISR Exit ADC interrupt
2
FW and MTPA Timer Out?
No
Yes
Calculate Reference and Feedback Vector Voltage Run FW Control
Run MTPA Get the current phase angle
Calculate the Reference Id and Iq
Update Id and Iq Reference for Rs Online Calculate Maximum Output Voltage
Run Id and Iq Controller
Run IPARK Transform on Voltage
Run SVGEN
Run PWM compensation for Over Modulation

Figure 2-28. Flow Chart for an InstaSPIN-FOC Project With FW and MTPA

2.4.2.4 Hardware Prerequisites for Motor Drive

The algorithm for controlling the motor makes use of sampled measurements of the motor conditions, including dc bus power supply voltage, the voltage on each motor phase, the current of each motor phase. There are a few hardware-dependent parameters that need to be set correctly to identify the motor properly and run the motor effectively using Field Oriented Control (FOC). The following sections show how to calculate the current scale value, voltage scale value, and voltage filter pole for motor control with FAST or eSMO.

2.4.2.4.1 Motor Current Feedback

Two techniques are supported to measure phase currents of the motor.

- Three-shunt current sensing
- Single-shunt current sensing

Either one of these two current sensing techniques can be selected in the build configuration of the project. The *Flash_MtrInv_3SC* build configuration supports three-shunt current sensing method, the *Flash_MtrInv_1SC* supports single-shunt current sensing method as described in Section 3.3.2.

2.4.2.4.1.1 Three-Shunt Current Sensing

The current through the motor is sampled by microcontroller as part of the motor control algorithm during every one PWM cycle. TMS320F2800137 daughter board supports 1-3 shunt current sensing, MSPM0G1507 daughterboard supports 1-2 shunt current sensing. To measure bidirectional currents of the motor phase, that is, positive and negative currents, the circuits below require a reference voltage of 1.65 V. This offset reference voltage is created by a voltage follower as shown in Figure 2-29.

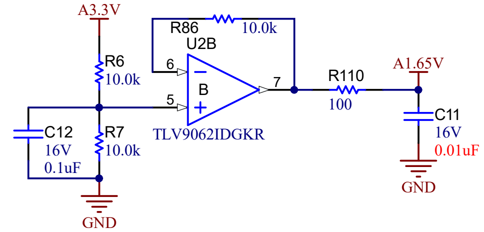


Figure 2-29. 1.65-V Reference From 3.3-V Input Circuit

Figure 2-30 shows how the motor current is represented as a voltage signal, with filtering, amplification, and offset to the center of the ADC input range for TMS320F2800137 daughterboard. This circuit is used for each phase of the 3-phase PMSM of compressor and fan. The transfer function of this circuit is given by Equation 52.

$$V_{OUT} = V_{OFFSET} + (I_{IN} \times R_{SHUNT} \times G_i) \tag{52}$$

where

- $R_{shunt} = 0.05\ \Omega$
- $V_{offset} = 1.65$ V

The calculated resistance values lead to the sensing circuit shown in Figure 2-36, G_i is given by Equation 53.

$$G_i = \frac{R_{fb}}{R_{in}} = \frac{R18}{(R97 + R15)} = \frac{10\ k\Omega}{20 + 2.4\ k\Omega} = 4.132 \tag{53}$$

The maximum peak-to-peak current measurable by the microcontroller is given by Equation 54.

$$I_{scale\_max} = \frac{V_{ADC\_max}}{R_{SHUNT} \times G_i} = \frac{3.3}{0.05 \times 4.132} = 15.97\ A \tag{54}$$

This has the peak-to-peak value of ±7.99 A. The following code snippet shows how this is defined for compressor motor in user_mtr1.h file:

```
//! \brief Defines the maximum current at the AD converter
#define USER_M1_ADC_FULL_SCALE_CURRENT_A         (15.97f)
```

Correct polarity of the current feedback is also important so that the microcontroller has an accurate current measurement. In this hardware board configuration, the negative pin of the shunt resistor, which is connected to ground, is also connected to the inverting pin of the operational amplifier. The highlighted sign is required to be configured to have the correct polarity for the current feedback in software as shown in the following code snippet in user.mtr1.h:

```
// define the sign of current feedback based on hardware board
#define USER_M1_SIGN_CURRENT_SF     (1.0f)
```

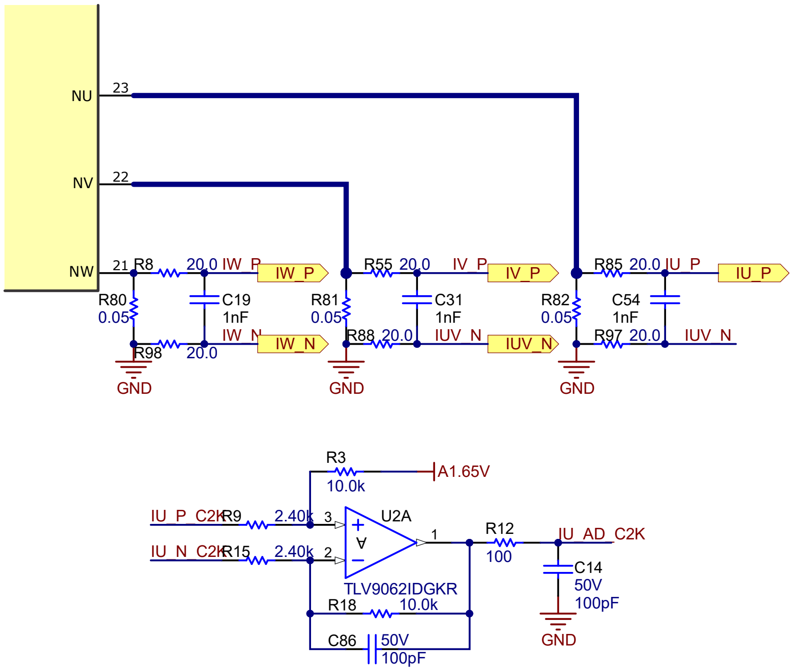


Figure 2-30. Three-Shunt Current Sensing Circuit for TMS320F2800137

On the MSPM0 daughterboard, two shunt current sensing are implemented with the two high-end internal amplifiers to save system cost. The amplifier gain is also 4.132, and the cutoff frequency is 70 kHz. Figure 2-31 shows the two-shunt current sensing circuit for the MSPM0G1507 daughterboard.

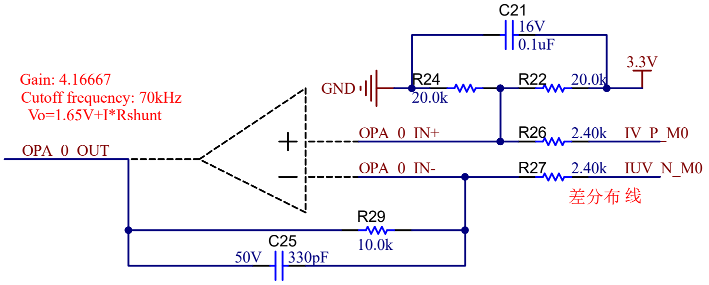


Figure 2-31. MSPM0G1507 Two-Shunt Current-Sensing Circuit

2.4.2.4.1.2 Single-Shunt Current Sensing

The single-shunt current-sensing technique measures the DC-link bus current, with knowledge of the power FET switching states and reconstructs the three-phase current of the motor. The detailed description of the single shunt technique is described in the *Sensorless-FOC for PMSM With Single DC-Link Shunt* application note.

On this reference board, implement the single-shunt current-sensing technique by removing two shunts and shorting the connection of the U, V, W ground of the power module as shown in Figure 2-32.

1. On the motherboard, remove current shunt resistors R81, and R82, just keep only shunt resistor R80 to sense the DC-Link current.
2. On TMS320F2800137 daughterboard, remove C86 to increase U2A bandwidth for single-shunt sampling.
3. On the MSPM0G1507 daughterboard, remove C29 to increase the bandwidth for single-shunt sampling.
4. Use a thick wire to connect the NU, NV, and NW pins together.

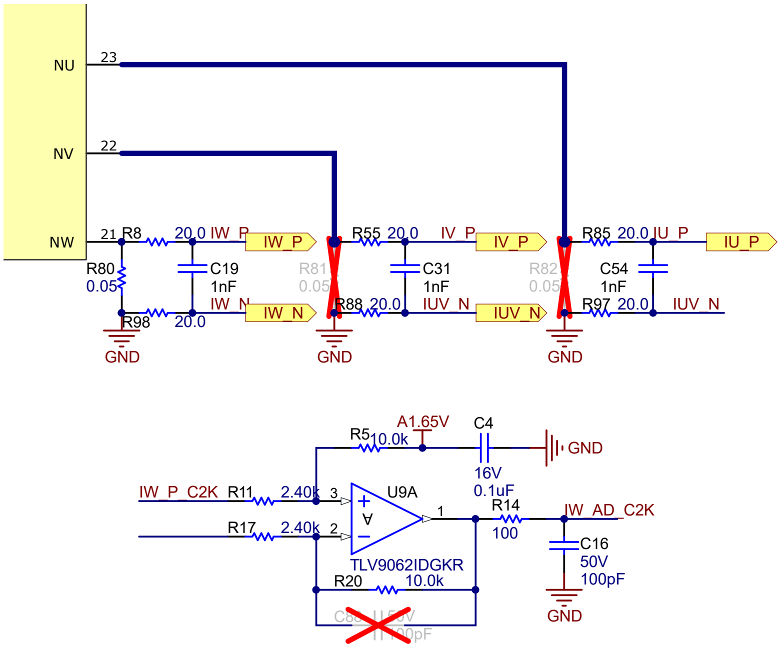


Figure 2-32. Single-Shunt Current-Sensing Circuit for TMS320F2800137

Figure 2-33 shows the single-shunt current-sensing circuit for the MSPM0G1507 daughterboard.

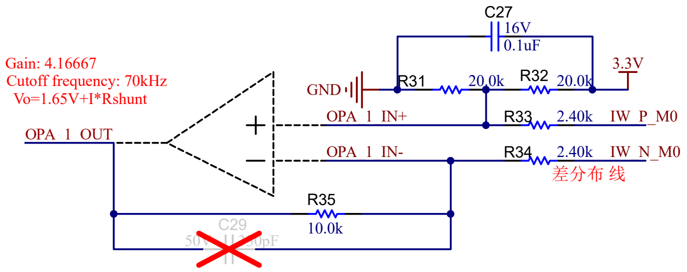


Figure 2-33. MSPM0G1507 Single-Shunt Current-Sensing Circuit

By default, the board has three shunt resistors, Figure 2-34 shows the layout of the shunt resistors. To run with a single-shunt resistor, remove R81 and R82 while keeping R80, solder NU, NV and NW (pin 2 of R80, R81, and R82) together, then all three phase currents flow through only R80.

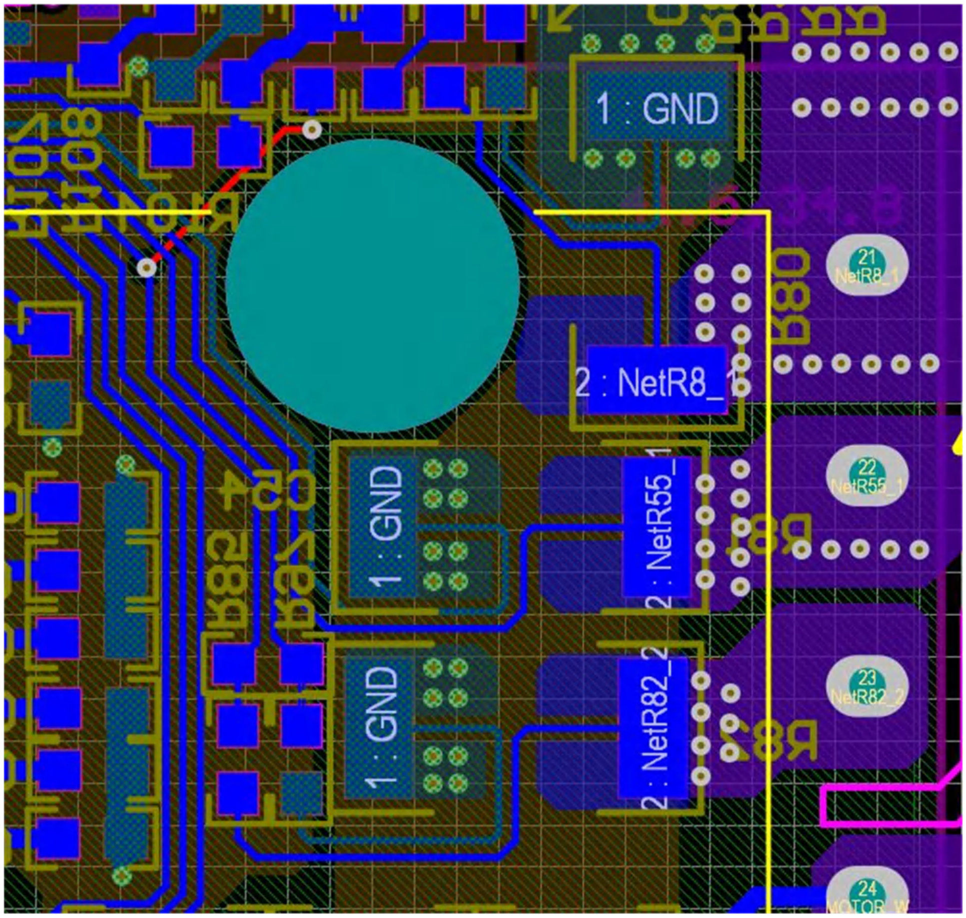


Figure 2-34. Shunt Resistors Layout

The DC-Link current is a unidirectional signal, so the DC current offset can be set to a minimum or maximum value to improve the ADC sampling range for the DC-Link current as Figure 2-35 shows. On the TMS320F2800137 daughterboard, change R7 from 10 kΩ to 1 kΩ/1% resistor for the reference voltage to have 0.3-V offset for DC current sensing.

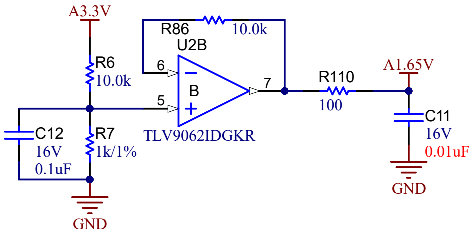


Figure 2-35. DC Offset Reference for Single Shunt of TMS320F2800137 Daughterboard

The transfer function of this current sampling circuit and the calculation for single shunt are the same as the three shunts.

For the MSPM0 daughterboard, offset for single-shunt current sensing can also be reduced to 0.3 V by reducing R31 from 20 kΩ to 2 kΩ, as shown in Figure 2-33.

2.4.2.4.2 Motor Voltage Feedback

Voltage feedback is needed in the FAST estimator to allow the best performance at the widest speed range, the phase voltages are measured directly from the motor phases instead of a software estimate. The eSMO relies on software estimation values to represent the voltage phases without using the motor phase voltage-sensing circuit. This software value (USER_ADC_FULL_SCALE_VOLTAGE_V) depends on the circuit that senses the voltage feedback from the motor phases. Figure 2-36 shows how the motor voltage is filtered and scaled for the ADC input range using a voltage feedback circuit based on resistor dividers. The similar circuit is used to measure all three of both compressor and fan motors, and dc bus.

The maximum phase voltage feedback measurable by the microcontroller in this reference design can be calculated as given in Equation 55, considering the maximum voltage for the ADC input is 3.3 V.

$$V_{FS} = V_{ADC\_FS} \times G_V = 3.3\ \text{V} \times 122.46 = 404.13\ \text{V} \tag{55}$$

where

- G_V is attenuation factor, G_V is calculated with Equation 56

$$G_V = \frac{(R62 + R67 + R70 + R74)}{R74} = \frac{(332\ \text{k}\Omega + 332\ \text{k}\Omega + 332\ \text{k}\Omega + 8.2\ \text{k}\Omega)}{8.2\ \text{k}\Omega} = 122.46 \tag{56}$$

With that voltage feedback circuit, the following setting is done in user_mtr1.h:

```
//! \brief Defines the maximum voltage at the AD converter
#define USER_M1_ADC_FULL_SCALE_VOLTAGE_V          (404.1292683f)
```

The voltage filter pole is needed by the FAST estimator to allow an accurate detection of the voltage feedback. Make the filter low enough to filter out the PWM signals, and at the same time allow a high-speed voltage feedback signal to pass through the filter. As a general guideline, a cutoff frequency of a few hundred Hz is enough to filter out a PWM frequency of 5 to 20 kHz. Change the hardware filter only when ultra-high-speed motors are run, which generate phase-voltage frequencies in the order of a few kHz.

In this reference design the filter pole setting can be calculated with Equation 57:

$$f_{\text{filter\_pole}} = \frac{1}{(2 \times \pi \times R_{\text{Parallel}} \times C)} = 405.15\ \text{Hz} \tag{57}$$

where,
$C = 47\text{nF}$

$$R_{\text{Parallel}} = \left(\frac{(332\ \text{k}\Omega + 332\ \text{k}\Omega + 332\ \text{k}\Omega) \times 8.2\ \text{k}\Omega}{(332\ \text{k}\Omega + 332\ \text{k}\Omega + 332\ \text{k}\Omega) + 8.2\ \text{k}\Omega}\right) = 8.133\ \text{k}\Omega$$

The following code example shows how this is defined in user_mtr1.h:

```
//! \brief Defines the analog voltage filter pole location, Hz
#define USER_M1_VOLTAGE_FILTER_POLE_Hz            (416.3602877f)
```

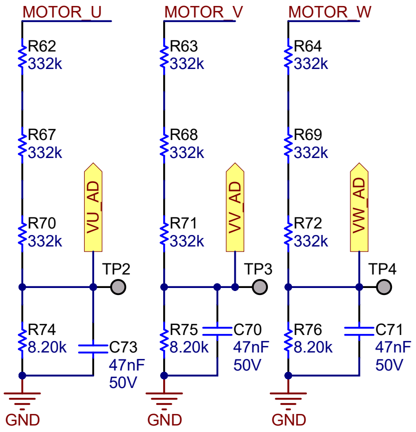


Figure 2-36. Motor Voltage Sensing Circuit

3 Hardware, Software, Testing Requirements, and Test Results

3.1 Getting Started Hardware

This section details the necessary equipment, test setup, and procedure instructions for the design board and software testing and validation.

3.1.1 Hardware Board Overview

Figure 3-1 shows an overview of a typical motor inverter system.

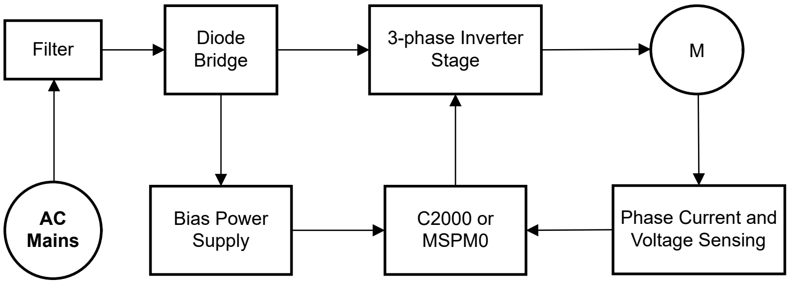


Figure 3-1. Hardware Board Block Diagram of TIDA-010265

The motor control board has functional groups that enable a complete motor drive system. The following is a list of the blocks (and functions) on the board, Figure 3-2 shows the top view of the board and different blocks of the TIDA-010265 PCB.

- Power line input filter
- 3-phase inverter
  - Up to 750-W 3-phase inverter supports PMSM or IPM
  - 15-kHz switching frequency
  - 1-3 shunts current sensing
- Control
  - Single TMS320F2800137 or MSPM0G1507 series MCU in 48-pin LQFP package
  - Amplify and input filters for the analog signals
- Interface for external motor temperature sensing
- Isolated UART port
- Auxiliary power supply
  - Onboard power supply +3.3 V, +5 V and +15 V

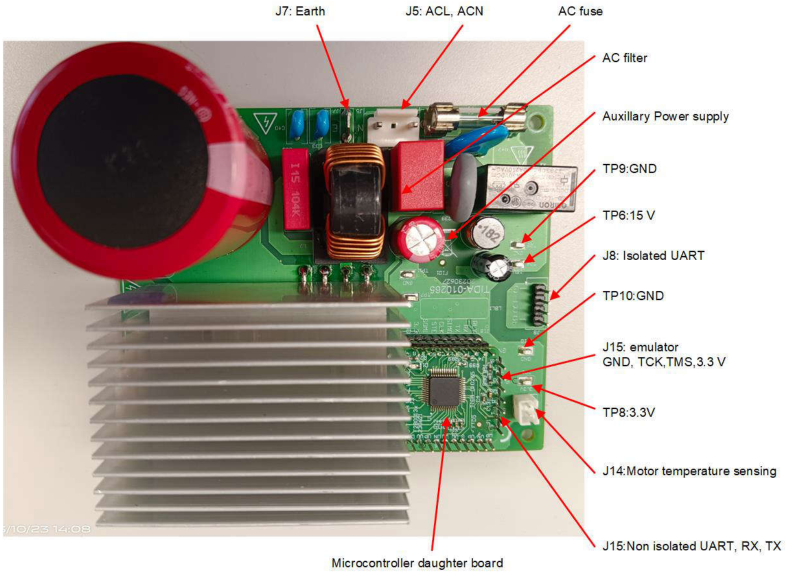


Figure 3-2. TIDA-010265 Reference Design Board Layout

TI recommends taking the following precautions when using the board:

WARNING

- Do not touch any part of the board or components connected to the board when the board is energized.
- Use the AC Mains or wall power supply to power the kit. TI recommends an isolation AC source.
- Do not touch any part of the board, the kit or the assembly when energized. (Though the power module heat sink is isolated from the board, high-voltage switching generates some capacitive coupled voltages over the heat sink body.)
- Control Ground can be hot.

3.1.2 Test Conditions

Observe the following for testing the reference design software:

- For input, the power supply source must range from 165-V to 265-V AC if using the AC source, or must range from 100 V to 400 V if using the DC power supply. Set the input current limit of input AC source to 10 A or DC power supply to 6.5 A, but start with a lower current limit during initial board bring up.
- For the output, use a 3-phase PMSM with dynamo meters.

3.1.3 Test Equipment Required for Board Validation

The designer must use the following equipment for board validation:

- Isolated AC source
- Single-phase power analyzer
- Digital oscilloscope
- Multimeters
- DC power supply
- 750-W, 3-phase PM synchronous motors
- Dynamo meters
- Three-phase power analyzer

3.2 Getting Started GUI

Source code for this reference design is provided so the designer can debug firmware directly, as explained in Section 3.3. However, software debugging needs more time. To speed up development time, a UART based GUI software is provided to help quickly tune parameters on any customized application. This section introduces how to debug and tune the motor control parameters with the GUI software.

For now, only the C2000 daughter-board firmware supports the GUI.

Use caution when connecting the host PC to this reference board via UART, because the AC rectifier generates the DC-output voltage, which has a **HOT Ground** floating from the protective earth ground. Isolation transformers must be used when connecting grounded equipment to the Kit.

3.2.1 Test Setup

The GUI software needs only a UART connection between the host PC and the reference design board. Figure 3-3 shows the hardware connection for testing with the GUI. Setup the hardware using the following steps:

1. Connect only TX, RX, and GND on J15 to the host PC through a UART-to-USB adapter. 3.3 V is not needed from the adapter.
2. Connect motor wires to J10.
3. Connect the multimeter, oscilloscope probes, and other measurement equipment to probe or analyze various signals and parameters
4. Power on the board with a DC bus power, AC power supply or AC mains power to the inverter at J5 and J7.
    a. The maximum output of the DC power supply is 380 VDC.
    b. The maximum output of the AC power supply is 265 VAC, 50/60 Hz.
    c. AC main power is 220 VAC, 50/60 Hz.

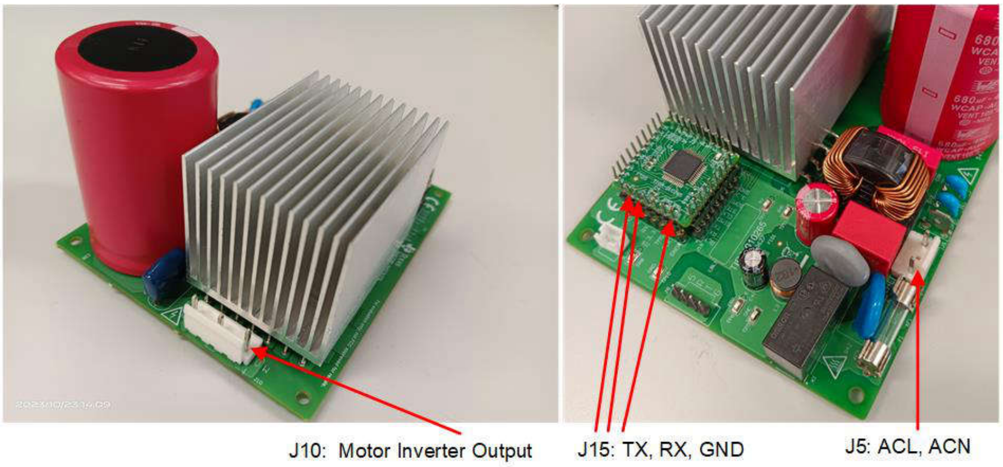


Figure 3-3. Hardware Connection for Testing With GUI Software

Note

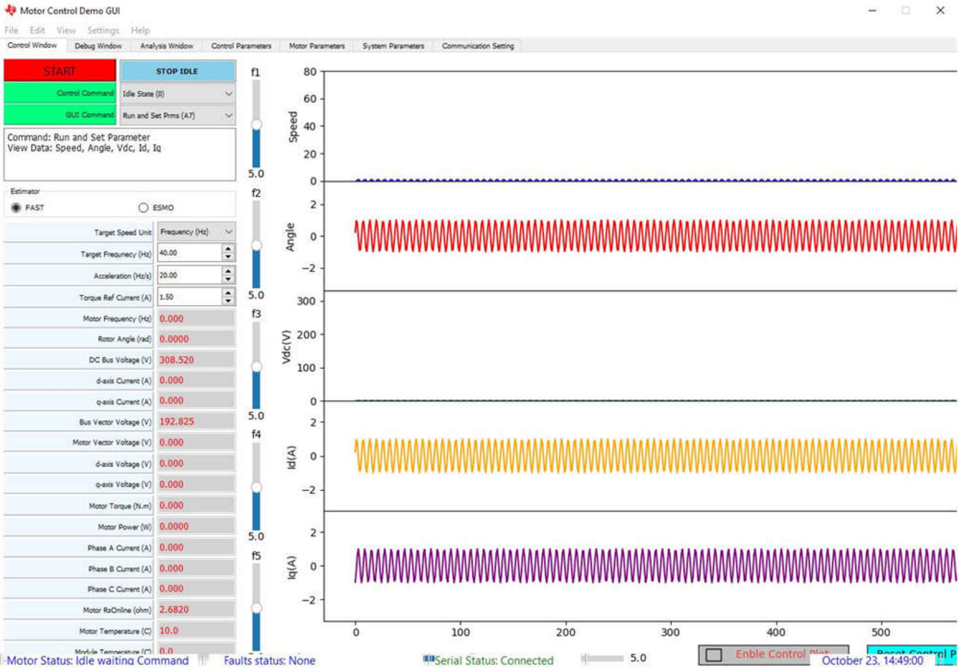
Add ferrite beads on JTAG signals and the USB cable if the external emulator has connectivity issues while testing. Make the connection lines as short as possible.

WARNING

The ground planes of both the power domains can be the same or different depending on the hardware configuration. Meet proper isolation requirements before connecting any test equipment with the board for personal safety and to prevent damaging equipment. Review the GND connections before powering the board. An isolator is required if measurement equipment is connected to the board.

3.2.2 Overview of GUI Software

The GUI software can be run on a Microsoft® Windows® based system. The GUI has 6 tabs: Control Window, Debug Windows, Control Parameters, Motor Parameters, System Parameters, and Communication Setting, as shown in Figure 3-4. The Analysis Windows is not available in this version GUI. Those tabs provides multiple functions, such as motor control, identification, control parameters tuning, virtual oscilloscope, read and write MCU flash, and so forth.



Motor Control Demo GUI

Command: Run and Set Parameter
View Data: Speed, Angle, Vdc, Id, Iq

Control Command: Idle State (0)

GUI Command: Run and Set Prms (A7)

Estimator: ● FAST ○ ESMO

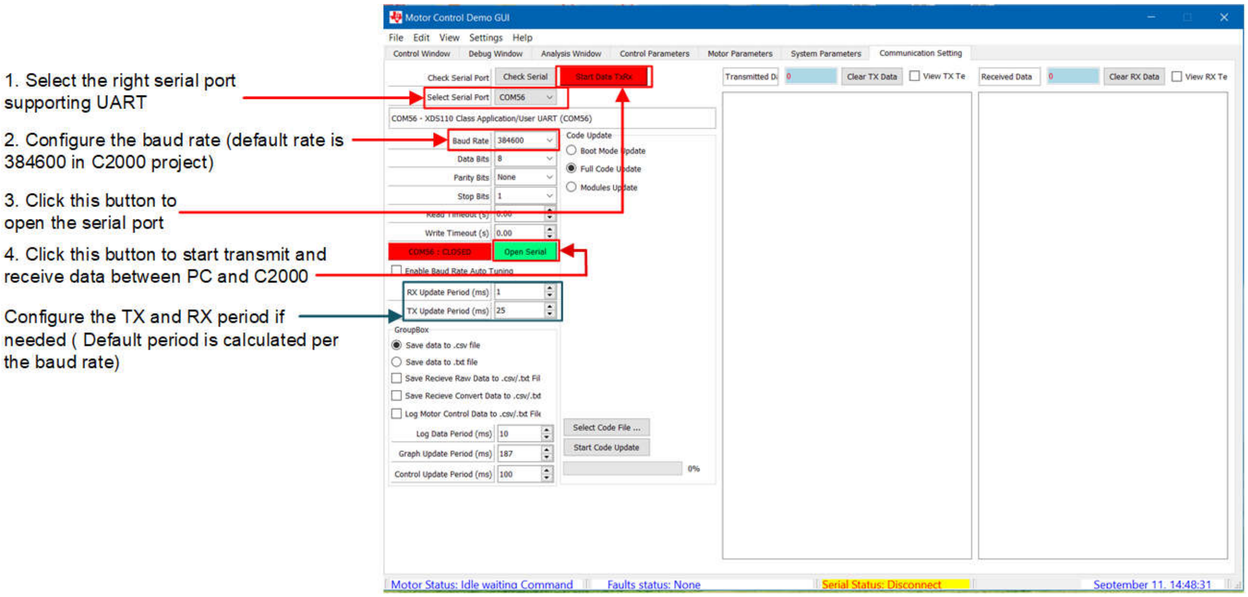
Target Speed Unit	Frequency (Hz)
Target Frequency (Hz)	40.00
Acceleration (Hz/s)	20.00
Torque Ref Current (A)	1.50
Motor Frequency (Hz)	0.000
Rotor Angle (rad)	0.0000
DC Bus Voltage (V)	308.520
d-axis Current (A)	0.000
q-axis Current (A)	0.000
Bus Vector Voltage (V)	192.825
Motor Vector Voltage (V)	0.000
d-axis Voltage (V)	0.000
q-axis Voltage (V)	0.000
Motor Torque (N.m)	0.000
Motor Power (W)	0.0000
Phase A Current (A)	0.000
Phase B Current (A)	0.000
Phase C Current (A)	0.000
Motor RsOnline (ohm)	2.6820
Motor Temperature (C)	10.0
Module Temperature (C)	0.0

Motor Status: Idle waiting Command | Faults status: None | Serial Status: Connected | October 23, 14:49:00

Figure 3-4. GUI Software

3.2.3 Setup Serial Port

Run the GUI software on the host PC, wait for GUI window to pop up. Figure 3-5 shows the steps to connect the host PC to this reference design board via UART. The default baud rate is 258600bps. The user must change the baud rate setting on both GUI and C2000 if they want to use the different UART speed.



1. Select the right serial port supporting UART
2. Configure the baud rate (default rate is 384600 in C2000 project)
3. Click this button to open the serial port
4. Click this button to start transmit and receive data between PC and C2000

Configure the TX and RX period if needed (Default period is calculated per the baud rate)

Motor Control Demo GUI

Select Serial Port: COM56

COM56 - XDS110 Class Application/User UART (COM56)

Baud Rate	384600
Data Bits	8
Parity Bits	None
Stop Bits	1
Read Timeout (s)	0.00
Write Timeout (s)	0.00
RX Update Period (ms)	1
TX Update Period (ms)	25
Log Data Period (ms)	10
Graph Update Period (ms)	187
Control Update Period (ms)	100

COM56 : CLOSED

Code Update: ○ Boot Mode Update ● Full Code Update ○ Modules Update

GroupBox: ● Save data to .csv file ○ Save data to .txt file

□ Enable Baud Rate Auto Tuning

□ Save Recieve Raw Data to .csv/.txt Fil

□ Save Recieve Convert Data to .csv/.txt

□ Log Motor Control Data to .csv/.txt File

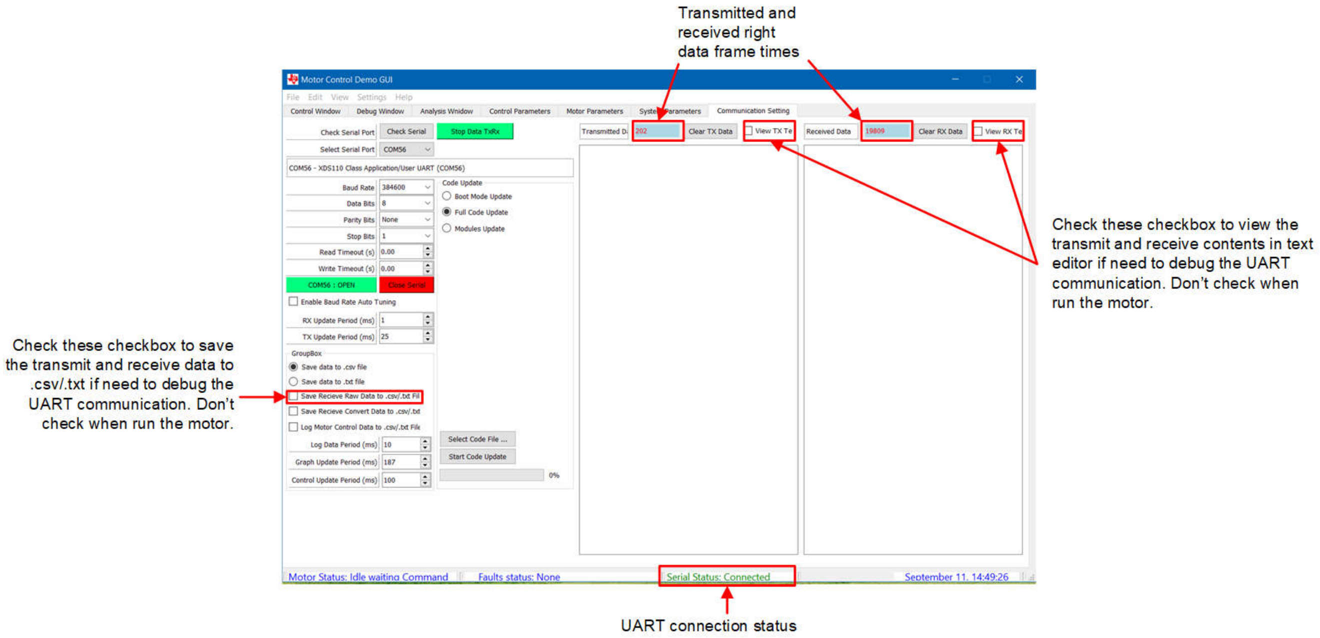
Transmitted Data: 0

Received Data: 0

Motor Status: Idle waiting Command | Faults status: None | Serial Status: Disconnect | September 11, 14:48:31

Figure 3-5. Setup Serial Port

A successful connection and communication can be verified by the number of *Transmitted Data* and *Received Data*. Figure 3-6 shows that both numbers increase continuously with a successful connection. Those transmitted and received data can be viewed by checking *View Tx Text* and *View Rx Text*; however, do not check these features to view when the motor is running.



Transmitted and received right data frame times

Check these checkbox to view the transmit and receive contents in text editor if need to debug the UART communication. Don't check when run the motor.

Check these checkbox to save the transmit and receive data to .csv/.txt if need to debug the UART communication. Don't check when run the motor.

UART connection status

Motor Control Demo GUI

Check Serial Port: Check Serial | Stop Data TxRx
Select Serial Port: COM56
COM56 - XDS110 Class Application/User UART (COM56)
Baud Rate: 384600
Data Bits: 8
Parity Bits: None
Stop Bits: 1
Read Timeout (s): 0.00
Write Timeout (s): 0.00
COM56 : OPEN | Close Serial
☐ Enable Baud Rate Auto Tuning
RX Update Period (ms): 1
TX Update Period (ms): 25
GroupBox
● Save data to .csv file
○ Save data to .txt file
☐ Save Recieve Raw Data to .csv/.txt Fil
☐ Save Recieve Convert Data to .csv/.txt
☐ Log Motor Control Data to .csv/.txt File
Log Data Period (ms): 10
Graph Update Period (ms): 187
Control Update Period (ms): 100
Code Update
○ Boot Mode Update
● Full Code Update
○ Modules Update
Select Code File ...
Start Code Update
0%
Transmitted Da: 202 | Clear TX Data | ☐ View TX Te
Received Data: 19809 | Clear RX Data | ☐ View RX Te
Motor Status: Idle waiting Command | Faults status: None | Serial Status: Connected | September 11, 14:49:26

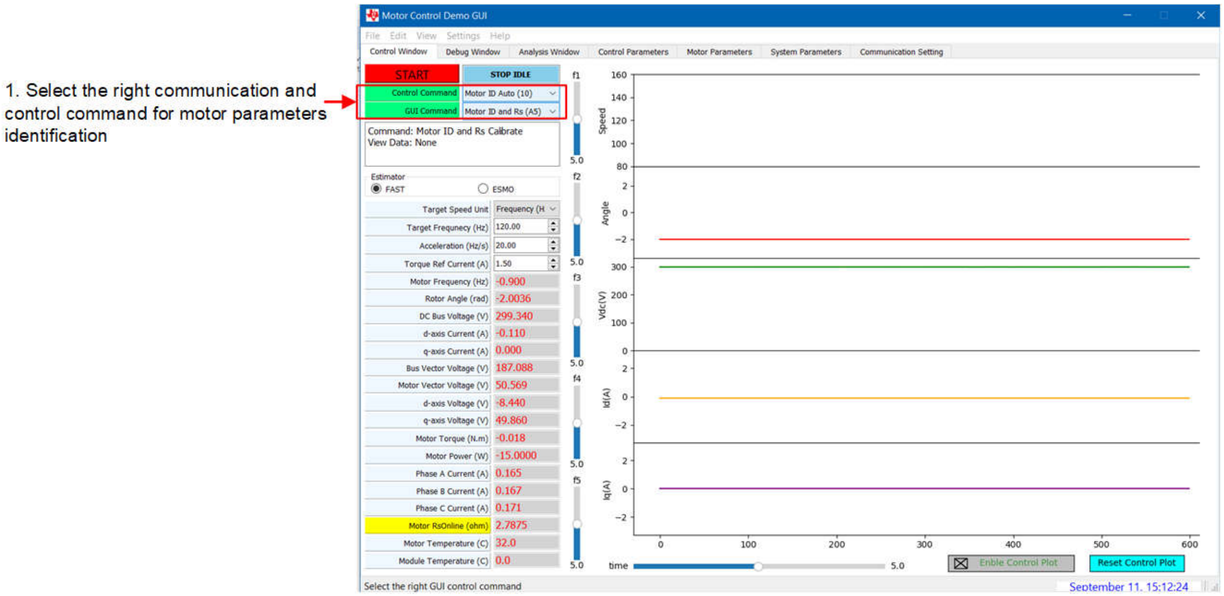
Figure 3-6. Successful Communication Status

3.2.4 Motor Identification

Implementing the correct motor parameters is critical for the firmware to control the motor successfully. The parameters include Stator resistance, Stator inductance, flux, and so forth, and those parameters have a default value for a default motor inside the firmware.

For a different PMSM motor, those parameters are usually found from the specification; however, if those parameters cannot be found, the GUI software can identify those parameters.

First, choose the motor identification command in the *Control Window* tab as shown in Figure 3-7.



1. Select the right communication and control command for motor parameters identification

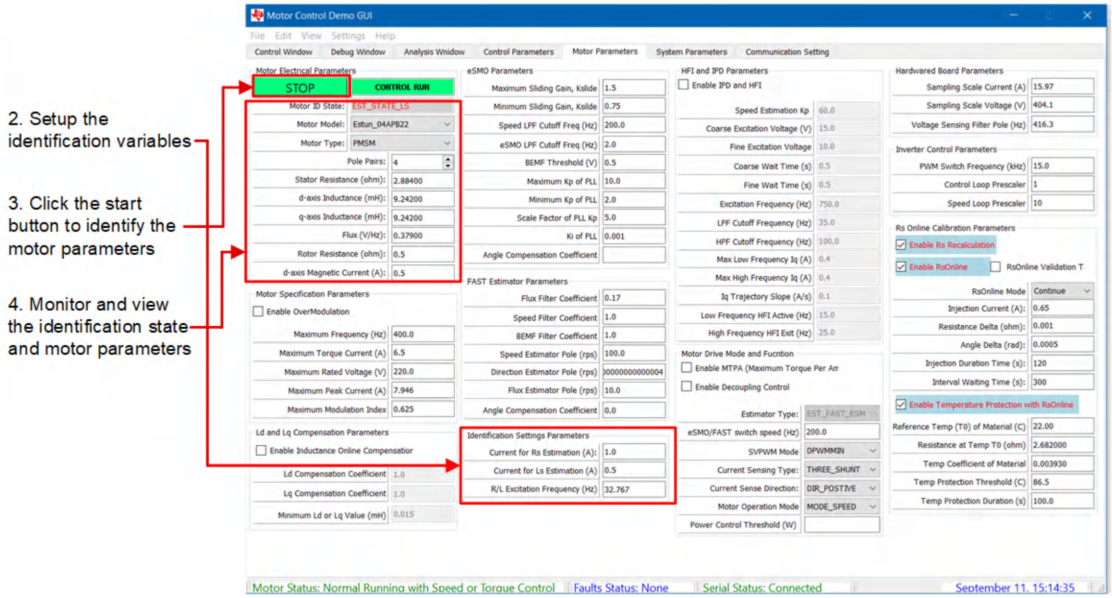
Motor Control Demo GUI

START | STOP IDLE
Control Command: Motor ID Auto (10)
GUI Command: Motor ID and Rs (A5)
Command: Motor ID and Rs Calibrate
View Data: None
Estimator: ● FAST ○ ESMO
Target Speed Unit: Frequency (H
Target Frequnecy (Hz): 120.00
Acceleration (Hz/s): 20.00
Torque Ref Current (A): 1.50
Motor Frequency (Hz): -0.900
Rotor Angle (rad): -2.0036
DC Bus Voltage (V): 299.340
d-axis Current (A): -0.110
q-axis Current (A): 0.000
Bus Vector Voltage (V): 187.088
Motor Vector Voltage (V): 50.569
d-axis Voltage (V): -8.440
q-axis Voltage (V): 49.860
Motor Torque (N.m): -0.018
Motor Power (W): -15.0000
Phase A Current (A): 0.165
Phase B Current (A): 0.167
Phase C Current (A): 0.171
Motor RsOnline (ohm): 2.7875
Motor Temperature (C): 32.0
Module Temperature (C): 0.0
Select the right GUI control command
September 11, 15:12:24

Figure 3-7. Motor Identification Command

Next, select the *Motor Parameters* tab. The motor identification applies current to the motor to estimate motor parameters, those identification parameters, such as current for stator resistor estimation, current for stator inductor estimation and *R/L Excitation Frequency (Hz)* can be changed or left on the default settings.

Click the *START* button to start motor identification. An audible noise is heard, and the motor spins at low speed during identification. Monitor the identification status and motor parameters, the overall identification time is about 2 minutes. Figure 3-8 shows the steps to start motor identification.



2. Setup the identification variables
3. Click the start button to identify the motor parameters
4. Monitor and view the identification state and motor parameters

Motor Electrical Parameters
Motor ID State: EST_STATE_LS
Motor Model: Estun_04APB22
Motor Type: PMSM
Pole Pairs: 4
Stator Resistance (ohm): 2.88400
d-axis Inductance (mH): 9.24200
q-axis Inductance (mH): 9.24200
Flux (V/Hz): 0.37900
Rotor Resistance (ohm): 0.5
d-axis Magnetic Current (A): 0.5

Motor Specification Parameters
☐ Enable OverModulation
Maximum Frequency (Hz) 400.0
Maximum Torque Current (A) 6.5
Maximum Rated Voltage (V) 220.0
Maximum Peak Current (A) 7.946
Maximum Modulation Index 0.625

Ld and Lq Compensation Parameters
☐ Enable Inductance Online Compensatior
Ld Compensation Coefficient 1.0
Lq Compensation Coefficient 1.0
Minimum Ld or Lq Value (mH) 0.015

eSMO Parameters
Maximum Sliding Gain, Kslide 1.5
Minimum Sliding Gain, Kslide 0.75
Speed LPF Cutoff Freq (Hz) 200.0
eSMO LPF Cutoff Freq (Hz) 2.0
BEMF Threshold (V) 0.5
Maximum Kp of PLL 10.0
Minimum Kp of PLL 2.0
Scale Factor of PLL Kp 5.0
Ki of PLL 0.001
Angle Compensation Coefficient

FAST Estimator Parameters
Flux Filter Coefficient 0.17
Speed Filter Coefficient 1.0
BEMF Filter Coefficient 1.0
Speed Estimator Pole (rps) 100.0
Direction Estimator Pole (rps))0000000000004
Flux Estimator Pole (rps) 10.0
Angle Compensation Coefficient 0.0

Identification Settings Parameters
Current for Rs Estimation (A): 1.0
Current for Ls Estimation (A) 0.5
R/L Excitation Frequency (Hz) 32.767

HFI and IPD Parameters
☐ Enable IPD and HFI
Speed Estimation Kp 60.0
Coarse Excitation Voltage (V) 15.0
Fine Excitation Voltage 10.0
Coarse Wait Time (s) 0.5
Fine Wait Time (s) 0.5
Excitation Frequency (Hz) 750.0
LPF Cutoff Frequency (Hz) 35.0
HPF Cutoff Frequency (Hz) 100.0
Max Low Frequency Iq (A) 0.4
Max High Frequency Iq (A) 0.4
Iq Trajectory Slope (A/s) 0.1
Low Frequency HFI Active (Hz) 15.0
High Frequency HFI Exit (Hz) 25.0

Motor Drive Mode and Fucntion
☐ Enable MTPA (Maximum Torque Per Am
☐ Enable Decoupling Control
Estimator Type: EST_FAST_ESM
eSMO/FAST switch speed (Hz) 200.0
SVPWM Mode DPWMMIN
Current Sensing Type: THREE_SHUNT
Current Sense Direction: DIR_POSTIVE
Motor Operation Mode MODE_SPEED
Power Control Threshold (W)

Hardwared Board Parameters
Sampling Scale Current (A) 15.97
Sampling Scale Voltage (V) 404.1
Voltage Sensing Filter Pole (Hz) 416.3

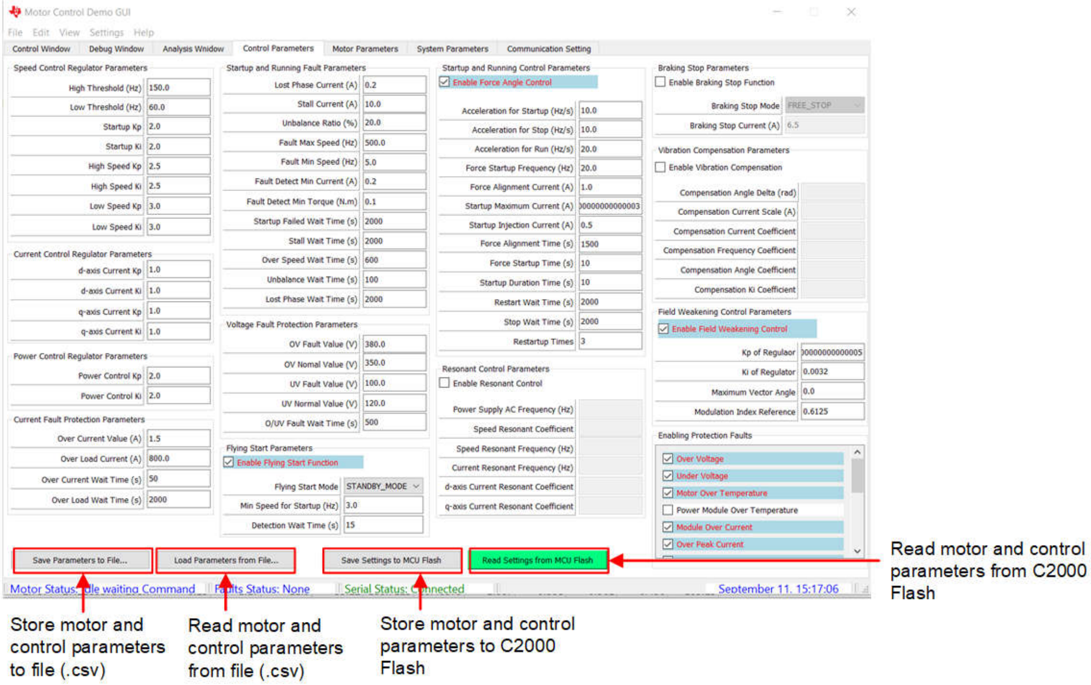
Inverter Control Parameters
PWM Switch Frequency (kHz) 15.0
Control Loop Prescaler 1
Speed Loop Prescaler 10

Rs Online Calibration Parameters
☑ Enable Rs Recalculation
☑ Enable RsOnline ☐ RsOnline Validation T
RsOnline Mode Continue
Injection Current (A): 0.65
Resistance Delta (ohm): 0.001
Angle Delta (rad): 0.0005
Injection Duration Time (s): 120
Interval Waiting Time (s): 300
☑ Enable Temperature Protection with RsOnline
Reference Temp (T0) of Material (C) 22.00
Resistance at Temp T0 (ohm) 2.682000
Temp Coefficient of Material 0.003930
Temp Protection Threshold (C) 86.5
Temp Protection Duration (s) 100.0

Motor Status: Normal Running with Speed or Torque Control | Faults Status: None | Serial Status: Connected | September 11, 15:14:35

Figure 3-8. Start Motor Identification

After identification is complete, the Motor Pairs, Stator resistance, Stator inductance, flux, and so forth, must be written to MCU flash to make sure those parameters are saved inside the MCU. Select the *Control Parameters* tab, choose to store motor and control parameters to MCU Flash, or save them to a file. To verify writing is successful, click the *Read Settings from MCU Flash* button, then select the *Motor Parameters* tab to make sure motor parameters are the same as what were written before. Figure 3-9 shows the locations of the buttons mentioned in this paragraph.


Motor Control Demo GUI
File Edit View Settings Help
Control Window
Debug Window
Analysis Wnidow
Control Parameters
Motor Parameters
System Parameters
Communication Setting
Save Parameters to File...
Load Parameters from File...
Save Settings to MCU Flash
Read Settings from MCU Flash

Speed Control Regulator Parameters
High Threshold (Hz) 150.0
Low Threshold (Hz) 60.0
Startup Kp 2.0
Startup Ki 2.0
High Speed Kp 2.5
High Speed Ki 2.5
Low Speed Kp 3.0
Low Speed Ki 3.0

Current Control Regulator Parameters
d-axis Current Kp 1.0
d-axis Current Ki 1.0
q-axis Current Kp 1.0
q-axis Current Ki 1.0

Power Control Regulator Parameters
Power Control Kp 2.0
Power Control Ki 2.0

Current Fault Protection Parameters
Over Current Value (A) 1.5
Over Load Current (A) 800.0
Over Current Wait Time (s) 50
Over Load Wait Time (s) 2000

Startup and Running Fault Parameters
Lost Phase Current (A) 0.2
Stall Current (A) 10.0
Unbalance Ratio (%) 20.0
Fault Max Speed (Hz) 500.0
Fault Min Speed (Hz) 5.0
Fault Detect Min Current (A) 0.2
Fault Detect Min Torque (N.m) 0.1
Startup Failed Wait Time (s) 2000
Stall Wait Time (s) 2000
Over Speed Wait Time (s) 600
Unbalance Wait Time (s) 100
Lost Phase Wait Time (s) 2000

Voltage Fault Protection Parameters
OV Fault Value (V) 380.0
OV Nomal Value (V) 350.0
UV Fault Value (V) 100.0
UV Normal Value (V) 120.0
O/UV Fault Wait Time (s) 500

Flying Start Parameters
☑ Enable Flying Start Function
Flying Start Mode STANDBY_MODE
Min Speed for Startup (Hz) 3.0
Detection Wait Time (s) 15

Startup and Running Control Parameters
☑ Enable Force Angle Control
Acceleration for Startup (Hz/s) 10.0
Acceleration for Stop (Hz/s) 10.0
Acceleration for Run (Hz/s) 20.0
Force Startup Frequency (Hz) 20.0
Force Alignment Current (A) 1.0
Startup Maximum Current (A))0000000000003
Startup Injection Current (A) 0.5
Force Alignment Time (s) 1500
Force Startup Time (s) 10
Startup Duration Time (s) 10
Restart Wait Time (s) 2000
Stop Wait Time (s) 2000
Restartup Times 3

Resonant Control Parameters
☐ Enable Resonant Control
Power Supply AC Frequency (Hz)
Speed Resonant Coefficient
Speed Resonant Frequency (Hz)
Current Resonant Frequency (Hz)
d-axis Current Resonant Coefficient
q-axis Current Resonant Coefficient

Braking Stop Parameters
☐ Enable Braking Stop Function
Braking Stop Mode FREE_STOP
Braking Stop Current (A) 6.5

Vibration Compensation Parameters
☐ Enable Vibration Compensation
Compensation Angle Delta (rad)
Compensation Current Scale (A)
Compensation Current Coefficient
Compensation Frequency Coefficient
Compensation Angle Coefficient
Compensation Ki Coefficient

Field Weakening Control Parameters
☑ Enable Field Weakening Control
Kp of Regulaor)0000000000005
Ki of Regulator 0.0032
Maximum Vector Angle 0.0
Modulation Index Reference 0.6125

Enabling Protection Faults
☑ Over Voltage
☑ Under Voltage
☑ Motor Over Temperature
☐ Power Module Over Temperature
☑ Module Over Current
☑ Over Peak Current

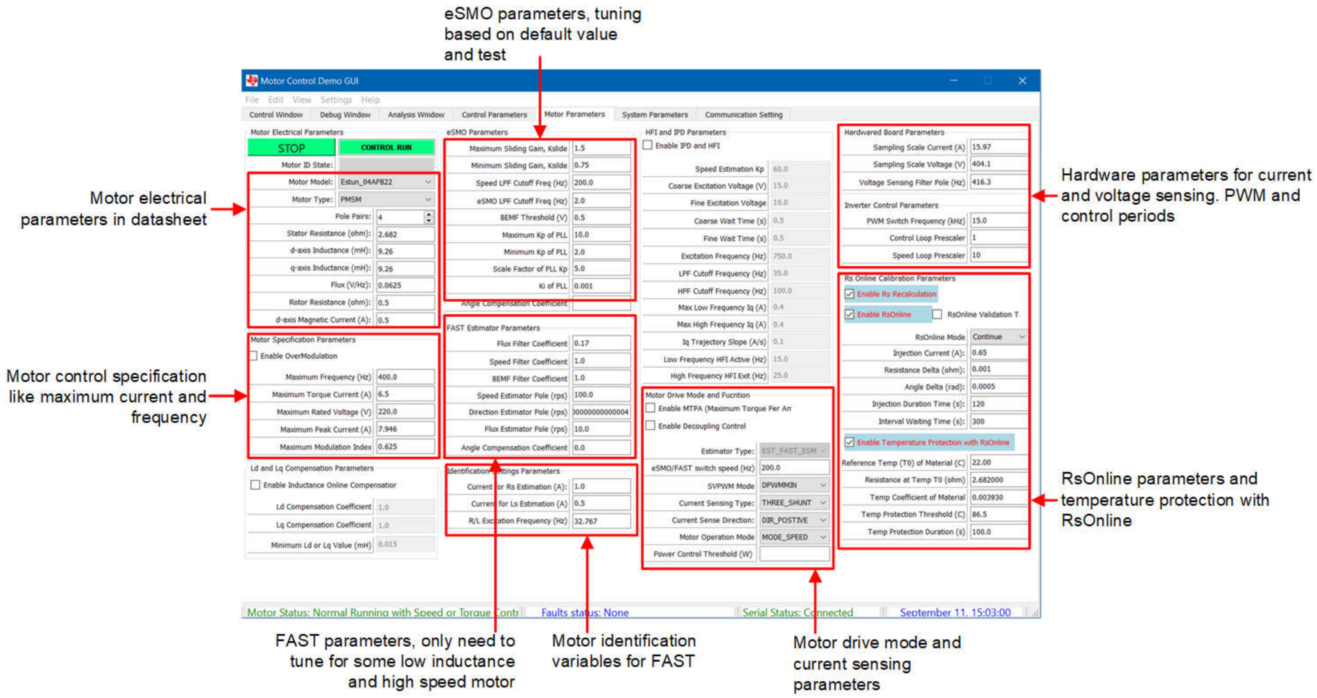
Motor Status: Idle waiting Command | Faults Status: None | Serial Status: Connected | September 11, 15:17:06

Store motor and control parameters to file (.csv)
Read motor and control parameters from file (.csv)
Store motor and control parameters to C2000 Flash
Read motor and control parameters from C2000 Flash

Figure 3-9. Store Motor Identification Result

3.2.5 Spin Motor

Figure 3-10 shows the *Motor Parameters* tab indicating the motor and control parameters. Make sure the motor electrical parameters are correct.



eSMO parameters, tuning based on default value and test

Motor electrical parameters in datasheet

Motor control specification like maximum current and frequency

Hardware parameters for current and voltage sensing. PWM and control periods

RsOnline parameters and temperature protection with RsOnline

FAST parameters, only need to tune for some low inductance and high speed motor

Motor identification variables for FAST

Motor drive mode and current sensing parameters

Motor Electrical Parameters

Motor ID State:	
Motor Model:	Estun_04APB22
Motor Type:	PMSM
Pole Pairs:	4
Stator Resistance (ohm):	2.682
d-axis Inductance (mH):	9.26
q-axis Inductance (mH):	9.26
Flux (V/Hz):	0.0625
Rotor Resistance (ohm):	0.5
d-axis Magnetic Current (A):	0.5

Motor Specification Parameters

□ Enable OverModulation

Maximum Frequency (Hz)	400.0
Maximum Torque Current (A)	6.5
Maximum Rated Voltage (V)	220.0
Maximum Peak Current (A)	7.946
Maximum Modulation Index	0.625

Ld and Lq Compensation Parameters

□ Enable Inductance Online Compensatior

Ld Compensation Coefficient	1.0
Lq Compensation Coefficient	1.0
Minimum Ld or Lq Value (mH)	0.015

eSMO Parameters

Maximum Sliding Gain, Kslide	1.5
Minimum Sliding Gain, Kslide	0.75
Speed LPF Cutoff Freq (Hz)	200.0
eSMO LPF Cutoff Freq (Hz)	2.0
BEMF Threshold (V)	0.5
Maximum Kp of PLL	10.0
Minimum Kp of PLL	2.0
Scale Factor of PLL Kp	5.0
Ki of PLL	0.001
Angle Compensation Coefficient	

FAST Estimator Parameters

Flux Filter Coefficient	0.17
Speed Filter Coefficient	1.0
BEMF Filter Coefficient	1.0
Speed Estimator Pole (rps)	100.0
Direction Estimator Pole (rps)	)000000000000004
Flux Estimator Pole (rps)	10.0
Angle Compensation Coefficient	0.0

Identification Settings Parameters

Current for Rs Estimation (A):	1.0
Current for Ls Estimation (A)	0.5
R/L Excitation Frequency (Hz)	32.767

HFI and IPD Parameters

□ Enable IPD and HFI

Speed Estimation Kp	60.0
Coarse Excitation Voltage (V)	15.0
Fine Excitation Voltage	10.0
Coarse Wait Time (s)	0.5
Fine Wait Time (s)	0.5
Excitation Frequency (Hz)	750.0
LPF Cutoff Frequency (Hz)	35.0
HPF Cutoff Frequency (Hz)	100.0
Max Low Frequency Iq (A)	0.4
Max High Frequency Iq (A)	0.4
Iq Trajectory Slope (A/s)	0.1
Low Frequency HFI Active (Hz)	15.0
High Frequency HFI Exit (Hz)	25.0

Motor Drive Mode and Fucntion

□ Enable MTPA (Maximum Torque Per Ar

□ Enable Decoupling Control

Estimator Type:	EST_FAST_ESM
eSMO/FAST switch speed (Hz)	200.0
SVPWM Mode	DPWMMIN
Current Sensing Type:	THREE_SHUNT
Current Sense Direction:	DIR_POSTIVE
Motor Operation Mode	MODE_SPEED
Power Control Threshold (W)	

Hardwared Board Parameters

Sampling Scale Current (A)	15.97
Sampling Scale Voltage (V)	404.1
Voltage Sensing Filter Pole (Hz)	416.3

Inverter Control Parameters

PWM Switch Frequency (kHz)	15.0
Control Loop Prescaler	1
Speed Loop Prescaler	10

Rs Online Calibration Parameters

☑ Enable Rs Recalculation

☑ Enable RsOnline □ RsOnline Validation T

RsOnline Mode	Continue
Injection Current (A):	0.65
Resistance Delta (ohm):	0.001
Angle Delta (rad):	0.0005
Injection Duration Time (s):	120
Interval Waiting Time (s):	300

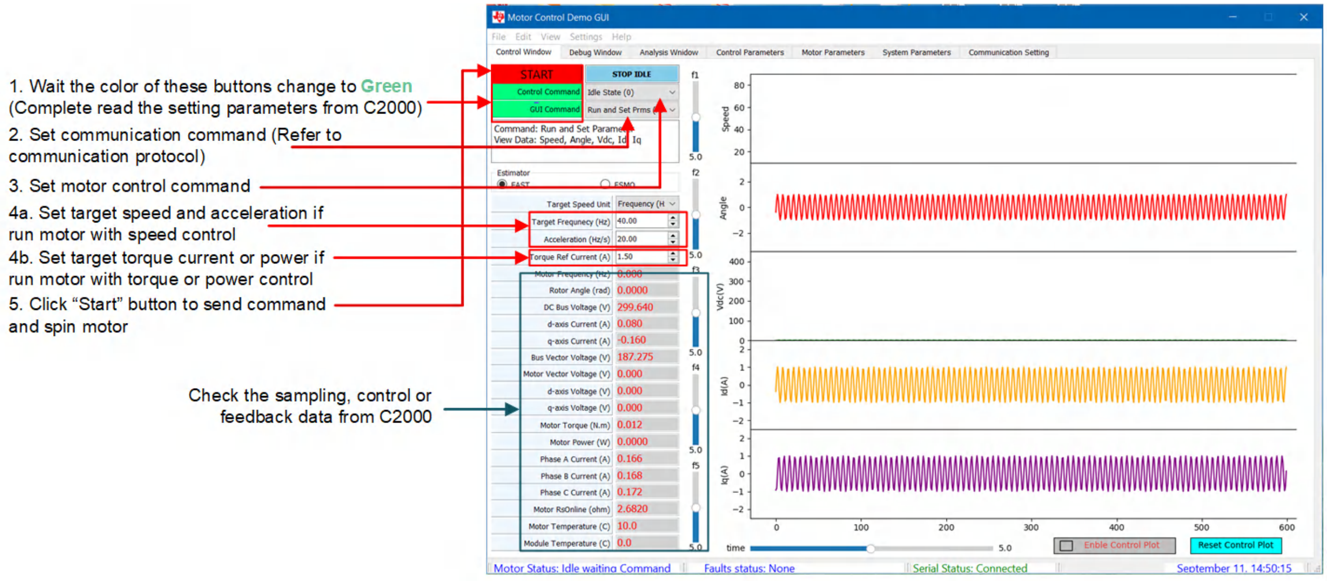
☑ Enable Temperature Protection with RsOnline

Reference Temp (T0) of Material (C)	22.00
Resistance at Temp T0 (ohm)	2.682000
Temp Coefficient of Material	0.003930
Temp Protection Threshold (C)	86.5
Temp Protection Duration (s)	100.0

Motor Status: Normal Running with Speed or Torque Contr | Faults status: None | Serial Status: Connected | September 11, 15:03:00

Figure 3-10. Motor Drive Parameters

On the *Control Window* tab, if the DC BUS voltage is high enough (> 230 VDC),the *Control Command* and *GUI Command* button are in green, the motor spin commands are effective. Follow the steps in Figure 3-11 to spin the motor.



1. Wait the color of these buttons change to Green (Complete read the setting parameters from C2000)
2. Set communication command (Refer to communication protocol)
3. Set motor control command
4a. Set target speed and acceleration if run motor with speed control
4b. Set target torque current or power if run motor with torque or power control
5. Click "Start" button to send command and spin motor

Check the sampling, control or feedback data from C2000

Control Command	Idle State (0)
GUI Command	Run and Set Prms (

Command: Run and Set Parameter
View Data: Speed, Angle, Vdc, Id, Iq

Estimator
● FAST ○ ESMO

Target Speed Unit	Frequency (H
Target Frequney (Hz)	40.00
Acceleration (Hz/s)	20.00
Torque Ref Current (A)	1.50
Motor Frequency (Hz)	0.000
Rotor Angle (rad)	0.0000
DC Bus Voltage (V)	299.640
d-axis Current (A)	0.080
q-axis Current (A)	-0.160
Bus Vector Voltage (V)	187.275
Motor Vector Voltage (V)	0.000
d-axis Voltage (V)	0.000
q-axis Voltage (V)	0.000
Motor Torque (N.m)	0.012
Motor Power (W)	0.0000
Phase A Current (A)	0.166
Phase B Current (A)	0.168
Phase C Current (A)	0.172
Motor RsOnline (ohm)	2.6820
Motor Temperature (C)	10.0
Module Temperature (C)	0.0

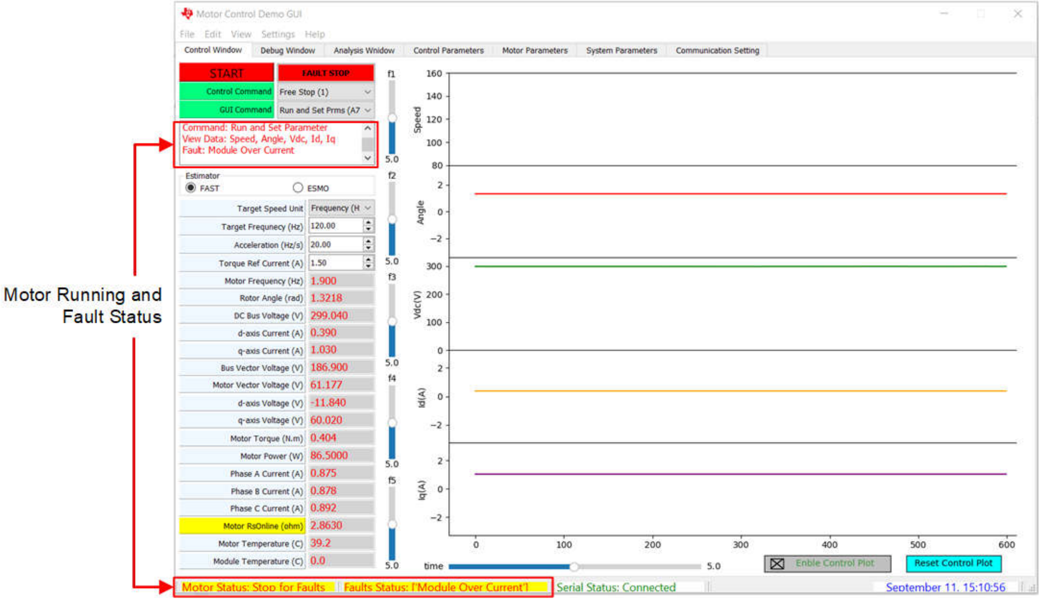
Motor Status: Idle waiting Command | Faults status: None | Serial Status: Connected | September 11, 14:50:15

z

Figure 3-11. Steps to Spin Motor

3.2.6 Motor Fault Status

Faults can be found while the motor is spinning, especially when motor parameters are incorrect or the motor control parameters are not well tuned. Watch the *Control Window* carefully, and pay attention to any faults reported as shown in Figure 3-12.



Command: Run and Set Parameter
View Data: Speed, Angle, Vdc, Id, Iq
Fault: Module Over Current

Estimator
● FAST ○ ESMO

Control Command	Free Stop (1)
GUI Command	Run and Set Prms (A7
Target Speed Unit	Frequency (H
Target Frequney (Hz)	120.00
Acceleration (Hz/s)	20.00
Torque Ref Current (A)	1.50
Motor Frequency (Hz)	1.900
Rotor Angle (rad)	1.3218
DC Bus Voltage (V)	299.040
d-axis Current (A)	0.390
q-axis Current (A)	1.030
Bus Vector Voltage (V)	186.900
Motor Vector Voltage (V)	61.177
d-axis Voltage (V)	-11.840
q-axis Voltage (V)	60.020
Motor Torque (N.m)	0.404
Motor Power (W)	86.5000
Phase A Current (A)	0.875
Phase B Current (A)	0.878
Phase C Current (A)	0.892
Motor RsOnline (ohm)	2.8630
Motor Temperature (C)	39.2
Module Temperature (C)	0.0

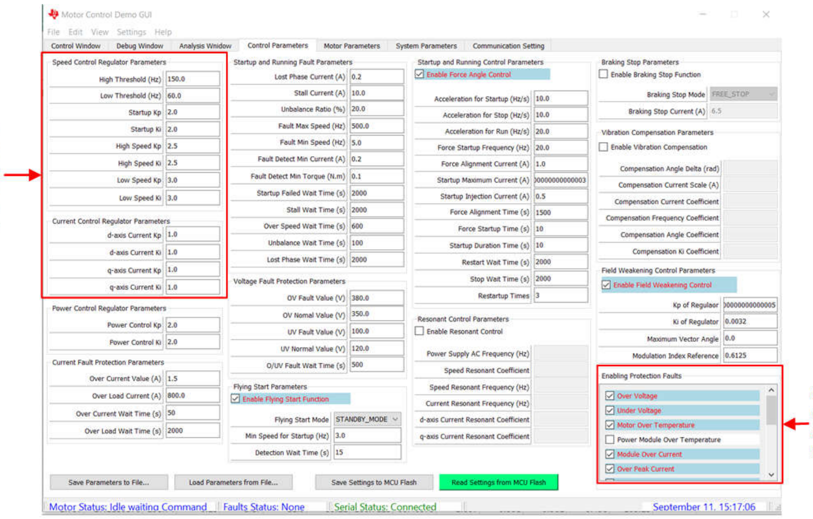
Motor Status: Stop for Faults | Faults Status: ['Module Over Current'] | Serial Status: Connected | September 11, 15:10:56

Figure 3-12. Fault Status Monitor

Faults detection can be enabled or disabled as shown in Figure 3-13.

> **CAUTION**
>
> Disabling some faults can cause the board can break. Do not disable fault protections without full knowledge of acceptable settings.

Configure or change the control parameters for flying start, braking, startup, fault protection according to motor or system. The default parameters value are read from the value set in C2000 controller


Motor Control Demo GUI
File Edit View Settings Help
Control Window
Debug Window
Analysis Wnidow
Control Parameters
Motor Parameters
System Parameters
Communication Setting
Save Parameters to File...
Load Parameters from File...
Save Settings to MCU Flash
Read Settings from MCU Flash

Speed Control Regulator Parameters

High Threshold (Hz)	150.0
Low Threshold (Hz)	60.0
Startup Kp	2.0
Startup Ki	2.0
High Speed Kp	2.5
High Speed Ki	2.5
Low Speed Kp	3.0
Low Speed Ki	3.0

Current Control Regulator Parameters

d-axis Current Kp	1.0
d-axis Current Ki	1.0
q-axis Current Kp	1.0
q-axis Current Ki	1.0

Power Control Regulator Parameters

Power Control Kp	2.0
Power Control Ki	2.0

Current Fault Protection Parameters

Over Current Value (A)	1.5
Over Load Current (A)	800.0
Over Current Wait Time (s)	50
Over Load Wait Time (s)	2000

Startup and Running Fault Parameters

Lost Phase Current (A)	0.2
Stall Current (A)	10.0
Unbalance Ratio (%)	20.0
Fault Max Speed (Hz)	500.0
Fault Min Speed (Hz)	5.0
Fault Detect Min Current (A)	0.2
Fault Detect Min Torque (N.m)	0.1
Startup Failed Wait Time (s)	2000
Stall Wait Time (s)	2000
Over Speed Wait Time (s)	600
Unbalance Wait Time (s)	100
Lost Phase Wait Time (s)	2000

Voltage Fault Protection Parameters

OV Fault Value (V)	380.0
OV Nomal Value (V)	350.0
UV Fault Value (V)	100.0
UV Normal Value (V)	120.0
O/UV Fault Wait Time (s)	500

Flying Start Parameters

☑ Enable Flying Start Function

Flying Start Mode	STANDBY_MODE
Min Speed for Startup (Hz)	3.0
Detection Wait Time (s)	15

Startup and Running Control Parameters

☑ Enable Force Angle Control

Acceleration for Startup (Hz/s)	10.0
Acceleration for Stop (Hz/s)	10.0
Acceleration for Run (Hz/s)	20.0
Force Startup Frequency (Hz)	20.0
Force Alignment Current (A)	1.0
Startup Maximum Current (A)	0000000000003
Startup Injection Current (A)	0.5
Force Alignment Time (s)	1500
Force Startup Time (s)	10
Startup Duration Time (s)	10
Restart Wait Time (s)	2000
Stop Wait Time (s)	2000
Restartup Times	3

Resonant Control Parameters

☐ Enable Resonant Control

Power Supply AC Frequency (Hz)	
Speed Resonant Coefficient	
Speed Resonant Frequency (Hz)	
Current Resonant Frequency (Hz)	
d-axis Current Resonant Coefficient	
q-axis Current Resonant Coefficient	

Braking Stop Parameters

☐ Enable Braking Stop Function

Braking Stop Mode	FREE_STOP
Braking Stop Current (A)	6.5

Vibration Compensation Parameters

☐ Enable Vibration Compensation

Compensation Angle Delta (rad)	
Compensation Current Scale (A)	
Compensation Current Coefficient	
Compensation Frequency Coefficient	
Compensation Angle Coefficient	
Compensation Ki Coefficient	

Field Weakening Control Parameters

☑ Enable Field Weakening Control

Kp of Regulaor	0000000000005
Ki of Regulator	0.0032
Maximum Vector Angle	0.0
Modulation Index Reference	0.6125

Enabling Protection Faults

☑ Over Voltage
☑ Under Voltage
☑ Motor Over Temperature
☐ Power Module Over Temperature
☑ Module Over Current
☑ Over Peak Current

Motor Status: Idle waiting Command | Faults Status: None | Serial Status: Connected | September 11, 15:17:06

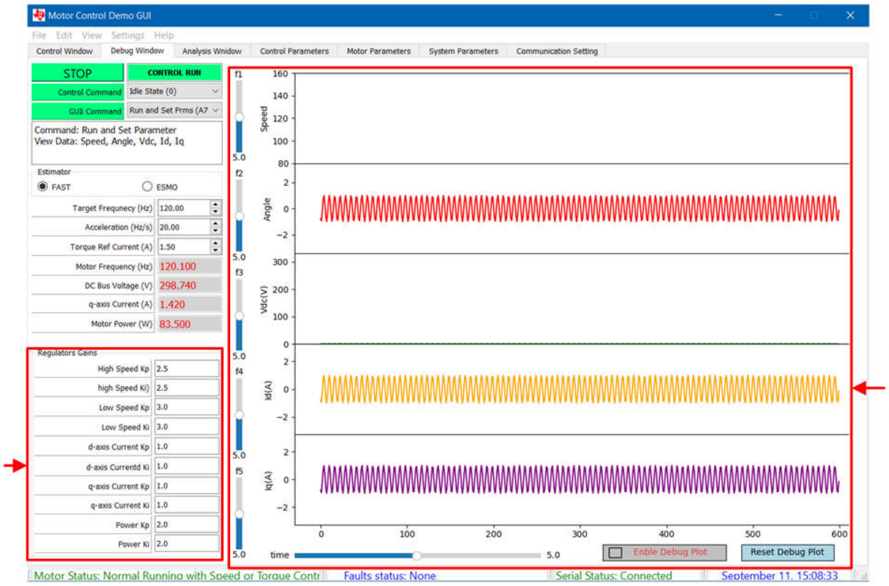
Check the checkbox to enable/ disable the related fault protection according to the system requirement

Configure or change the gain of PI regulators for speed, current, and power control. The values are coefficient, the final gains are these coefficient multiply the setting value in C2000 controller. The setting gains are calculated per the motor electrical parameters.

Figure 3-13. Motor Control Parameters and Faults Detection Disable

3.2.7 Tune Control Parameters

On the *Debug Window* tab, the PI regulators can be tuned for speed, current, and power control as shown in Figure 3-14. The values are coefficient, the final gains are that these coefficients multiply the setting value in the C2000 controller. The setting gains are calculated per the motor electrical parameters. As Figure 3-9 shows, those tuned values can be written to MCU flash.



Control Command	Idle State (0)
GUI Command	Run and Set Prms (A7

Command: Run and Set Parameter
View Data: Speed, Angle, Vdc, Id, Iq

Estimator
● FAST ○ ESMO

Target Frequnecy (Hz)	120.00
Acceleration (Hz/s)	20.00
Torque Ref Current (A)	1.50
Motor Frequency (Hz)	120.100
DC Bus Voltage (V)	298.740
q-axis Current (A)	1.420
Motor Power (W)	83.500

Regulators Gains

High Speed Kp	2.5
high Speed Ki)	2.5
Low Speed Kp	3.0
Low Speed Ki	3.0
d-axis Current Kp	1.0
d-axis Currentd Ki	1.0
q-axis Current Kp	1.0
q-axis Current Ki	1.0
Power Kp	2.0
Power Ki	2.0

Motor Status: Normal Running with Speed or Torque Contr | Faults status: None | Serial Status: Connected | September 11, 15:08:33

Tune the gain of PI regulators for speed, current, and power control. The values are coefficient, the final gains are these coefficient multiply the setting value in C2000 controller. The setting gains are calculated per the motor electrical parameters.

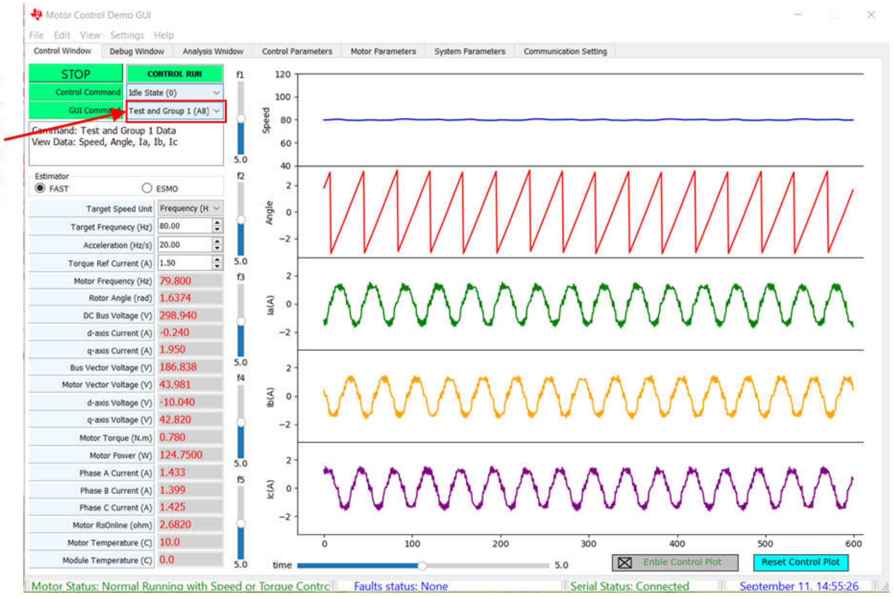
Enable data plot to monitor the speed and current response for tuning the PI regulators

Figure 3-14. Tune Control Parameters

3.2.8 Virtual Oscilloscope

The GUI software has a virtual oscilloscope function, and can show the waveform of angle, phase current, and phase voltage. Figure 3-15 shows how to configure the command to show the rotor angle and phase current.

Using "Test and Group 1 (A8)" communication command to view the feedback speed, rotor angle and the three-phase current (Ia, Ib, and Ic) The target speed and acceleration can't be changed in this mode.



Control Command	Idle State (0)
GUI Command	Test and Group 1 (A8)

Command: Test and Group 1 Data
View Data: Speed, Angle, Ia, Ib, Ic

Estimator ● FAST ○ ESMO

Target Speed Unit	Frequency (H
Target Frequney (Hz)	80.00
Acceleration (Hz/s)	20.00
Torque Ref Current (A)	1.50
Motor Frequency (Hz)	79.800
Rotor Angle (rad)	1.6374
DC Bus Voltage (V)	298.940
d-axis Current (A)	-0.240
q-axis Current (A)	1.950
Bus Vector Voltage (V)	186.838
Motor Vector Voltage (V)	43.981
d-axis Voltage (V)	-10.040
q-axis Voltage (V)	42.820
Motor Torque (N.m)	0.780
Motor Power (W)	124.7500
Phase A Current (A)	1.433
Phase B Current (A)	1.399
Phase C Current (A)	1.425
Motor RsOnline (ohm)	2.6820
Motor Temperature (C)	10.0
Module Temperature (C)	0.0

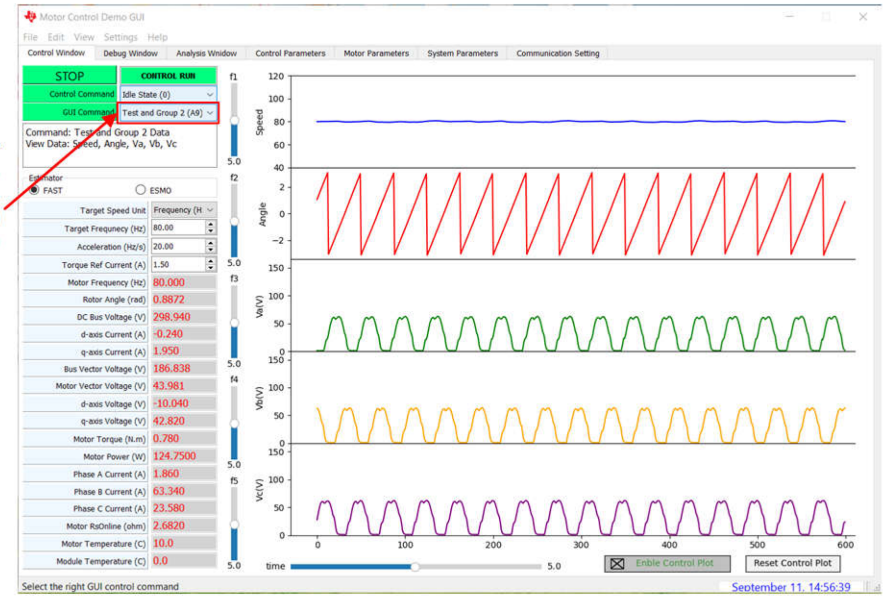
Motor Status: Normal Running with Speed or Torque Contrc | Faults status: None | Serial Status: Connected | September 11, 14:55:26

The graph quality is dependent on the maximum baud rate supporting by the hardware board. Much higher baud rate much better graph view

Figure 3-15. Virtual Oscilloscope for Rotor Angle and Phase Current

Figure 3-16 illustrates the command *Test and Group 2 (A9)* and shows the rotor angle and phase voltages.

Using "Test and Group 2 (A9)" communication command to view the feedback speed, rotor angle and three-phase voltage (Va, Vb, and Vc) The target speed and acceleration can't be changed in this mode.



Control Command	Idle State (0)
GUI Command	Test and Group 2 (A9)

Command: Test and Group 2 Data
View Data: Speed, Angle, Va, Vb, Vc

Estimator ● FAST ○ ESMO

Target Speed Unit	Frequency (H
Target Frequney (Hz)	80.00
Acceleration (Hz/s)	20.00
Torque Ref Current (A)	1.50
Motor Frequency (Hz)	80.000
Rotor Angle (rad)	0.8872
DC Bus Voltage (V)	298.940
d-axis Current (A)	-0.240
q-axis Current (A)	1.950
Bus Vector Voltage (V)	186.838
Motor Vector Voltage (V)	43.981
d-axis Voltage (V)	-10.040
q-axis Voltage (V)	42.820
Motor Torque (N.m)	0.780
Motor Power (W)	124.7500
Phase A Current (A)	1.860
Phase B Current (A)	63.340
Phase C Current (A)	23.580
Motor RsOnline (ohm)	2.6820
Motor Temperature (C)	10.0
Module Temperature (C)	0.0

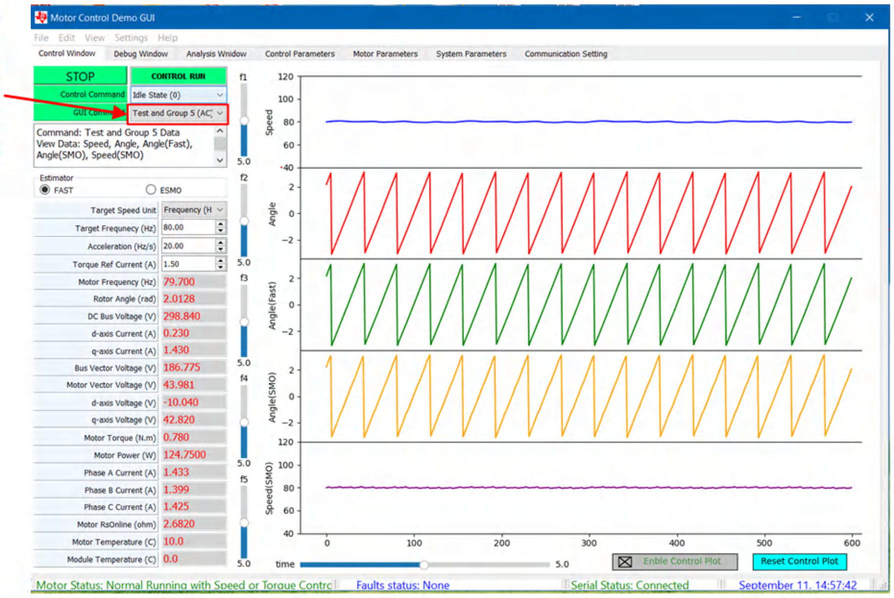
Select the right GUI control command | September 11, 14:56:39

The graph quality is dependent on the maximum baud rate supporting by the hardware board. Much higher baud rate much better graph view

Figure 3-16. Virtual Oscilloscope for Rotor Angle and Phase Voltages

Command *Test and Group 5 (AC)* can show the rotor angle of FAST and EMO as shown in Figure 3-17.

Using "Test and Group 5 (AC)" communication command to view the rotor angle and speed from FAST or eSMO The target speed and acceleration can't be changed in this mode.



Control Command	Idle State (0)
GUI Comm	Test and Group 5 (AC)

Command: Test and Group 5 Data
View Data: Speed, Angle, Angle(Fast), Angle(SMO), Speed(SMO)

Estimator ● FAST ○ ESMO

Target Speed Unit	Frequency (H
Target Frequency (Hz)	80.00
Acceleration (Hz/s)	20.00
Torque Ref Current (A)	1.50
Motor Frequency (Hz)	79.700
Rotor Angle (rad)	2.0128
DC Bus Voltage (V)	298.840
d-axis Current (A)	0.230
q-axis Current (A)	1.430
Bus Vector Voltage (V)	186.775
Motor Vector Voltage (V)	43.981
d-axis Voltage (V)	-10.040
q-axis Voltage (V)	42.820
Motor Torque (N.m)	0.780
Motor Power (W)	124.7500
Phase A Current (A)	1.433
Phase B Current (A)	1.399
Phase C Current (A)	1.425
Motor RsOnline (ohm)	2.6820
Motor Temperature (C)	10.0
Module Temperature (C)	0.0

Motor Status: Normal Running with Speed or Torque Contrc | Faults status: None | Serial Status: Connected | September 11, 14:57:42

The graph quality is dependent on the maximum baud rate supporting by the hardware board. Much higher baud rate much better graph view

Figure 3-17. Virtual Oscilloscope for Rotor Angle of FAST and eSMO

3.3 Getting Started C2000 Firmware

Download and install C2000WARE-MOTORCONTROL-SDK v5.01.00.00 or newer software from the link provided by TI. Install this Motor Control SDK software in the default folder. The software project then resides inside the C2000Ware Motor Control SDK folder at `<install_location>\solutions\tida_010265_wminv\`. Follow these steps to build and run this code with different incremental builds.

3.3.1 Download and Install Software Required for Board Test

1. Download and install Code Composer Studio™ IDE from the Code Composer Studio (CCS) Integrated Development Environment (IDE) tools folder. Version 12.5 or newer is recommended.
2. Install C2000WARE-MOTORCONTROL-SDK in one of two ways:
   - Download the software through the C2000Ware MotorControl SDK tools folder
   - Go to CCS and under *View → Resource Explorer*. Under the TI Resource Explorer, go to *Software → C2000Ware_MotorControl_SDK*, and click the install button.
3. Once installation is complete, close CCS, and create a new workspace for importing the project.

Note

This reference design supports SysConfig for configuring device pins and initializing device peripherals in an easy-to-use graphical interface. This feature is just for a reference in the current release. The designer can download SysConfig, and refer to C2000 SysConfig Software Guide to implement the SysConfig to migrate the reference design to the board for configuring the device.

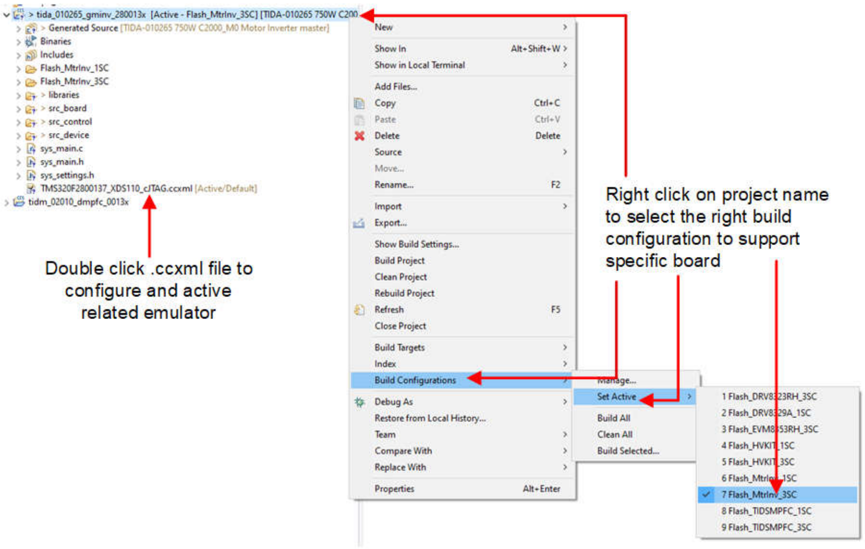
3.3.2 Opening Project Inside CCS

The projectspec file for F280013x based reference design is in the directory below

```
<install_location>\solutions\tida_010265_wminv\f280013x\ccs\motor_control
```

Import the project within CCS and select the right build configurations by right-clicking on project name as shown in Figure 3-18. Select the right build configuration for the HVAC reference design. The *Flash_MtrInv_3SC* build configuration supports the three-shunt current sensing method, *Flash_MtrInv_1SC* supports the single-shunt current sensing method.

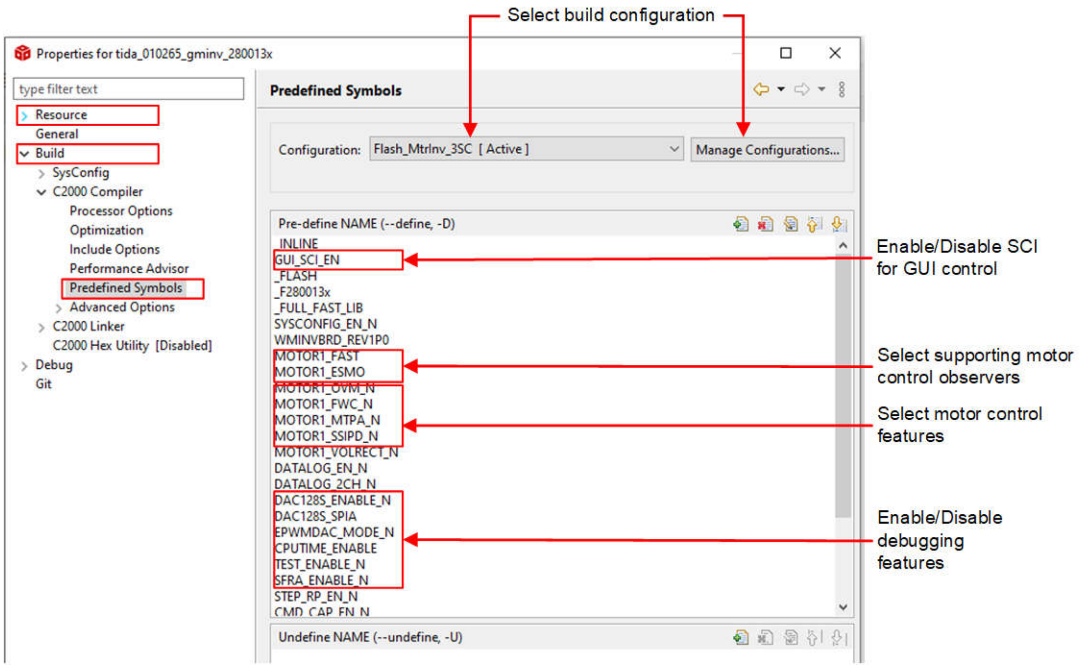
Configure the project to select the supporting functions in the project by clicking *Project* → *Import CCS Projects*, and browse to `<install_location>\solutions\tida_010265_wminv\f280013x\ccs\motor_control` for F280013x based reference design, and then right-click on the imported project name, click *Properties* command to set the pre-define symbols for the project as shown in Figure 3-19.



Double click .ccxml file to configure and active related emulator

Right click on project name to select the right build configuration to support specific board

Figure 3-18. Select the Correct Build Configurations


Select build configuration
Properties for tida_010265_gminv_280013x
type filter text
Resource
General
Build
SysConfig
C2000 Compiler
Processor Options
Optimization
Include Options
Performance Advisor
Predefined Symbols
Advanced Options
C2000 Linker
C2000 Hex Utility [Disabled]
Debug
Git
Predefined Symbols
Manage Configurations...
Enable/Disable SCI for GUI control
Select supporting motor control observers
Select motor control features
Enable/Disable debugging features

Configuration: Flash_MtrInv_3SC [Active]

Pre-define NAME (--define, -D)

```
INLINE
GUI_SCI_EN
_FLASH
_F280013x
_FULL_FAST_LIB
SYSCONFIG_EN_N
WMINVBRD_REV1P0
MOTOR1_FAST
MOTOR1_ESMO
MOTOR1_OVM_N
MOTOR1_FWC_N
MOTOR1_MTPA_N
MOTOR1_SSIPD_N
MOTOR1_VOLRECT_N
DATALOG_EN_N
DATALOG_2CH_N
DAC128S_ENABLE_N
DAC128S_SPIA
EPWMDAC_MODE_N
CPUTIME_ENABLE
TEST_ENABLE_N
SFRA_ENABLE_N
STEP_RP_EN_N
CMD_CAP_EN_N
```

Undefine NAME (--undefine, -U)

Figure 3-19. Select the Correct Predefined Symbols in Project Properties

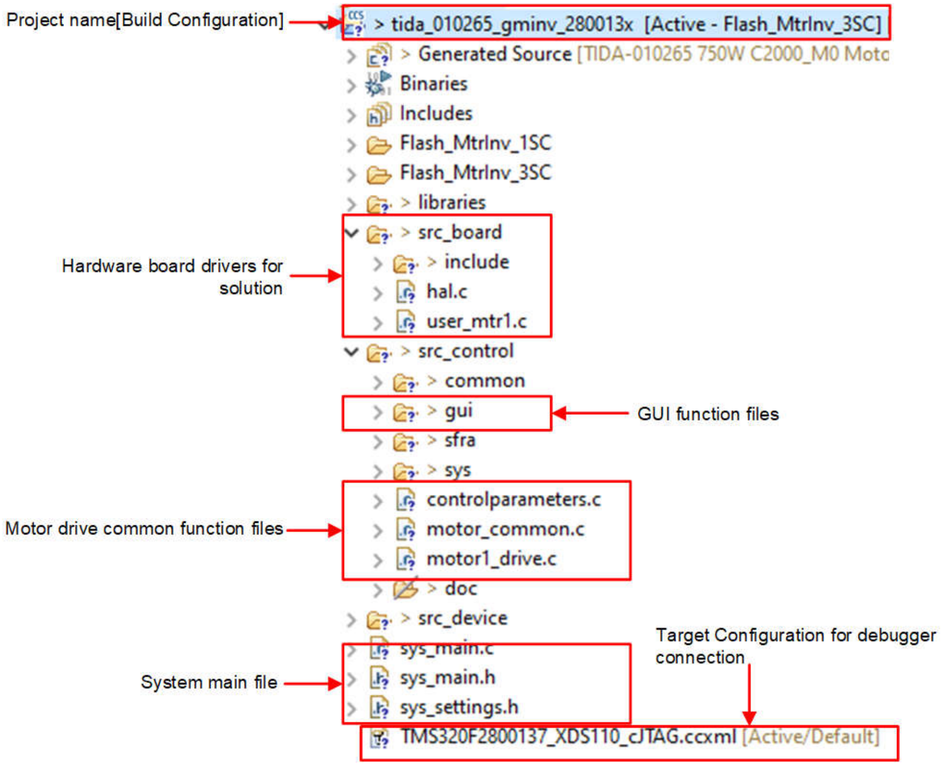
3.3.3 Project Structure

Once the project is imported, the project explore appears inside CCS as shown in Figure 3-20. The device peripherals configuration is based on C2000Ware driverlib. The users only need to change the codes and definitions in *hal.c* and *hal.h*.

The folder *src_control* includes *hal.c* and *user_mtr1.c*, in which users can change codes and definitions.

The folder *src_board* includes board drivers for this hardware board.

The folder *src_control* includes motor drive files that call motor control core algorithm functions within the interrupt service routines and background tasks.



Project name[Build Configuration]

Hardware board drivers for solution

GUI function files

Motor drive common function files

Target Configuration for debugger connection

System main file

Figure 3-20. TIDA-010265 Project Explorer View

Figure 3-21 shows the project software flow diagram of ISR for motor control, a main loop for motor control parameters update in background loop.

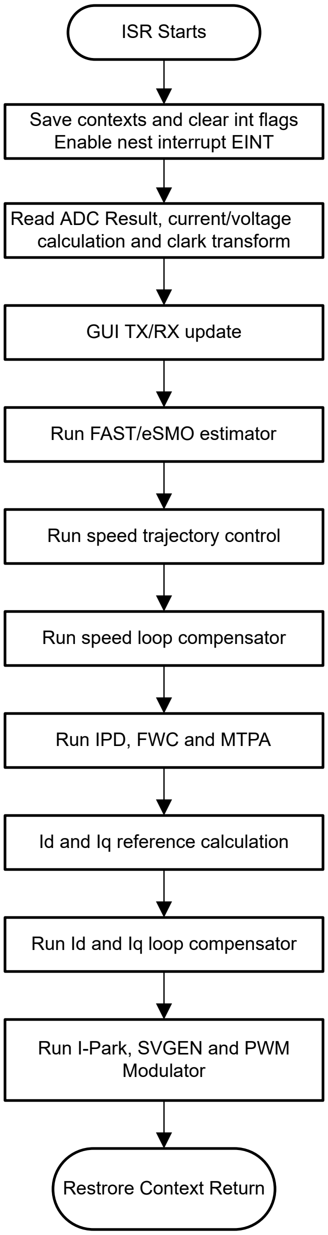

ISR Starts
Save contexts and clear int flags
Enable nest interrupt EINT
Read ADC Result, current/voltage calculation and clark transform
GUI TX/RX update
Run FAST/eSMO estimator
Run speed trajectory control
Run speed loop compensator
Run IPD, FWC and MTPA
Id and Iq reference calculation
Run Id and Iq loop compensator
Run I-Park, SVGEN and PWM Modulator
Restrore Context Return

Figure 3-21. Firmware Project Flow Diagram

The project consists of a motor control interrupt service routine, which are called every PWM cycle. A few background tasks are called in a loop forever in main() and can be used to run slow tasks for which absolute timing accuracy is not required, motor control parameters update, and so on. A CPU timer is used to trigger slow background tasks.

motor1CtrlISR is reserved for calling the motor drive control algorithms to spin the motor 1 that is periodically triggered at USER_M1_ISR_FREQ_Hz.

To simplify the system, the bring up and design of the software for this reference design is organized in four labs with incremental builds (DMC_BUILDLEVEL), which makes learning and getting familiar with the board and software easier. This approach is also good for debugging and testing boards. Table 3-1 lists the detailed incremental build options. To select a particular build option, select the corresponding BUILDLEVEL option in *sys_settings.h*. Once the build option is selected, compile the project by selecting the *rebuild all* compiler option. Section 3.3.4 provides more details to run each of the build level options.

Table 3-1. Incremental Build Options

OPERATION	BUILD OPTION	DESCRIPTION
MOTOR DRIVE	DMC_LEVEL_1	50% PWM duty, verify ADC offset calibration, PWM output and phase shift
	DMC_LEVEL_2	Open-loop v/f control to check current and voltage sensing signals for motor
	DMC_LEVEL_3	Closed current loop to check the hardware settings
	DMC_LEVEL_4	Motor parameters identify and run with InstaSPIN-FOC or eSMO

3.3.4 Test Procedure

WARNING

There are **high voltages** present on the board. To safely evaluate this board, use an appropriate isolated and current limited power source. Before power is applied to the board, an appropriate resistive or electronic load must be connected at the output. Do not handle the unit when power is applied. Only use appropriately rated equipment and follow proper isolation and safety practices.

CAUTION

Use caution when connecting scopes and other test equipment to the board because the AC rectifier generates the DC-output voltage, which has a **HOT Ground** floating from the protective earth ground. Isolation transformers must be used when connecting grounded equipment to the kit.

3.3.4.1 Build Level 1: CPU and Board Setup

Objectives learned in this build level:

- Evaluate the open-loop operation of the system
- Use the HAL object to set up the MCU controller and initialize the inverter
- Verify the PWM and ADC driver modules
- Become familiar with the operation of CCS

Because this system is running with open-loop control, the ADC measured values are only used for instrumentation purposes in this build level. Only bias power supply for MCU controller and gate drivers is used in this build level. The high-voltage AC and DC power supply are not implemented on the inverter.

In this build level, the board is executed in open-loop fashion with a fixed duty cycle. The duty cycles are set to 50% for the motor. This build level verifies the sensing of feedback values from the power stage and also operation of the PWM gate driver and makes sure there are no hardware issues. Additionally calibration of input and output voltage sensing can be performed in this build level. The software flow for this build level is shown in Figure 3-22.

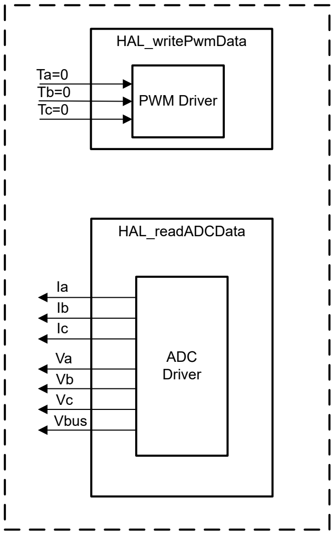


Figure 3-22. Control Software Block Diagram: Build Level 1 – Offset Validation

3.3.4.1.1 Start CCS and Open Project

To start CCS and open the project, complete the following steps:

1. Connect the emulator at J15.
2. Connect AC or DC power supply to J5 as shown in Figure 3-23.
3. Open CCSv12.5 (or newer). A project contains all the files and build options needed to generate an executable output file (.out), which can be run on the C2000 controller-based hardware. On the menu bar, click *Project* → *Import CCS Projects*. Below *Select search-directory:*, browse to the C2000Ware Motor Control SDK folder and select `<install_location>\solutions\tida_010265_wminv`. Click *Finish* to import the related project into CCS. This project invokes all the necessary tools (compiler, assembler, linker) to build the project.
4. In the project window on the left, click the plus sign (+) to the left of Project. An example project window is shown in Figure 3-20.

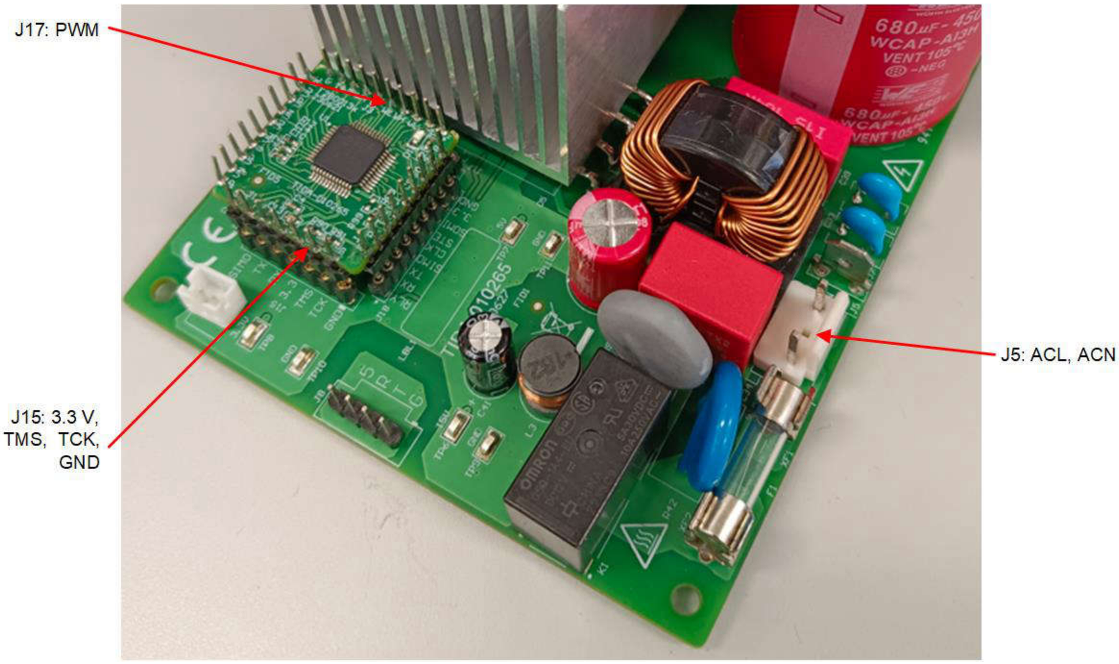


Figure 3-23. Connect the External AC or DC Power Supply to Verify the Hardware

3.3.4.1.2 Build and Load Project

To build and load the project, complete the following steps:

1. Right-click on the project name, click the *Properties* command, move to pre-defined symbols to change *GUI_SCI_EN* to *GUI_SCI_N* to disable the SCI function for the GUI as shown in Figure 3-19.

> **Note**
>
> The SCI function can be enabled again to allow GUI control through SCI, as described in Section 3.2.

2. Open the sys_settings.h file, set DMC_BUILDLEVEL to DMC_LEVEL_1.
3. If another build option was built previously, right click on the project name and click on *Clean Project*, and then click on *Build Project*. Watch the tools run in the build window. The project builds successfully.
4. In the *Project Explorer* make sure the correct target configuration file is set as Active as shown in Figure 3-20.
5. Turn on the AC or DC power supply to apply 30 VAC or 40 VDC to J5, to create the +15 V and 3.3 V for the controller and gate driver. Click on the Debug button or click *Run* → *Debug*. The build level 1 code can be compiled and loaded on the C2000 device. Notice the CCS Debug icon in the upper right-hand corner, indicating that the user is now in the *Debug Perspective* view. The program can be stopped at the start of main().

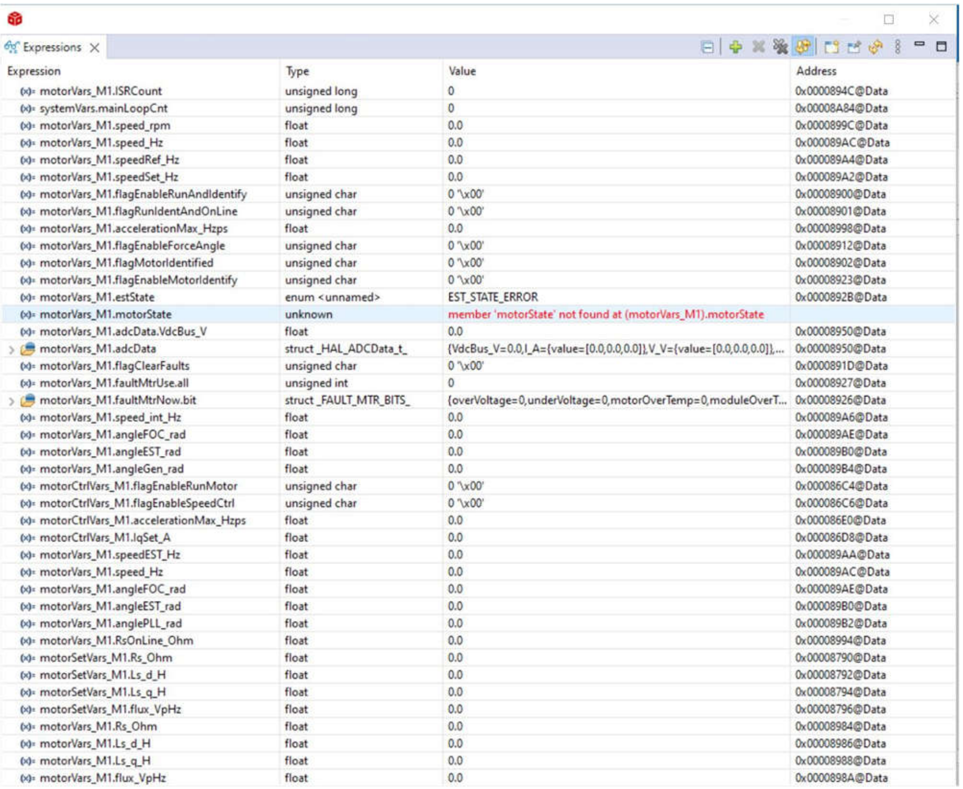
3.3.4.1.3 Setup Debug Environment Windows

To watch local and global variables while debugging code is a standard debug practice. There are various methods for doing this in CCS, such as memory views and watch views. Additionally, CCS has the ability to make time (and frequency) domain plots. This ability allows the user to view waveforms using the graph tool.

1. Click *View* → *Expressions* on the menu bar to open an *Expressions* watch window . Move the mouse to the *Expressions* window to view the variables being used in the project. Add variables to the *Expressions* window as shown in Figure 3-24. The window uses the number format associated with variables during declaration, and Figure 3-24 shows an example of the *Expressions* window. Select a desired number format for the variable by right clicking on expressions and choosing.
2. Alternately, a group of variables can be imported into the *Expressions* window by right clicking within the *Expressions* window and clicking *Import*, and browse to the directory of the project at `<install_location>\solutions\tida_010265_wminv\src_control\common\debug` and pick *BuildLevel1.txt* and click the *OK* button to import the variables shown in Figure 3-24.

> **Note**
>
> Some of the variables have not been initialized at this point in the main code and can contain some useless values.

3. The structure variables *motorVars_M1[]* have references to most variables that are related to controlling the motor. By expanding this variable, you can see and edit all the variables, as needed.
4. Click on the *Continuous Refresh* button in the expressions window. This enables the window to run with real-time mode. By clicking the down arrow in this *Expressions* window, you can select *Customize Continuous Refresh Interval* and edit the refresh rate of the expressions window. Choosing too fast an interval can affect performance.



Expression	Type	Value	Address
motorVars_M1.ISRCount	unsigned long	0	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	0	0x00008A84@Data
motorVars_M1.speed_rpm	float	0.0	0x0000899C@Data
motorVars_M1.speed_Hz	float	0.0	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	0.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	0.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	0 '\x00'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	0 '\x00'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	0.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	0 '\x00'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	0 '\x00'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ERROR	0x0000892B@Data
motorVars_M1.motorState	unknown	member 'motorState' not found at (motorVars_M1).motorState	
motorVars_M1.adcData.VdcBus_V	float	0.0	0x00008950@Data
› motorVars_M1.adcData	struct _HAL_ADCData_t_	{VdcBus_V=0.0,I_A={value=[0.0,0.0,0.0]},V_V={value=[0.0,0.0,0.0]},...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUse.all	unsigned int	0	0x00008927@Data
› motorVars_M1.faultMtrNow.bit	struct _FAULT_MTR_BITS_	{overVoltage=0,underVoltage=0,motorOverTemp=0,moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	0.0	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	0.0	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.0	0x000089B0@Data
motorVars_M1.angleGen_rad	float	0.0	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E0@Data
motorCtrlVars_M1.IqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	0.0	0x000089AA@Data
motorVars_M1.speed_Hz	float	0.0	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	0.0	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.0	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	0.0	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	0.0	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	0.0	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.0	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.0	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0	0x00008796@Data
motorVars_M1.Rs_Ohm	float	0.0	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.0	0x00008986@Data
motorVars_M1.Ls_q_H	float	0.0	0x00008988@Data
motorVars_M1.flux_VpHz	float	0.0	0x0000898A@Data

Figure 3-24. Build Level 1: Expressions Watch Window at Reset

3.3.4.1.4 Run the Code

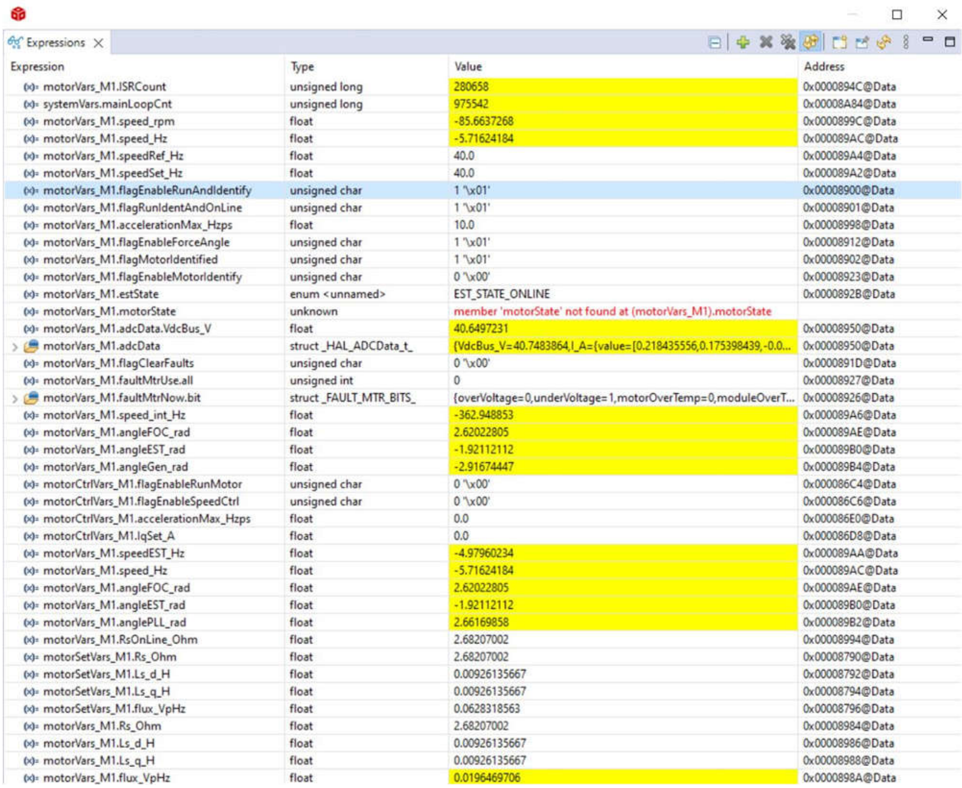
To run the code, complete the following steps:

1. Run the project by clicking the button , or click *Run* → *Resume* in the *Debug* tab.
2. In the *Expressions* window, set the variables *motorVars_M1.flagEnableRunAndIdentify* to "1" after *systemVars.flagEnableSystem* was automatically set to "1" in the watch window.
3. The project can now run, and the values in the graphs and *Expressions* window can continuously update as shown in Figure 3-25 while using this project. The windows can be resized according to user preference.
4. In the watch view, the variables *motorVars_M1.flagRunIdentAndOnLine* can be set to "1" automatically. The *ISRCount* is increasing continuously.
5. Check calibration offsets of the motor, the offset value of the motor phase current sensing can be equal to approximately half of the scale current of ADC as shown in Figure 3-25.
6. Probe the PWM output for motor drive control with an oscilloscope at J15 as shown in Figure 3-23. All of the PWM duty are set to 50% in this build level, the PWM output waveforms are as shown in Figure 3-26. The PWM switching frequency of motor_1 is 15 kHz.
7. The controller can now be halted, and the debug connection terminated. Fully halt the controller by first clicking the *Halt* button on the toolbar or by clicking *Target* → *Halt*. Finally, reset the controller by clicking on or clicking *Run* → *Reset*.
8. Erase the code in the controller for the next build level by clicking *Tools* → *On-Chip Flash*, and click *Erase Flash* in the *On-Chip Flash* tab (make sure that all of the flash banks are checked) as shown in Figure 3-27. This operation erases all of the program code stored in flash. (This step is optional, the user can ignore this step to load the new program code in next build level.)

Note

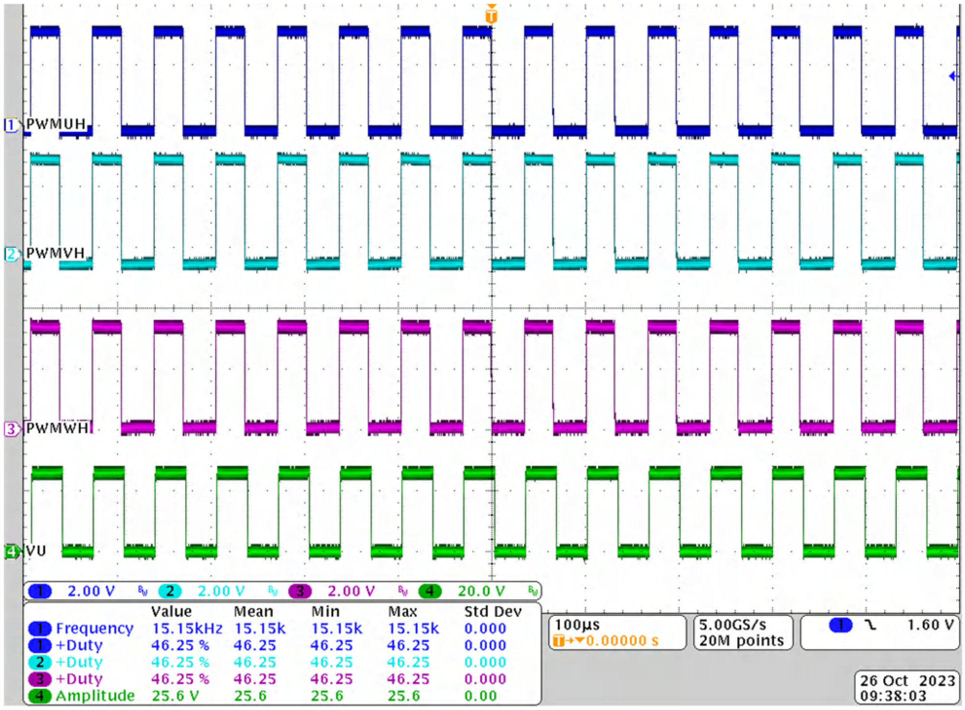
Do not click *Cancel*, turn off the power of the board, or disconnect the emulator when erasing flash.

9. Close the CCS debug session by clicking the *Terminate Debug Session* button or clicking *Run* → *Terminate*.



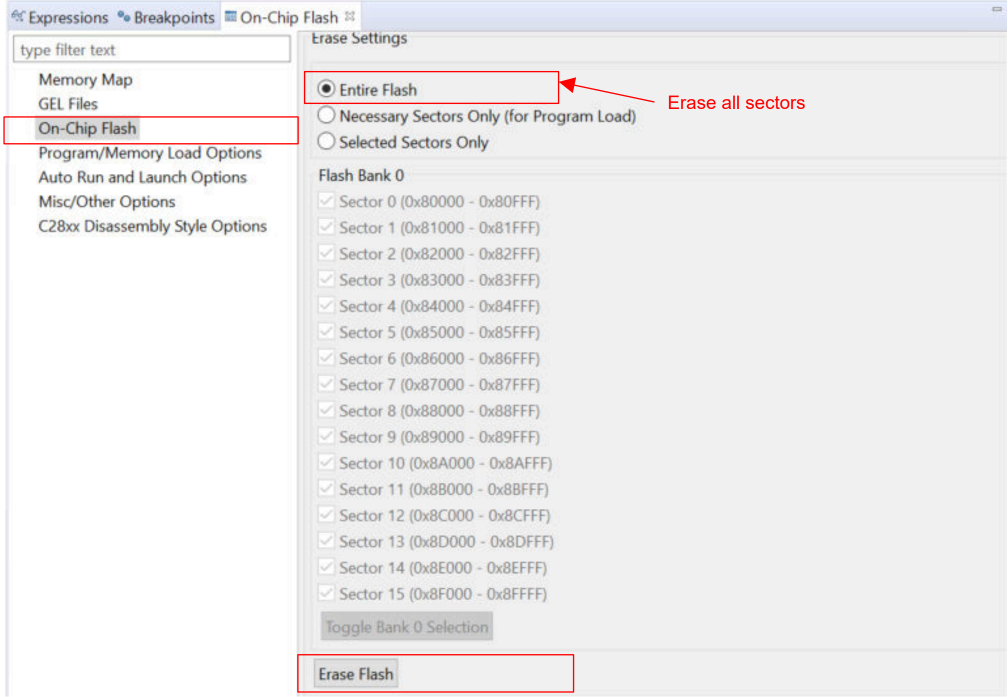
Expression	Type	Value	Address
motorVars_M1.ISRCount	unsigned long	280658	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	975542	0x00008A84@Data
motorVars_M1.speed_rpm	float	-85.6637268	0x0000899C@Data
motorVars_M1.speed_Hz	float	-5.71624184	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	40.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	40.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	1 '\x01'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\x01'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	10.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\x01'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\x01'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ONLINE	0x0000892B@Data
motorVars_M1.motorState	unknown	member 'motorState' not found at (motorVars_M1).motorState	
motorVars_M1.adcData.VdcBus_V	float	40.6497231	0x00008950@Data
motorVars_M1.adcData	struct _HAL_ADCData_t_	{VdcBus_V=40.7483864,I_A={value=[0.218435556,0.175398439,-0.0...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUse.all	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNow.bit	struct _FAULT_MTR_BITS_	{overVoltage=0,underVoltage=1,motorOverTemp=0,moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	-362.948853	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	2.62022805	0x000089AE@Data
motorVars_M1.angleEST_rad	float	-1.92112112	0x000089B0@Data
motorVars_M1.angleGen_rad	float	-2.91674447	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E0@Data
motorCtrlVars_M1.IqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	-4.97960234	0x000089AA@Data
motorVars_M1.speed_Hz	float	-5.71624184	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	2.62022805	0x000089AE@Data
motorVars_M1.angleEST_rad	float	-1.92112112	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	2.66169858	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	2.68207002	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	2.68207002	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.00926135667	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.00926135667	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0628318563	0x00008796@Data
motorVars_M1.Rs_Ohm	float	2.68207002	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008986@Data
motorVars_M1.Ls_q_H	float	0.00926135667	0x00008988@Data
motorVars_M1.flux_VpHz	float	0.0196469706	0x0000898A@Data

Figure 3-25. Build Level 1: Expressions Window at Run Time



	Value	Mean	Min	Max	Std Dev
1 Frequency	15.15kHz	15.15k	15.15k	15.15k	0.000
1 +Duty	46.25 %	46.25	46.25	46.25	0.000
2 +Duty	46.25 %	46.25	46.25	46.25	0.000
3 +Duty	46.25 %	46.25	46.25	46.25	0.000
4 Amplitude	25.6 V	25.6	25.6	25.6	0.00

Figure 3-26. Build Level 1: MCU PWM Output and IPM Output



Erase Settings

● Entire Flash
○ Necessary Sectors Only (for Program Load)
○ Selected Sectors Only

Flash Bank 0

☑ Sector 0 (0x80000 - 0x80FFF)
☑ Sector 1 (0x81000 - 0x81FFF)
☑ Sector 2 (0x82000 - 0x82FFF)
☑ Sector 3 (0x83000 - 0x83FFF)
☑ Sector 4 (0x84000 - 0x84FFF)
☑ Sector 5 (0x85000 - 0x85FFF)
☑ Sector 6 (0x86000 - 0x86FFF)
☑ Sector 7 (0x87000 - 0x87FFF)
☑ Sector 8 (0x88000 - 0x88FFF)
☑ Sector 9 (0x89000 - 0x89FFF)
☑ Sector 10 (0x8A000 - 0x8AFFF)
☑ Sector 11 (0x8B000 - 0x8BFFF)
☑ Sector 12 (0x8C000 - 0x8CFFF)
☑ Sector 13 (0x8D000 - 0x8DFFF)
☑ Sector 14 (0x8E000 - 0x8EFFF)
☑ Sector 15 (0x8F000 - 0x8FFFF)

Figure 3-27. Build Level 1: Erase Program Code in Flash for Next Build Level

3.3.4.2 Build Level 2: Open-Loop Check With ADC Feedback

Objectives learned in this build level:

- Implements a simple scalar v/f control of motor to drive motor for validating current and voltage sensing circuit, and IPM circuit.
- Test InstaSPIN-FOC FAST or eSMO modules for motor control.

This system is running with open-loop control, the ADC measured values are only used for instrumentation purposes in this build level. The high-voltage DC power supply is implemented on the inverter, the bias power supply for the MCU controller and IPM is provided by the auxiliary power-supply module. Figure 3-28 shows the software flow for this build level.

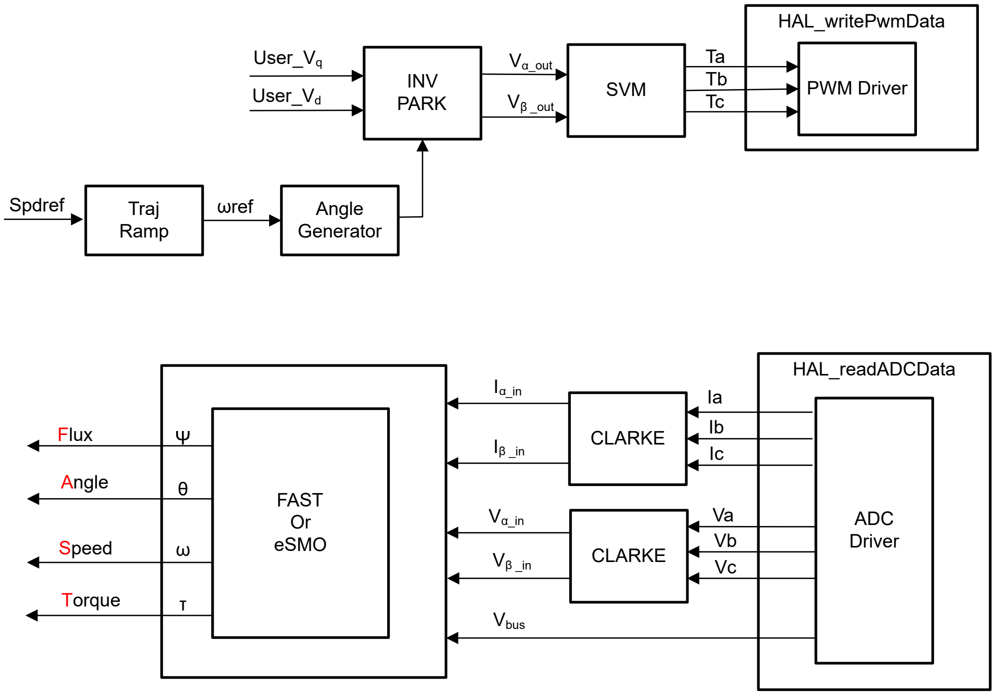


Figure 3-28. Control Software Block Diagram: Build Level 2 – Open-Loop Control

3.3.4.2.1 Start CCS and Open Project

To start CCS and open the project, complete the following steps:

1. Connect an isolated AC power source capable of providing universal input up to 750 W to the input terminals (connector J5) of the reference board as shown in Figure 3-23. Set the power supply current limit to 2 A. Do not turn the power supply on at this time.
2. Connect a motor to J10.
3. Follow Steps 2 and 3 of Section 3.3.4.1.1 to open the project.

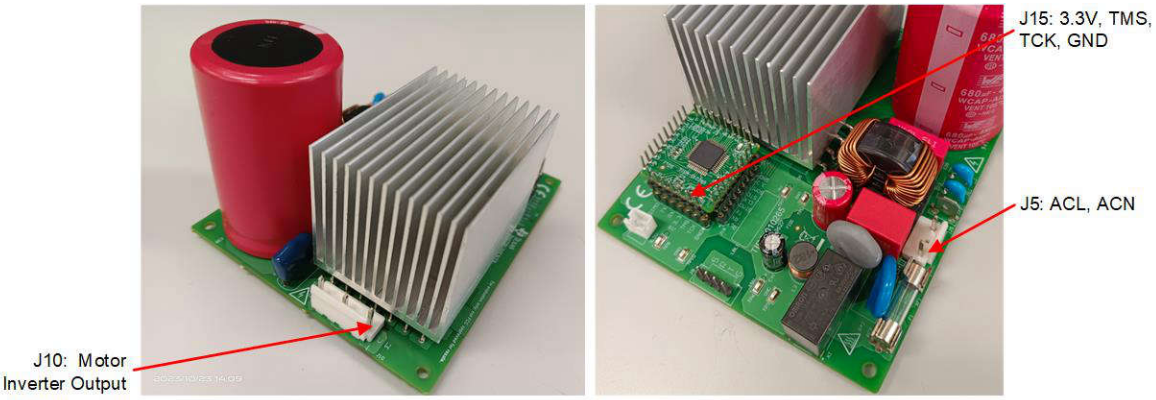


Figure 3-29. Connect the External AC or DC Power Supply to Verify the Hardware

3.3.4.2.2 Build and Load Project

To build and load the project, complete the following steps:

1. Set DMC_BUILDLEVEL to DMC_LEVEL_2.
2. Follow Steps 2 to 4 of Section 3.3.4.1.2 to build the project and load code into controller.

3.3.4.2.3 Setup Debug Environment Windows

Follow Steps 1 to 4 of Section 3.3.4.1.3 to import the variables into the *Expressions* window by picking *BuildLevel2.txt*. The *Expressions* window appears as shown in Figure 3-30.

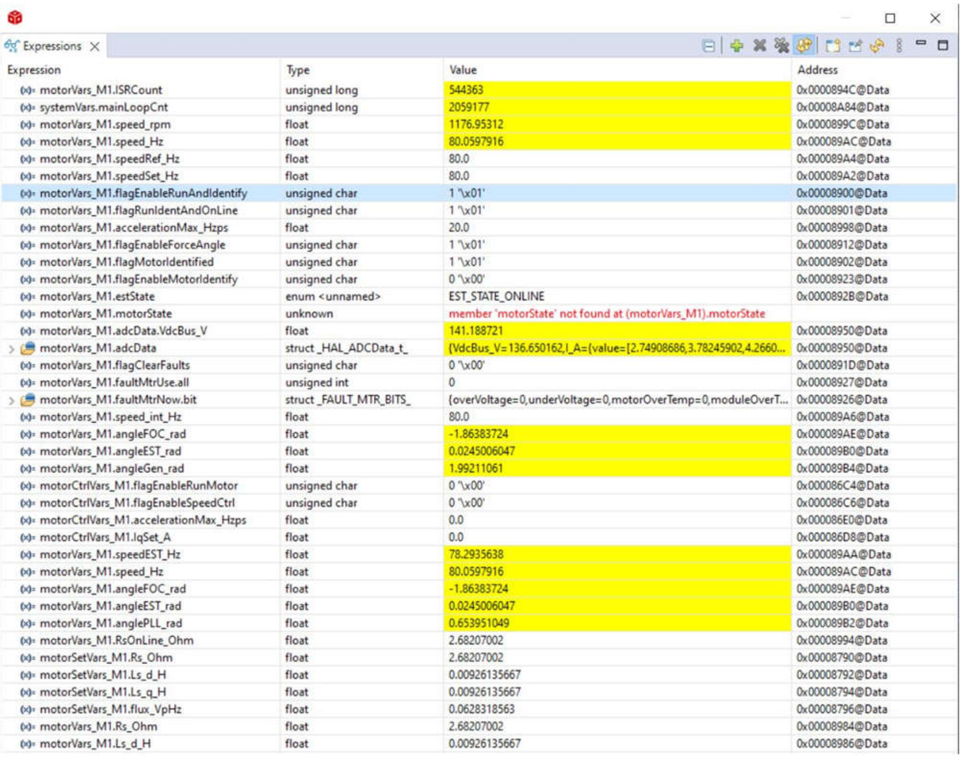
3.3.4.2.4 Run the Code

To run the code, complete the following steps:

1. Set the AC power source output to 0 V, turn on the AC power source, slowly increase the output voltage from 0-V to 100-VAC.
2. Run the project by clicking on the ▶ button, or click *Run* → *Resume* in the *Debug* tab. The motor fault flags *motorVars_M1.faultMtrUse.all* need to be equal to "0" , if not, the user must check the current and voltage sensing circuit as described in Section 3.3.4.1.
3. To verify the current and voltage-sensing circuit of the inverter for the motor, set the variable *motorVars_M1.flagEnableRunAndIdentify* to "1" in the *Expressions* window as shown in Figure 3-30. The motor_1 needs to run with v/f open loop, tune the v/f profile parameters in *user_mtr1.h* as below according to the specification of the motor if the motor does not spin smoothly.

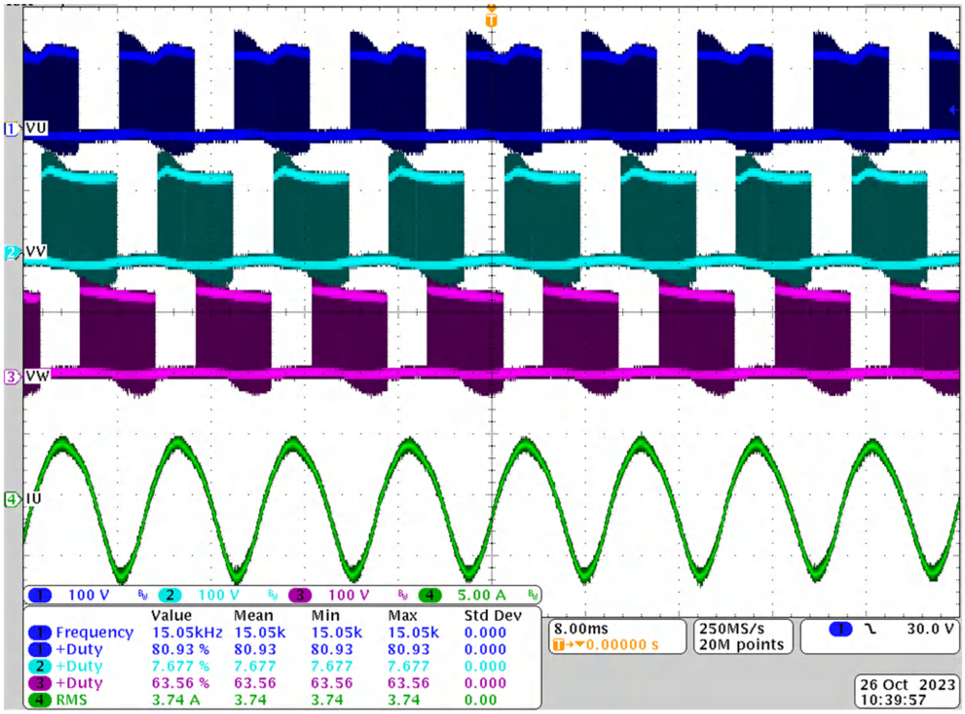
```
#define USER_MOTOR1_FREQ_LOW_Hz            (10.0f)         // Hz
#define USER_MOTOR1_FREQ_HIGH_Hz           (200.0f)        // Hz
#define USER_MOTOR1_VOLT_MIN_V             (10.0f)         // Volt
#define USER_MOTOR1_VOLT_MAX_V             (200.0f)        // Volt
```

4. The motor now spins with a setting speed in the variable *motorVars_M1.speedRef_Hz,* check the value of *motorVars_M1.speed_Hz* in the *Expressions* window. The value needs to be very close, as shown in Figure 3-30.
5. Connect the oscilloscope voltage and current probes to watch the motor phase voltage and current as shown in Figure 3-31.
6. Verify the overcurrent fault protection by decreasing the value of the variable *motorVars_M1.overCurrent_A*, the overcurrent protection is implemented by the CMPSS modules. The overcurrent fault is triggered if the *motorVars_M1.overCurrent_A* is set to a value less than the actual current, the PWM output is disabled, the *motorVars_M1.flagEnableRunAndIdentify* is cleared to "0", and the *motorVars_M1.faultMtrUse.all* is set to "0x10".
7. The controller can now be halted, and the debug connection terminated. Fully halting the controller by first clicking the *Halt* button on the toolbar ⏸ or by clicking *Target* → *Halt*. Finally, reset the controller by clicking on or clicking *Run* → *Reset*.
8. Close the CCS debug session by clicking on *Terminate Debug Session* ■ or clicking *Run* → *Terminate*.



Expression	Type	Value	Address
motorVars_M1.ISRCount	unsigned long	544363	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	2059177	0x00008A84@Data
motorVars_M1.speed_rpm	float	1176.95312	0x0000899C@Data
motorVars_M1.speed_Hz	float	80.0597916	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	80.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	80.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	1 '\x01'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\x01'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	20.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\x01'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\x01'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ONLINE	0x0000892B@Data
motorVars_M1.motorState	unknown	member 'motorState' not found at (motorVars_M1).motorState	
motorVars_M1.adcData.VdcBus_V	float	141.188721	0x00008950@Data
motorVars_M1.adcData	struct _HAL_ADCData_t_	{VdcBus_V=136.650162,I_A={value=[2.74908686,3.78245902,4.2660...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUse.all	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNow.bit	struct _FAULT_MTR_BITS_	{overVoltage=0,underVoltage=0,motorOverTemp=0,moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	80.0	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	-1.86383724	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.0245006047	0x000089B0@Data
motorVars_M1.angleGen_rad	float	1.99211061	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E0@Data
motorCtrlVars_M1.IqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	78.2935638	0x000089AA@Data
motorVars_M1.speed_Hz	float	80.0597916	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	-1.86383724	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.0245006047	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	0.653951049	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	2.68207002	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	2.68207002	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.00926135667	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.00926135667	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0628318563	0x00008796@Data
motorVars_M1.Rs_Ohm	float	2.68207002	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008986@Data

Figure 3-30. Build Level 2: Expressions Window at Run Time



	Value	Mean	Min	Max	Std Dev
1 Frequency	15.05kHz	15.05k	15.05k	15.05k	0.000
1 +Duty	80.93 %	80.93	80.93	80.93	0.000
2 +Duty	7.677 %	7.677	7.677	7.677	0.000
3 +Duty	63.56 %	63.56	63.56	63.56	0.000
4 RMS	3.74 A	3.74	3.74	3.74	0.00

Figure 3-31. Build Level 2: Motor Phase Voltage and Current

3.3.4.3 Build Level 3: Closed Current Loop Check

Objectives learned in this build level:

- Evaluate the closed current loop of motor operation.

In this build level, the motor is controlled using i/f control that the rotor angle is generated from ramp generator module. Figure 3-32 shows the software flow for this build level.

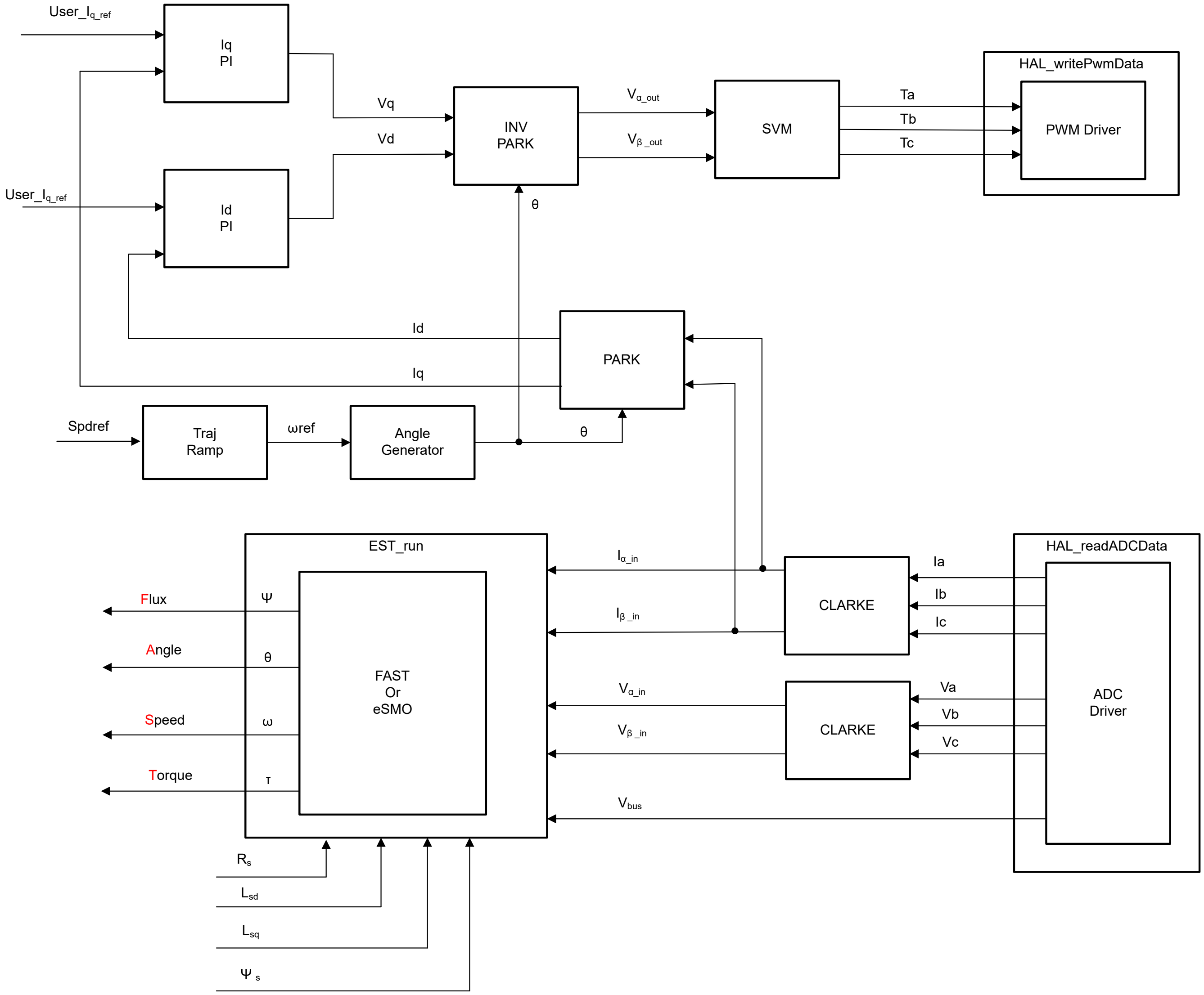


Figure 3-32. Control Software Block Diagram: Build Level 3 – Current Close-Loop Control

3.3.4.3.1 Start CCS and Open Project

To start CCS and open the project, complete the following steps:

1. Connect a programmable, isolated AC power supply capable of providing universal AC input up to 750 W to the input terminals (connector J5) of the reference board as shown in Figure 3-29. Set the power supply current limit to 2 A and the output frequency to 50/60 Hz. Do not turn the power supply on at this time.
2. Connect a motor (compressor) to J10.
3. Follow Steps 2 and 3 of Section 3.3.4.1.1 to open the project.

3.3.4.3.2 Build and Load Project

To build and load the project, complete the following steps:

1. Open the sys_settings.h file, set DMC_BUILDLEVEL to DMC_LEVEL_3.

2. Follow Steps 2 to 4 of Section 3.3.4.1.2 to build the project and load code into controller.

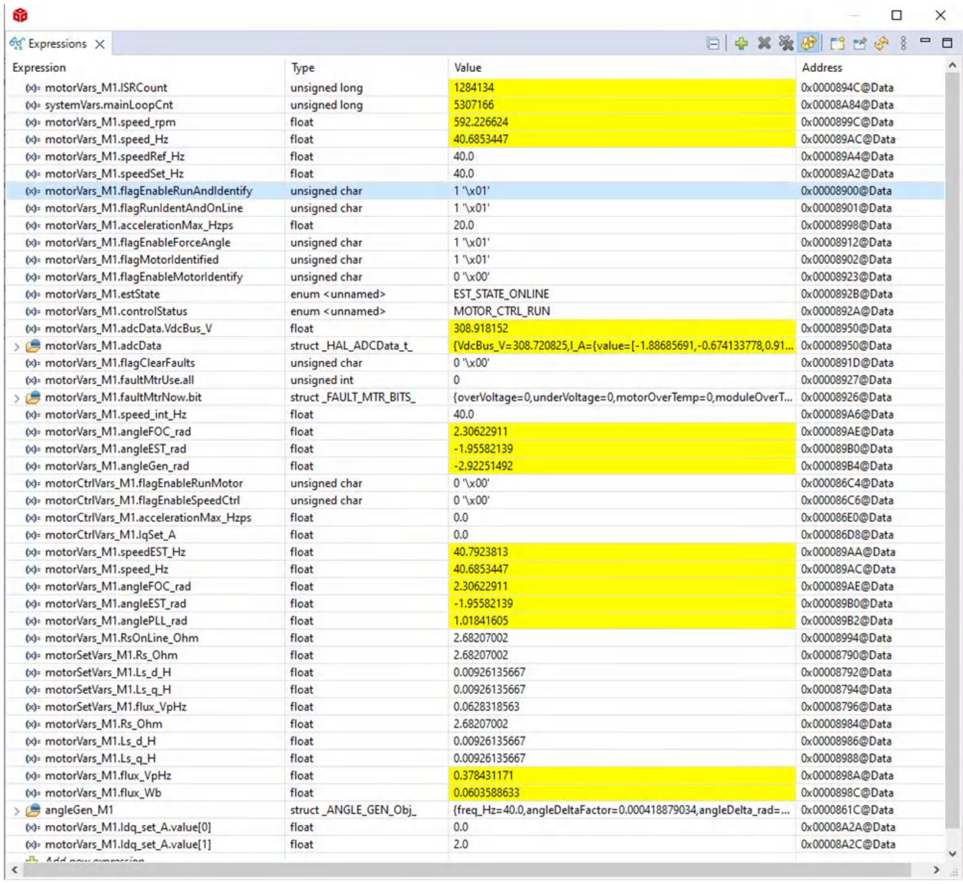
3.3.4.3.3 Setup Debug Environment Windows

Follow Steps 1 to 4 of Section 3.3.4.1.3 to import the variables into the *Expressions* window by picking *BuildLevel3.txt*. The *Expressions* window appears as shown in Figure 3-30.

3.3.4.3.4 Run the Code

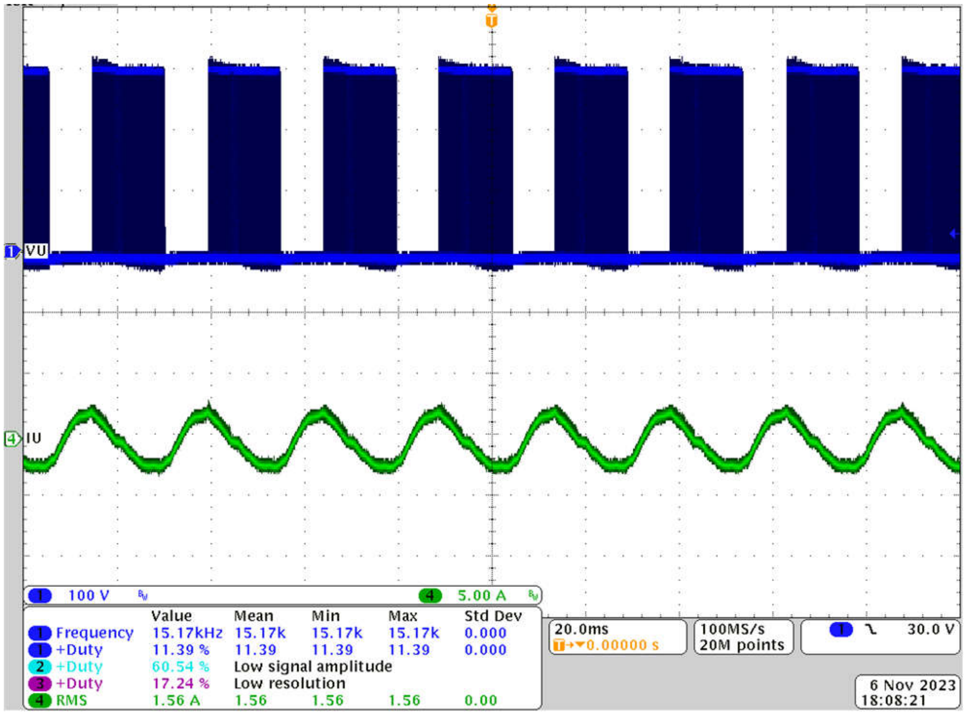
To run the code, complete the following steps:

1. Set the AC source output to 0 V at 50/60 Hz, turn on the AC power supply, slowly increase the input voltage from 0-V to 220-V AC.
2. Run the project by clicking the button, or click *Run* → *Resume* in the *Debug* tab. Set *systemVars.flagEnableSystem* to "1" after a fixed time, that means the offsets calibration has been done and the power relay for inrush is turned on. The motor fault flags for *motorVars_M1.faultMtrUse.all* need to equal to "0" , if the values do not, check the current and voltage sensing circuit as described in Section 3.3.4.1.
3. To verify current closed-loop control for motor, set the variable *motorVars_M1.flagEnableRunAndIdentify* to "1" in the *Expressions* window as shown in Figure 3-33. The motor needs to run with a closed-loop control using the angle from the angle generator at a setting speed in the variable *motorVars_M1.speedRef_Hz,* check the value of *motorVarsM1.speed_Hz* in Expressions window, both variables value need to be very close.
4. The motor current Iq can be set and changed with *motorVars_M1.Idq_Set_A.value[1]*
5. Connect oscilloscope probes to IPM output to watch the motor phase voltage and current as shown in Figure 3-34. Change the *Idq_set_A[0].value[1]* in the *Expressions* window, the motor phase current needs to be increasing accordingly.
6. If the motor cannot run with current-closed loop and appears to experience an overcurrent fault, check if the sign of *adcData[0].current_sf* and the value of *userParams[0].current_sf* are set correctly according to the hardware board.
7. The controller can now be halted before setting the *motorVars_M1.flagEnableRunAndIdentify* to "0", and the debug connection terminated. Fully halting the controller by first clicking the *Halt* button on the toolbar or by clicking *Target* → *Halt*. Finally, reset the controller by clicking on or clicking *Run* → *Reset*.
8. Close CCS debug session by clicking on *Terminate Debug Session* or clicking *Run* → *Terminate*.



Expression	Type	Value	Address
motorVars_M1.ISRCount	unsigned long	1284134	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	5307166	0x00008A84@Data
motorVars_M1.speed_rpm	float	592.226624	0x0000899C@Data
motorVars_M1.speed_Hz	float	40.6853447	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	40.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	40.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	1 '\x01'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\x01'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	20.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\x01'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\x01'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ONLINE	0x0000892B@Data
motorVars_M1.controlStatus	enum <unnamed>	MOTOR_CTRL_RUN	0x0000892A@Data
motorVars_M1.adcData.VdcBus_V	float	308.918152	0x00008950@Data
motorVars_M1.adcData	struct _HAL_ADCData_t_	{VdcBus_V=308.720825,I_A={value=[-1.88685691,-0.674133778,0.91...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUse.all	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNow.bit	struct _FAULT_MTR_BITS_	{overVoltage=0,underVoltage=0,motorOverTemp=0,moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	40.0	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	2.30622911	0x000089AE@Data
motorVars_M1.angleEST_rad	float	-1.95582139	0x000089B0@Data
motorVars_M1.angleGen_rad	float	-2.92251492	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E0@Data
motorCtrlVars_M1.IqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	40.7923813	0x000089AA@Data
motorVars_M1.speed_Hz	float	40.6853447	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	2.30622911	0x000089AE@Data
motorVars_M1.angleEST_rad	float	-1.95582139	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	1.01841605	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	2.68207002	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	2.68207002	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.00926135667	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.00926135667	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0628318563	0x00008796@Data
motorVars_M1.Rs_Ohm	float	2.68207002	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008986@Data
motorVars_M1.Ls_q_H	float	0.00926135667	0x00008988@Data
motorVars_M1.flux_VpHz	float	0.378431171	0x0000898A@Data
motorVars_M1.flux_Wb	float	0.0603588633	0x0000898C@Data
angleGen_M1	struct _ANGLE_GEN_Obj_	{freq_Hz=40.0,angleDeltaFactor=0.000418879034,angleDelta_rad=...	0x0000861C@Data
motorVars_M1.Idq_set_A.value[0]	float	0.0	0x00008A2A@Data
motorVars_M1.Idq_set_A.value[1]	float	2.0	0x00008A2C@Data

Figure 3-33. Build Level 3: Expressions Window at Run Time



	Value	Mean	Min	Max	Std Dev
1 Frequency	15.17kHz	15.17k	15.17k	15.17k	0.000
1 +Duty	11.39 %	11.39	11.39	11.39	0.000
2 +Duty	60.54 %	Low signal amplitude			
3 +Duty	17.24 %	Low resolution			
4 RMS	1.56 A	1.56	1.56	1.56	0.00

Figure 3-34. Build Level 3: Motor Current Under 2-A I_Q Setting

3.3.4.4 Build Level 4: Full Motor Drive Control

Objectives learned in this build level:

- Evaluate the complete motor drive
- Evaluate the additional features, field weakening control for motor
- Evaluate the completed system

In this build level, the outer speed loop is closed with the inner current loop for the motor such that the rotor angle is from the FAST or eSMO estimator module. Figure 3-35 shows the software flow for this build level.

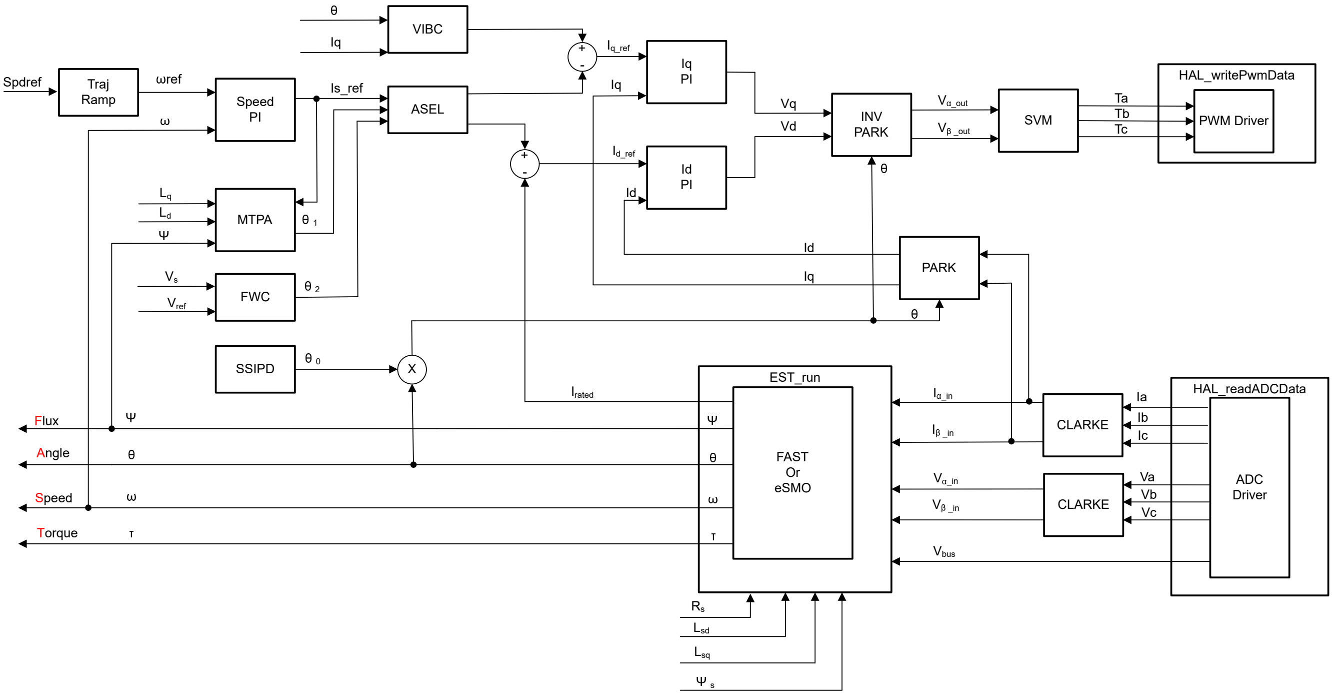


Figure 3-35. Control Software Block Diagram: Build Level 4 – Speed and Current Close-Loop Control

3.3.4.4.1 Start CCS and Open Project

To start CCS and open the project, complete the following steps:

1. Connect a programmable, isolated AC source capable of providing universal AC input up to 750 W to the input terminals (connector J5) of the reference board as shown in Figure 3-29. Set the AC source current limit to 8 A. Do not turn the power supply on at this time.
2. Connect a motor (compressor) to J10.
3. Follow Steps 2 and 3 of Section 3.3.4.1.1 to open the project.

3.3.4.4.2 Build and Load Project

To build and load the project, complete the following steps:

1. Open the sys_settings.h file, set DMC_BUILDLEVEL to DMC_LEVEL_4.
2. Follow Steps 2 to 4 of Section 3.3.4.1.2 to build the project and load code into controller.

3.3.4.4.3 Setup Debug Environment Windows

Follow Steps 1 to 4 of Section 3.3.4.1.3 to import the variables into the *Expressions* window by picking *BuildLevel4.txt*. The *Expressions* window appears as shown in Figure 3-30.

3.3.4.4.4 Run the Code

To run the code, complete the following steps:

1. Set the AC source output to 0 V at 50/60 Hz, turn on the AC power supply, slowly increase the input voltage from 0-V to 220-V AC.

2. The required motor parameters must be recorded in the header files (*user_mtr1.h*) as shown in the following example codes. If the motor parameters are not well known, the motor identification can be used to achieve the motor parameters if the FAST estimator is implemented in the reference design.

```
#define USER_MOTOR1_Rs_Ohm                    (2.68207002f)
#define USER_MOTOR1_Ls_d_H                    (0.00926135667f)
#define USER_MOTOR1_Ls_q_H                    (0.00926135667f)
#define USER_MOTOR1_RATED_FLUX_VpHz           (0.381890297f)
```

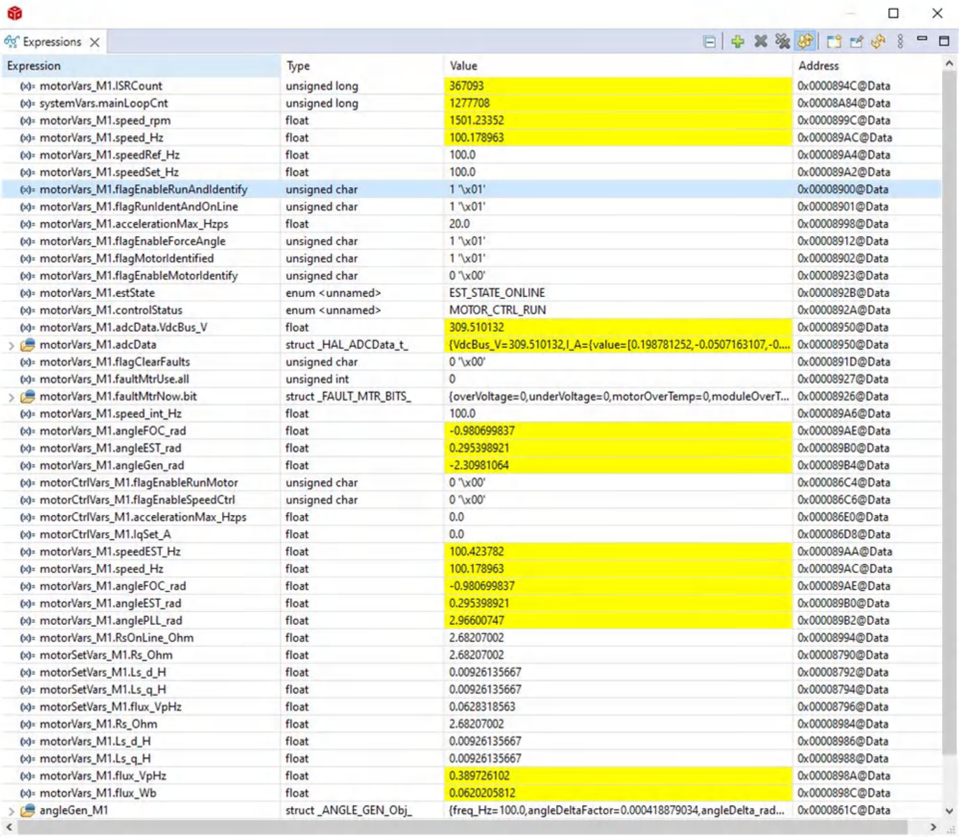
3. Change the *userParams_M1.flag_bypassMotorId* value to "false" to enable the motor identification as the following example code for motor.

```
// true->enable identification, false->disable identification
    userParams[MTR_1].flag_bypassMotorId = false;
```

4. Set the right identification variables value in the *user_mtr1.h* according to the specification of the motor.

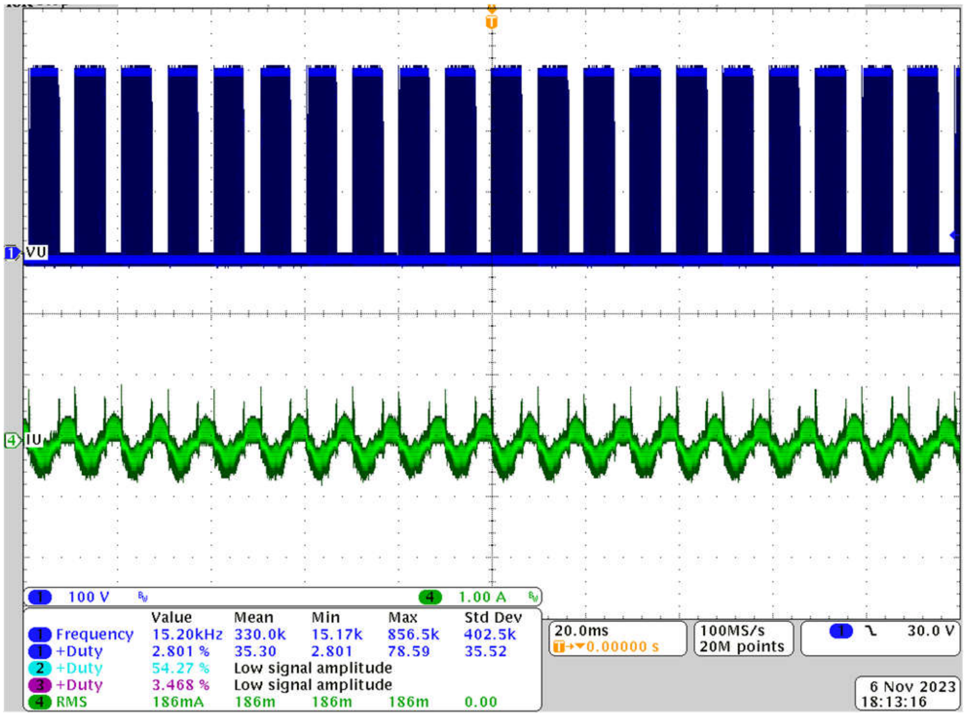
```
 #define USER_MOTOR1_RES_EST_CURRENT_A         (1.0f)      // A - 10~30% of rated current of the
motor
#define USER_MOTOR1_IND_EST_CURRENT_A         (-1.0f)     // A - 10~30% of rated current of the
motor, just enough to enable rotation
#define USER_MOTOR1_MAX_CURRENT_A             (6.5f)      // A - 30~150% of rated current of the
motor
#define USER_MOTOR1_FLUX_EXC_FREQ_Hz          (40.0f)     // Hz - 10~30% of rated frequency of
the motor
```

5. Rebuild the project and load the code into the controller, run the project by clicking on the ▶ button, or click *Run* → *Resume* in the *Debug* tab. The *systemVars.flagEnableSystem* needs to be set to "1" after a fixed time, that means the offsets calibration have been done and the power relay for inrush is turned on. The motor fault flags *motorVars_M1.faultMtrUse.all* need to be equal to "0", if not, check the current and voltage sensing circuit as described in Section 3.3.4.1.
6. Set the variable *motorVars_M1.flagEnableRunAndIdentify* to "1" in the *Expressions* window as shown in Figure 3-36, the motor identification can be executed, the whole process takes about 150 s. Once *motorVars_M1.flagEnableRunAndIdentify* is equal to "0", the motor parameters have been identified. Record the watch window values with the newly-defined motor parameters in user_mtr1.h as follows:
    - USER_MOTOR1_Rs = motorVars_M1.Rs_Ohm's value
    - USER_MOTOR1_Ls_d = motorVars_M1.Ls_d_H's value
    - USER_MOTOR1_Ls_q = motorVars_M1.Ls_q_H's value
    - USER_MOTOR_RATED_FLUX = motorVars_M1.flux_VpHz's value
7. Set both *userParams_M1.flag_bypassMotorId* to "true" after successfully identify the motors parameters, rebuild the project and load the code into the controller.
    - Set the variables *motorVars_M1.flagEnableRunAndIdentify* equal to "1" again for starting to run the motor.
    - Set the variables *motorVars_M1.speedRef_Hz* to a different value and watch how the motor shaft speed follows.
    - To change the acceleration, enter a different acceleration value for the variables *motorVars_M1.accelerationMax_Hzps* and *motorVars_M1.accelerationMax_Hzps*.
8. The controller can now be halted before setting the *motorVars_M1.flagEnableRunAndIdentify* to "0", and the debug connection terminated. Fully halting the controller by first clicking the *Halt* button ⏸ on the toolbar or by clicking *Target* → *Halt*. Finally, reset the controller by clicking on or clicking *Run* → *Reset*.
9. Close the CCS debug session by clicking on *Terminate Debug Session* ■ or clicking *Run* → *Terminate*.



Expression	Type	Value	Address
motorVars_M1.ISRCount	unsigned long	367093	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	1277708	0x00008A84@Data
motorVars_M1.speed_rpm	float	1501.23352	0x0000899C@Data
motorVars_M1.speed_Hz	float	100.178963	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	100.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	100.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	1 '\x01'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\x01'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	20.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\x01'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\x01'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ONLINE	0x0000892B@Data
motorVars_M1.controlStatus	enum <unnamed>	MOTOR_CTRL_RUN	0x0000892A@Data
motorVars_M1.adcData.VdcBus_V	float	309.510132	0x00008950@Data
motorVars_M1.adcData	struct _HAL_ADCData_t_	{VdcBus_V=309.510132,I_A={value=[0.198781252,-0.0507163107,-0....	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUse.all	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNow.bit	struct _FAULT_MTR_BITS_	{overVoltage=0,underVoltage=0,motorOverTemp=0,moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	100.0	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	-0.980699837	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.295398921	0x000089B0@Data
motorVars_M1.angleGen_rad	float	-2.30981064	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E0@Data
motorCtrlVars_M1.IqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	100.423782	0x000089AA@Data
motorVars_M1.speed_Hz	float	100.178963	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	-0.980699837	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.295398921	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	2.96600747	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	2.68207002	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	2.68207002	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.00926135667	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.00926135667	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0628318563	0x00008796@Data
motorVars_M1.Rs_Ohm	float	2.68207002	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008986@Data
motorVars_M1.Ls_q_H	float	0.00926135667	0x00008988@Data
motorVars_M1.flux_VpHz	float	0.389726102	0x0000898A@Data
motorVars_M1.flux_Wb	float	0.0620205812	0x0000898C@Data
angleGen_M1	struct _ANGLE_GEN_Obj_	{freq_Hz=100.0,angleDeltaFactor=0.000418879034,angleDelta_rad...	0x0000861C@Data

Figure 3-36. Build Level 4: Expressions Window at Run Time



	Value	Mean	Min	Max	Std Dev
1 Frequency	15.20kHz	330.0k	15.17k	856.5k	402.5k
1 +Duty	2.801 %	35.30	2.801	78.59	35.52
2 +Duty	54.27 %	Low signal amplitude			
3 +Duty	3.468 %	Low signal amplitude			
4 RMS	186mA	186m	186m	186m	0.00

Figure 3-37. Build Level 4: Rotor Angle, Phase Current of Motor

3.3.4.4.5 Tuning Motor Drive FOC Parameters

The sliding mode current observer consists of a model-based current observer and a bang-bang control generator driven by error between estimated motor currents and actual motor currents. The F and G parameters are calculated based on the motor parameters Rs, and Ls as described in Section 2.4.2.2.1.2. The observer gain k for bang-bang control, the cutoff frequency for LPF, and the K_p and K_i for PLL angle tracker must be tuned according to the testing state, and try to get the best parameters. The user can run the FAST estimator and eSMO in parallel to validate the angle from the eSMO for tuning the parameters. The initial parameters are defined in the user-mtr1.h files.

```
// Only for eSMO
#define USER_MOTOR1_KSLIDE_MAX              (1.50f)
#define USER_MOTOR1_KSLIDE_MIN              (0.75f)

#define USER_MOTOR1_PLL_KP_MAX              (10.0f)
#define USER_MOTOR1_PLL_KP_MIN              (2.0f)
#define USER_MOTOR1_PLL_KP_SF               (5.0f)

#define USER_MOTOR1_BEMF_THRESHOLD          (0.5f)
#define USER_MOTOR1_BEMF_KSLF_FC_Hz         (2.0f)
#define USER_MOTOR1_THETA_OFFSET_SF         (1.0f)
#define USER_MOTOR1_SPEED_LPF_FC_Hz         (200.0f)
```

The speed and current PI regulator gains are calculated according to the motor parameters, the user can tune these gains online to optimize the control performance of the system.

- Adding the motorVars[0].Kp_spd, motorVars[0].Ki_spd, motorVars[0].Kp_Iq, motorVars[0].Ki_Iq, motorVars[0].Kp_Id, and motorVars[0].Ki_Id to the *Expressions* window in CCS Debug Perspective. Change the PI gains for the compressor motor drive and record the values.

3.3.4.4.6 Tuning Field Weakening and MTPA Control Parameters

The FWC and MTPA functions are added and called in the motor drive ISR to calculate current angle, and then compute the reference currents of the d-axis and q-axis.

1. Adding the pre-define symbols *MOTOR1_FWC* and *MOTOR1_MTPA* in the build configuration of the project as described in Section 3.3.2 for enabling the FWC and MTPA, respectively.
2. In the *user_mtr1.h* file, make sure the motor parameters are known and correctly set. In *mtpa.h*, make sure the tables are set properly for and calculations are set according to the specification of the motor.
3. Add the variables VsRef_pu, Kp_fwc, and Ki_fwc to the *Expressions* window in CCS Debug Perspective, and tune these parameters to achieve the expected performance for the field weakening control according to the motor and the system.
4. After tuning and fixing these variables, record the watch window values with the newly-defined parameters in *user_mtr1.h* file.

   USER_M1_FWC_VREF = VsRef_pu's value. The factor of the reference voltage for Field Weakening Control.

   USER_M1_FWC_KP = Kp_fwc's value. The Kp gain of PI regulator for Field Weakening Control

   USER_M1_FWC_KI = Ki_fwc's value. The Ki gain of PI regulator for Field Weakening Control
5. MTPA control parameters are calculated according to the motor parameters, L_d , L_q , and ψ_m , so there are not any additional parameters to be tuned online.

3.3.4.4.7 Tuning Current Sensing Parameters

Accurate current sensing is important to estimate the rotor angle and speed, and also have the best dynamic motor control. The current sensing parameters must match the hardware by setting the following related parameters:

- Dead-band time, the rising edge delay time must be greater than (high-side turn on time) + (low side turn-off time) of the power module, and the falling edge delay time must be greater than (high-side turn-off time) +

(low-side turn-on time) of the power module as shown in the following setting for a power module used in the reference design.

```
//! \brief Defines the PWM deadband falling edge delay count (system clocks)
#define MTR1_PWM_DBFED_CNT (uint16_t)(2.5f * 120.0f)  // 2.5us, (>2.0us)

//! \brief Defines the PWM deadband rising edge delay count (system clocks)
#define MTR1_PWM_DBRED_CNT (uint16_t)(2.5f * 120.0f)  // 2.50us, (>2.0us)
```

- Minimum duration of pulse width PWM, specifies to be greater than (Hardware delay time + Dead band time + Ringing duration + ADC sampling time).

```
//! \brief Defines the minimum duration, Clock Cycle
#define USER_M1_DCLINKSS_MIN_DURATION   (450U)
```

- Sample and hold delay time, specifies the time delay from PWM output to ADC sample time for current sensing. The delay time is dependent on the hardware and includes the propagation delay of the gate driver circuit and turn on and turn off delay of the power FET, and is less than or equal to (Minimum duration – ADC sampling time).

```
//! \brief Defines the sample delay, Clock Cycle

#define USER_M1_DCLINKSS_SAMPLE_DELAY   (430U)
```

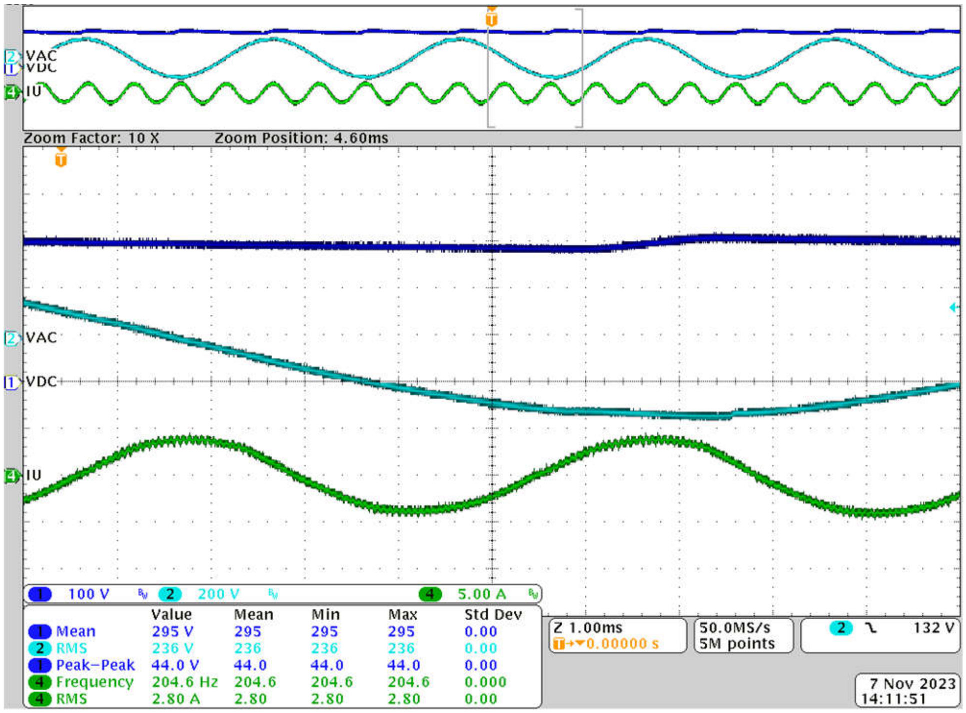
3.4 Test Results

The following sections show the test data from characterizing the design. The test results are divided in multiple sections that cover the steady-state performance and data, functional performance waveforms, and transient performance waveforms of the fan and compressor motor.

3.4.1 Load and Thermal Test

Figure 3-38 is a waveform at 3000 RPM (200 Hz) under 500-W dyno load. The waveform includes the following display:

- CH1 (Blue): DCBUS voltage
- CH2 (Light Blue): AC Input Voltage
- CH4 (Green): Current of phase U

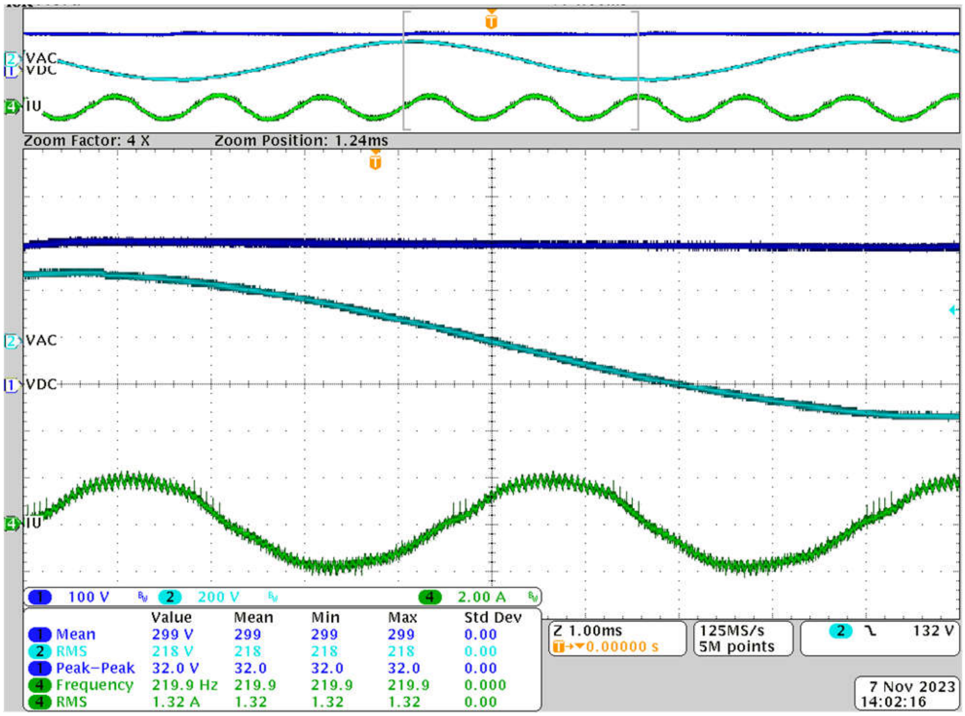


	Value	Mean	Min	Max	Std Dev
1 Mean	295 V	295	295	295	0.00
2 RMS	236 V	236	236	236	0.00
1 Peak−Peak	44.0 V	44.0	44.0	44.0	0.00
4 Frequency	204.6 Hz	204.6	204.6	204.6	0.000
4 RMS	2.80 A	2.80	2.80	2.80	0.00

Figure 3-38. Phase Current and Voltage Waveforms of Motor at 500 W, 200 Hz

Figure 3-38 shows a waveform at 3300 RPM (220 Hz) under 300-W dyno load with field weakening enabled. The motor tested is rated at 3000 RPM (200 Hz) and now works at field-weakening status.

- CH1 (Blue): DCBUS voltage
- CH2 (Light Blue): AC Input Voltage
- CH4 (Green): Current of phase U



	Value	Mean	Min	Max	Std Dev
1 Mean	299 V	299	299	299	0.00
2 RMS	218 V	218	218	218	0.00
1 Peak−Peak	32.0 V	32.0	32.0	32.0	0.00
4 Frequency	219.9 Hz	219.9	219.9	219.9	0.000
4 RMS	1.32 A	1.32	1.32	1.32	0.00

Figure 3-39. Field Weakening Test at 300 W, 220 Hz

This board is designed to work at 750 W for a short amount of time (≤ 1 minute), pay attention to rising temperatures. If running the board at high power or for a long time, use an external cooling fan to cool down the heat sink. Figure 3-38 shows the board temperature rising at 500 W, 3000 RPM (200 Hz).

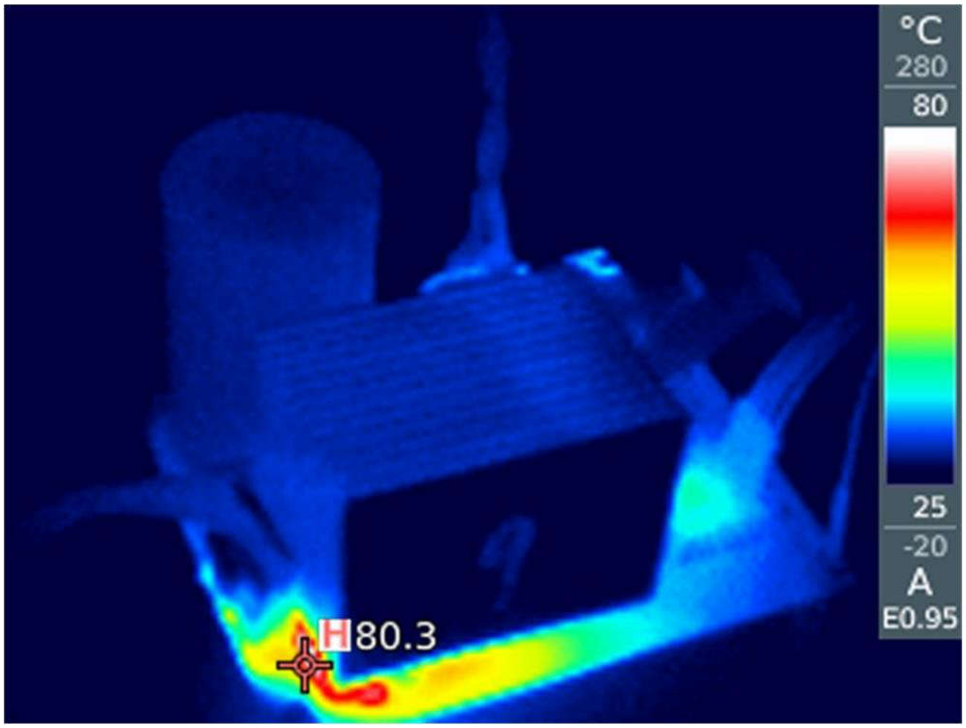
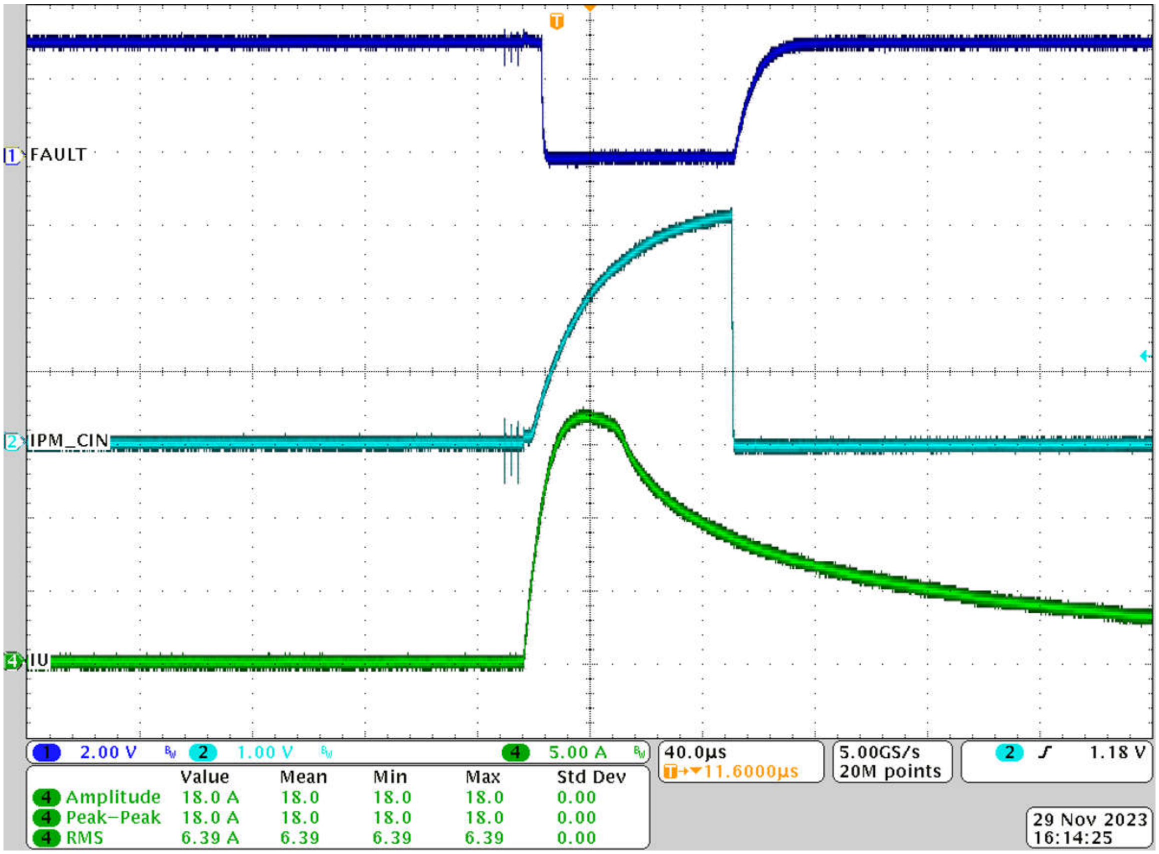


Figure 3-40. Thermal Test Under 220 VAC, 500 W, 200 Hz

3.4.2 Overcurrent Protection by External Comparator

As explained in Section 2.4.1.6, there is a comparator U10 for external overcurrent protection. Figure 3-41 shows the eternal overcurrent protection waveform. Output (net IPM_CIN) of U10 is high when current on R80 exceeds the reference point set by negative input of U10, high-level IPM_CIN then triggers IPM fault protect to output a low-level signal at IPM_FAULT, which is connected to the microcontroller.

- CH1 (Blue):IPM_FAULT
- CH2 (Light Blue): IPM_CIN
- CH4 (Green): Current of R80



	Value	Mean	Min	Max	Std Dev
4 Amplitude	18.0 A	18.0	18.0	18.0	0.00
4 Peak−Peak	18.0 A	18.0	18.0	18.0	0.00
4 RMS	6.39 A	6.39	6.39	6.39	0.00

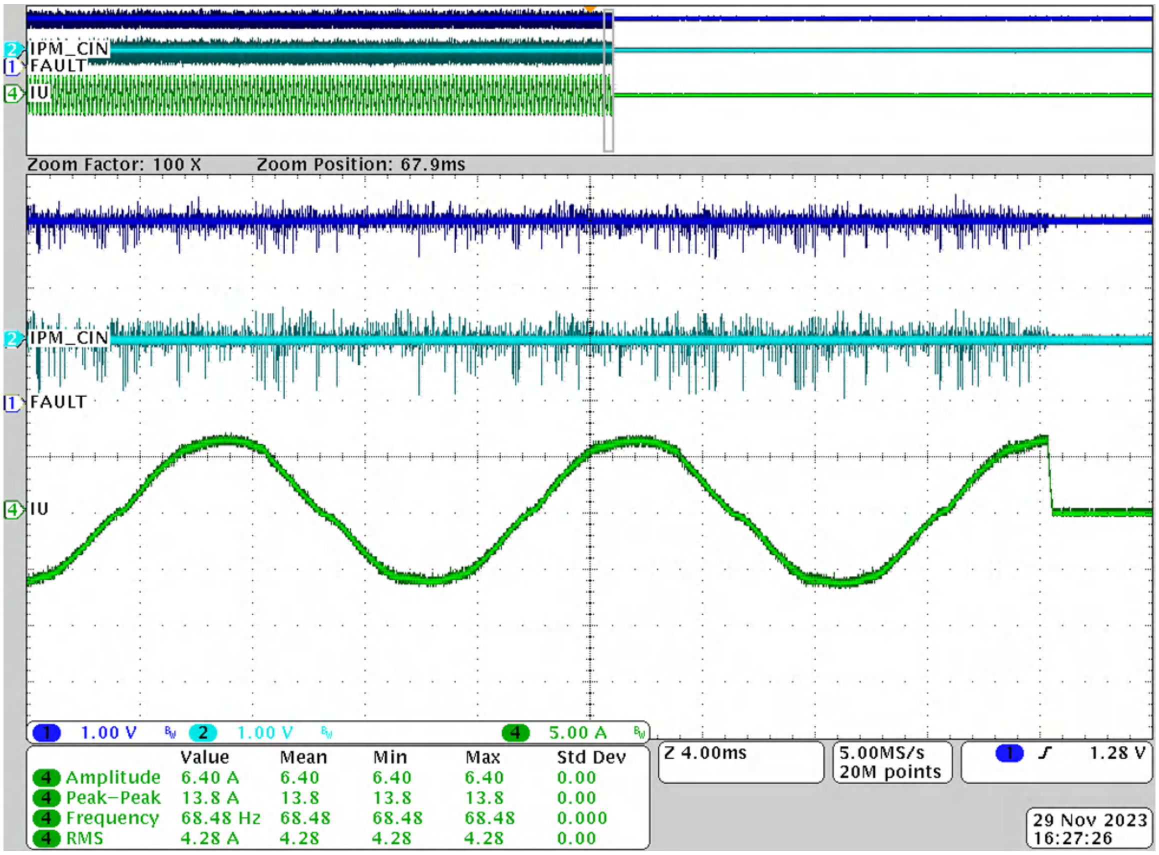
Figure 3-41. Overcurrent Protection by External Comparator

3.4.3 Overcurrent Protection by Internal CMPSS

As explained in Section 2.4.1.7, the internal CMPSS can be configured for overcurrent protection. Figure 3-42 shows the internal overcurrent protection waveform, is triggered by internal CMPSS, since both IPM_FALUT and IPM_CIN are not triggered. Overcurrent can be set with the following codes.

*objSets->maxPeakCurrent_A = USER_M1_ADC_FULL_SCALE_CURRENT_A * 0.4975f;*

- CH1 (Blue):IPM_FAULT
- CH2 (Light Blue): IPM_CIN
- CH4 (Green): Current of phase U



	Value	Mean	Min	Max	Std Dev
4 Amplitude	6.40 A	6.40	6.40	6.40	0.00
4 Peak−Peak	13.8 A	13.8	13.8	13.8	0.00
4 Frequency	68.48 Hz	68.48	68.48	68.48	0.000
4 RMS	4.28 A	4.28	4.28	4.28	0.00

Figure 3-42. Overcurrent Protection by Internal CMPSS

3.5 Migrate Firmware to a New Hardware Board

If the designer wants to migrate the reference design to a hardware board, change the motor related PWM, CMPSS, ADC peripherals configuration, hardware parameters, and motor parameters, accordingly in the *hal.c*, *hal.h*, and *user_mtr1.h* files as described in the following sections.

3.5.1 Configure the PWM, CMPSS, and ADC Modules

The application parameters to control the motor are written as `#define` configuring the PWM, CMPSS, and ADC modules base address in *hal.h* according to the hardware. The PWM, CMPSS, and ADC of the compressor motor defines are shown in the following codes.

Configure PWM and CMPSS base address for motor drive:

```
// EPWM
#define MTR1_PWM_U_BASE         EPWM2_BASE
#define MTR1_PWM_V_BASE         EPWM3_BASE
#define MTR1_PWM_W_BASE         EPWM4_BASE

// CMPSS->Iu/Iv/Iw
#define MTR1_CMPSS_U_BASE       CMPSSLITE4_BASE
#define MTR1_CMPSS_V_BASE       CMPSSLITE2_BASE
#define MTR1_CMPSS_W_BASE       CMPSSLITE3_BASE
```

Configure ADC base address and channels for motor drive:

```
// Three shunts
// Using ADCA/ADCC for current sensing
#define MTR1_ADC_TRIGGER_SOC     ADC_TRIGGER_EPWM2_SOCA  // EPWM2_SOCA

#define MTR1_ADC_I_SAMPLEWINDOW     14
#define MTR1_ADC_V_SAMPLEWINDOW     20
```

```
#define MTR1_IU_ADC_BASE    ADCC_BASE       // ADCC-A7/C3*/CMP4  -SOC1
#define MTR1_IV_ADC_BASE    ADCA_BASE       // ADCA-A4*/C14/CMP2 -SOC2
#define MTR1_IW_ADC_BASE    ADCA_BASE       // ADCA-A0*/C15/CMP3 -SOC1

#define MTR1_IU_ADCRES_BASE     ADCCRESULT_BASE         // ADCC-A7/C3*
#define MTR1_IV_ADCRES_BASE     ADCARESULT_BASE         // ADCA-A4*/C14
#define MTR1_IW_ADCRES_BASE     ADCARESULT_BASE         // ADCA-A0*/C15

#define MTR1_IU_ADC_CH_NUM      ADC_CH_ADCIN3           // ADCC-A7/C3*
#define MTR1_IV_ADC_CH_NUM      ADC_CH_ADCIN4           // ADCA-A4*/C14
#define MTR1_IW_ADC_CH_NUM      ADC_CH_ADCIN0           // ADCA-A0*/C15

#define MTR1_IU_ADC_SOC_NUM  ADC_SOC_NUMBER1  // ADCC-A7/C3*  -SOC1-PPB1
#define MTR1_IV_ADC_SOC_NUM  ADC_SOC_NUMBER2  // ADCA-A4*/C14 -SOC2-PPB2
#define MTR1_IW_ADC_SOC_NUM  ADC_SOC_NUMBER1  // ADCA-A0*/C15 -SOC1-PPB1

#define MTR1_IU_ADC_PPB_NUM  ADC_PPB_NUMBER1  // ADCC-A7/C3*  -SOC1-PPB1
#define MTR1_IV_ADC_PPB_NUM  ADC_PPB_NUMBER2  // ADCA-A4*/C14 -SOC2-PPB2
#define MTR1_IW_ADC_PPB_NUM  ADC_PPB_NUMBER1  // ADCA-A0*/C15 -SOC1-PPB1
```

Configure peripheral interrupt for motor drive control:

```
// Interrupt
#define MTR1_PWM_INT_BASE  MTR1_PWM_U_BASE       // EPWM1
#define MTR1_ADC_INT_BASE  ADCC_BASE             // ADCC-A11/C0*-SOC4
#define MTR1_ADC_INT_NUM   ADC_INT_NUMBER1       // ADCC_INT1   -SOC4
#define MTR1_ADC_INT_SOC   ADC_SOC_NUMBER4       // ADCC_INT1   -SOC4

#define MTR1_PIE_INT_NUM   INT_ADCC1             // ADCC_INT1   -SOC4
#define MTR1_CPU_INT_NUM   INTERRUPT_CPU_INT1    // ADCC_INT1-CPU_INT1
#define MTR1_INT_ACK_GROUP INTERRUPT_ACK_GROUP1  // ADCC_INT1-CPU_INT1
```

Configure the connections between the ADC pin and CMPSS modules in *hal.h* based on the hardware, the details refer to the Table, Analog Pins, and Internal Connections in the TMS320F280013x Real-Time Microcontrollers Technical Reference Manual

```
// CMPSS->Iu/Iv/Iw
#define MTR1_CMPSS_U_BASE       CMPSSLITE4_BASE
#define MTR1_CMPSS_V_BASE       CMPSSLITE2_BASE
#define MTR1_CMPSS_W_BASE       CMPSSLITE3_BASE

#define MTR1_IU_CMPHP_SEL  ASYSCTL_CMPHPMUX_SELECT_4  // CMPSS4H-A7/C3*
#define MTR1_IU_CMPLP_SEL  ASYSCTL_CMPLPMUX_SELECT_4  // CMPSS4L-A7/C3*
#define MTR1_IV_CMPHP_SEL  ASYSCTL_CMPHPMUX_SELECT_2  // CMPSS2H-A4*/C14
#define MTR1_IV_CMPLP_SEL  ASYSCTL_CMPLPMUX_SELECT_2  // CMPSS2L-A4*/C14

#define MTR1_IW_CMPHP_SEL  ASYSCTL_CMPHPMUX_SELECT_3  // CMPSS3H-A0*/C15
#define MTR1_IW_CMPLP_SEL  ASYSCTL_CMPLPMUX_SELECT_3  // CMPSS3L-A0*/C15

#define MTR1_IU_CMPHP_MUX       1                     // CMPSS4H-A7/C3*
#define MTR1_IU_CMPLP_MUX       1                     // CMPSS4L-A7/C3*

#define MTR1_IV_CMPHP_MUX       0                     // CMPSS2H-A4*/C14
#define MTR1_IV_CMPLP_MUX       0                     // CMPSS2L-A4*/C14

#define MTR1_IW_CMPHP_MUX       2                     // CMPSS3H-A0*/C15
#define MTR1_IW_CMPLP_MUX       2                     // CMPSS3L-A0*/C15
```

Configure the trip signals from CMPSS to be passed to EPWM and GPIO output in *hal.h* based on the hardware, the details refer to Table, ePWM X-BAR MUX Configuration Table and Table, OUTPUT X-BAR MUX Configuration Table in TMS320F280013x Real-Time Microcontrollers Technical Reference Manual.

```
// XBAR-EPWM
#define MTR1_XBAR_TRIP_ADDRL    XBAR_O_TRIP8MUX0TO15CFG
#define MTR1_XBAR_TRIP_ADDRH    XBAR_O_TRIP8MUX16TO31CFG

#define MTR1_XBAR_INPUT1        XBAR_INPUT1
#define MTR1_TZ_OSHT1           EPWM_TZ_SIGNAL_OSHT1

#define MTR1_XBAR_TRIP          XBAR_TRIP8
#define MTR1_DCTRIPIN           EPWM_DC_COMBINATIONAL_TRIPIN8y

// XBAR-EPWM->Iu/Iv/Iw
```

```
#define MTR1_IU_XBAR_EPWM_MUX   XBAR_EPWM_MUX06_CMPSS4_CTRIPH_OR_L      // CMPSS4-HP&LP, A7/C3*
#define MTR1_IV_XBAR_EPWM_MUX   XBAR_EPWM_MUX02_CMPSS2_CTRIPH_OR_L      // CMPSS2-HP&LP, A4*/C14
#define MTR1_IW_XBAR_EPWM_MUX   XBAR_EPWM_MUX04_CMPSS3_CTRIPH_OR_L      // CMPSS3-HP&LP, A0*/C15

#define MTR1_IU_XBAR_MUX        XBAR_MUX06     // CMPSS4-HP&LP, A7/C3*
#define MTR1_IV_XBAR_MUX        XBAR_MUX02     // CMPSS2-HP&LP, A4*/C14
#define MTR1_IW_XBAR_MUX        XBAR_MUX04     // CMPSS3-HP&LP, A0*/C15
```

The related ADC channels are used for motor-current sensing which pins are internally connected to the Comparator Subsystem (CMPSS), configure the CMPSS registers in the HAL_setupCMPSSs() function in the *hal.c* file as shown in the following codes. Three CMPSS modules are used to implement positive and negative overcurrent protection of U-phase, V-phase, and W-phase of the motor.

```
void HAL_setupCMPSSsMTR(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;

#if defined(DMCPFC_REV3P2) || defined(DMCPFC_REV3P1)
#if !defined(MOTOR1_DCLINKSS) || !defined(MOTOR2_DCLINKSS)
    uint16_t cmpsaDACH;
#endif  // !(MOTOR1_DCLINKSS || MOTOR2_DCLINKSS)
    uint16_t cmpsaDACL;
    ... ...
#else   // !MOTOR1_DCLINKSS, Three-shunt
        cmpsaDACH = MTR1_CMPSS_DACH_VALUE;
        cmpsaDACL = MTR1_CMPSS_DACL_VALUE;

        ASysCtl_selectCMPHPMux(MTR1_IU_CMPHP_SEL, MTR1_IU_CMPHP_MUX);

        ASysCtl_selectCMPHPMux(MTR1_IV_CMPHP_SEL, MTR1_IV_CMPHP_MUX);

        ASysCtl_selectCMPLPMux(MTR1_IW_CMPLP_SEL, MTR1_IW_CMPLP_MUX);
    ... ...
    return;
} // end of HAL_setupCMPSSs() function
```

The CMPSS-generated signals go to the X-Bar, where signals can be combined in different and unique fashions to flag unique trip events from multiple sources including external TZ signal from IPM #Fault to implement the fault protection. The faults include the overcurrent signals from the CMPSS and the fault indicator output from the power module. Configure the XBAR registers in HAL_setupMtrFaults() function in the *hal.c* file as shown in the following codes.

```
void HAL_setupMtrFaults(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;
    uint16_t cnt;

    // Configure TRIP 7 to OR the High and Low trips from both
    // comparator 5, 3 & 1, clear everything first
    EALLOW;
    HWREG(XBAR_EPWM_CFG_REG_BASE + MTR1_XBAR_TRIP_ADDRL) = 0;
    HWREG(XBAR_EPWM_CFG_REG_BASE + MTR1_XBAR_TRIP_ADDRH) = 0;
    EDIS;
    ... ...
        // What do we want the OST/CBC events to do?
        // TZA events can force EPWMxA
        // TZB events can force EPWMxB
        EPWM_setTripZoneAction(obj->pwmHandle[cnt],
                               EPWM_TZ_ACTION_EVENT_TZA,
                               EPWM_TZ_ACTION_LOW);

        EPWM_setTripZoneAction(obj->pwmHandle[cnt],
                               EPWM_TZ_ACTION_EVENT_TZB,
                               EPWM_TZ_ACTION_LOW);
    ... ...
    // Clear any spurious fault
    EPWM_clearTripZoneFlag(obj->pwmHandle[0], HAL_TZFLAG_INTERRUPT_ALL);
    EPWM_clearTripZoneFlag(obj->pwmHandle[1], HAL_TZFLAG_INTERRUPT_ALL);
    EPWM_clearTripZoneFlag(obj->pwmHandle[2], HAL_TZFLAG_INTERRUPT_ALL);

    return;
}
```

Configure the GPIOs based on the hardware in HAL_setupGPIOs() in the *hal.c* file as shown in the following codes.

```
void HAL_setupGPIOs(HAL_Handle handle)
{
    ... ...
    // GPIO2->EPWM2A->M1_UH
    GPIO_setPinConfig(GPIO_2_EPWM2_A);
    GPIO_writePin(2, 0);
    GPIO_setDirectionMode(2, GPIO_DIR_MODE_OUT);
    GPIO_setPadConfig(2, GPIO_PIN_TYPE_STD);

    // GPIO3->EPWM2B->M1_UL
    GPIO_setPinConfig(GPIO_3_EPWM2_B);
    GPIO_writePin(3, 0);
    GPIO_setDirectionMode(3, GPIO_DIR_MODE_OUT);
    GPIO_setPadConfig(3, GPIO_PIN_TYPE_STD);
    ... ...
    return;
}  // end of HAL_setupGPIOs() function
```

The configuration codes need to be changed in HAL_enableMtrPWM() and HAL_clearMtrFaultStatus() in the *hal.h* file as below marked in **bold** according to the used CMPSS for motor control.

```
static inline void HAL_enableMtrPWM(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;

    obj->flagEnablePWM = true;

#if defined(DMCPFC_REV3P2) || defined(DMCPFC_REV3P1)
    if(obj->motorNum == MTR_1)
    {
#if defined(MOTOR1_DCLINKSS)
        // Clear any comparator digital filter output latch
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[0]);
#else   // !MOTOR1_DCLINKSS
        // Clear any comparator digital filter output latch
        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[0]);

        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[1]);

        CMPSS_clearFilterLatchLow(obj->cmpssHandle[2]);
    ... ...
    return;
} // end of HAL_enableMtrPWM() function
```

```
static inline void HAL_clearMtrFaultStatus(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;
    ... ...
#if defined(HVMTRPFC_REV1P1) || defined(WMINVBRD_REV1P0) || defined(TIDSMPFC_REV3P2)
    // Clear any comparator digital filter output latch
    CMPSS_clearFilterLatchHigh(obj->cmpssHandle[0]);
    CMPSS_clearFilterLatchLow(obj->cmpssHandle[0]);

    CMPSS_clearFilterLatchHigh(obj->cmpssHandle[1]);
    CMPSS_clearFilterLatchLow(obj->cmpssHandle[1]);

    CMPSS_clearFilterLatchHigh(obj->cmpssHandle[2]);
    CMPSS_clearFilterLatchLow(obj->cmpssHandle[2]);
    ... ...
    return;
} // end of HAL_clearMtrFaultStatus() function
```

3.5.2 Setup Hardware Board Parameters

The *user_mtr1.h* file is where all user parameters are stored for motor control. The maximum phase current and phase voltage at the input to the AD converter, these values are hardware-dependent and need to be based on the current and voltage sensing and scaling to the ADC input. The number of current sensors and voltage (phase) sensors used are defined in *user_mtr1.h* that are hardware dependent.

All of the configurable parameters are defined in the *user_mtr1.h* file. These parameters can be calculated using the *Motor_Drive_Parameters_Calculation.xlsx* Microsoft® Excel® spreadsheet. This file is included with the TIDA-010265 archive file at the folder `..\solutions\tida_010265\docs` to calculate these values and copy these parameters marked ***bold*** to *user_mtr1.h* as shown in the following codes.

```
//! \brief Defines the maximum voltage at the AD converter
//  Full scale voltage of AD converter, not the current voltage
#define USER_M1_ADC_FULL_SCALE_VOLTAGE_V         (404.1292683f)

//! \brief Defines the analog voltage filter pole location, Hz
//!
#define USER_M1_VOLTAGE_FILTER_POLE_Hz           (416.3602877f)

//! \brief Defines the maximum current at the AD converter
#define USER_M1_ADC_FULL_SCALE_CURRENT_A         (15.972f)
```

3.5.3 Configure Faults Protection Parameters

Fault management is implemented in this system that includes overcurrent, overvoltage, undervoltage, stall, overload, start-up failed. The faults protection parameters are defined in *user_mtr1.h* as shown in the following codes, which are hardware board, motors, and system dependent.

```
//! \brief motor over current threshold
#define USER_MOTOR1_OVER_CURRENT_A          (6.5f)          // A

//! \brief motor lost phase current threshold
#define USER_M1_LOST_PHASE_CURRENT_A        (0.2f)

//! \brief motor unbalance ratio percent threshold
#define USER_M1_UNBALANCE_RATIO             (0.2f)
```

```
//! \brief DC bus over voltage threshold
#define USER_M1_OVER_VOLTAGE_FAULT_V        (380.0f)

//! \brief DC bus over voltage threshold
#define USER_M1_OVER_VOLTAGE_NORM_V         (350.0f)

//! \brief DC bus under voltage threshold
#define USER_M1_UNDER_VOLTAGE_FAULT_V       (100.0f)
```

3.5.4 Setup Motor Electrical Parameters

The parameters provided in *user_mtr1.h* for PMSM and BLDC motors are listed as shown in the following codes. The motor parameters can be identified if the FAST technique is implemented on this motor or by getting the parameters from the motor data sheet.

```
#define USER_MOTOR1_TYPE                    MOTOR_TYPE_PM
#define USER_MOTOR1_NUM_POLE_PAIRS          (4)
#define USER_MOTOR1_Rr_Ohm                  (0.0f)
#define USER_MOTOR1_Rs_Ohm                  (2.68207002f)
#define USER_MOTOR1_Ls_d_H                  (0.00926135667f)
#define USER_MOTOR1_Ls_q_H                  (0.00926135667f)
#define USER_MOTOR1_RATED_FLUX_VpHz         (0.381890297f)
```

3.6 Getting Started MSPM0 Firmware

Contact the local TI sales representative for firmware for the MSPM0G1507 daughterboard.

4 Design and Documentation Support

4.1 Design Files

4.1.1 Schematics

To download the schematics, see the design files at TIDA-010265.

4.1.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDA-010265.

4.1.3 PCB Layout Recommendations

This reference design is implemented using a PCB with two-layer, 2-oz copper with the bottom-side SMD component placement to save cost and board area. There are several important aspects to remember while designing the PCB. In the following bullets, system-level placement and layout of each block is explained.

- Separate components (or singles) into high and low voltage, high and low current, high and low independence groups. Place and route low-voltage and high-impedance components and signals together, such as microcontroller-related signals and the input side of IPM. Use a whole copper pour to provide an integrated GND plane for these area. AC input, filter, and rectifiers and IPM output sides are high-voltage, high-current and low-impedance parts and signals, route them with wider traces or copper pours to provide high-current paths, and also separate them from above low voltage and high-impedance signals to reduce interference.
- Components in the high power path are kept on the outer edges of the PCB with the minimum distance possible. The microcontroller is placed at the center considering the optimum distance from all the power blocks that need to be controlled. Pin assignment is set to minimize the control signal or feedback signal trace distance and the crossing between analog and digital signals.
- AC Line Protection and EMI Filter
  - AC line protection components are closely placed within the minimum distance to the connection path. Earth connection guarding is provided around protection and EMI filter circuit.
- Motor Drive
  - For the high ripple requirement, the motor drive is placed as close as possible to the film capacitor and DC bus capacitor bank.
  - The low-side shunt resistor method with four-wire sensing is implemented for current sensing. A differential pair with impedance matching resistor is used to connect the sensing signal from the shunt resistors to the op-amp circuit. Shunt resistors are placed near the module with an immediate ground copper plane connection.
- Auxiliary Power Supply
  - The GND of the auxiliary power supply connects the DC bus capacitor bank directly and independently to separate low current from high current and high frequency GND trace for the inverter.

4.1.4 Altium Project

To download the Altium project files, see the design files at TIDA-010265.

4.1.5 Gerber Files

To download the Gerber files, see the design files at TIDA-010265.

4.2 Software Files

Download the Code Composer Studio Integrated Development Environment at CCSTUDIO.

Download the TIDA-010265 hardware-specific software design files from the design files at C2000WARE-MOTORCONTROL-SDK.

4.3 Documentation Support

1. Texas Instruments, *TMS320F280013x Microcontrollers* data sheet
2. Texas Instruments, *TMS320F280013x Real-Time Microcontrollers* technical reference manual
3. Texas Instruments, *InstaSPIN-FOC and InstaSPIN-MOTION* user's guide
4. Texas Instruments, *Motor Control SDK Universal Project and Lab* user's guide
5. Texas Instruments, *C2000™ Software Frequency Response Analyzer (SFRA) Library and Compensation Designer* user's guide
6. Texas Instruments, *Sensorless-FOC for PMSM With Single DC-Link Shunt* application note
7. Texas Instruments, *C2000 SysConfig* application note

4.4 Support Resources

TI E2E™ support forums are an engineer's go-to source for fast, verified answers and design help — straight from the experts. Search existing answers or ask your own question to get the quick design help you need.

Linked content is provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

4.5 Trademarks

FAST™, C2000™, TI E2E™, InstaSPIN™, Code Composer Studio™, and are trademarks of Texas Instruments.
Arm® and Cortex® are registered trademarks of Arm Limited.
Microsoft®, Windows®, and Excel® are registered trademarks of Microsoft Corporation.
All trademarks are the property of their respective owners.

5 About the Author

HELY ZHANG is a System Application Engineer at Texas Instruments, where he is responsible for developing home appliance related power delivery and motor inverters. Hely earned his masters degree from Anhui University of Science and Technology with Power electronics in 2002, and worked in SolarEdge and General Electric before joining TI.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265