

TIEVM-MTR-HVINV 750W High-Voltage Motor Inverter Evaluation Module



Description

The TIEVM-MTR-HVINV is a 750-W development board for high-voltage motor drive applications. This EVM implements sensorless FOC control for a 3-phase PMSM motor with the InstaSPIN-FOC FAST™ and eSMO sensorless observers. The modular design allows for plug-and-play support of different daughterboard attachments to the same motherboard. The hardware and firmware of this EVM are tested and ready-to-use to help accelerate development time, with design details and test results available in this user's guide.

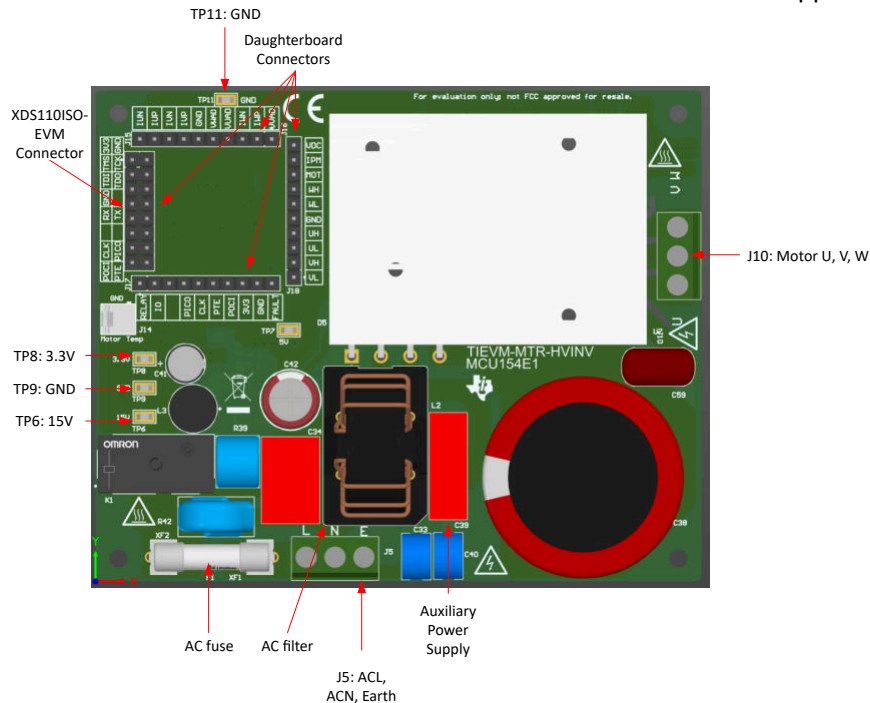
Features

- Wide voltage input operating range: 165 to 265 VAC at 50/60 Hz, or up to 400VDC

- Up to 750-W inverter stage, 15-kHz switching frequency, torque compensation, and automatic field weakening control (FWC)
- Modular design for use of different MCU daughterboards on the same motherboard
- Sensorless Field Oriented Control (FOC) motor control, supporting both FAST™ and eSMO observers
- User-friendly graphical user interface (GUI) to facilitate code-free development and to control, identify, and monitor the motor

Applications

- [Washer and dryer](#)
- [Air conditioner indoor unit](#)
- [Refrigerator and freezer](#)
- [Appliances: compressor](#)
- Other HV motor control applications



TIEVM-MTR-HVINV EVM Board Layout

1 Evaluation Module Overview

1.1 Introduction

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

The TIEVM-MTR-HVINV provides a 750-W inverter motherboard and a control daughterboard with an MCU, such as the TIEVM-MC-F280013x, making it convenient for users to evaluate the C2000 series of microcontrollers in a high-voltage environment.

The software for the TIEVM-MTR-HVINV supports both the FAST™ and eSMO sensorless observers, so performance of the two algorithms can be compared. A user-friendly GUI also helps both identify the motor and tune the control parameters, accelerating development time.

1.2 Kit Contents

The TIEVM-MTR-HVINV Development Kit contains these items:

- TIEVM-MTR-HVINV Power Board/motherboard
- TIEVM-MC-MODULE-F280013x Control Board/daughter board
- 4x stand-offs and 4x screws, to be installed on the TIEVM-MTR-HVINV in each corner

While not included in the TIEVM-MTR-HVINV Development Kit, the following items are also required to use the EVM:

- XDS110ISO-EVM plug-in isolated emulator board
- USB Type-A to USB Type-C® cable or USB Type-C® to USB Type-C® cable to connect PC to XDS110ISO-EVM
- 1x HV motor
 - A variety of motors are supported, including completely custom motors, through the user_mtr1.h file and/or the MCU-MOTOR-CONTROL-GUI custom motor interface.
 - The Estun EMJ-04APB22 motor spec implementation is known good for this system, and has an integrated option in the MCU-MOTOR-CONTRL-GUI interface.

As part of the TIEVM-MTR-HVINV Power board/motherboard, an 8A 250V fuse has been provided.

In the event that fuse replacement is required, refer to the board BOM for fuse details and ratings, listed as component F1. Ensure that ALL motherboard and daughterboard elements have been safely de-energized. Gently remove the pre-existing fuse held by XF1 and XF2. Replace with the new fuse.

1.3 Specification

The TIEVM-MTR-HVINV specifications are listed in [Table 1-1](#).

Table 1-1. Key System Specifications

Parameters	TEST CONDITIONS	MIN	NOM	MAX	UNIT
SYSTEM INPUT/EXTERNAL POWER SUPPLY OUTPUT CHARACTERISTICS					
Input Voltage (V_{INAC})	–	165	230	265	VAC
Input Frequency (f_{LINE})	–	47	50	63	Hz
No Load Standby Power (P_{NL})	$V_{INAC} = 230\text{ V}$, $I_{out} = 0\text{ A}$	–	3.0	–	W
Input Current (I_{IN})	$V_{INAC} = 230\text{ V}$, $I_{out} = I_{MAX}$	–	8	–	A
MOTOR INVERTER CHARACTERISTICS					
PWM switching frequency (f_{SW})	–	–	15	20	kHz
Rated output power (P_{OUT})	$V_{INAC} = \text{nom}$	–	500	750	W
Output current (I_{RMS}) ¹	$V_{INAC} = \text{nom}$	–	3	–	A

¹ Refer to output power and individual motor characteristics to determine output current.

Table 1-1. Key System Specifications (continued)

Parameters	TEST CONDITIONS	MIN	NOM	MAX	UNIT
Inverter efficiency (η)	$V_{INAC} = \text{nom}$, $P_{OUT} = \text{nom}$	–	98	–	%
Motor electrical frequency (f)	$V_{INAC} = \text{min to max}$	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				

Table 1-1. Key System Specifications (continued)

Parameters	TEST CONDITIONS	MIN	NOM	MAX	UNIT
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 Temperature	Open frame	-10	25	55	°C
Board size	Length × width × height	105 mm × 85 mm × 60 mm			mm ³

TI intends this EVM 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 EVM 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! The board operates at voltages and currents that can cause shock, fire, or injury if not properly handled or applied. Electric shock possible when connecting board to live wire. Board should be handled with care by a professional.

For safety, use of **isolated test equipment** with overvoltage/overcurrent protection is **required**. If possible, tests should be conducted in an enclosure rated for high voltage testing.

WARNING

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 EVM powered when unattended.

Note

TI recommends using an external power supply or power accessory which complies with applicable regional safety standards such as (by example) UL, CSA, VDE, CCC, PSE, and so forth.

1.4 Device Information

- 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.
- The [TLV740P](#) 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.

- 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 μ A. 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.
- 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.

- 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.

- 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.

General Texas Instruments High Voltage Evaluation (TI HV EVM) User Safety Guidelines



Always follow TI's set-up and application instructions, including use of all interface components within the recommended electrical rated voltage and power limits. Always use electrical safety precautions to help ensure your personal safety and those working around you. Contact TI's Product Information Center <http://ti.com/customer-support> for further information.

Save all warnings and instructions for future reference.

WARNING

Failure to follow warnings and instructions can result in personal injury, property damage or death due to electrical shock and burn hazards.

The term TI HV EVM refers to an electronic device typically provided as an open framed, unenclosed printed circuit board assembly. It is *intended strictly for use in development laboratory environments, solely for qualified professional users having training, expertise and knowledge of electrical safety risks in development and application of high voltage electrical circuits. Any other use and/or application are strictly prohibited by Texas Instruments.* If you are not suitably qualified, you should immediately stop from further use of the HV EVM.

1. Work Area Safety:
 - a. Keep work area clean and orderly.
 - b. Qualified observers must be present anytime circuits are energized.
 - c. Effective barriers and signage must be present in the area where the TI HV EVM and the interface electronics are energized, indicating operation of accessible high voltages can be present, for the purpose of protecting inadvertent access.
 - d. All interface circuits, power supplies, evaluation modules, instruments, meters, scopes, and other related apparatus used in a development environment exceeding 50Vrms/75VDC must be electrically located within a protected Emergency Power Off EPO protected power strip.
 - e. Use stable and non-conductive work surface.
 - f. Use adequately insulated clamps and wires to attach measurement probes and instruments. No freehand testing whenever possible.
2. Electrical Safety:
 - a. As a precautionary measure, a good engineering practice is to assume that the entire EVM can have fully accessible and active high voltages.
 - b. De-energize the TI HV EVM and all the inputs, outputs and electrical loads before performing any electrical or other diagnostic measurements. Revalidate that TI HV EVM power has been safely de-energized.
 - c. With the EVM confirmed de-energized, proceed with required electrical circuit configurations, wiring, measurement equipment hook-ups and other application needs, while still assuming the EVM circuit and measuring instruments are electrically live.
 - d. Once EVM readiness is complete, energize the EVM as intended.

WARNING

While the EVM is energized, never touch the EVM or the electrical circuits, as the EVM or the electrical circuits can be at high voltages capable of causing electrical shock hazard.

3. Personal Safety
 - a. Wear personal protective equipment e.g. latex gloves or safety glasses with side shields or protect EVM in an adequate lucent plastic box with interlocks from accidental touch.

Limitation for safe use:

EVMs are not to be used as all or part of a production unit.

2 Hardware

2.1 Hardware Description

The TIEVM-MTR-HVINV, when coupled with a control board (such as the TIEVM-MC-MODULE-F280013x), enables a complete motor drive system. Figure 2-1 shows an overview of the system.

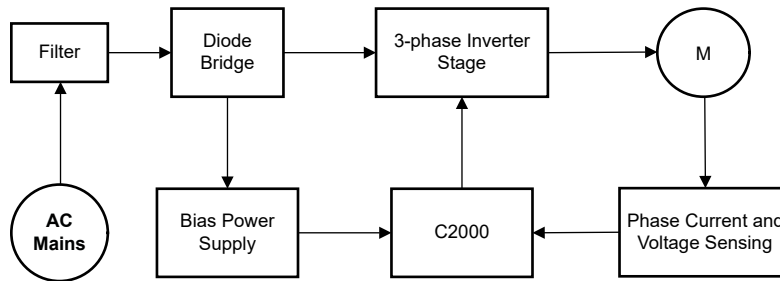


Figure 2-1. Hardware Board Block Diagram of TIEVM-MTR-HVINV

The following is a list of the major functionalities of the TIEVM-MTR-HVINV, organized into blocks for ease of reference.

- Power line 220VAC input filter and diode bridge rectifier
- Bias power supply
 - +15 V and +3.3V generated from rectified AC input.
- 3-phase inverter
 - Up to 750-W 3-phase inverter
 - 15-kHz switching frequency
- Phase current shunt resistors and phase voltage signal conditioning
- TIEVM-MC-MODULE-F280013x daughter card
 - Single C2000 family MCU in a 48-pin LQFP package
 - Analog sensing amplification and filtering
 - XDS110ISO-EVM UART-capable isolated debugger emulator board interface

2.1.1 Auxiliary Power Supply

A non-isolated UCC28881-based high-voltage buck supply provides auxiliary power, delivering up to 200 mA for 15 VDC for the TIEVM-MTR-HVINV. Figure 2-2 shows the UCC28881 high-voltage buck supply circuit.

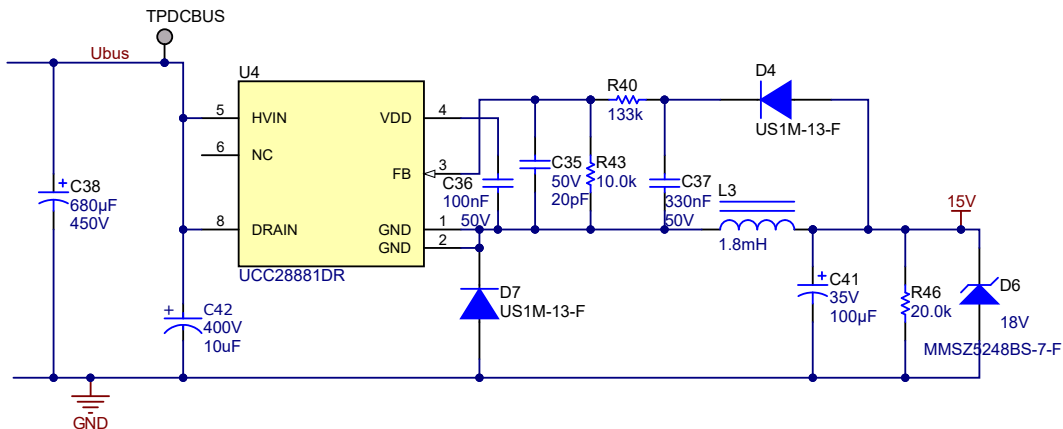


Figure 2-2. High-Voltage Buck Power Supply Circuit

A TPS54202 step-down converter is used to generate 5 VDC from the 15 VDC auxiliary power. A TLV74033P LDO then regulates the 5 V power rail to provide 3.3 VDC to the MCU daughterboard.

2.1.2 DC Link Voltage Sensing

The DC voltage sensing circuit is used to convert the rectified DC bus voltage signal into a low-voltage signal. The circuit is implemented by a low-cost resistor network, as shown in Figure 2-3. The DC bus voltage can also be used to estimate AC input voltage, as the physical parameters of the AC input circuit are known constants.

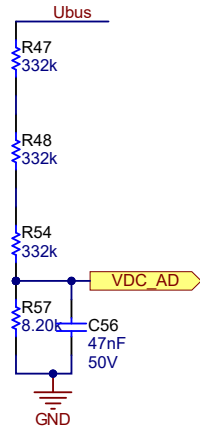


Figure 2-3. DC Bus Voltage Sensing Circuit

2.1.3 Motor Phase Voltage Sensing

The software for the TIEVM-MTR-HVINV allows for both the enhanced Sliding Mode Observer (eSMO) and the Flux, Angle, Speed, and Torque (FAST™) sensorless observers. While the eSMO only requires the commonly used 3-phase motor current sensing, the FAST observer additionally requires 3-phase motor voltage sensing to improve low-speed performance and increase the precision of the motor speed estimation.

Note that the software parameter (USER_ADC_FULL_SCALE_VOLTAGE_V) depends on the circuit that senses the voltage feedback from the motor phases. Figure 2-4 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 1, considering the maximum voltage for the ADC input is 3.3 V.

$$V_{FS} = V_{ADC_FS} \times G_V = 3.3 \text{ V} \times 122.46 = 404.13 \text{ V} \quad (1)$$

where

- G_V is attenuation factor, G_V is calculated with Equation 2

$$G_V = \frac{(R62 + R67 + R70 + R74)}{R74} = \frac{(332 \text{ k}\Omega + 332 \text{ k}\Omega + 332 \text{ k}\Omega + 8.2 \text{ k}\Omega)}{8.2 \text{ k}\Omega} = 122.46 \quad (2)$$

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 3:

$$f_{\text{filter_pole}} = \frac{1}{(2 \times \pi \times R_{\text{Parallel}} \times C)} = 405.15 \text{ Hz} \quad (3)$$

where,

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

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

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)
```

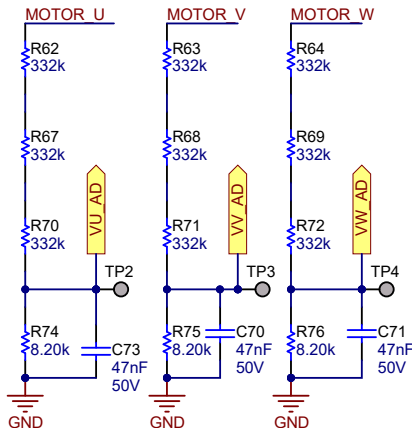


Figure 2-4. Motor Voltage Sensing Circuit

2.1.4 Motor Phase Current Sensing

The TIEVM-MTR-HVINV design supports phase current measurements of single or three-shunt sensing schemes. Either of these two current sensing techniques can be selected in the build configuration of the project. The default *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 2.1.4.2.

2.1.4.1 Three-Shunt Current Sensing

The current through the motor is sampled by the microcontroller as part of the motor control algorithm once during each PWM cycle. The TIEVM-MC-MODULE daughter board supports single and 3-shunt current sensing.

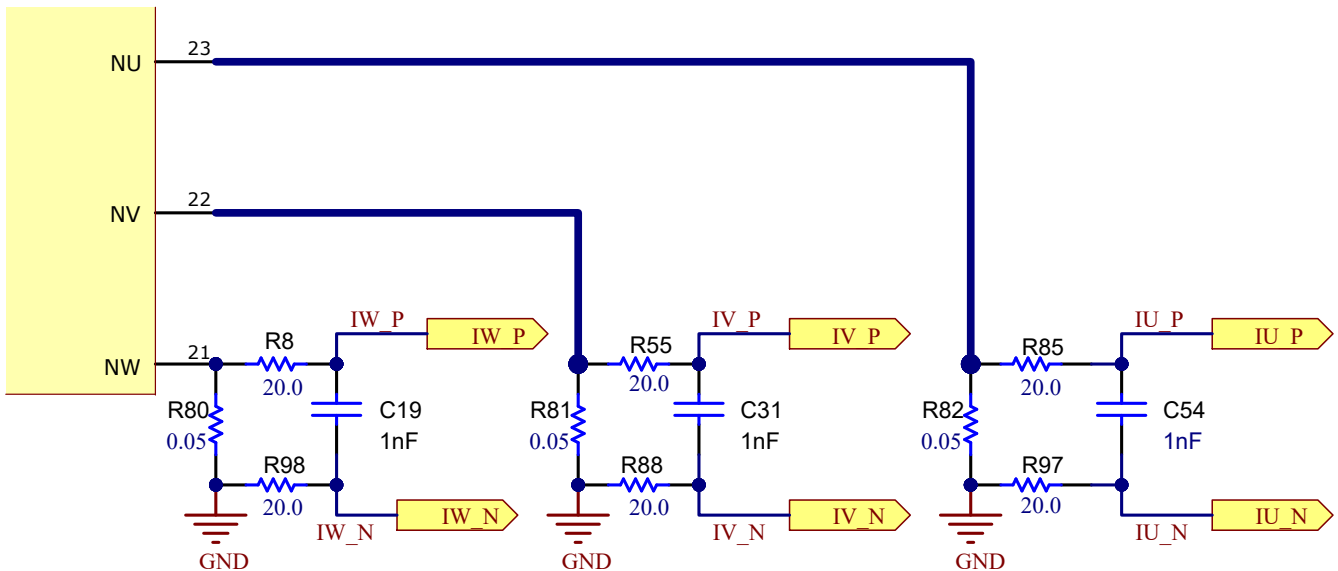


Figure 2-5. Three-Shunt Resistor Circuit

To measure both positive and negative currents, the measurement circuits require an offset reference voltage of half of the maximum input allowable by the ADC. This 1.65 V offset reference voltage is created by a voltage follower, as shown in Figure 2-6:

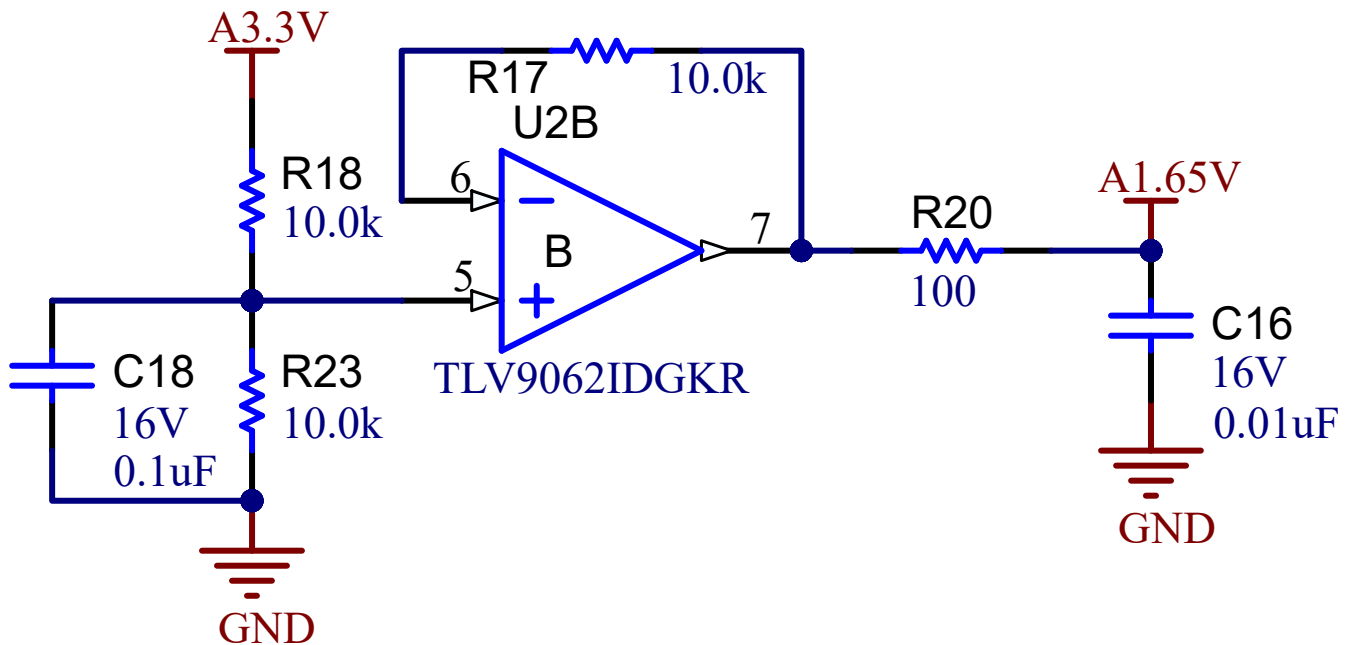


Figure 2-6. 1.65-V Reference From 3.3-V Input Circuit

Figure 2-7 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 daughterboard. This circuit is used for each of the three phases of the PMSM. The transfer function of this circuit is given by Equation 4.

$$V_{OUT} = V_{OFFSET} + (I_{IN} \times R_{SHUNT} \times G_i) \quad (4)$$

where

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

The calculated resistance values lead to the sensing circuit shown in Figure 2-4, G_i is given by Equation 5.

$$G_i = \frac{R_{fb}}{R_{in}} = \frac{R_{18}}{(R_{97} + R_{15})} = \frac{10 \text{ k}\Omega}{20 + 2.4 \text{ k}\Omega} = 4.132 \quad (5)$$

The maximum peak-to-peak current measurable by the microcontroller is given by Equation 6.

$$I_{scale_max} = \frac{V_{ADC_max}}{R_{SHUNT} \times G_i} = \frac{3.3}{0.05 \times 4.132} = 15.97 \text{ A} \quad (6)$$

This has the peak-to-peak value of $\pm 7.99 \text{ A}$. 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)
```

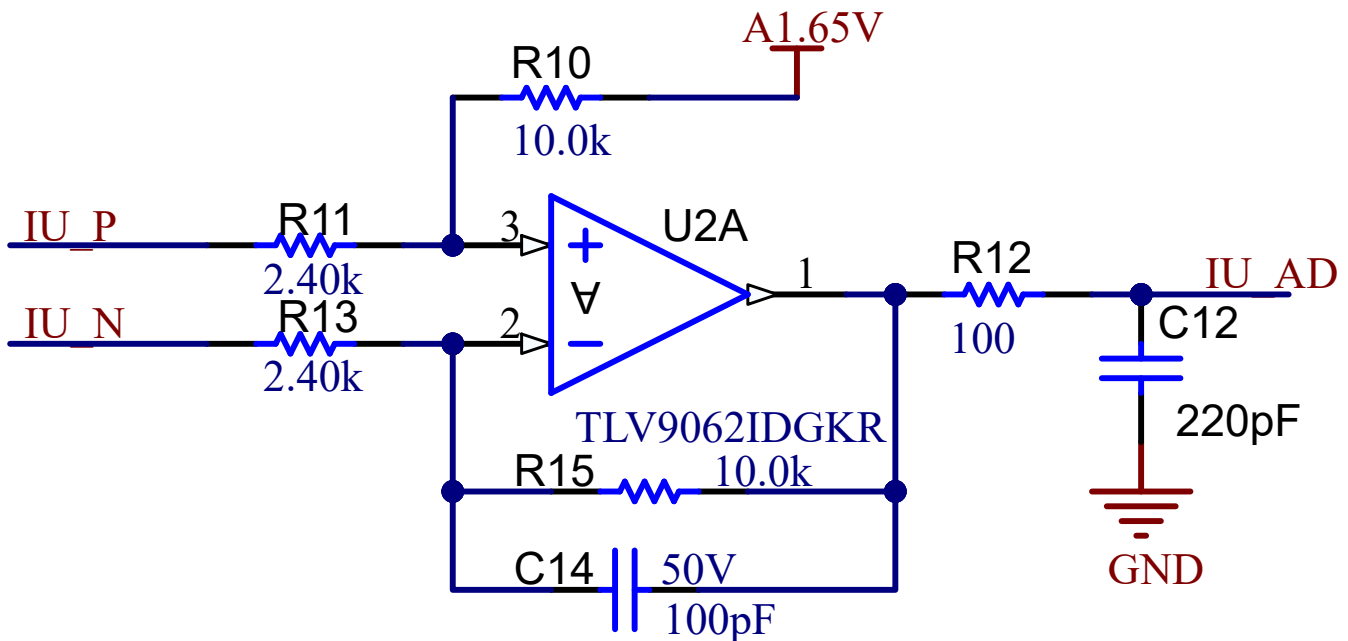


Figure 2-7. Three-Shunt Current Sensing Circuit for TMS320F2800137

2.1.4.2 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 2-8.

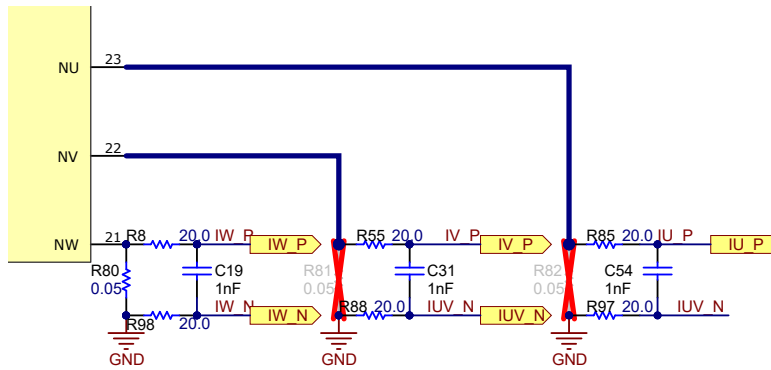


Figure 2-8. Single-Shunt Current-Sensing Resistor Configuration

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 the TMS320F2800137 daughterboard, remove C23 to increase U3A bandwidth for single-shunt sampling.
3. Use a thick wire to connect the NU, NV, and NW pins together.

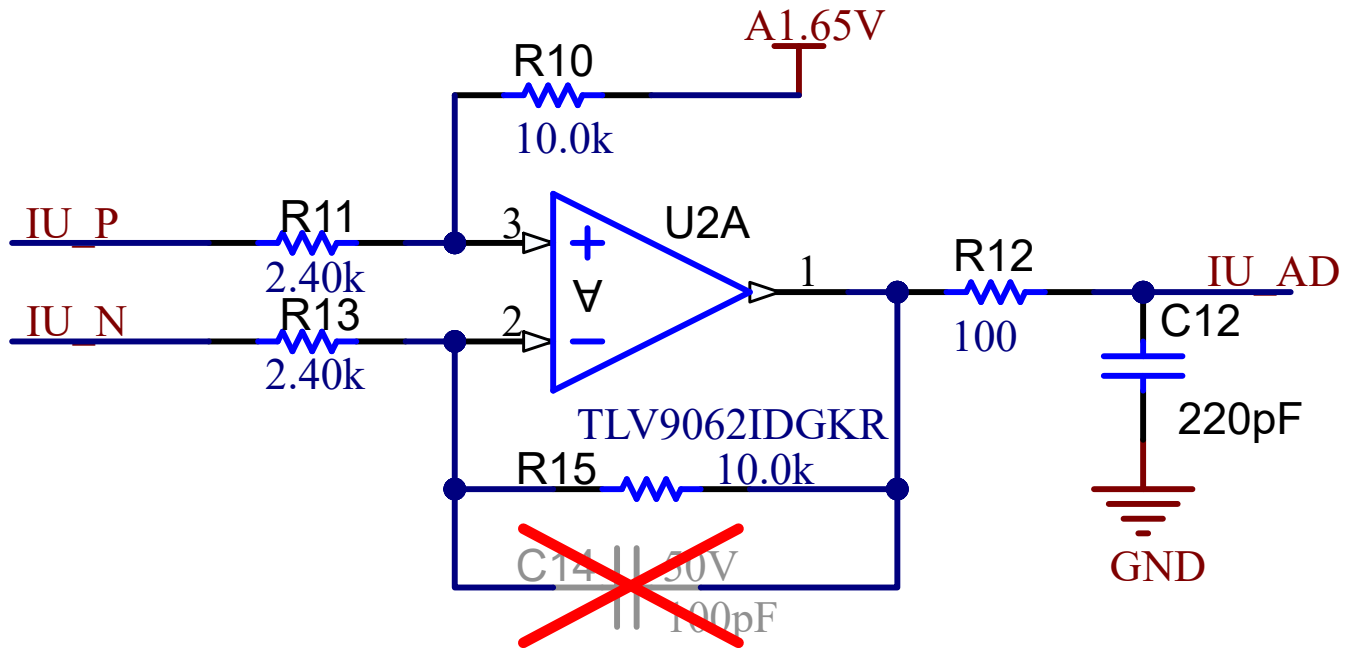


Figure 2-9. Single-Shunt Current-Sensing Circuit for TMS320F2800137

By default, the board has three shunt resistors. [Figure 2-10](#) shows the layout of the shunt resistors. To implement single-shunt resistor sensing, remove R81 and R82 while keeping R80. Solder NU, NV and NW (pin 2 of R80, R81, and R82) together. All three phase currents now flow through only R80.

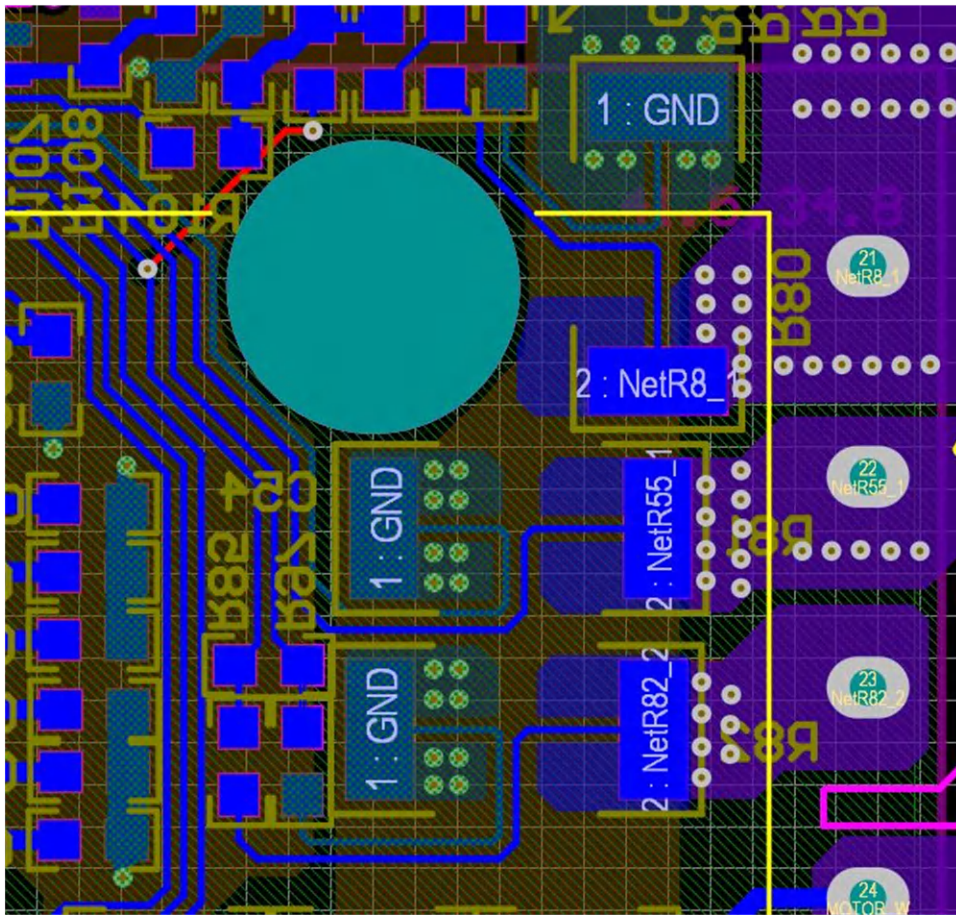


Figure 2-10. Shunt Resistors Layout

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 2-11](#) shows. On the TMS320F2800137 daughterboard, change R23 from 10 k Ω to 1 k Ω /1% resistor for the reference voltage to have 0.3-V offset for DC current sensing.

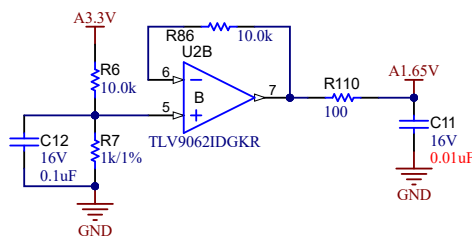


Figure 2-11. DC Offset Reference for Single Shunt of TMS320F2800137 Daughterboard

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

2.1.5 External Overcurrent Protection

The TIEVM-MTR-HVINV implements two redundant overcurrent protection (OCP) circuits. First, an external comparator is capable of reporting an overcurrent fault to the 3-phase inverter. Second, the daughterboard MCU is capable of reporting an overcurrent fault through the use of internal comparators. If either OCP mechanism reports a fault, both the 3-phase inverter and the MCU transition into a fault state. The system remains in a fault state until the state is manually cleared.

Figure 2-12 shows the external OCP circuit. This circuit sums the three current phases, the compares that sum to a reference value. If the summation of the three phase currents ever exceeds the reference value, then the 'IPM_CIN' signal switches and the IPM reports an overcurrent fault to the microcontroller. The exact overcurrent protection current can be calculated by below equations.²

First, the reference voltage V_- is generated via a voltage divider from the 3.3V power rail.

$$V_- = 3.3 \text{ V} \times \frac{R_{108}}{R_{107} + R_{108}} = 3.3 \text{ V} \times \frac{1 \text{ k}}{20 \text{ k} + 1 \text{ k}} = 0.15714 \text{ V} \quad (7)$$

Next, note that the resistance from any IPM phase to the junction point U10₊ is 5.12kΩ. In this scenario, two of the phases (for example, V and W) are at approximately 0A, while the third phase (U) is experiencing a current spike and triggering the OCP circuit. As such, consider R_V and R_W in parallel.

$$R_U = R_V = R_W = R_{87} + R_8 = 5.12 \text{ k} \quad (8)$$

$$R_{V||W} = \left(\frac{1}{R_V} + \frac{1}{R_W} \right)^{-1} = \left(\frac{2}{5.12 \text{ k}} \right)^{-1} = 2.56 \text{ k} \quad (9)$$

The voltage at the junction point U10₊ can be referred to as V_+ . V_+ is generated via voltage division relative to the voltage of each phase of the IPM (V_U , V_V , V_W). Note that V_V and V_W are both assumed to be 0V in this scenario, as I_V and I_W are both stated to be at or approaching 0A.

$$V_+ = \left(V_U - V_{VW} \right) \times \frac{R_{V||W}}{R_U + R_{V||W}} = V_U \times \frac{R_{V||W}}{R_U + R_{V||W}} \quad (10)$$

$$V_+ = V_U \times \frac{R_{V||W}}{R_U + R_{V||W}} = V_U \times \frac{2.56 \text{ k}}{5.12 \text{ k} + 2.56 \text{ k}} = \frac{V_U}{3} \quad (11)$$

V_U is also defined as the Voltage across the shunt resistor R_{shunt} during the overcurrent event.

$$V_U = R_{\text{shunt}} \times I_{\text{OCP}} = 0.05 \times I_{\text{OCP}} \quad (12)$$

For the OCP circuit to trigger, V_+ must be greater than or equal to the reference voltage V_- .

$$V_+ = \frac{0.05 \times I_{\text{OCP}}}{3} \geq V_- \quad (13)$$

$$I_{\text{ocp}} \geq \frac{V_-}{\left(\frac{0.05}{3} \right)} = \frac{0.15714}{\left(\frac{0.05}{3} \right)} = 9.42857 \text{ A} \quad (14)$$

² Certain revisions of the board may have higher OCP limits for the external circuit. In these instances, the internal (CMPSS) protection is still fully functional at the intended levels. Refer to [Section 6.1](#) for more information.

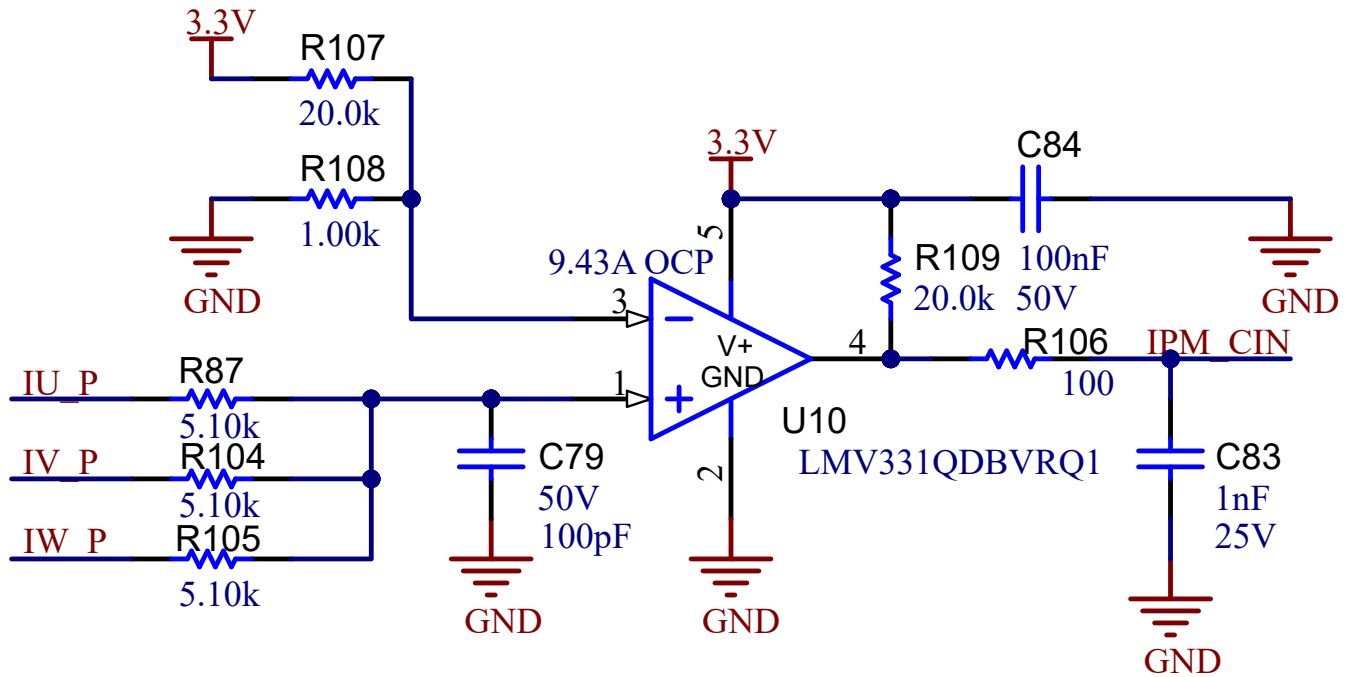


Figure 2-12. External Overcurrent Protection Circuit

2.1.6 Internal Overcurrent Protection for TMS320F2800F137

The daughter board MCU has internal window comparators which can be configured to monitor the three phases of the current. All comparator and PWM trip tasks operate independently of the software, and there is no interrupt delay between the comparators reporting an overcurrent fault and the PWM transitioning into a fault state. This allows for quick protection for external IPM or power devices.

2.2 Getting Started Hardware

This section details the necessary equipment and test setup for the board hardware.

2.2.1 Test Conditions and Equipment

Verify the following for testing the TIEVM-MTR-HVINV :

- 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 a DC power supply. Set the input current limit of input AC source to 10 A or DC power supply to 6.5 A.
- For the output, use a 3-phase PMSM with a dynamometer to apply counter-rotation.
 - As described in [Kit Contents](#), Texas Instruments loosely recommends the Estun EMJ-04APB22 for testing the TIEVM-MTR-HVINV system.
 - For motor selection, note that the maximum output power of the TIEVM-MTR-HVINV is 750-W.

For board testing, the HV-qualified engineer must use the following equipment:

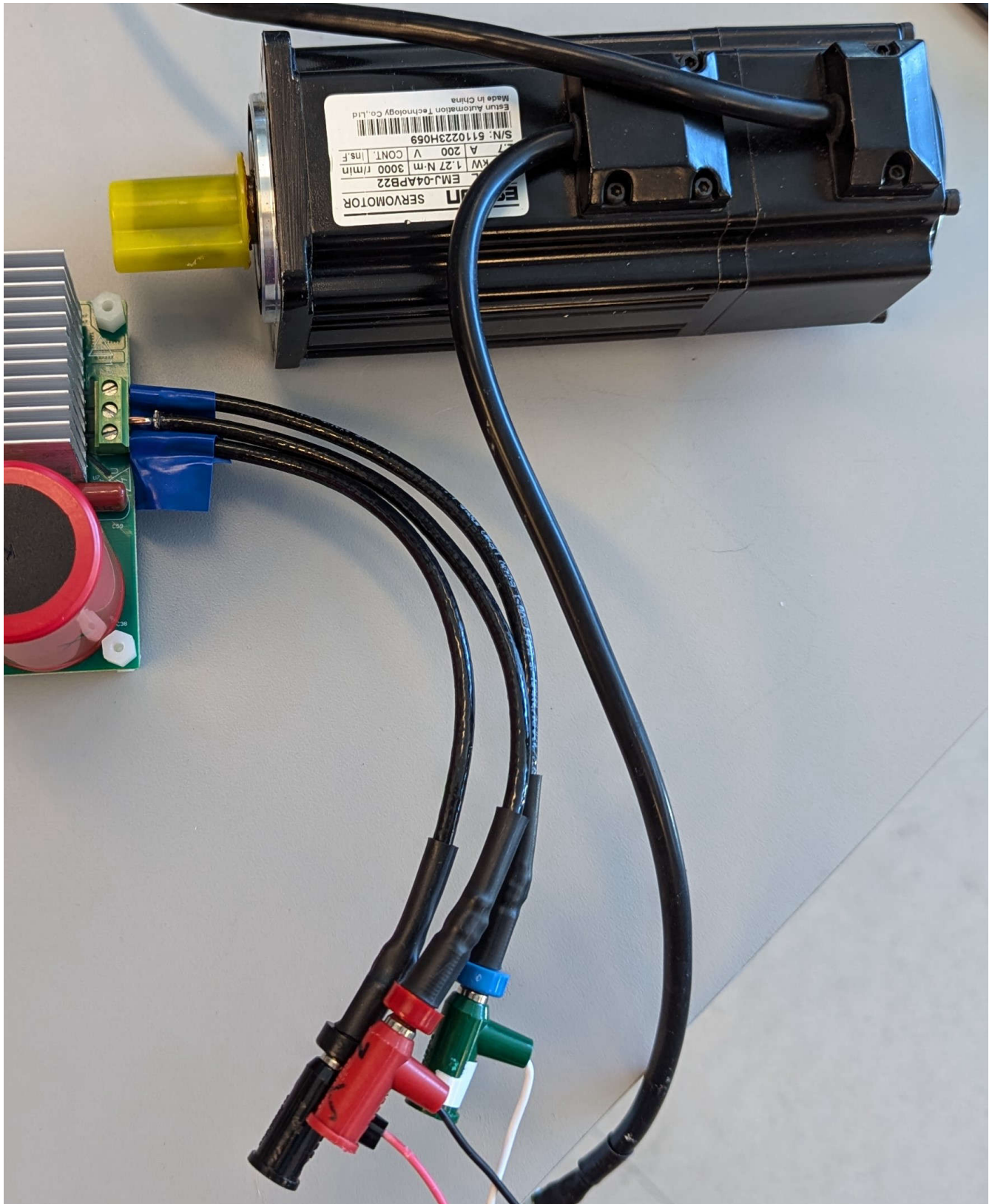
- Isolated AC source

The following equipment is also highly recommended for full testing, but is not necessarily required for basic operation:

- DC power supply
- Dynamometer
- Single-phase power analyzer
- Three-phase power analyzer
- Oscilloscope
 - HV probes
 - At least 1 current probe
- Multimeter

2.2.2 Test Setup

1. Optionally, use 4 included screws to attach the board to the standoffs. Each corner of the TIEVM-MTR-HVINV has a circular hole for this purpose.
2. If not already connected, connect the daughterboard to the top of the motherboard. The 2-row header of the motherboard (J15) should align with the 2-row connector of the daughterboard (J3).
3. Key pin 8 of daughterboard connector J3 should be clipped if it is not already done.
4. Observe the bottom of the XDS110ISO-EVM and locate the key pin. Connect the XDS110ISO-EVM to daughterboard connector J3, such that the key pins of each board are aligned.
5. Verify the desired state of the daughterboard boot mode switch (S1).
6. Connect motor wires to J10.
 - a. For the Estun EMJ-04APB22, the following is an example image showing how it may be connected:



7. Connect measurement equipment, such as multimeter, oscilloscope probes, and so forth, to probe or analyze various signals and parameters. Available test points can be viewed in the board schematic (TPx).
8. Power on the board with isolated AC or DC power supply to the J5 ACL, ACN, and Earth connector.
 - a. The maximum output of the AC power supply is 265-VAC, 50/60Hz.
 - b. The maximum output of the DC power supply is 380-VDC.

The following options are also available.

- Daughterboard connector J4 pins 4 through 7 are available as SPI signals, should the user require external SPI. When doing so, the XDS110ISO-EVM on-board DAC is not available. This connection can be used by either:
 - Connecting directly to the SPI signals on daughterboard connector J4. This connection is NOT isolated, and an external isolator must be used.
 - Connecting to isolated UART interface J14, which may not be available on certain board revisions.
- If using this connection, depopulate daughterboard resistor R8 to ensure that the XDS110ISO-EVM on-board DAC does not interfere with the SPI peripheral-to-controller signals.
- The default board state utilizes three-phase current sensing. Should single-phase be required, refer to [Section 2.1.4.2](#).

3 Motor Control Software

3.1 Three-Phase PMSM Drive System Design Theory

In order to properly explain the software of the TIEVM-MTR-HVINV, background knowledge is required. Broadly speaking, it is important to understand both the basic theory of a three-phase PMSM motor drive system, as well as how sensorless FOC control fits into that system.

A Permanent Magnet Synchronous motor (PMSM) has a wound stator, a permanent magnet rotor assembly, and internal or external mechanisms to sense rotor position. The sensing mechanisms 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 rated power range and 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 the flux gradients and number of poles 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 3-1](#).
- 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 90° 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.

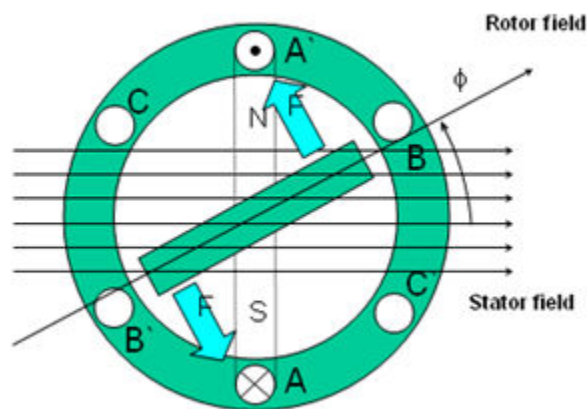


Figure 3-1. Interaction Between the Rotating Stator Flux and the Rotor Flux Produces Torque

3.1.1 Field-Oriented Control of PMSM

To achieve better dynamic performance, a more complex control scheme needs to be applied to control the motor. With the mathematical processing power offered by microcontrollers, advanced control strategies which use mathematical transformations to decouple the torque generation and the magnetization functions in PM motors can be implemented. Such de-coupled torque and magnetization control is commonly called rotor flux oriented control, vector 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 3-2](#). 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.

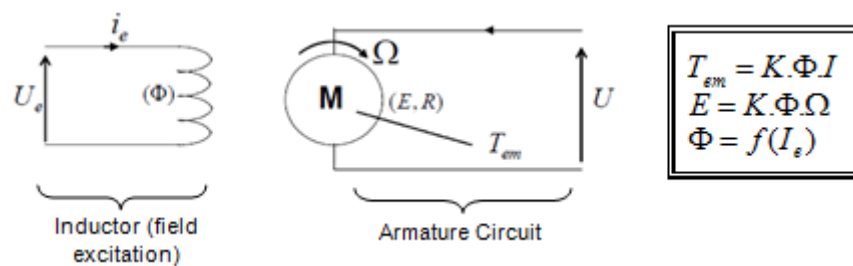


Figure 3-2. Flux and Torque are Independently Controlled in DC Motor Model

The goal of FOC on synchronous and asynchronous machines is to be able to separately control the torque-producing and magnetizing flux components. FOC algorithms allow 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 a microcontroller adds the most value. The processing capability provided by a microcontroller enables these mathematical transformations to be carried out very quickly. This, in turn, means 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 can also be used for the computation of many values such as rotor flux angle and rotor speed, improving overall quality of control.

According to the laws of electromagnetism, the torque produced in a synchronous machine is equal to the vector cross product of the two existing magnetic fields as in [Equation 15](#).

$$\tau_{em} = \vec{B}_{stator} \times \vec{B}_{rotor} \quad (15)$$

This expression shows that the torque is maximum if stator and rotor magnetic fields are *orthogonal*, meaning 90 degrees apart. If this condition can be maintained all the time and if the flux can be oriented correctly, the torque ripple is reduced and a better dynamic response is provided. However, for this to be true, the rotor position must be known: this can be achieved either with a physical position sensor (such as an incremental encoder) or a sensorless rotor position observer.

In order to achieve the goal of aligning the stator flux orthogonally to the rotor flux, the d-axis component of the stator current in the (direct, quadrature) rotating reference frame is set to zero. The (d, q) rotating reference frame is explained more fully in [Section 3.1.1.1.2](#). When this condition is true, the stator flux and the rotor flux are orthogonally aligned. The d-axis component of the stator current can also be used in some cases for Field Weakening, which allows for reduction of back-emf and for the motor to operate at higher speeds.

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 problems of classical motor control schemes, 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 16](#).

$$\tau_{em} \propto \psi_R \times i_{sq} \tag{16}$$

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.

3.1.1.1 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 17](#).

$$\bar{i}_s = i_a + \alpha i_b + \alpha^2 i_c \tag{17}$$

where

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

[Figure 3-3](#) shows the stator current complex space vector.

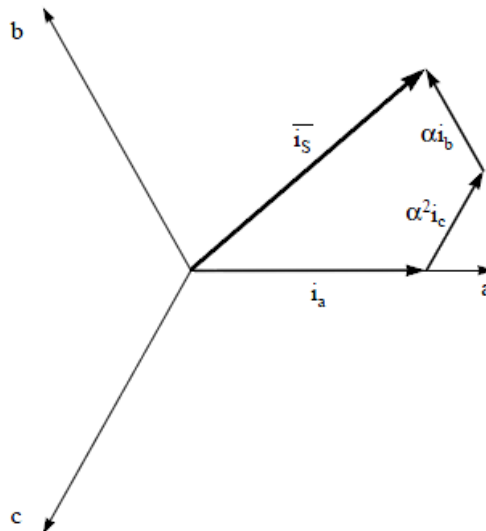


Figure 3-3. Stator Current Space Vector and Component in (a, b, c) Frame

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:

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

3.1.1.1.1 $(a, b) \Rightarrow (\alpha, \beta)$ **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 are in the same direction yields the vector diagram shown in [Figure 3-4](#).

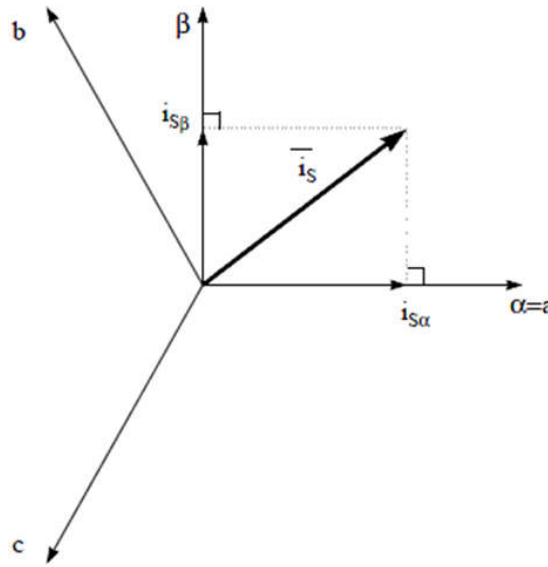


Figure 3-4. Stator Current Space Vector in the Stationary Reference Frame

The projection that modifies the 3-phase system into the (α, β) 2-dimension orthogonal system is presented in [Equation 18](#).

$$\begin{aligned} i_{s\alpha} &= i_a \\ i_{s\beta} &= \frac{1}{\sqrt{3}}i_a + \frac{2}{\sqrt{3}}i_b \end{aligned} \tag{18}$$

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

3.1.1.1.2 $(\alpha, \beta) \Rightarrow (d, q)$ **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 3-5](#) shows the relationship for the current vector from the two reference frame.

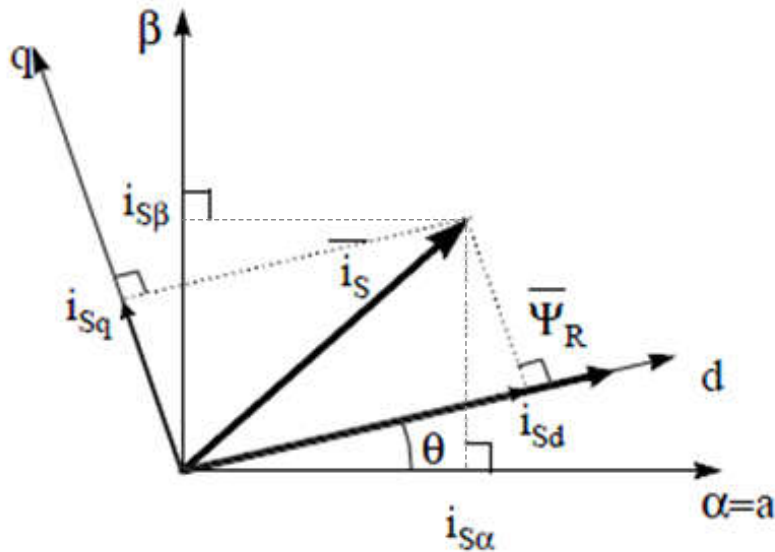


Figure 3-5. Stator Current Space Vector in the d,q Rotating Reference Frame

The flux and torque components of the current vector are determined by [Equation 19](#).

$$\begin{aligned} i_{sd} &= i_{s\alpha}\cos(\theta) + i_{s\beta}\sin(\theta) \\ i_{sq} &= -i_{s\alpha}\sin(\theta) + i_{s\beta}\cos(\theta) \end{aligned} \tag{19}$$

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.

3.1.1.2 Basic Scheme of FOC for AC Motor

[Figure 3-6](#) summarizes the basic scheme of torque control with FOC. Note that this description assumes a two-shunt sensing system to align with the figure, but single and three-shunt sensing schemes are also common. Three-shunt control uses a somewhat different implementation of the Clarke transformation, while single-shunt requires programmatic phase current reconstruction prior to the Clarke transform. In any of these instances, the scheme is identical beginning with the α and β output of the Clarke transform.

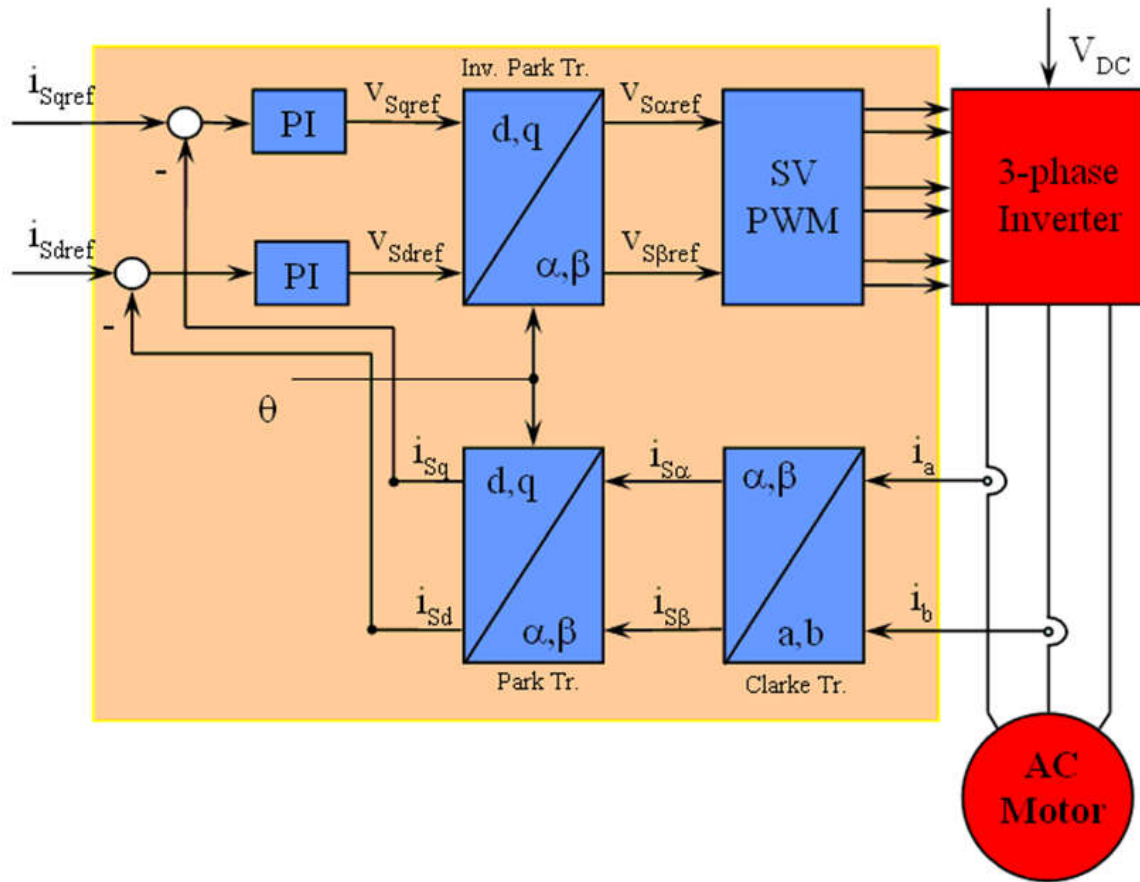


Figure 3-6. Basic Scheme of FOC for AC Motor

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 calculates 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.

In a PMSM, the rotor flux is fixed and determined by the magnets; there is no need to generate any flux. 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).

3.1.1.3 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 3-7 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.

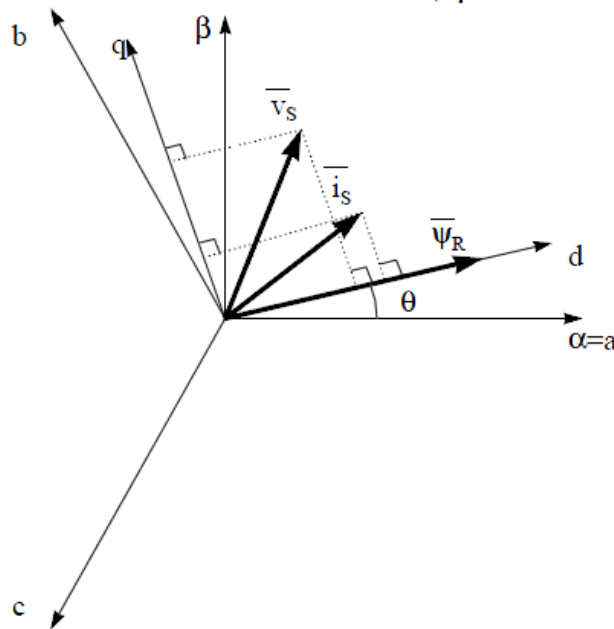


Figure 3-7. Current, Voltage and Rotor Flux Space Vectors in the (d, q) Rotating Reference Frame

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 3-8 shows the overall block diagram of sensorless FOC of fan PMSM using eSMO with flying start in this reference design.

Figure 3-9 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 3-10 shows the overall block diagram of sensorless FOC of fan PMSM using FAST with flying start in this reference design.

Figure 3-11 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.

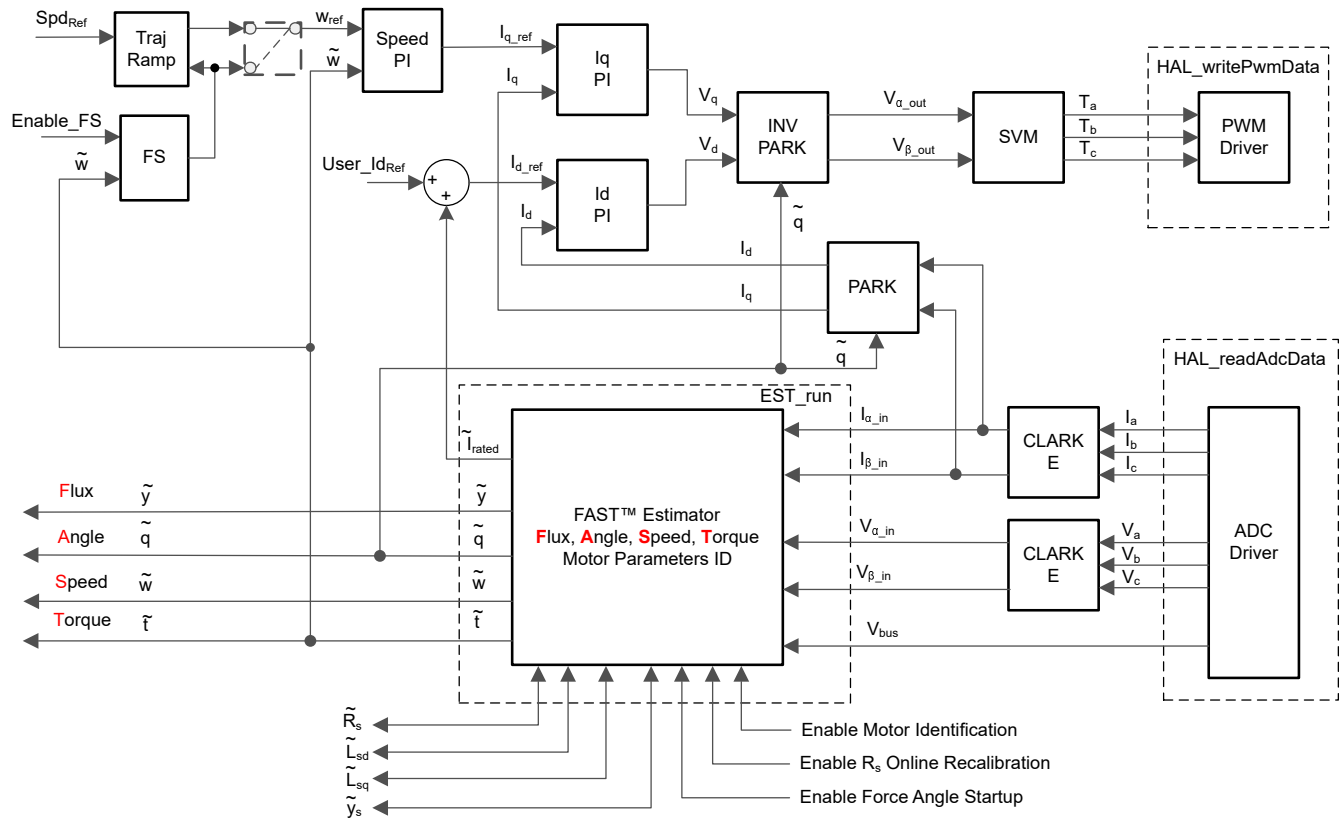


Figure 3-10. Sensorless FOC of PMSM Using FAST With Flying Start (FS)

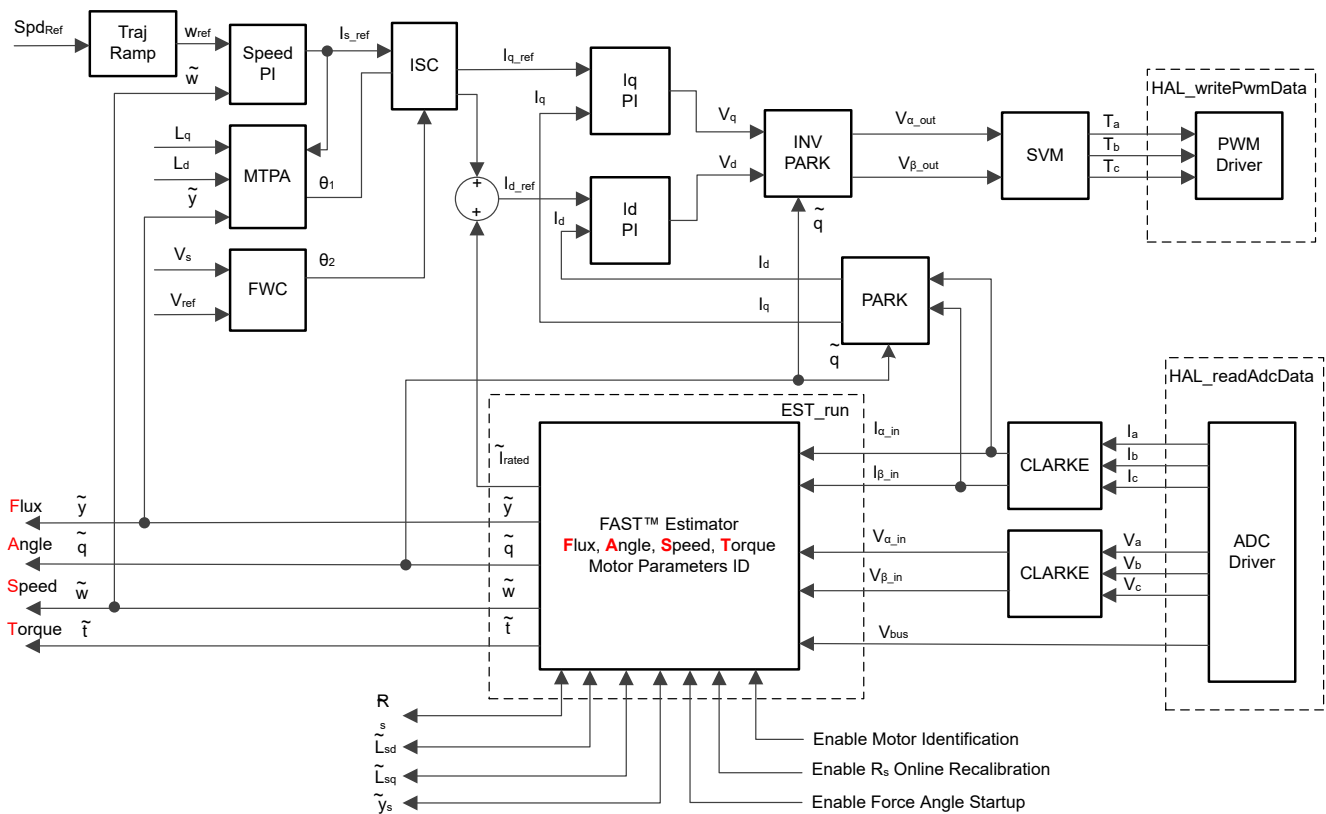


Figure 3-11. Sensorless FOC of PMSM Using FAST With FWC and MTPA

3.1.2 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.

3.1.2.1 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.

3.1.2.1.1 Mathematical Model and FOC Structure of an IPMSM

The sensorless FOC structure for an IPMSM is illustrated in [Figure 3-12](#). 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.

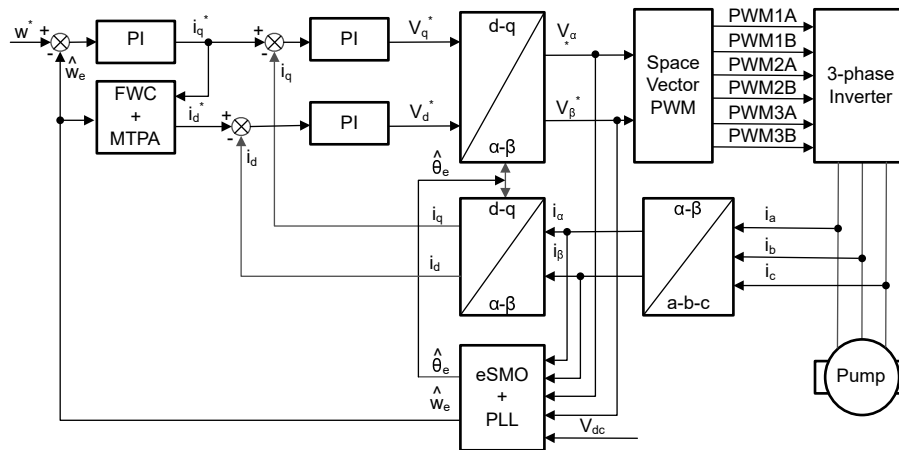


Figure 3-12. Sensorless FOC Structure of an IPMSM System

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 20](#). The dynamic model of a generic PMSM can be written in the d-q rotor reference frame as:

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} R_s + pL_d & -\omega_e L_q \\ \omega_e L_d & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_e \lambda_{pm} \end{bmatrix} \quad (20)$$

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

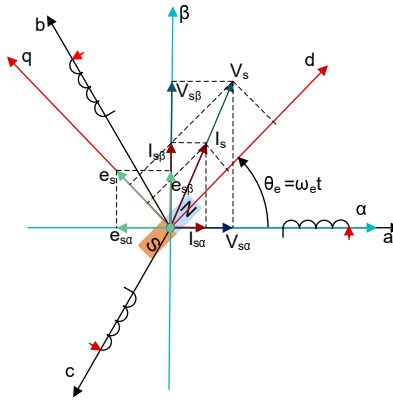


Figure 3-13. Definitions of Coordinate Reference Frames for PMSM Modeling

By using the inverse Park transformation as shown in [Figure 3-13](#), the dynamics of the PMSM can be modeled in the α - β stationary reference frame as shown in [Equation 21](#):

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} R_s + pL_d & \omega_e(L_d - L_q) \\ -\omega_e(L_d - L_q) & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} \quad (21)$$

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 22](#):

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = (\lambda_{pm} + (L_d - L_q)i_d)\omega_e \begin{bmatrix} -\sin(\theta_e) \\ \cos(\theta_e) \end{bmatrix} \quad (22)$$

According to [Equation 21](#) and [Equation 22](#), 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 21](#) as a state equation using the stator current as a state variable:

$$\begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} = \frac{1}{L_d} \begin{bmatrix} -R_s & -\omega_e(L_d - L_q) \\ \omega_e(L_d - L_q) & -R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} V_\alpha - e_\alpha \\ V_\beta - e_\beta \end{bmatrix} \quad (23)$$

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

$$s(x) = \begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix} = \begin{bmatrix} \tilde{i}_\alpha \\ \tilde{i}_\beta \end{bmatrix} \quad (24)$$

where

- \hat{i}_α and \hat{i}_β are the estimated currents
- the superscript $\hat{}$ indicates the estimated value
- the superscript “ $\tilde{}$ ” indicates the variable error which refers to the difference between the observed value and the actual measurement value

3.1.2.1.2 Design of ESMO for the IPMS

Figure 3-14 shows the conventional PLL integrated into the SMO.

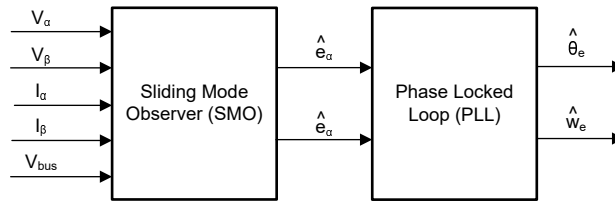


Figure 3-14. Block Diagram of eSMO With PLL for a PMSM

The traditional reduced-order sliding-mode observer is constructed, with the mathematical model shown in Equation 25 and the block diagram shown in Figure 3-15.

$$\begin{bmatrix} \dot{\hat{i}}_{\alpha} \\ \dot{\hat{i}}_{\beta} \end{bmatrix} = \frac{1}{L_d} \begin{bmatrix} -R_s & -\hat{\omega}_e(L_d - L_q) \\ \hat{\omega}_e(L_d - L_q) & -R_s \end{bmatrix} \begin{bmatrix} \hat{i}_{\alpha} \\ \hat{i}_{\beta} \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} V_{\alpha} - \hat{e}_{\alpha} + z_{\alpha} \\ V_{\beta} - \hat{e}_{\beta} + z_{\beta} \end{bmatrix} \quad (25)$$

where

- z_{α} and z_{β} are sliding-mode feedback components and are defined as shown in Equation 26:

$$\begin{bmatrix} z_{\alpha} \\ z_{\beta} \end{bmatrix} = \begin{bmatrix} k_{\alpha} \text{sign}(\hat{i}_{\alpha} - i_{\alpha}) \\ k_{\beta} \text{sign}(\hat{i}_{\beta} - i_{\beta}) \end{bmatrix} \quad (26)$$

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 $k_{\beta} > \max(|e_{\beta}|)$.

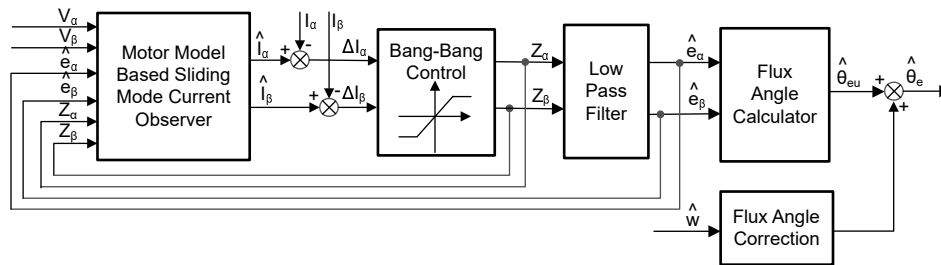


Figure 3-15. Block Diagram of Traditional Sliding-Mode Observer

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

$$\begin{bmatrix} \hat{e}_{\alpha} \\ \hat{e}_{\beta} \end{bmatrix} = \frac{\omega_c}{s + \omega_c} \begin{bmatrix} z_{\alpha} \\ z_{\beta} \end{bmatrix} \quad (27)$$

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 28](#) defines:

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) \quad (28)$$

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 29](#):

$$\Delta\theta_e = -\tan^{-1}\left(\frac{\omega_e}{\omega_c}\right) \quad (29)$$

Then the estimated rotor position by using SMO method is found with [Equation 30](#):

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) + \Delta\theta_e \quad (30)$$

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 25](#) in α - β coordinates is given by [Equation 31](#) as:

$$\begin{bmatrix} \hat{i}_\alpha(n+1) \\ \hat{i}_\beta(n+1) \end{bmatrix} = \begin{bmatrix} F_\alpha \\ F_\beta \end{bmatrix} \begin{bmatrix} \hat{i}_\alpha(n) \\ \hat{i}_\beta(n) \end{bmatrix} + \begin{bmatrix} G_\alpha \\ G_\beta \end{bmatrix} \begin{bmatrix} V_\alpha^*(n) - \hat{e}_\alpha(n) + z_\alpha(n) \\ V_\beta^*(n) - \hat{e}_\beta(n) + z_\beta(n) \end{bmatrix} \quad (31)$$

where

- the matrix [F] and [G] are given by [Equation 32](#) and [Equation 33](#) as:

$$\begin{bmatrix} F_\alpha \\ F_\beta \end{bmatrix} = \begin{bmatrix} e^{-\frac{R_s}{L_d}} \\ e^{-\frac{R_s}{L_q}} \end{bmatrix} \quad (32)$$

$$\begin{bmatrix} G_\alpha \\ G_\beta \end{bmatrix} = \frac{1}{R_s} \begin{bmatrix} 1 - e^{-\frac{R_s}{L_d}} \\ 1 - e^{-\frac{R_s}{L_q}} \end{bmatrix} \quad (33)$$

The time-discrete form of [Equation 27](#) is given by [Equation 34](#) as:

$$\begin{bmatrix} \hat{e}_\alpha(n+1) \\ \hat{e}_\beta(n+1) \end{bmatrix} = \begin{bmatrix} \hat{e}_\alpha(n) \\ \hat{e}_\beta(n) \end{bmatrix} + 2\pi f_c \begin{bmatrix} z_\alpha(n) - \hat{e}_\alpha(n) \\ z_\beta(n) - \hat{e}_\beta(n) \end{bmatrix} \quad (34)$$

3.1.2.1.2.1 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 3.1.2.1.2](#) illustrates the PLL structure used with SMO. The back-EMF estimations \hat{e}_α and \hat{e}_β can be used with a PLL model to estimate the motor angular velocity and position as shown in [Figure 3-16](#).

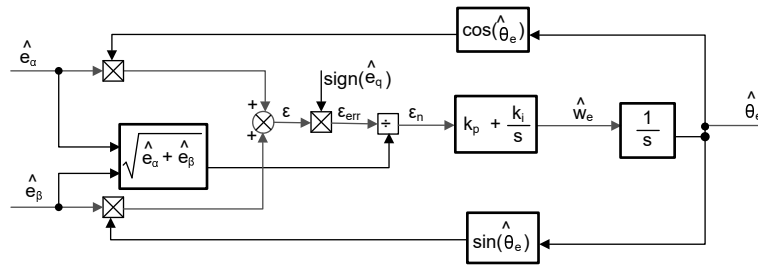


Figure 3-16. Block Diagram of Phase-Locked Loop Position Tracker

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 35](#):

$$\varepsilon = \hat{e}_\beta \cos(\hat{\theta}_e) - \hat{e}_\alpha \sin(\hat{\theta}_e) = E \sin(\theta_e) \cos(\hat{\theta}_e) - E \cos(\theta_e) \sin(\hat{\theta}_e) = E \sin(\theta_e - \hat{\theta}_e) \quad (35)$$

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 35](#) can be simplified as [Equation 36](#).

$$\varepsilon = E(\theta_e - \hat{\theta}_e) \quad (36)$$

Further, the position error after the normalization of the EEMF can be obtained ([Equation 37](#)):

$$\varepsilon_n = \theta_e - \hat{\theta}_e \quad (37)$$

According to the analysis, the simplified block diagram of the quadrature phase-locked loop position tracker can be obtained as shown in [Figure 3-17](#). The closed-loop transfer functions of the PLL can be expressed as [Equation 38](#):

$$\frac{\hat{\theta}_e}{\theta_e} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} = \frac{2\xi\omega_n s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \quad (38)$$

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 39](#):

$$k_p = 2\xi\omega_n, \quad k_i = \omega_n^2 \quad (39)$$

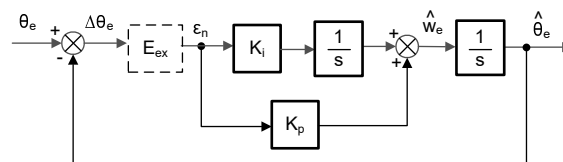


Figure 3-17. Simplified Block Diagram of Phase-Locked Loop Position Tracker

3.1.3 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 40 and Equation 41.

$$v_d = L_d \frac{di_d}{dt} + R_s i_d - p \omega_m L_q i_q \quad (40)$$

$$v_q = L_q \frac{di_q}{dt} + R_s i_q + p \omega_m L_d i_d + p \omega_m \psi_m \quad (41)$$

Figure 3-18 shows the dynamic equivalent circuit of an IPM synchronous motor.

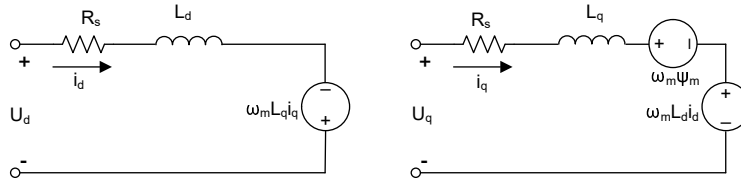


Figure 3-18. Equivalent Circuit of an IPM Synchronous Motor

The total electromagnetic torque generated by the IPMSM can be expressed as Equation 42 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.

$$T_e = \frac{3}{2} p [\psi_m i_q + (L_d - L_q) i_d i_q] \quad (42)$$

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 43 and Equation 44.

$$I_a = \sqrt{i_d^2 + i_q^2} \leq I_{\max} \quad (43)$$

$$V_a = \sqrt{v_d^2 + v_q^2} \leq V_{\max} \quad (44)$$

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 45 if Space Vector Modulation (SVPWM) is adopted.

$$\sqrt{v_d^2 + v_q^2} \leq v_{\max} = \frac{v_{dc}}{\sqrt{3}} \quad (45)$$

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 46 is obtained as shown.

$$\sqrt{L_d^2 \left(i_d + \frac{\psi_{pm}}{L_d} \right)^2 + L_q^2 i_q^2} \leq \frac{V_{\max}}{\omega_m} \quad (46)$$

The current limitation of Equation 43 produces a circle of radius I_{\max} in the d-q plane, and the voltage limitation of Equation 44 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 3-19](#).

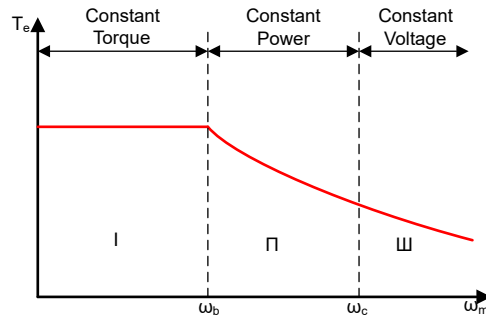


Figure 3-19. IPMSM Control Operation Regions

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 42](#), 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.

$$I_s = \sqrt{i_d^2 + i_q^2} \quad (47)$$

$$I_d = I_s \cos \beta \quad (48)$$

$$I_q = I_s \sin \beta \quad (49)$$

where

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

[Equation 42](#) can be expressed as [Equation 50](#) where I_s substituted for i_d and i_q .

[Equation 50](#) shows that motor torque depends on the angle of the stator current vector:

$$T_e = \frac{3}{2} p I_s \sin \beta [\psi_m + (L_d - L_q) I_s \cos \beta] \quad (50)$$

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 51](#).

$$\frac{dT_e}{d\beta} = \frac{3}{2} p [\psi_m I_s \cos \beta + (L_d - L_q) I_s^2 \cos 2\beta] = 0 \quad (51)$$

Following this equation, the current angle of the MTPA control can be derived as in [Equation 52](#).

$$\beta_{\text{mtpa}} = \cos^{-1} \frac{-\psi_m + \sqrt{\psi_m^2 + 8 \times (L_d - L_q)^2 \times I_s^2}}{4 \times (L_d - L_q) \times I_s} \quad (52)$$

Thus, the effective d-axis and q-axis reference currents can be expressed by [Equation 53](#) and [Equation 54](#) using the current angle of the MTPA control.

$$I_d = I_s \times \cos \beta_{\text{mtpa}} \quad (53)$$

$$I_q = I_s \times \sin \beta_{\text{mtpa}} \quad (54)$$

However, as shown in [Equation 52](#), 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 L_d and 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 L_d changes with i_d only, and L_q changes with 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 55](#) and [Equation 56](#).

$$L_d = f_1(i_d, i_q) = f_1(i_d) \quad (55)$$

$$L_q = f_2(i_q, i_d) = f_2(i_q) \quad (56)$$

Reduce the ISR calculation burden by simplifying [Equation 52](#). The motor-parameter-based constant, K_{mtpa} is expressed instead as [Equation 57](#), where K_{mtpa} is computed in the background loop using the updated L_d and L_q .

$$K_{\text{mtpa}} = \frac{\psi_m}{4 \times (L_q - L_d)} = 0.25 \times \frac{\psi_m}{(L_q - L_d)} \quad (57)$$

$$\beta_{\text{mtpa}} = \cos^{-1} \left(K_{\text{mtpa}} / I_s - \sqrt{(K_{\text{mtpa}} / I_s)^2 + 0.5} \right) \quad (58)$$

A second intermediate variable, G_{mtpa} described in [Equation 59](#), is defined to further simplify the calculation. Using G_{mtpa} , the angle of the MTPA control, β_{mtpa} can be calculated as [Equation 60](#). These two calculations are performed in the ISR to achieve a real current angle β_{mtpa} .

$$G_{\text{mtpa}} = K_{\text{mtpa}} / I_s \quad (59)$$

$$\beta_{\text{mtpa}} = \cos^{-1} \left(G_{\text{mtpa}} - \sqrt{G_{\text{mtpa}}^2 + 0.5} \right) \quad (60)$$

In all cases, the magnetic flux can be weakened to extend the achievable speed range by acting on the direct axis current 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 i_d . Considering the voltage and current constraints, the armature current and the terminal voltage are limited as [Equation 43](#) and [Equation 44](#). 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 46](#),

the IPMSM has two factors: one is a permanent magnet value and the other is made by inductance and current of flux.

Figure 3-20 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 i_d and 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.

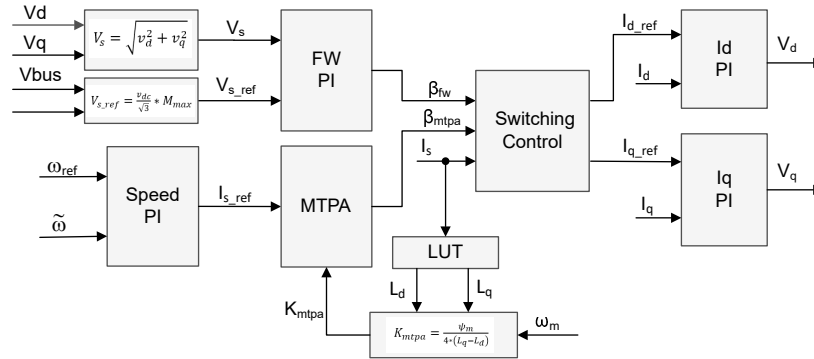


Figure 3-20. Block Diagram of Field-Weakening and Maximum Torque per Ampere Control

Figure 3-9 and Figure 3-11 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 3-21.

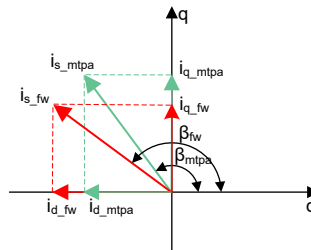


Figure 3-21. Current Phasor Diagram of an IPMSM During FW and MTPA

The switching control module is used to determine angle of application, and then calculate the reference i_d and i_q as shown in Equation 48 and Equation 54. The current angle is chosen as in the following: Equation 61 and Equation 62.

$$\beta = \beta_{fw} \text{ if } \beta_{fw} > \beta_{mtpa} \tag{61}$$

$$\beta = \beta_{mtpa} \text{ if } \beta_{fw} < \beta_{mtpa} \tag{62}$$

The flow chart in Figure 3-22 shows the steps required to run InstaSPIN™-FOC with FW and MPTA in the main loop and interrupt.

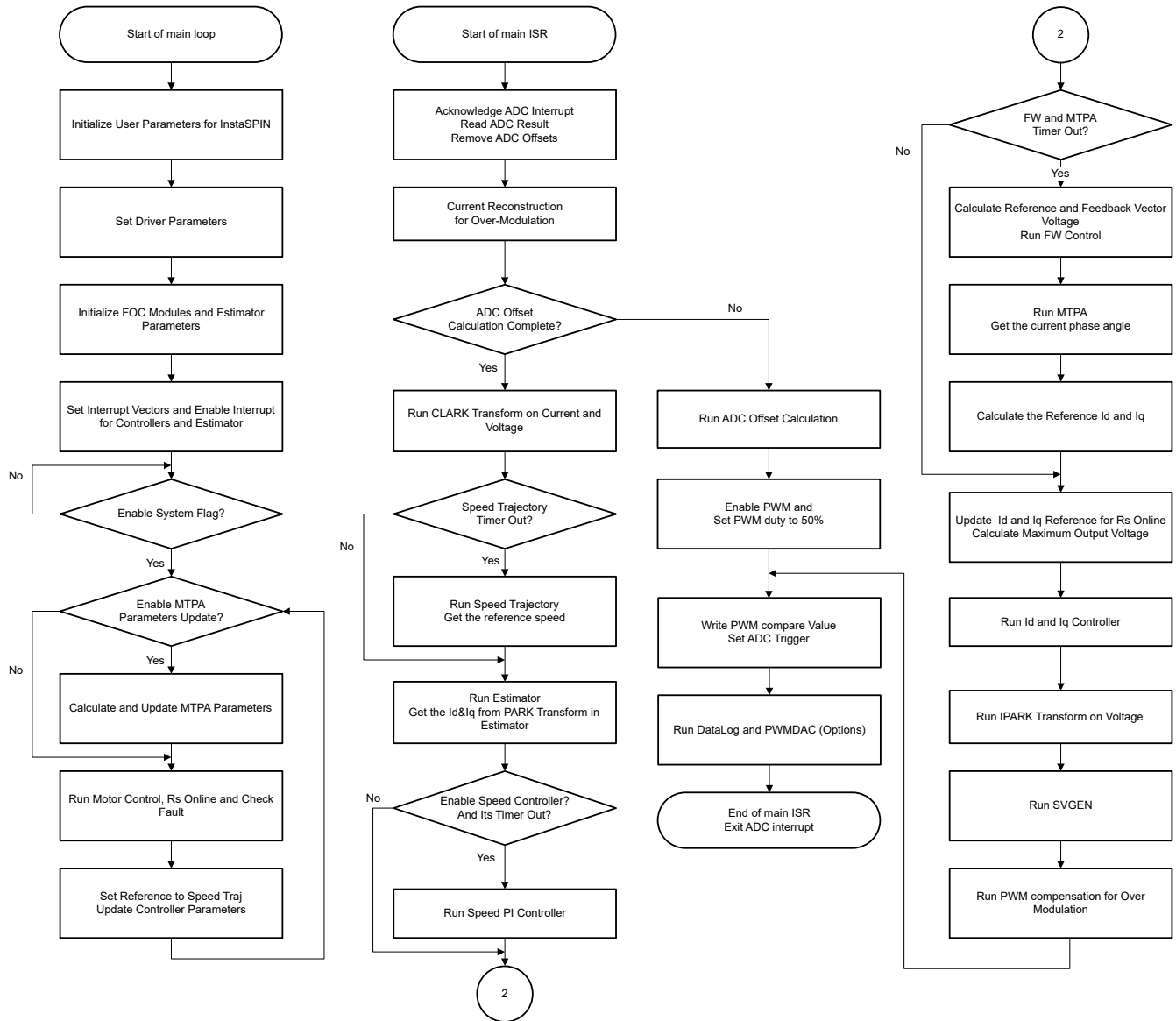


Figure 3-22. Flow Chart for an InstaSPIN-FOC Project With FW and MTPA

3.2 Getting Started Software

3.2.1 GUI

For an in-depth getting started guide for the MCU Motor Control GUI, refer to [this document](#).

Source code for this EVM is provided to allow direct debugging of the firmware. However, software debugging is often a lengthy process. To speed up development time, a UART-based GUI software is provided to help quickly tune parameters on any customized application.

Please note that the intent of the design is for the GUI to connect via the XDS110ISO-EVM UART connection or the isolated UART connection J14, present on most board revisions. If instead connecting directly to the UART pins available on the daughterboard, **the ground is HOT** and an external isolator must be used when connecting grounded equipment to the kit.

3.2.2 Download and Install C2000 Software

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. Download and install the [C2000WARE-MOTORCONTROL-SDK](#). Version v5.01.00.00 or newer is recommended.
3. Once installation is complete, open CCS and create a new workspace for importing the project.

Note

This EVM 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. Refer to the [C2000 SysConfig Software Guide](#) for implementation details.

3.2.3 Using the Software

The project folder for the F280013x daughterboard is in the C2000Ware Motor Control SDK, located at: <install_location>\solutions\tida_010265_wminv\f280013x. Follow these steps to build and run this code with different incremental builds.

1. Click *Project* → *Import CCS Projects*. Click browse and navigate to the project folder for the daughterboard controller.
2. Select the correct build configuration, as shown in [Figure 3-23](#).
 - a. Use the default *Flash_MtrInv_3SC* build configuration for three-shunt current sensing.
 - b. *Flash_MtrInv_1SC* supports single-shunt current sensing.
3. Configure the project to enable and disable the supporting functionalities by selecting the correct pre-defined symbols, as shown in [Figure 3-24](#).

If using board revision E1, you **must** add *TIEVM_MTR-HVINV_REV_E1* as a pre-defined symbol. Without this symbol, the board will not function as expected.

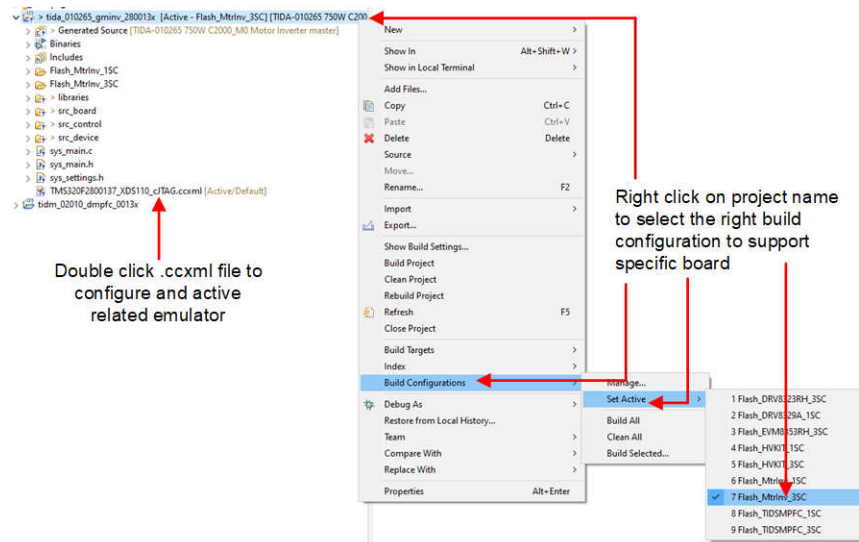


Figure 3-23. Select the Correct Build Configurations

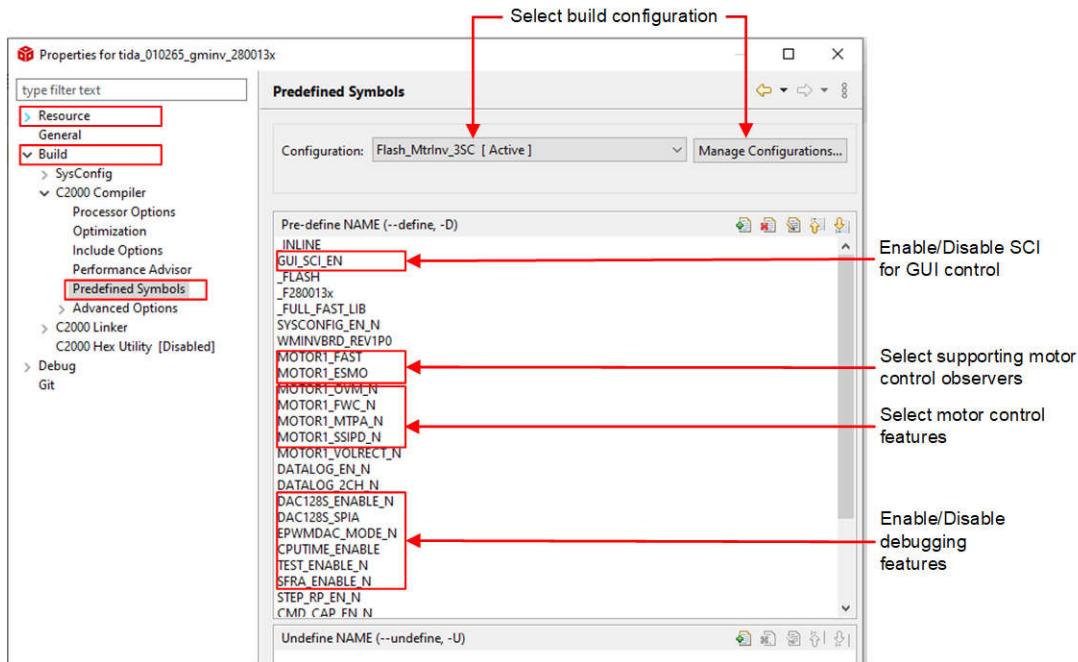


Figure 3-24. Select the Correct Predefined Symbols in Project Properties

3.2.4 Project Structure

Once the project is imported, the project explore appears inside CCS as shown in [Figure 3-25](#).

The folder *src_board* includes device peripheral configuration. This configuration is done with driverlib from C2000Ware. MCU configuration is generally located in the **Hardware Access Layer** (HAL) files, *hal.c* and *hal.h*. This folder also contains *user_mtr1.c*, in which various parameters are calculated or assigned. In most use cases, any parameters fully defined or derived in this file will not need to be adjusted by the user.

The folder *src_control* includes motor drive files that call motor control core algorithm functions within the interrupt service routines and background tasks. This folder also holds various secondary functionalities, such as SFRA, the GUI interface, and so forth.

The folder *src_control/common/include* holds various header files useful to the system. Of these, the most notable is *user_mtr1.h*, which contains the motor and system parameter definitions.

The folder *src_device* holds driverlib, used by the HAL.

The folder *src_sta* holds self-test application code.

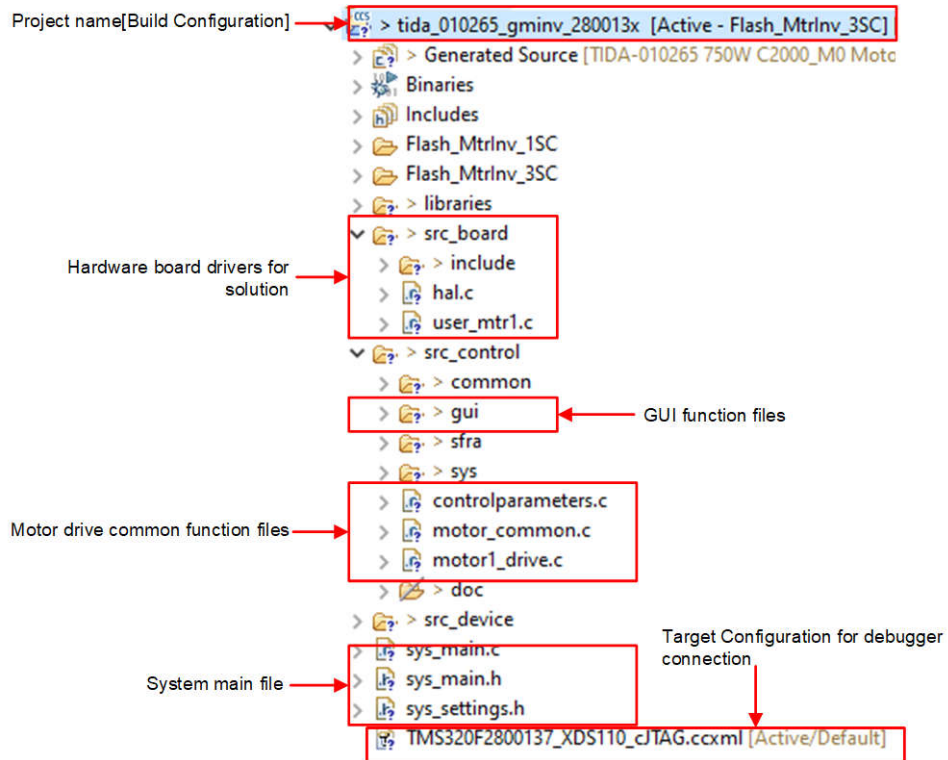


Figure 3-25. TIEVM-MTR-HVINV Project Explorer View

Figure 3-26 shows the project software flow diagram of ISR for motor control, a main loop for motor control parameters update in background loop.

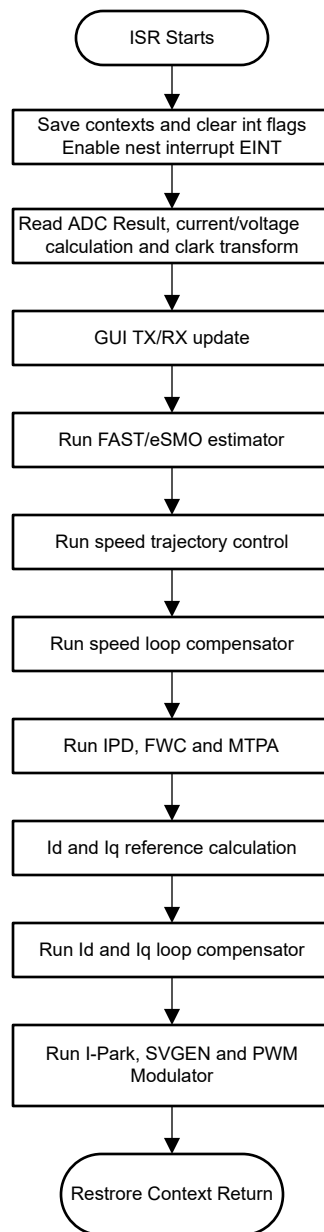


Figure 3-26. Firmware Project Flow Diagram

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 the user-defined rate of `USER_M1_ISR_FREQ_Hz`.

To simplify the system, the design of the software for this EVM is organized into 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 4.5](#) 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

4 Test Procedure and Results

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 4-1](#).

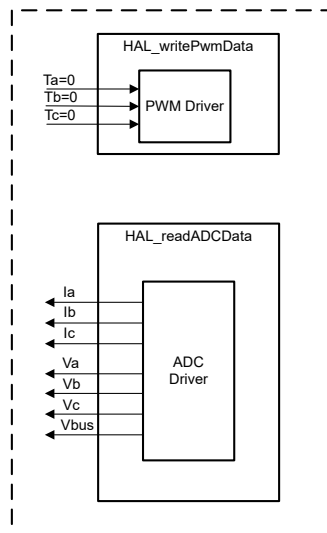


Figure 4-1. Control Software Block Diagram: Build Level 1 – Offset Validation

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 4-2](#) shows the software flow for this build level.

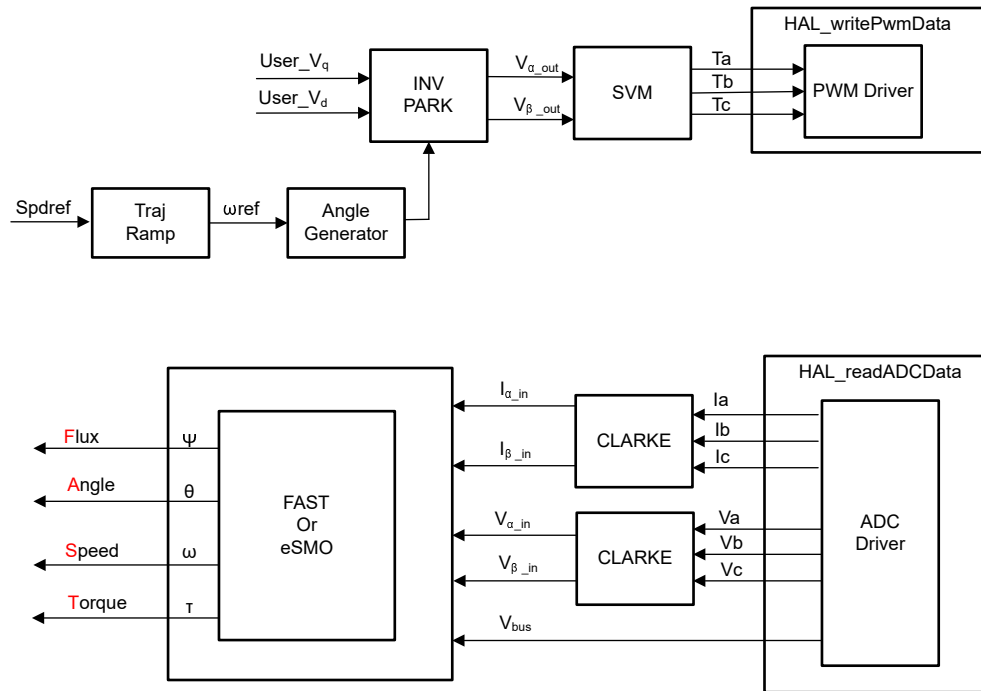


Figure 4-2. Control Software Block Diagram: Build Level 2 – Open-Loop Control

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 and the rotor angle is generated from ramp generator module. [Figure 4-3](#) shows the software flow for this build level.

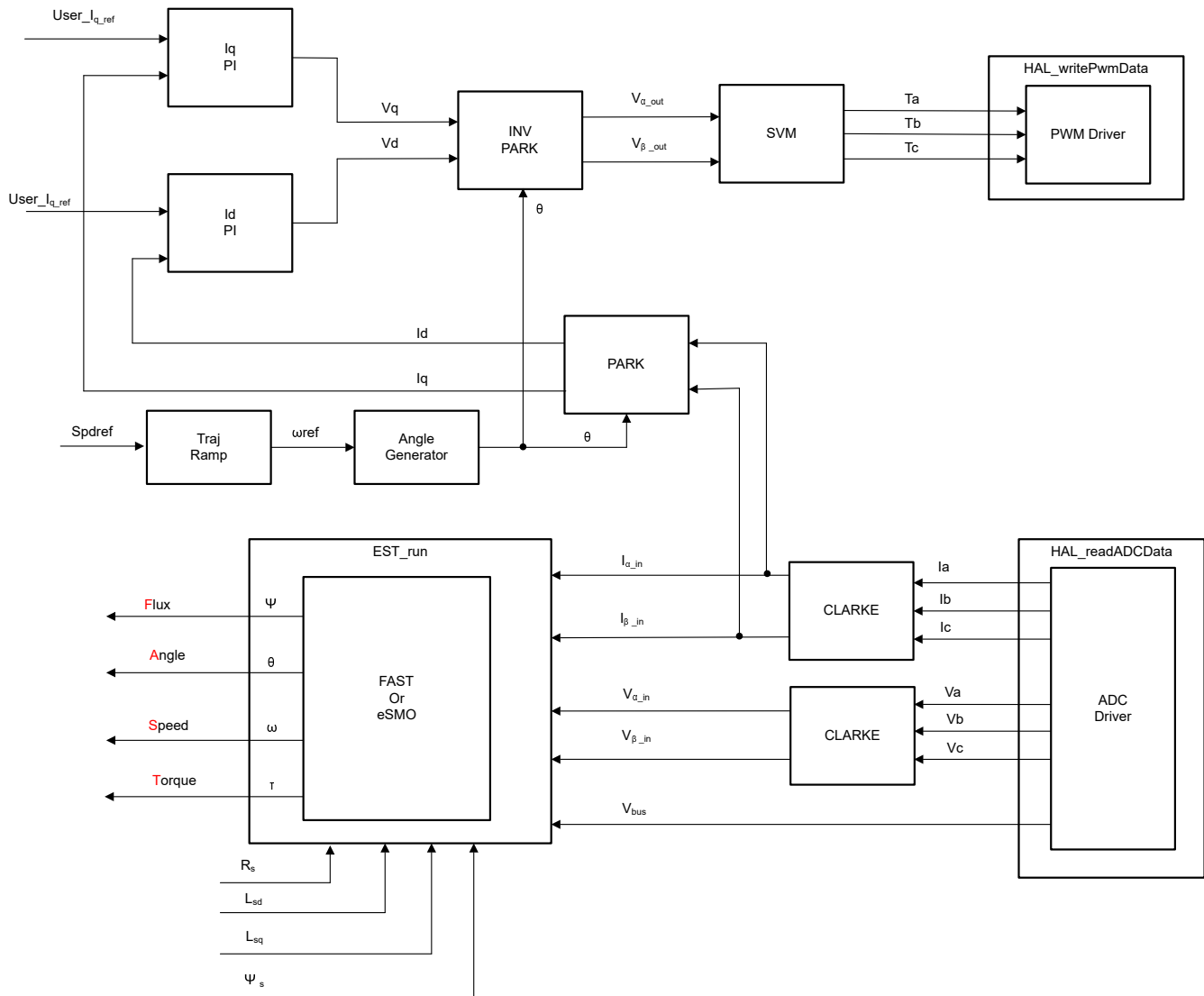


Figure 4-3. Control Software Block Diagram: Build Level 3 – Current Close-Loop Control

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 4-4](#) shows the software flow for this build level.

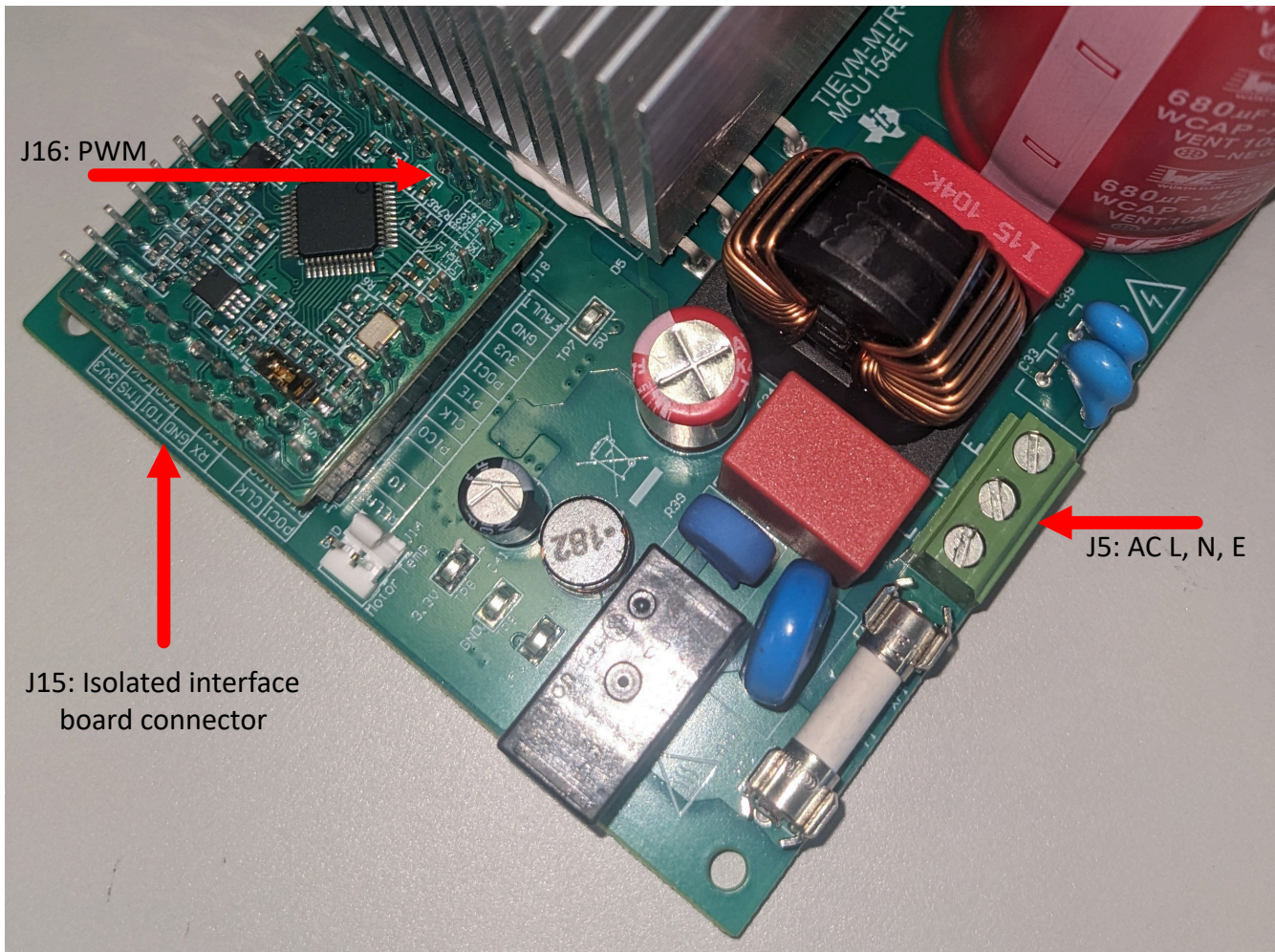


Figure 4-5. Connect the External AC or DC Power Supply to Verify the Hardware

4.5.2 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 described in [Figure 3-24](#).
2. Open the *sys_settings.h* file and set *DMC_BUILDLEVEL* to the selected build level. The build levels are described in [Table 3-1](#).
 - a. *DMC_LEVEL_1*
 - b. *DMC_LEVEL_2*
 - c. *DMC_LEVEL_3*
 - d. *DMC_LEVEL_4*
3. Build Levels 2, 3, and 4 require the motor physical parameters to be accurately known.
 - a. The required motor parameters must be recorded in the header files (*user_mtr1.h*) as shown in the following example codes. These are not the only motor parameters, this is an excerpt for example purposes.

```
#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)
```

- b. If the four specific motor parameters listed above are not accurately known, FAST motor identification can be used in Build Level 4 to determine the motor parameters if the FAST estimator is utilized. Do the following, then refer to the Build Level 4 instructions for further information.


- i. In `main()`, change the `userParams_M1.flag_bypassMotorId` value to "false" to enable motor identification, as shown in the following example code.

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

Set the following identification parameters to appropriate values in the `user_mtr1.h` header file according to the specifications of the motor, if known.

```
#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
```

4. 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. Wait until the project builds successfully.
5. In the *Project Explorer* window, right-click on the .ccxml file and select 'Set as Active Target Configuration'
6. Turn on the AC or DC power supply to apply 30 VAC or 40 VDC to J5, generating the +15 V and +3.3 V for

the controller and gate driver. Click on the Debug button  or click *Run* → *Debug*. The code for the selected build level can be compiled and loaded onto 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. When launched, the program stops at the start of `main()`.


4.5.3 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. The primary debugger tool in CCS is the Expressions window.

1. Click *View* → *Expressions* on the menu bar to open an *Expressions* watch window.
2. 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`. There are 5 different .txt files available. Select the one corresponding to your selected build level and click the *OK* button to import the variables into the expressions window.


Note



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

3. Alternatively, variables can also be manually added to and removed from the expressions window.
4. The structure variable `motorVars_M1` has references to most variables that are related to controlling the motor. By expanding this variable, you can see and edit all of these variables as needed.
5. 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.

4.5.4 Run the Code


To run the project code, complete the following steps:

1. Slowly adjust the output voltage of the AC supply from 30-VAC to the build-dependent test value.
 - a. Build 1: Do not adjust the output voltage.
 - b. Build 2: 100-VAC
 - c. Build 3 and Build 4: 220-VAC
2. Run the project by clicking the button , or click *Run* → *Resume* in the *Debug* tab.

3. In the *Expressions* window, wait until `systemVars.flagEnableSystem` is automatically set to "1". Set `motorVars_M1.flagEnableRunAndIdentify` to "1".
4. In the *Expressions* window, the variable `motorVars_M1.flagRunIdentAndOnLine` is set to "1" automatically. *ISRCount* is increasing continuously.
5. The project can now run with the values in the *Expressions* window continuously updating. All CCS windows can be resized according to user preference.
6. Perform build-dependent test procedures.
 - a. [Section 4.5.4.1](#)
 - b. [Section 4.5.4.2](#)
 - c. [Section 4.5.4.3](#)
 - d. [Section 4.5.4.4](#)
7. 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 4-6](#). This operation erases all of the program code stored in flash. (This step is optional).

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*.

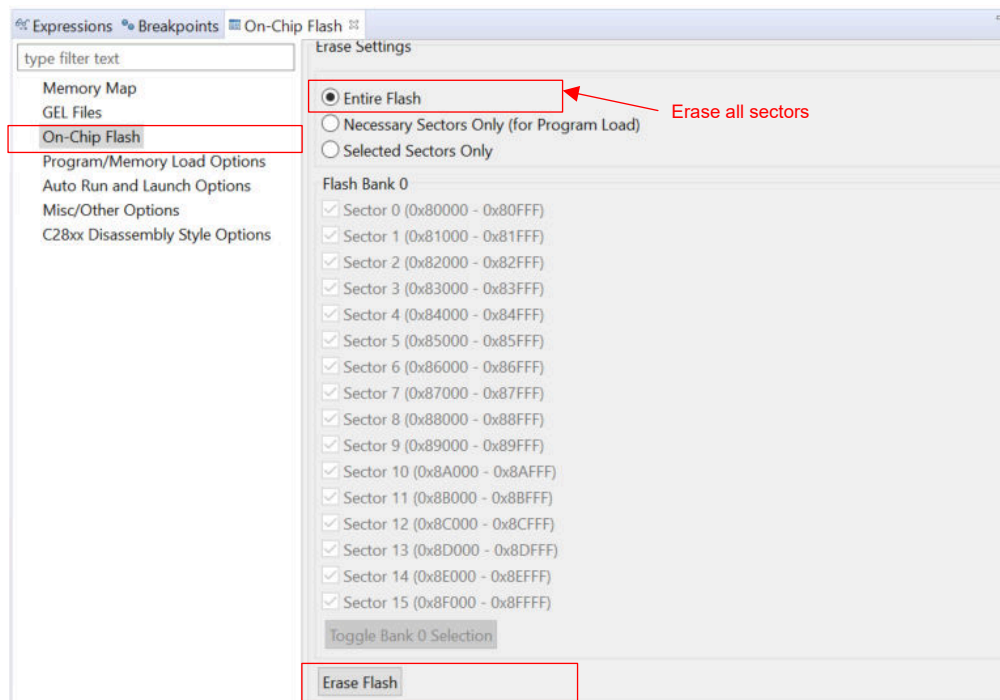


Figure 4-6. Erase Program Code in Flash for Next Build Level

4.5.4.1 Build Level 1 Test Procedure

1. Ensure initial steps listed in [Section 4.5.4](#) have been completed.
2. 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 4-7](#).
3. Probe the PWM output for motor drive control with an oscilloscope at J15 as shown in [Figure 4-8](#). All of the PWM duty are set to 50% in this build level, the PWM output waveforms are as shown in [Figure 4-9](#). The PWM switching frequency of `motor_1` is 15 kHz.

Expression	Type	Value	Address
motorVars_M1.ISRCCount	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 '\u001'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\u001'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	10.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\u001'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\u001'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\u000'	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_ADCCData_t	{VdcBus_V=40.7483864, I_A=(value=[0.218435556,0.175398439,-0.0...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\u000'	0x0000891D@Data
motorVars_M1.faultMtrUseAll	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 '\u000'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\u000'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E0@Data
motorCtrlVars_M1.lqSet_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 4-7. Build Level 1: Expressions Window at Run Time

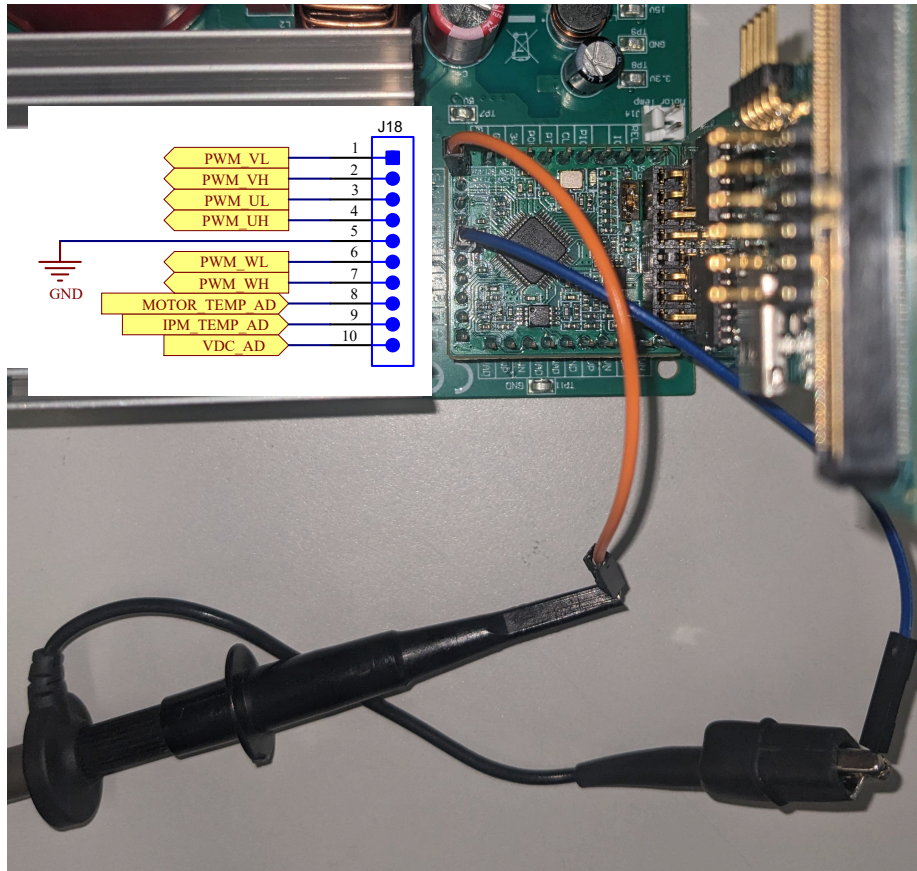


Figure 4-8. Probing PWM Outputs

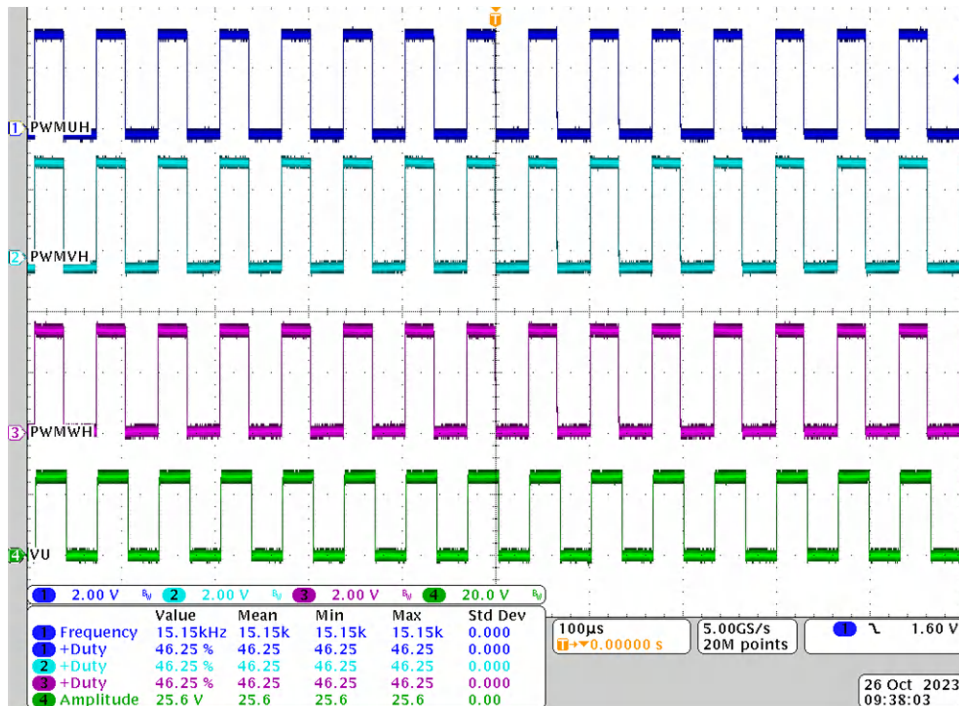


Figure 4-9. Build Level 1: MCU PWM Output and IPM Output

4.5.4.2 Build Level 2 Test Procedure

1. Ensure initial steps listed in Section 4.5.4 have been completed.
2. To verify the current and voltage-sensing circuit of the inverter for the motor, 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
```

3. 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 4-10.
4. Connect the oscilloscope voltage and current probes to watch the motor phase voltage and current as shown in Figure 4-11.
5. 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".

Expression	Type	Value	Address
motorVars_M1.ISRCCount	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.faultMtrUseAll	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	0x000086ED@Data
motorCtrlVars_M1.lqSet_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 4-10. Build Level 2: Expressions Window at Run Time

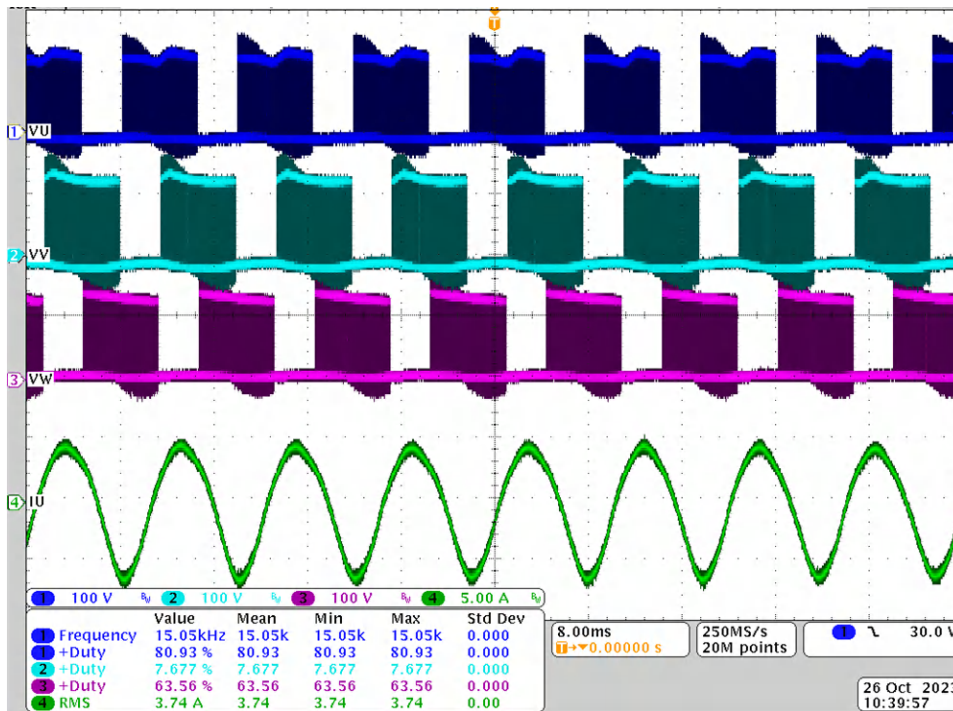


Figure 4-11. Build Level 2: Motor Phase Voltage and Current

4.5.4.3 Build Level 3 Test Procedure

1. Ensure initial steps listed in Section 4.5.4 have been completed.
2. 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, the value needs to be very close to the target *motorVars_M1.speedRef_Hz*.
3. The motor current Iq can be adjusted with *motorVars_M1.Idq_Set_A.value[1]*.
4. Connect oscilloscope probes to IPM output to watch the motor phase voltage and current. Change the *Idq_set_A[0].value[1]* in the Expressions window, and observe that the motor phase current increases accordingly.

Expression	Type	Value	Address
motorVars_M1.ISRCCount	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=308.7208251_A=(values=[-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	0x00008994@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008996@Data
motorVars_M1.Ls_q_H	float	0.00926135667	0x00008998@Data
motorVars_M1.flux_VpHz	float	0.378431171	0x0000899A@Data
motorVars_M1.flux_Vb	float	0.0603588633	0x0000899C@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 4-12. Build Level 3: Expressions Window at Run Time

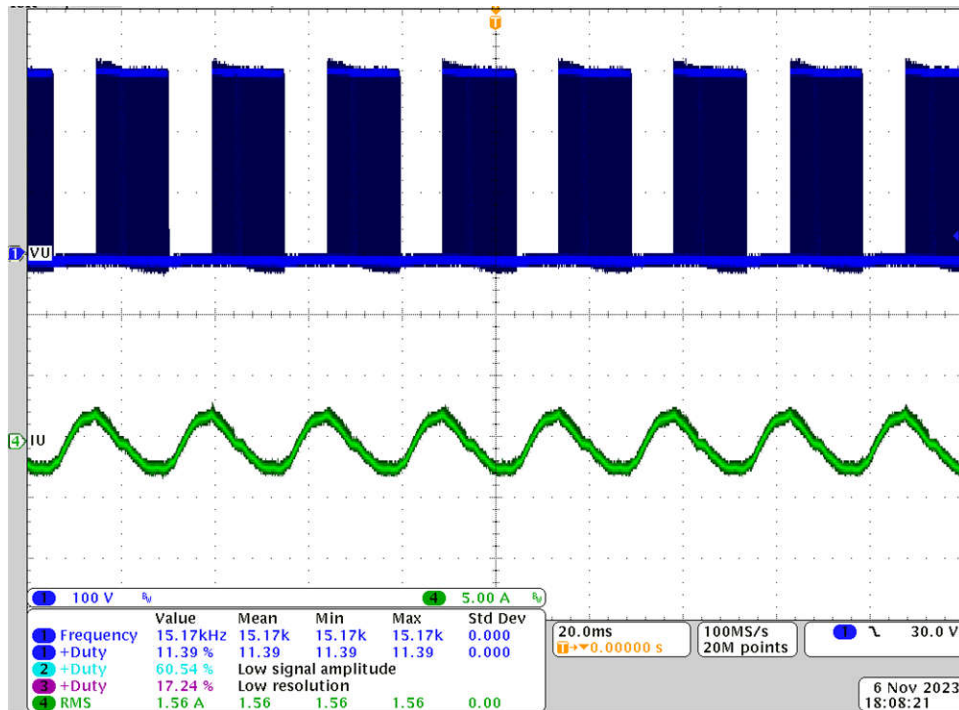


Figure 4-13. Build Level 3: Motor Current Under 2-A I_Q Setting

4.5.4.4 Build Level 4 Test Procedure

1. Ensure initial steps listed in Section 4.5.4 have been completed.
2. If the motor identification routine is being utilized, as described in Section 4.5.2, the motor identification routine begins execution immediately upon setting `motorVars_M1.flagEnableRunAndIdentify` to "1" in the Expressions window. This process takes about 150 seconds.
 - a. 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
 - b. Set `userParams_M1.flag_bypassMotorId` to "true" after successfully identify the motors parameters, rebuild the project and load the code into the controller.
3. Set the variables `motorVars_M1.speedRef_Hz` to a different value and watch how the motor shaft speed follows.
4. To change the acceleration, enter a different acceleration value for the variables `motorVars_M1.accelerationMax_Hzps` and `motorVars_M1.accelerationMax_Hzps`.

Expression	Type	Value	Address
motorVars_M1.ISRCCount	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	0x00008928@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	0x00008984@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.lqSet_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	0x00008990@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	0x00008994@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008996@Data
motorVars_M1.Ls_q_H	float	0.00926135667	0x00008998@Data
motorVars_M1.flux_VpHz	float	0.389726102	0x0000899A@Data
motorVars_M1.flux_Wb	float	0.0620205812	0x0000899C@Data
angleGen_M1	struct_ANGLE_GEN_Obj_	{freq_Hz=100.0,angleDeltaFactor=0.000418879034,angleDelta_rad...	0x0000861C@Data

Figure 4-14. Build Level 4: Expressions Window at Run Time

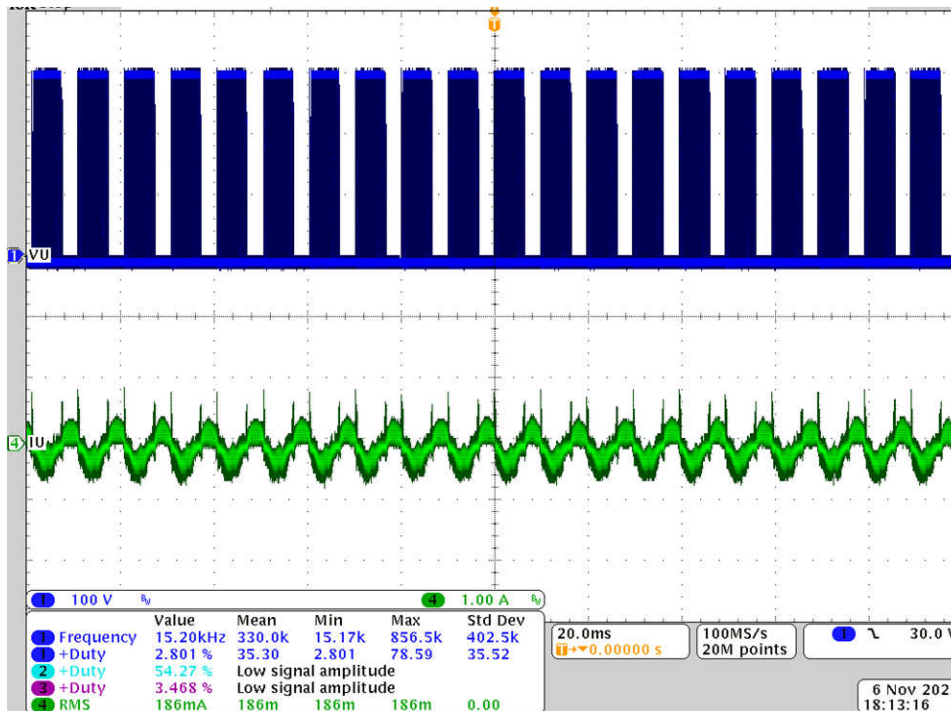


Figure 4-15. Build Level 4: Rotor Angle, Phase Current of Motor

After these initial tests, Build Level 4 is also where any other test conditions and tuning are optionally performed.

1. [Section 4.5.4.4.1](#)
2. [Section 4.5.4.4.2](#)
3. [Section 4.5.4.4.3](#)

4.5.4.4.1 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 R_s , and L_s as described in [Section 3.1.2.1.2](#). 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_lq, motorVars[0].Ki_lq, motorVars[0].Kp_ld, and motorVars[0].Ki_ld to the *Expressions* window in CCS Debug Perspective. Change the PI gains for the compressor motor drive and record the values.

4.5.4.4.2 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 3.2.3](#) 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.

4.5.4.4.3 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)

```

4.6 Performance Data and 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.

4.6.1 Load and Thermal Test

Figure 4-16 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

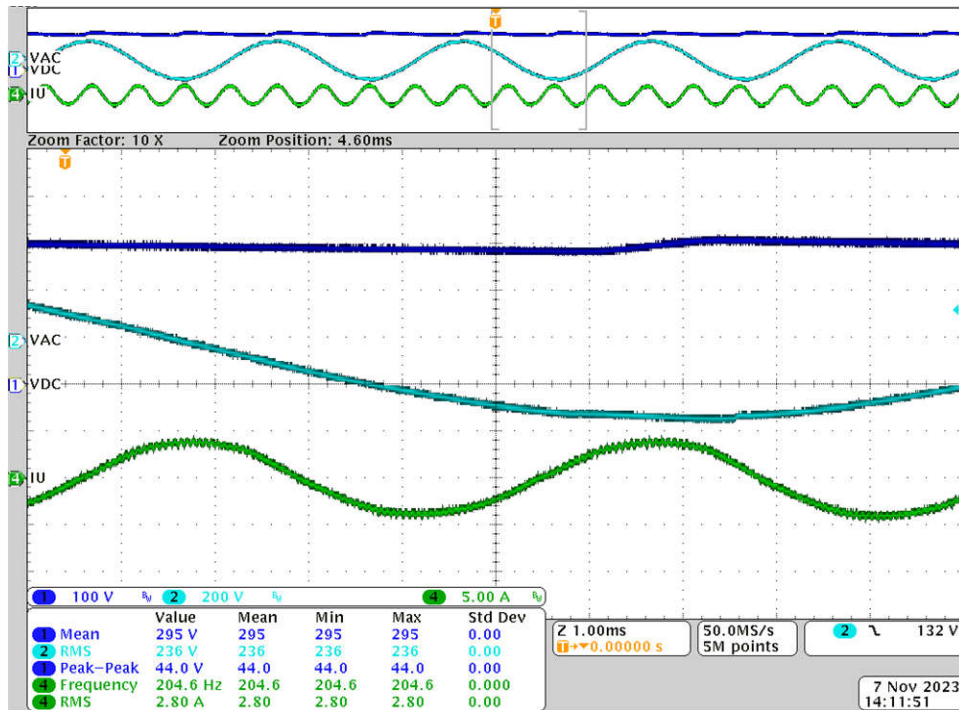


Figure 4-16. Phase Current and Voltage Waveforms of Motor at 500 W, 200 Hz

Figure 4-17 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

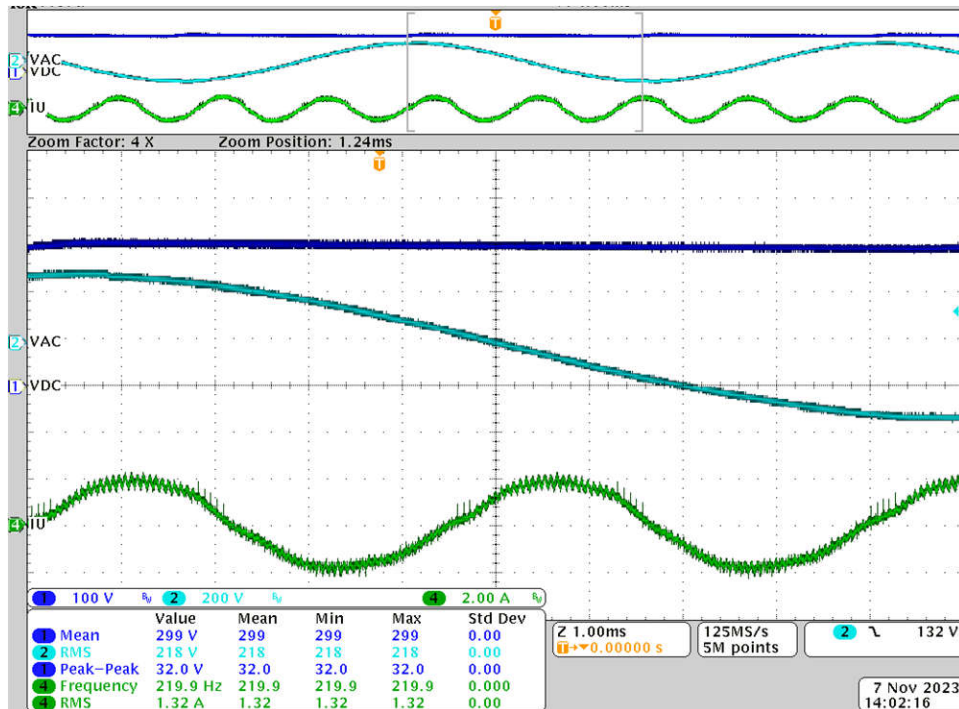


Figure 4-17. Field Weakening Test at 300 W, 220 Hz

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 4-18](#) shows the board temperature rising at 500 W, 3000 RPM (200 Hz).

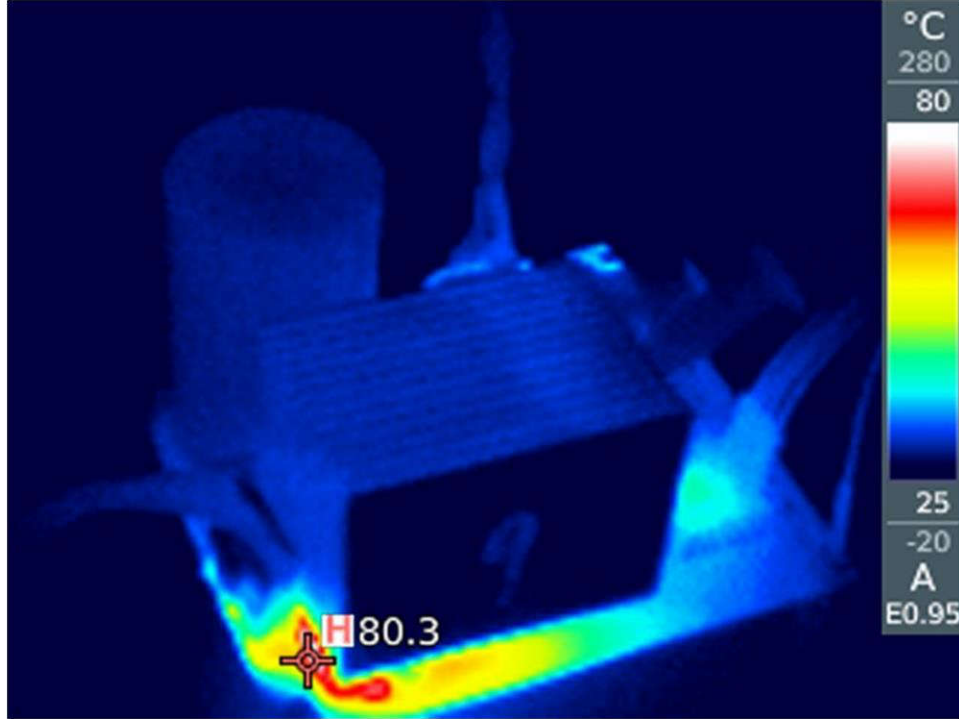


Figure 4-18. Thermal Test Under 220 VAC, 500 W, 200 Hz

4.6.2 Overcurrent Protection by External Comparator

As explained in [Section 2.1.5](#), there is a comparator U10 for external overcurrent protection. [Figure 4-19](#) shows the external 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

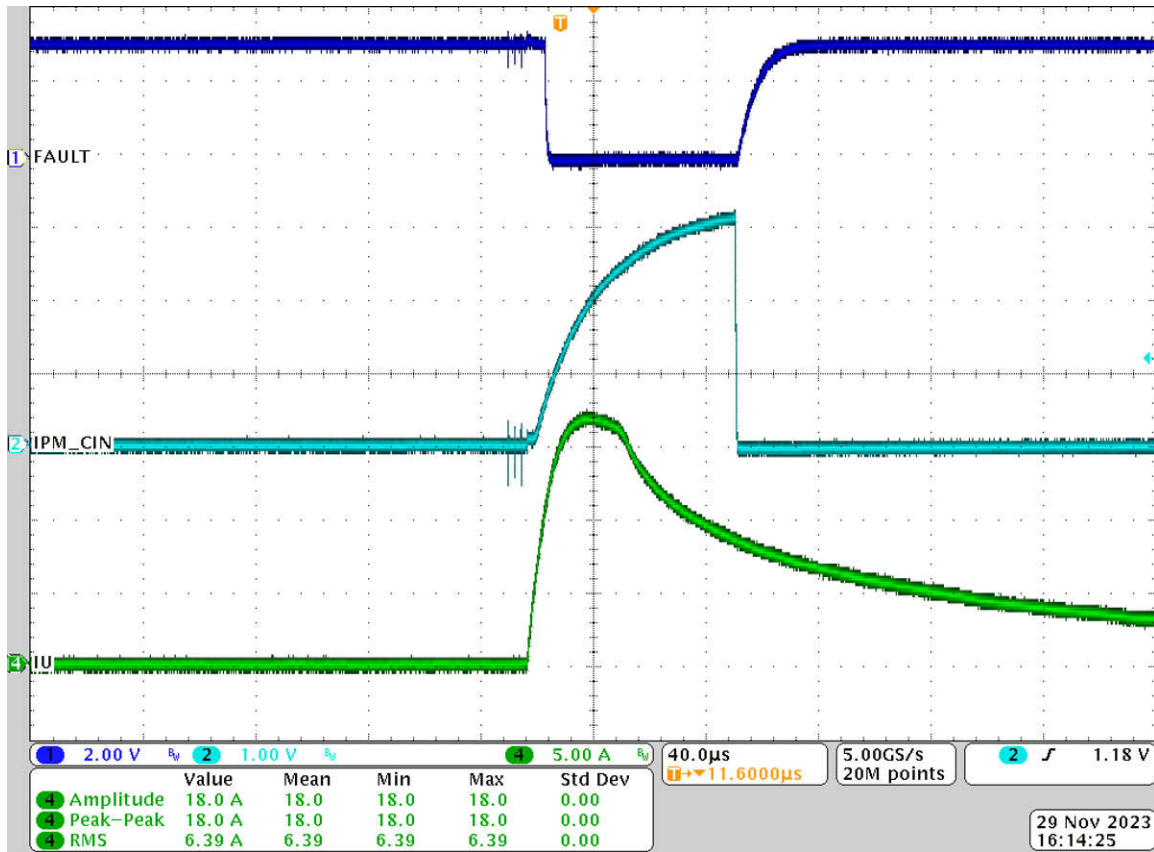


Figure 4-19. Overcurrent Protection by External Comparator

4.6.3 Overcurrent Protection by Internal CMPSS

As explained in Section 2.1.6, the internal CMPSS can be configured for overcurrent protection. Figure 4-20 shows the internal overcurrent protection waveform, is triggered by internal CMPSS, since both IPM_FAULT 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

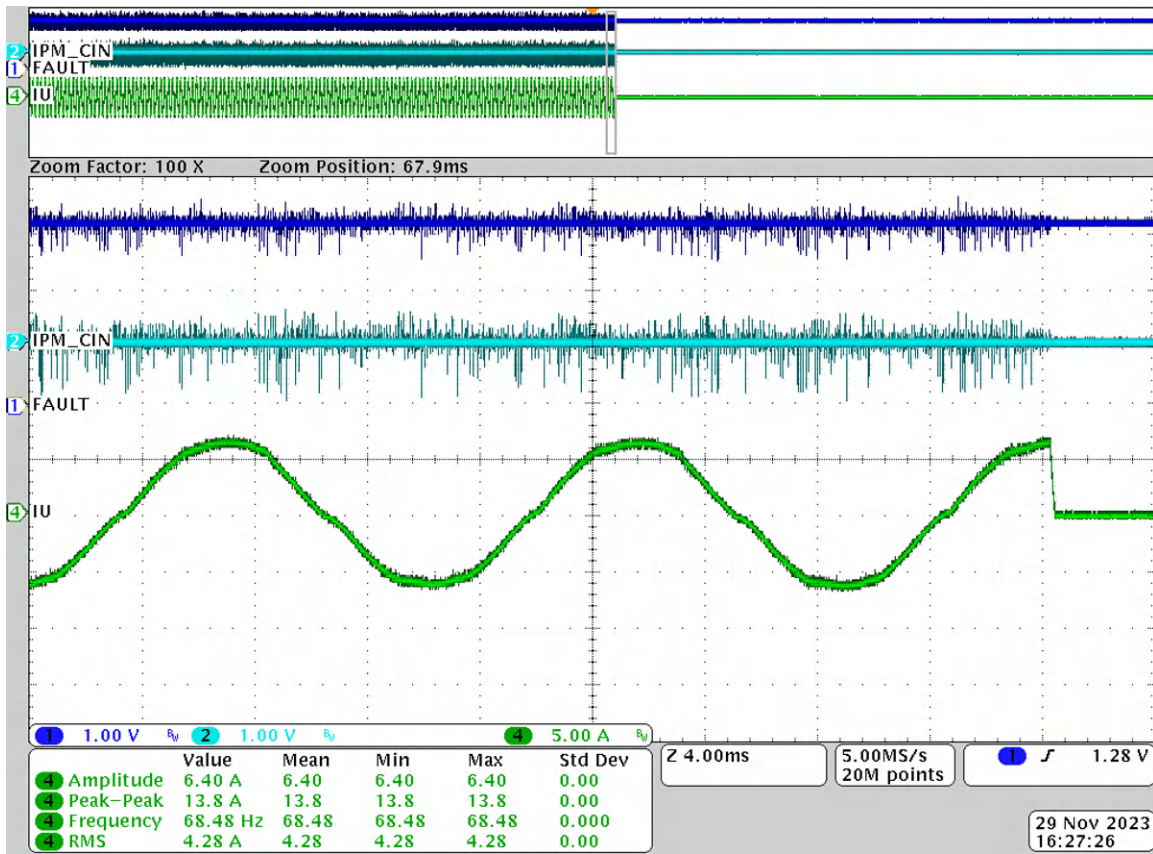


Figure 4-20. Overcurrent Protection by Internal CMPSS

5 Hardware Design Files

5.1 Schematics

The TIEVM-MTR-HVINV hardware design files, including the schematics, are available on the product page for the TIEVM-MTR-HVINV on ti.com: [TIEVM-MTR-HVINV](#)

5.2 PCB Layouts

The TIEVM-MTR-HVINV hardware design files, including the PCB layout, are available on the product page for the TIEVM-MTR-HVINV on ti.com: [TIEVM-MTR-HVINV](#)

5.3 Bill of Materials (BOM)

The TIEVM-MTR-HVINV hardware design files, including the BOM, are available on the product page for the TIEVM-MTR-HVINV on ti.com: [TIEVM-MTR-HVINV](#)

6 Additional Information

6.1 Known Hardware or Software Issues

Certain board revisions are known to have specific hardware issues.

- Daughterboard external precision R resistor R6, which is not populated by default, must not be populated to avoid spurious illegal ISR entries. This is applicable to all revisions in which R6 (connected to GPIO19) is present, as well as to any and all designs which utilize the external R functionality.
- Board revision E1 has the following issues. Later board revisions, such as revision A, do not have these issues.
 - Known Hardware Issues
 - Resistor R108 is known to have a value of 2.4kΩ. This is incorrect, and should be 1kΩ. This causes the OCP limit of the external OCP circuit to increase from the designed value (~9.42A) to approx. 21A.
 - In this revision, J14 (motor temp external sensor connector) is present, and J14 (Isolated UART connector) is not. To utilize the UART signals on J17, an external isolator must be used.
 - Known Software Issues
 - To utilize revision E1, the pre-defined symbol *TIEVM_MTR_HVINV_REV_E1* **must** be added. Without this pre-defined symbol, current sensing of the U and W legs are swapped, and the motor cannot run.

6.2 Trademarks

FAST™, InstaSPIN™, and Code Composer Studio™ are trademarks of Texas Instruments. All trademarks are the property of their respective owners.

6.3 Terminology

SLYZ022	TI Glossary: This glossary lists and explains common electrical 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 Torque observer

7 References

1. Texas Instruments, [TMS320F280013x Real-Time 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

8 Revision History

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

Changes from October 1, 2024 to November 30, 2024 (from Revision * (October 2024) to Revision A (November 2024))

	Page
• Added motor recommendation	2
• Updated for revA and standoffs added.....	2
• Added motor recommendation.....	15
• Added motor recommendation used as an example.....	16
• Added standoff instructions.....	16
• Added section	37
• Added revE1-specific instructions.....	38
• Updated link.....	62
• Updated link.....	62
• Updated link.....	62
• Updated revE1-specific instructions	63

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](https://www.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
Copyright © 2024, Texas Instruments Incorporated