Design Guide: TIDM-02010

Dual Motor Control with Digital Interleaved PFC for HVAC Reference Design

Description

The TIDM-02010 reference design is a 1.5-kW dual motor drive and PFC control reference design for variable frequency air conditioner outdoor unit controller in HVAC applications, which illustrates a method to implement sensor less 3-phase PMSM vector control for compressor and fan motor drive, and digital interleaved boost PFC for meeting new efficiency standards with a single C2000™ microcontroller. The hardware and software available with this reference design are tested and ready-to-use to help accelerate development time to market. The hardware design details and test results can be found in this design guide.

Resources

TIDM-02010	Design Folder
TMS320F2800137, TMS320F280025C	Product Folder
TMS320F280039C	Product Folder
UCC28740, UCC27517, TLV9062	Product Folder
C2000WARE-MOTORCONTROL-SDK	Tool Folder

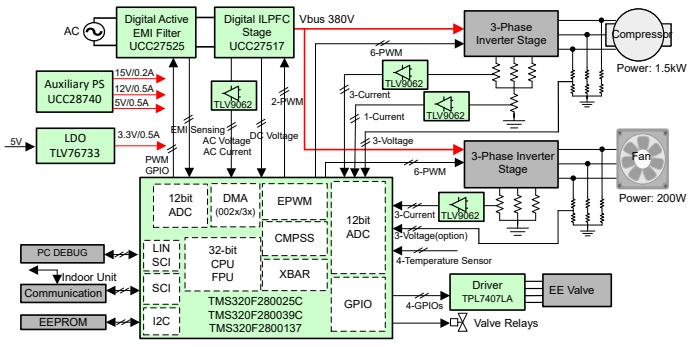
Ask our TI E2E™ support experts

Features

- Wide operating voltage input range: 165 V to 265 VAC, 50/60 Hz.
- Sensorless-FOC for compressor and fan motors plus digital interleaved PFC boost converter control using a single C2000 controller.
- Up to 1.5 kW inverter stage, 6 kHz switching frequency, torque compensation and automatic field weakening control for compressor PMSM.
- Up to 150 W inverter stage, 18kHz switching frequency for fan PMSM, stall detection with auto recovery, flying start and over current protections.
- Up to 1.5 kW power output, 72kHz switching frequency, digital control interleaved boost PFC with power factor >0.95 and <5% THD from medium to full load over entire operating voltage range, robust adjustable voltage output with voltage and current fault protection.

Applications

- Air Conditioner Outdoor Unit
- Refrigerator and Freezer
- Washer and Dryer
- Appliance Compressor



1 System Description

Today’s HVAC applications must meet a growing list of demands on low cost, smaller size, more comfort, more powerful and high energy efficiency. It is well known that PMSM is becoming increasingly popular over induction motor in HVAC applications. This reference design provides a true single chip solution for dual motor drives, including the front-end interleaved boost PFC. Implementation of controlling two motors using field oriented control (FOC) without position sensor, and an interleaved boost PFC using a single controller is illustrated. The overall system helps users to reduce the number of critical components in the bill of materials, reduce the size of bulky passive components, improve efficiency, minimize input-current harmonic content, maximize power factor, precisely regulate the DC bus, and save development time. This reference design based on TMS320F28002x or TMS320F28003x real-time controller family makes it easy to get started on an HVAC system outdoor unit controller design.

The ability to control multiple motors not only reduces system cost but also improves overall power efficiency and performance. For applications that operate dual motors, having both motors controlled by a single MCU enables the controller to coordinate how quickly it ramps one motor up relative to the speed of the other motor. And because both motors draw from the same current source, the PFC implementation can be coordinated as well for better results.

The reference design provides flexibility to support many different types of air conditioner systems, the hardware capabilities for more optional features beside motor drive and PFC like serial communication with indoor unit, stepper motor drive for electronic expansion valve, and temperature sensors for HVAC system control.

1.1 Key System Specifications

The dual motor control with PFC reference design specifications are listed in Table 1-1.

Table 1-1. Key System Specifications

PARAMETERS	TEST CONDITIONS	MIN	NOM	MAX	UNIT
SYSTEM INPUT CHARACTERISTICS					
Input voltage (V_{INAC})	-	165	230	265	VAC
Input Frequency (f_{LINE})	-	47	50	63	Hz
No load standby power (P_{NL})	V_{INAC}=230V, I_{out}=0A	-	3.0	-	W
Input current (I_{IN})	V_{INAC}=230V, I_{out} =I_{MAX}	-	15	-	A
PFC CONVERTER CHARACTERISTICS					
PWM switching frequency (f_{SW})	-	60	72	96	kHz
Output voltage (V_{OUT})	V_{IN}=nom, I_{OUT}=min to max	360	375	385	V
Output current (I_{OUT})	V_{IN}=min to max	-	-	5	A
Line regulation	V_{INAC}=min to max , I_{OUT}=nom	-	-	2	%
Load regulation	V_{INAC}=nom , I_{OUT}=min to max	-	-	3	%
Output voltage ripple	V_{INAC}=nom , I_{OUT}=max	-	-	15	V
Output over voltage	V_{INAC}=min to max	-	-	430	V
DC-Link peak over current (I_{OCP})	V_{INAC}=min	-	-	10	A
Output power at high line	V_{INAC}=250V	-	-	1.8	kW
Output power at low line	V_{INAC}=165V	-	-	1.2	kW
Efficiency (Π)	V_{INAC}=nom at full load	-	97	-	%
COMPRESSOR INVERTER CHARACTERISTICS					
PWM switching frequency (f_{SW})	-	-	6	-	kHz
Rated output power (P_{OUT})	V_{INAC}=nom	-	1.4	1.5	kW
Output current (I_{RMS})	V_{INAC}=nom	-	10	-	A
Inverter efficiency (Π)	V_{INAC}=nom, P_{OUT} =nom	-	98	-	%
Motor electrical frequency (f)	V_{INAC}=min to max	20	200	400	Hz
Fault protections	Over current, Stall, Over temperature, Under voltage, Over voltage				
Drive control method and features	Sensorless-FOC with three shunt resistors for current sensing				
FAN INVERTER CHARACTERISTICS					
PWM switching frequency (f_{SW})	-	-	18	-	kHz
Rated power (P)	V_{INAC}=nom	-	150	200	W
Output current (I_{RMS})	V_{INAC}=nom	-	1	-	A
Inverter efficiency (Π)	V_{INAC}=nom, P_{OUT} =nom	-	98	-	%
Motor electrical frequency (f)	V_{INAC}=min to max	30	50	400	Hz
Fault protections	Over current, Stall with recovery, Under voltage, Over voltage				
Drive control method and features	Sensorless-FOC with three shunt resistors for current sensing				
SYSTEM CHARACTERISTICS					
Build in auxiliary power supply	V_{INAC}=min to max	15V±10%, 300mA / 12V±10%, 500mA / 5V±10%, 500mA			
Operating ambient	Open frame	-10	25	55	ºC
Standards and norms	Power line harmonics	EC 61000-3-2 Class A			
Board size	Length × width × height	197mm * 197mm * 55mm			mm^2

WARNING

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

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

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

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

CAUTION

Do not leave the design powered when unattended.

2 System Overview

2.1 Block Diagram

Figure 2-1 shows the block diagram of this reference design with key TI components highlighted.

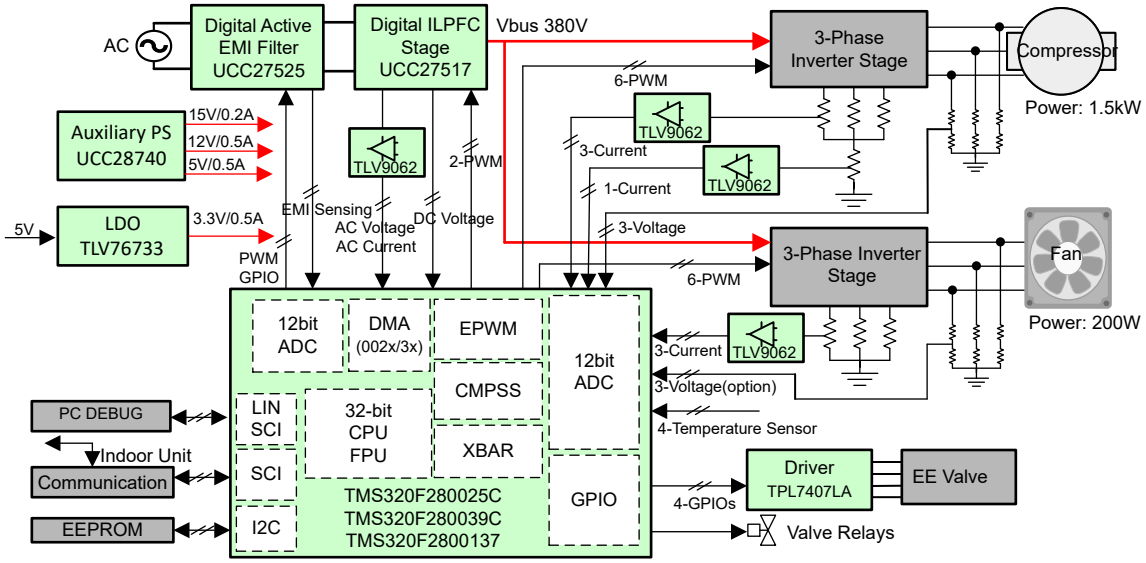


Figure 2-1. TIDM-02010 Dual Motor Control with Interleaved PFC Block Diagram

The entire system is represented in seven blocks:

- Digital active EMI filter
- Digital interleaved PFC
- 3-phase inverter for motor_1 (Compressor)
- 3-phase inverter for motor_2 (Fan)
- MCU Controller
- Valve and relay drive for system control
- Auxiliary power supply

2.2 Design Considerations

The design uses a single C2000 controller for dual motor control and PFC. Highly noise-immune current and voltage sensing solutions are necessary for precise motor drive and PFC control. The following detail the sensing and drive circuit that are used on this design. The hardware design files are available under the C2000Ware Motor Control SDK Install directory at <install_location>\solutions \tidm_02010_dmpfc\hardware.

2.3 Highlighted Products

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

2.3.1 TMS320F2800137

The TMS320F280013x is a member of the C2000™ real-time microcontroller family of scalable, ultra-low latency devices designed for efficiency in power electronics applications. The real-time control subsystem is based on TI's 32-bit C28x 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 VCRC (Cyclical Redundancy Check) 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 optimal 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 multi-level 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 optimal signal placement.

2.3.2 TMS320F280025C

The TMS320F28002x 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 DSP core, which provides 100 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 VCRC (Cyclical Redundancy Check) extended instruction sets, speeding up common algorithms key to real-time control systems. High-performance analog blocks are integrated on the F28002x real-time microcontroller (MCU) and are closely coupled with the processing and PWM units to provide optimal 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 multi-level power topologies. The inclusion of the Configurable Logic Block (CLB) allows the user to add custom logic and potentially integrate FPGA-like functions into the C2000 real-time MCU. Interfacing is supported through various industry-standard communication ports (such as FSI, SPI, SCI, I2C, PMBus, LIN, and CAN) and offers multiple pin-muxing options for optimal signal placement.

2.3.3 TMS320F280039C

The TMS320F28003x (F28003x) 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 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 Floating-Point Unit (FPU), Trigonometric Math Unit (TMU), and VCRC (Cyclical Redundancy Check) extended instruction sets, speeding up common algorithms key to real-time control systems. The CLA allows significant offloading of common tasks from the main C28x CPU. The CLA is an independent 32-bit floating-point math accelerator that executes in parallel with the CPU. The F28003x supports up to 384KB (192KW) of flash memory divided into three 128KB (64KW) banks, which enable programming and execution in parallel. Up to 69KB (34.5KW) of on-chip SRAM is also available to supplement the flash memory. High-performance analog blocks are integrated on the F28003x real-time microcontroller (MCU) and are closely coupled with the processing and PWM units to provide optimal real-time signal chain performance. Sixteen PWM channels, all supporting frequency-independent resolution modes, enable control of various power stages from a 3-phase inverter to power factor correction and advanced multi-level power topologies. The inclusion of the Configurable Logic Block (CLB) allows the user to add custom logic and potentially integrate FPGA-like functions into the C2000 real-time MCU. Interfacing is supported through various industry-standard communication ports (such as SPI, SCI, I2C, PMBus, LIN, CAN and CAN FD) and offers multiple pin-muxing options for optimal signal placement.

2.3.4 UCC28740

The UCC28740 isolated-flyback power-supply controller provides Constant-Voltage (CV) using an optical coupler to improve transient response to large-load steps. Constant-Current (CC) regulation is accomplished through Primary-Side Regulation (PSR) techniques. This device processes information from opto-coupled feedback and an auxiliary flyback winding for precise high-performance control of output voltage and current. An internal 700-V startup switch, dynamically controlled operating states, and a tailored modulation profile support ultra-low standby power without sacrificing startup time or output transient response. Control algorithms in the UCC28740 allow operating efficiencies to meet or exceed applicable standards.

2.3.5 UCC27517

The UCC27517 is a single-channel, high-speed, low-side gate driver device can effectively drive MOSFET and IGBT power switches. Using a design that inherently minimizes shoot-through current, UCC27517 can source and sink high peak-current pulses into capacitive loads offering rail-to-rail drive capability and extremely small propagation delay, typically 13 ns. The UCC27517 provides 4-A source, 4-A sink (symmetrical drive) peak-drive current capability at VDD = 12 V. The UCC27517 is designed to operate over a wide VDD range of 4.5 to 18 V and wide temperature range of –40°C to 140°C.

2.3.6 TLV9062

The TLV9062 is dual-low-voltage (1.8 V to 5.5 V) operational amplifiers (op amps) with rail-to-rail input- and output-swing capabilities. These devices are highly cost-effective solutions for applications where low-voltage

operation, a small footprint, and high capacitive load drive are required. Although the capacitive load drive of the TLV906x is 100 pF, the resistive open-loop output impedance makes stabilizing with higher capacitive loads simpler. The TLV906xS devices include a shutdown mode that allow the amplifiers to switch into standby mode with typical current consumption less than 1 µA. The TLV906xS family helps simplify system design, because the family is unity-gain stable, integrates the RFI and EMI rejection filter, and provides no phase reversal in overdrive condition.

2.3.7 TLV76733

The TLV76733 is a wide input linear voltage regulator supporting an input voltage range from 2.5 V to 16 V and up to 1 A of load current, which has a 1% output accuracy that can meet the needs of low voltage microcontrollers (MCUs) and processors. The TLV767 is designed to have a much lower IQ than traditional wide-VIN regulators, thus making the device well positioned to meet the needs of increasingly stringent standby power requirements.

2.4 System Design Theory

The main focus of this reference design is dual motor control using sensorless-FOC with a low EMI, high efficiency, high power factor, and protected power rail for targeted HVAC or other appliance applications.

2.4.1 Interleaved PFC

Boost converter is the most popular topology for PFC application. This is because boost converters have continuous input current that can be controlled with average current mode control to force input current to track changes in line voltage. Also higher inherent efficiency and wide duty cycle range makes it more attractive than buck-boost converter.

While basic single phase boost converter is adequate for low power non-isolated application, interleaving techniques provide additional advantages in high power motor drive application. The two phase interleaved boost converter is simply two boost converters in parallel operating 180° out of phase. The input current is the sum of the two inductor currents I_{L1} and I_{L2}. Because the inductor's ripple currents are out of phase, they tend to cancel each other and reduce the input ripple current caused by the boost inductors. The best input inductor ripple current cancellation occurs at 50 percent duty cycle. The output capacitor current is the sum of the two diode currents (I_{D1} + I_{D2}) less the dc output current. Interleaving reduces the output capacitor ripple current (I_{OUT}) as a function of duty cycle. As the duty cycle approaches 0 percent, 50 percent and 100 percent duty cycle, the sum of the two diode currents approaches dc. At these points, the output capacitor only has to filter the inductor ripple current.

2.4.1.1 Full Bridge Diode Rectifier Rating

- Current Rating of Bridge
  - Due to interleaving, rms value of input current is also reduced.
  - Maximum RMS and Peak Input line current assuming efficiency of 95% and corrected power factor 0.99

$$I_{LINE\_RMS\_MAX} = \frac{P_O}{\eta \times V_{ACMIN} \times PF} \quad (1)$$

  - Maximum peak line current is addition of peak of line current and sum of both inductor peak ripple current in worst case. Calculation of peak inductor current is shown in inductor rating section.

$$I_{LINE\_PEAK\_MAX} = \left(\sqrt{2} \times I_{LINE\_RMS\_MAX}\right) + \left(\frac{\Delta i_L}{2} \times 2\right) \quad (2)$$

  - Maximum Average Rectified Current

$$I_{RCT\_AVG\_MAX} = \frac{2 \times I_{LINE\_PEAK\_MAX}}{\pi} \quad (3)$$

- Reverse Blocking Voltage Rating of Diode Bridge
  - Maximum reverse blocking is peak of maximum line voltage

$$V_{LINE\_PEAK\_MAX} = \sqrt{2} \times V_{LINE\_RMS\_MAX} \quad (4)$$

- Considering a safety factor of approximately 50%, selected bridge has 600 V reverse blocking voltage rating
- Power Dissipation in Bridge

– For diode bridge, find forward voltage at maximum average current

$$P_{Bridge\_MAX} = \sqrt{2} \times V_f \times I_{RCT\_AVG\_MAX} \quad (5)$$

2.4.1.2 Inductor Ratings

This section provides procedure to calculate inductor value and current rating.

- Value of Inductor

$$L_1 = L_2 = \frac{\sqrt{2} \times V_{ACMIN} \times D_{MIN}}{\Delta I_L \times F_{SW}} \quad (6)$$

Where ΔI_L is peak to peak ripple current and most common design practices recommend to limit peak to peak ripple current at 20-25% of average inductor current.

– Peak to peak inductor current rating can be determined by $\Delta I_L = \% \ Ripple \times I_{L(AVG)}$

– Average inductor current can be determined by $I_{L(AVG)} = \frac{\sqrt{2} \times I_{LINE\_RMS\_MAX}}{2}$

- Inductor Current Rating

– Peak inductor current rating can be determined by $I_{L2_{PEAK}} = I_{L2_{PEAK}} = I_{L(AVG)} + \left(\frac{\Delta i_L}{2}\right)$

2.4.1.3 AC Voltage Sensing

The AC voltage sensing circuit is used to convert the grid voltage signal into low voltage signal, and then to condition these sensed signals to meet the requirements of the ADC input port. The line and the neutral voltages are sensed by resistor divider to the ground of the board as shown in Figure 2-2. The two signals are subtracted by the amplifier to the ADC input port to get the Vac sensing value.

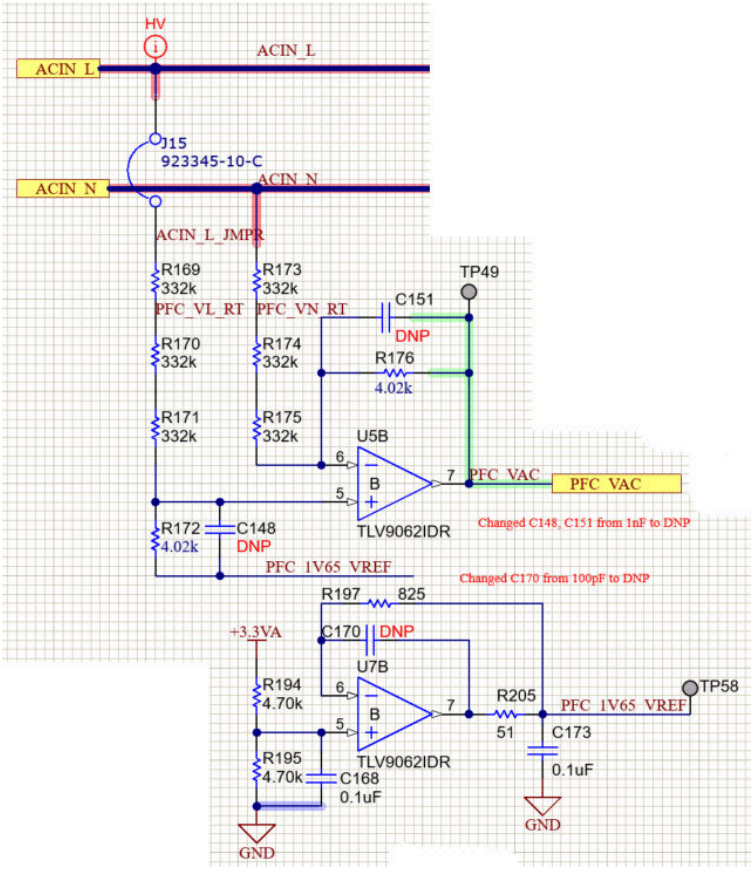


Figure 2-2. Input AC Voltage Sensing Circuit

2.4.1.4 DC Link Voltage Sensing

Similarly, the DC bus voltage sensing circuit is used to convert the DC link voltage signal into low voltage signal, is sensed by a resistor divider network as shown in Figure 2-3.

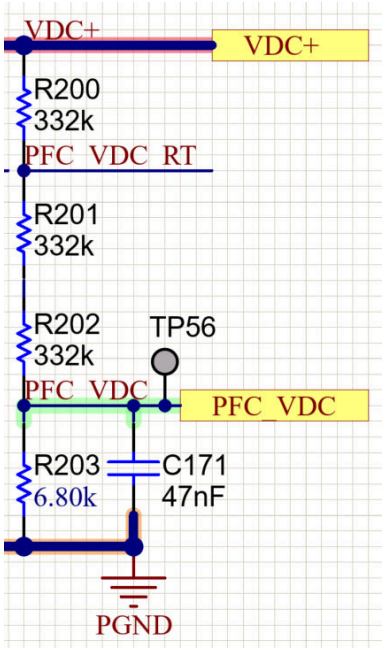


Figure 2-3. DC Bus Voltage Sensing Circuit

2.4.1.5 Bus Current Sensing

The current sensing circuit is used to convert the PFC inductor current signal into low voltage signal, and then to condition the sensed signal to meet the requirements of the ADC input port. As part of IPFC inner current loop control, AC current sensing is crucial. To do that, this design perform current sensing on returning point to diode bridge. It is rectified version of AC current. This method offers the benefit of current sensing with respect to system ground. There are two parts in the current sensing circuit as shown in Figure 2-4:

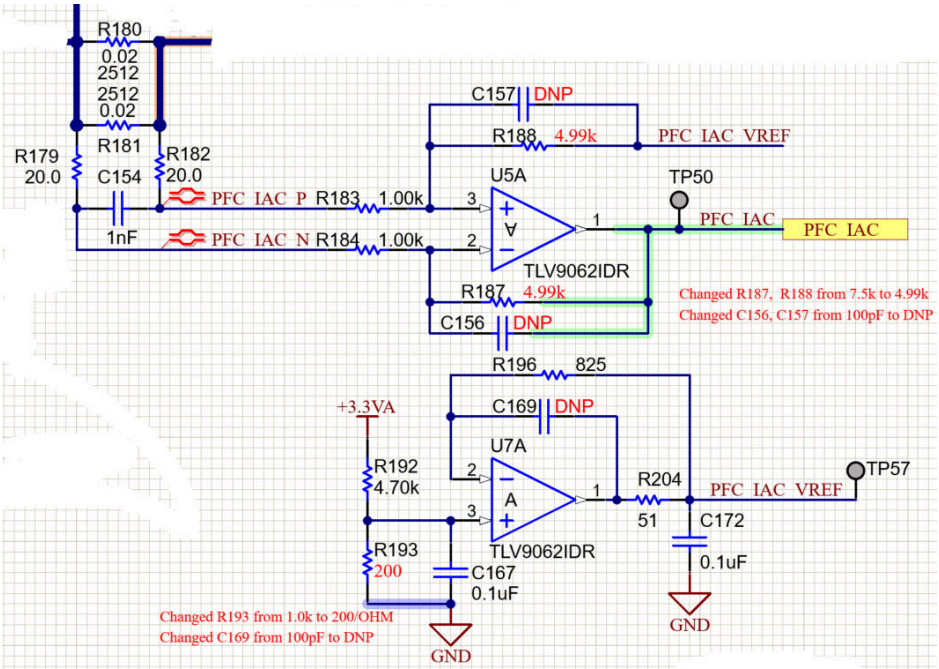


Figure 2-4. PFC Bus Current Sensing Circuit

- Shunt Resistor Calculation

  Recommended voltage across shunt resistor is decided as 0.2 to 0.25 V to optimize the power dissipation and better noise performance. Power dissipation across shunt has been limited to less than 0.05% of total power Exact value of full scale voltage is trial and error value depend on standard Shunt resistor available

$$V_{SHUNT\_PFC} = I_{SHUNT\_PFC} \times R_{SHUNT\_PFC} \times G_i \quad (7)$$

  Where G_i = 4.892 as show in Figure 2-4

$$R_{SHUNT\_PFC} = 0.02\Omega \parallel 0.02\Omega = 0.01\Omega \quad (8)$$

$$I_{SHUNT\_PFC} = 2 \times I_{L(MAX)} \quad (9)$$

2.4.1.6 DC Link Capacitor Rating

- TIDM02010 uses aluminum electrolyte capacitor for DC link. Although polymer electrolytic capacitors with the significantly longer lifetime has proven its high reliability, it is not optimal choice when comes to high vibration environment like compressor outdoor unit. Unlike liquid electrolyte in aluminum electrolyte capacitor, polymer electrolyte capacitor cannot absorb vibration because of its solid polymer structure. To absorb high frequency component on DC link, metal film capacitor is also used in DC link. These caps are placed very close boost diode to reduce high frequency loop.
- Capacitance Value Calculation
  - Value of DC link capacitor based on holdup charge

$$C_{OUT(MIN)} \geq \frac{2 \times P_{OUT} \times \frac{1}{f_{LINE}}}{V_{OUT}^2 - \left(\frac{V_{OUT}}{2}\right)^2} \quad (10)$$

  - Value of DC link capacitor value based upon voltage ripple requirement

$$C_{OUT(MIN)} = \frac{1}{4} \times \frac{\left(\frac{P_{OUT}}{V_{OUT} \times \eta \times 0.637}\right)}{V_{RIPPLE} \times 0.8 \times \pi \times f_{LINE}} \quad (11)$$

  - Capacitor Derating

    Actual capacitor value changes based upon factors like initial capacitor tolerance, temperature and aging. Consider 20% for each factor.

$$C_{OUT} = \frac{C_{OUT(MIN)}}{\left(1 - \eta_{tolerance}\right) \times \left(1 - \eta_{temp}\right) \times \left(1 - \eta_{aging}\right)} \quad (12)$$

- Capacitance Voltage Rating
  - Considering peak ripple voltage

$$WV_{DC\_BUS\_CAP\_MIN} = V_{DCBUS} + \frac{\Delta V_{PP}}{2} \quad (13)$$

  - Considering 10% safety margin for transient and overvoltage

$$WV_{DC\_BUS\_CAP} = 1.1 \times WV_{DC\_BUS\_CAP\_MIN} \quad (14)$$

- Capacitance Current RMS Value

  To estimate the capacitor RMS current is not straight forward and needs following steps.

  - Angular frequency is given by

$$\omega = 2 \times \pi \times f_{LINE} \quad (15)$$

  - Number of iteration

$$iteration = \frac{F_{SW\_PFC}}{2 \times F_{LINE}} \quad (16)$$

– Value of one step in seconds:

$$Step = \frac{\frac{1}{f_{LINE}}}{Iteration} \quad (17)$$

– Line voltage is sine function and can be determined by

$$V_{IN}(t) = \sqrt{2} \times V_{IN(RMS)} \times \sin(\omega t) \quad (18)$$

– Duty Ratio is also function of input voltage and can be calculated as

$$D(t) = \frac{V_{OUT} - V_{IN}(\omega t)}{V_{OUT}} \quad (19)$$

– With power factor correction, input current is also sine function

$$I_{IN}(t) = \frac{\sqrt{2} \times P_{OUT}}{\eta_{System} \times V_{IN(MIN)}} \times \sin(\omega t) \quad (20)$$

$$I_{Crms} = \sqrt{\frac{1}{Iteration} \times \sum_{n=1}^{Iteration} \left\{ [I_{IN}(n \times Step)]^2 \times \left[\frac{1}{2} \times \sqrt{(2 - 2 \times D(n \times Step)) - (2 - 2 \times D(n \times Step))^2} \right]^2 \right\}} \quad (21)$$

- Capacitor ESR Rating

$$ESR_{MAX} \geq \frac{V_{RIPPLE} \times 0.2}{\left(\frac{P_{OUT} \times \sqrt{2}}{V_{IN(MIN)} \times \eta} \right)} \quad (22)$$

2.4.1.7 MOSFET Ratings

- MOSFET Voltage Rating

  When MOSFET is in OFF state, it will have to block DC bus voltage which can rise up to 400V maximum. S0, with leaving 50% room as safety factor, 600V MOSFET is selected in this design.
- MOSFET Current Rating

  When MOSFET is ON, it will conduct peak value of inductor current when inductance is at its lowest value

$$I_{FET(PEAK)} = I_{L(PEAK)} \quad (23)$$

- Power Dissipation in MOSFET

  Total power dissipation in MOSFET consists of conduction loss, turn off loss, turn on loss, C_{OSS} loss and gate drive loss. It can be estimated using following formula

$$P_{FET} = P_{R_{DS(ON)}} + P_{SWITCH(t_{ON})} + P_{SWTICH(t_{OFF})} + P_{C_{OSS}} + P_{GATE} \quad (24)$$

- MOSFET Conduction Loss

  Conduction loss in MOSFET can be calculated by following formula

$$P_{R_{DS(ON)}} = I_{FET(RMS)}^2 \times R_{DS(ON)} \quad (25)$$

where FET RMS current can be calculated using following formula:

$$I_{FET(RMS)} = \sqrt{f_{LINE} \times \sum_{n=1}^{Iteration} \left\{ \int_{0}^{\left[\frac{D(n \times Step)}{f_{SW}}\right]} \left[\frac{I_{IN}(n \times Step)}{2}\right]^2 dt \right\}} \quad (26)$$

- MOSFET Switch ON Loss
  - MOSFET switching loss due to turn on operation

$$P_{SWITCH(t_{ON})} = f_{LINE} \times \sum_{n=1}^{Iteration} \left[I_{IN_{t_{ON}}}\left(n \times Step\right) \times V_{OUT} \times t_{ON(delay)} \right] \quad (27)$$

  - Where current at ON state is given by

$$I_{IN_{t_{ON}}}(t) = \left(I_L - \frac{\Delta I_{L\_MAX}}{2}\right)\sin(\omega \times t) \quad (28)$$

  - MOSFET turn ON delay time can be given by

$$t_{ON(delay)} = \frac{Q_{GS(miller)}}{I_{GATE(ON)}} \quad (29)$$

  - $I_{GATE(ON)}$ is average FET gate drive current during turn ON operation

$$I_{GATE(ON)} = \frac{V_{GS(MAX)}}{2 \times R_{GATE(ON)}} \quad (30)$$

    Where, $R_{GATE(ON)}$ = R70 = R71. That is resistance in gate current path during turn ON operation.
- MOSFET Switch OFF
  - MOSFET switching loss due to turn off operation

$$P_{SWITCH(t_{OFF})} = f_{LINE} \times \sum_{n=1}^{Iteration} \left[I_{IN_{t_{OFF}}}\left(n \times Step\right) \times V_{OUT} \times t_{OFF(delay)} \right] \quad (31)$$

  - Where current at OFF state is given by

$$I_{IN_{t_{OFF}}}(t) = \left(I_L + \frac{\Delta I_{L\_MAX}}{2}\right)\sin(\omega \times t) \quad (32)$$

  - MOSFET turn OFF delay time can be given by

$$t_{OFF(delay)} = \frac{Q_{GS(miller)}}{I_{GATE(OFF)}} \quad (33)$$

  - $I_{GATE(OFF)}$ is average FET gate drive current during turn OFF operation. Current in this mode flows through diode and resistor in series with diode. So, considering forward voltage drop, $I_{GATE(OFF)}$ can be given by:

$$I_{GATE(OFF)} = \frac{V_{GS(MAX)} - V_{f\_Diode}}{2 \times R_{GATE(OFF)}} \quad (34)$$

- Part of the total FET losses are contributor C_{OSS} ($C_{OSS(AVG)}$) during charging and discharging during a PWM switching cycle and can be given by

$$P_{C_{OSS}} = \frac{1}{2} \times C_{OSS(AVG)} \times V_{OUT}^2 \times f_{SW\_PFC} \quad (35)$$

  Where C_{OSS} varies with line voltage and is not a linear function. The following equation and information from the FETs data sheet can be used to calculate $C_{OSS(AVG)}$. $C_{OSS(SPEC)}$is the typical C_{OSS} measured at a specified voltage $V_{DS(SPEC)}$.

$$C_{OSS(AVG)} = 2 \times C_{OSS(SPEC)} \times \sqrt{\frac{V_{DS(SPEC)}}{V_{OUT}}} \quad (36)$$

- MOSFET Gate Loss can be determined by

$$P_{GATE} = Q_{GS} \times V_{GS(MAX)} \quad \times \quad f_{SW\_PFC} \quad (37)$$

2.4.1.8 Diode Ratings

- Diode Type Selection
  - In this design, Silicon Carbide (SiC) Schottky Diode is used as boost diode.
  - As compared to standard ultra-fast silicon power diodes, silicon Schottky devices offer improved performance because of their lower forward drops and reduced forward and reverse recovery characteristics. But their low breakdown voltages have limited their use to low-voltage applications.
  - Silicon carbide overcomes these limitations with its high reverse breakdown voltage and fast recovery.
- Diode Reverse Voltage Rating
  - Diode reverse voltage is similar to MOSFET blocking voltage. When diode is reverse biased, it will have to block DC bus voltage which can rise up to 400 V maximum. Leaving 50% room as safety factor, 600 V reverse blocking rating is selected.
- Diode Current Rating
  - When diode conducts average and peak value of current passing through the diode can be determined by

$$I_{DIODE(AVG)} = \frac{P_{OUT}}{2 \times V_{OUT(MIN)} \times \eta_{DIODE}} \quad (38)$$

$$I_{DIODE(PEAK)} = I_{L(MAX)} \quad (39)$$

- Power Dissipation in diode
  - There are two type of losses in diode rectifier, for example, conduction loss and reverse recovery loss. Major loss is contributed by forward voltage drop can be given by

$$P_{DIODE} = I_{DIODE(AVG)} \times V_F \quad (40)$$

2.4.2 Three-Phase PMSM Drive

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

- Synchronous motor construction: Permanent magnets are rigidly fixed to the rotating axis to create a constant rotor flux. This rotor flux usually has a constant magnitude. The stator windings when energized create a rotating electromagnetic field. To control the rotating magnetic field, it is necessary to control the stator currents.
- The actual structure of the rotor varies depending on the power range and rated speed of the machine. Permanent magnets are suitable for synchronous machines ranging up-to a few Kilowatts. For higher power ratings the rotor usually consists of windings in which a DC current circulates. The mechanical structure of the rotor is designed for number of poles desired, and the desired flux gradients desired.
- The interaction between the stator and rotor fluxes produces a torque. Since the stator is firmly mounted to the frame, and the rotor is free to rotate, the rotor will rotate, producing a useful mechanical output as shown in Figure 2-5.
- 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 a fine tuning is needed after closing the speed loop using sensorless algorithm to draw 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 will experience rapidly alternating positive and negative torque. This will result in less than optimal 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 will stop rotating at the synchronous frequency, and respond 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 90ºC to obtain the highest mutual torque production. This synchronization requires knowing the rotor position to generate the right stator field.

- The stator magnetic field can be made to have any direction and magnitude by combining the contribution of different stator phases to produce the resulting stator flux.

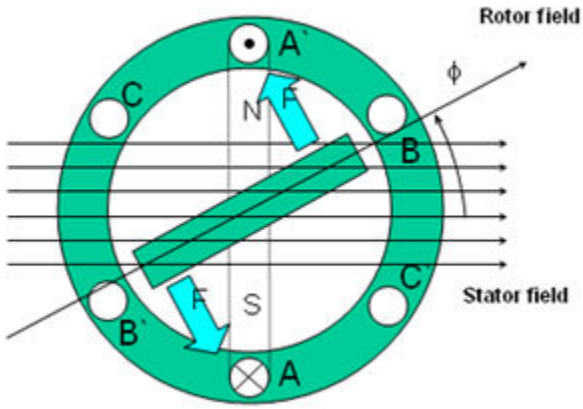


Figure 2-5. The Interaction Between the Rotating Stator Flux, and the Rotor Flux Produces a Torque

2.4.2.1 Field Oriented Control of PM Synchronous Motor

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

In a direct current (DC) Motor, the excitation for the stator and rotor is independently controlled, the produced torque and the flux can be independently tuned as shown in Figure 2-6. 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 optimal all the time. The key point here is that the windings are managed to keep the flux produced by the rotor windings orthogonal to the stator field.

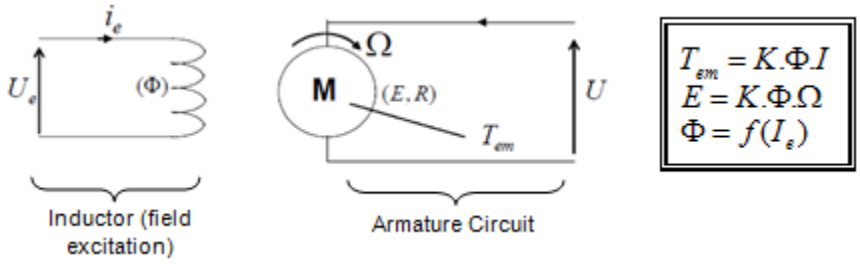


Figure 2-6. Flux and torque are independently controlled in DC motor model

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

According to the electromagnetic laws, the torque produced in the synchronous machine is equal to vector cross product of the two existing magnetic fields as Equation 41.

$$\tau_{em} = \vec{B}_{stator} \times \vec{B}_{rotor} \quad (41)$$

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

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

The Field Orientated Control 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 co-ordinate (d and q co-ordinates) time invariant system. These projections lead to a structure similar to that of a DC machine control. Field orientated controlled machines need two constants as input references: the torque component (aligned with the q co-ordinate) and the flux component (aligned with d co-ordinate). As Field Orientated Control is simply based on projections, the control structure handles instantaneous electrical quantities. This makes the control accurate in every working operation (steady state and transient) and independent of the limited bandwidth mathematical model. The FOC thus solves the classic scheme problems, in the following ways:

- The ease of reaching constant reference (torque component and flux component of the stator current)
- The ease of applying direct torque control because in the (d, q) reference frame the expression of the torque is defined in Equation 42.

$$\tau_{em} \propto \psi_R \times i_{sq} \quad (42)$$

By maintaining the amplitude of the rotor flux (ψ_R) at a fixed value we have a linear relationship between torque and torque component (i_{Sq}). We can then control the torque by controlling the torque component of stator current vector.

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 43.

$$\bar{i}_s = i_a + \alpha i_b + \alpha^2 i_c \quad (43)$$

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

Figure 2-7 shows the stator current complex space vector.

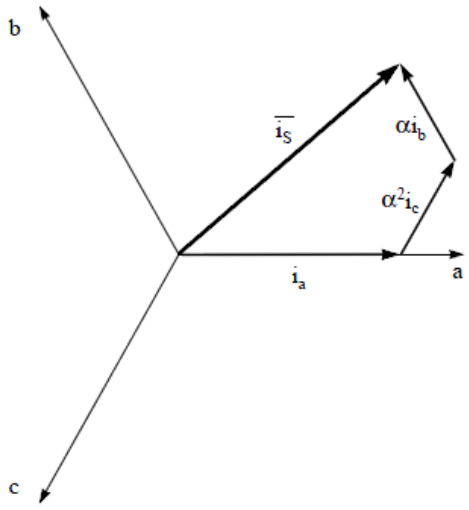


Figure 2-7. Stator Current Space Vector and its Component in (a,b,c) Frame

Where (a,b,c) are the three phase system axes. This current space vector depicts the three phase sinusoidal system. It 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

The $(a,\ b) \Rightarrow (\alpha, \beta)$ 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*lpha* are in the same direction we have the following vector diagram as shown in Figure 2-8.

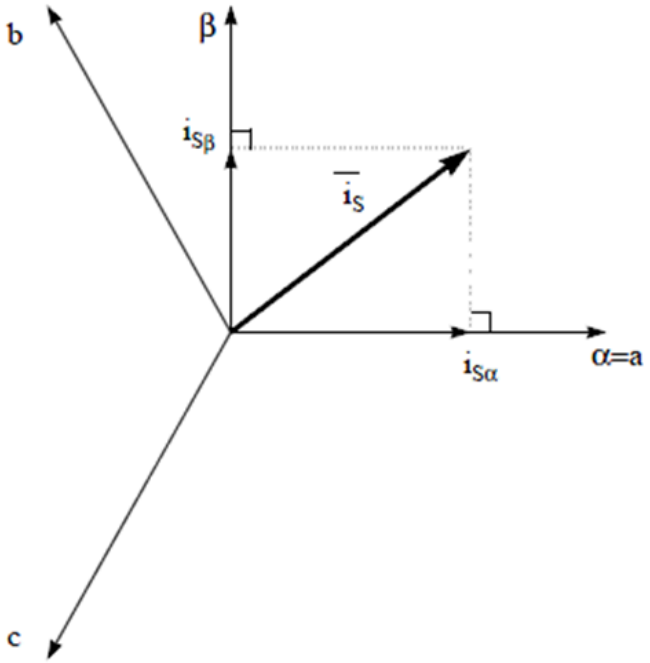


Figure 2-8. Stator Current Space Vector in the Stationary Reference Frame

The projection that modifies the 3-phase system into the (α, β) 2-dimension orthogonal system is presented in Equation 44.

$$i_{s\alpha} = i_a$$
$$i_{s\beta} = \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b \quad (44)$$

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

The $(\alpha, \beta) \Rightarrow (d,\ q)$ 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. If we consider the d axis aligned with the rotor flux, Figure 2-9 shows the relationship for the current vector from the two reference frame.

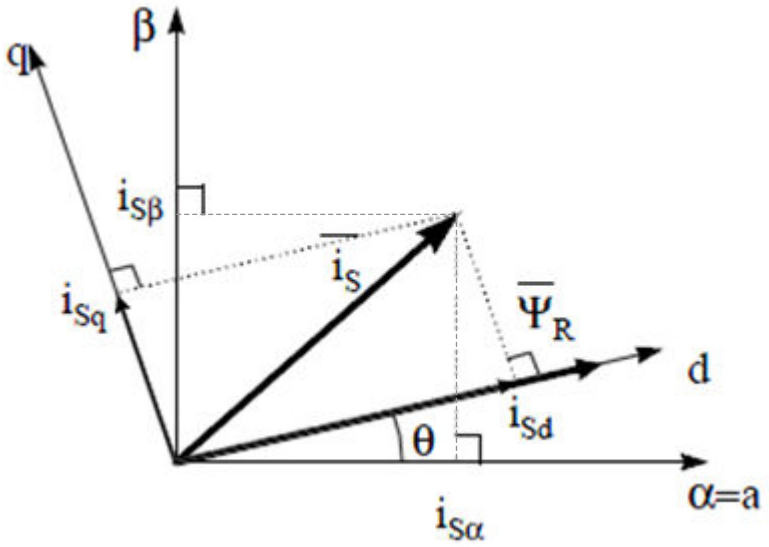


Figure 2-9. Stator Current Space Vector in The d,q Rotating Reference Frame

The flux and torque components of the current vector are determined by Equation 45.

$$i_{sd} = i_{s\alpha}\cos(\theta) + i_{s\beta}\sin(\theta)$$
$$i_{sq} = -i_{s\alpha}\sin(\theta) + i_{s\beta}\cos(\theta) \quad (45)$$

where θ is the rotor flux position

These components depend on the current vector (α, β) components and on the rotor flux position; if we know the right rotor flux position 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.

The Basic Scheme of FOC for AC Motor

Figure 2-10 summarizes the basic scheme of torque control with FOC:

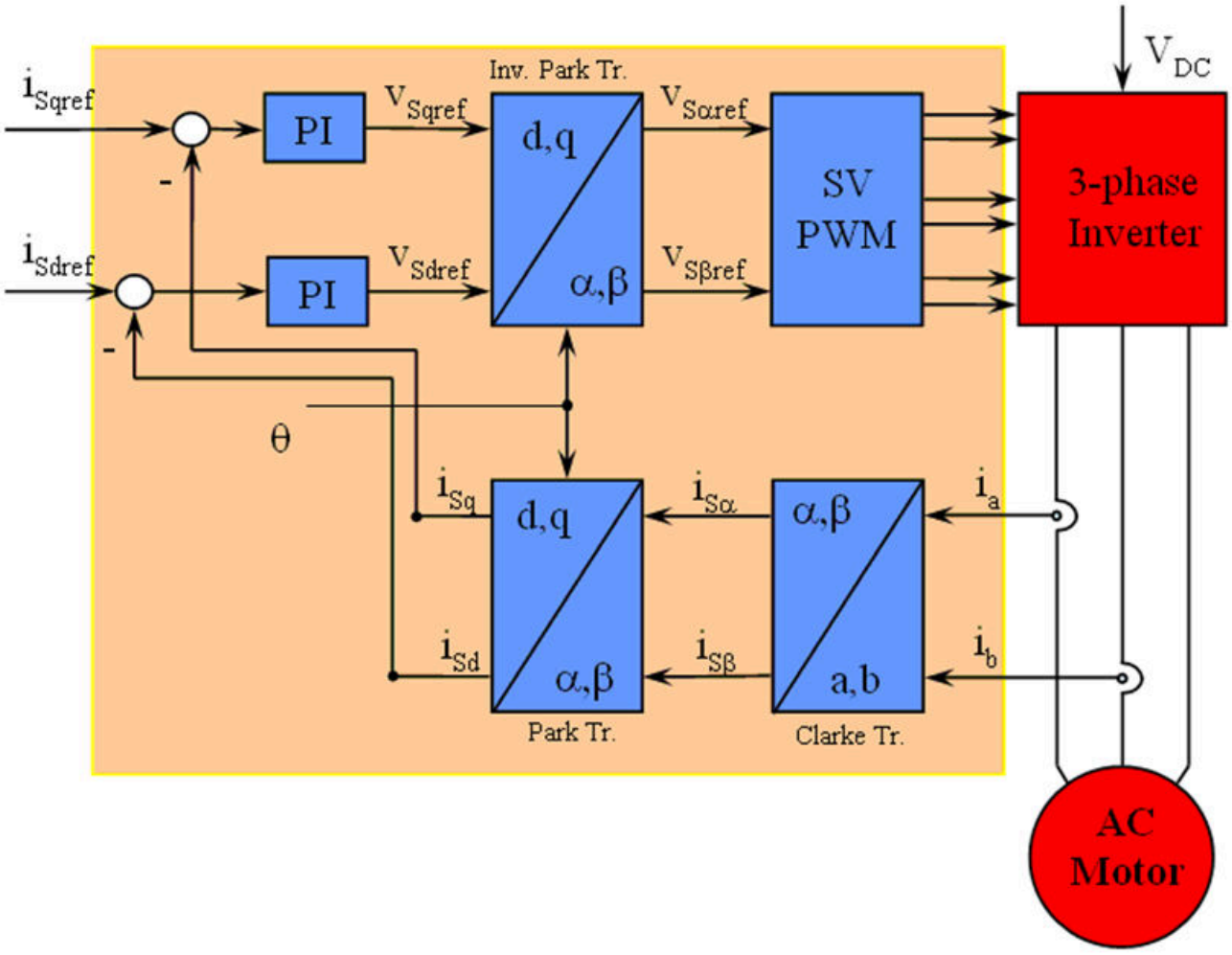


Figure 2-10. Basic Scheme of FOC for AC Motor

Two motor phase currents are measured. These measurements feed the Clarke transformation module. The outputs of this projection are designated $i_{s\alpha}$ and $i_{s\beta}$. These two components of the current are the inputs of the Park transformation that gives the current in the d,q rotating reference frame. The i_{sd} and i_{sq} components are compared to the references i_{sdref} (the flux reference component) and i_{sqref} (the torque reference component). At this point, this control structure shows an interesting advantage: it can be used to control either synchronous or induction machines by simply changing the flux reference and obtaining rotor flux position. As in synchronous permanent magnet a motor, the rotor flux is fixed determined by the magnets; there is no need to create one. Hence, when controlling a PMSM, i_{sdref} should be set 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} could be the output of the speed regulator when we use a speed FOC. The outputs of the current regulators are Vsdref and V_{sqref}; they 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).

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 d-axis and i_{sd} and i_{sq} are incorrect flux and torque components of the stator current. Figure 2-11 shows the (a, b, c), (α, β) and (*d, q*) reference frames, and the correct position of the rotor flux, the stator current and stator voltage space vector that rotates with d,q reference at synchronous speed.

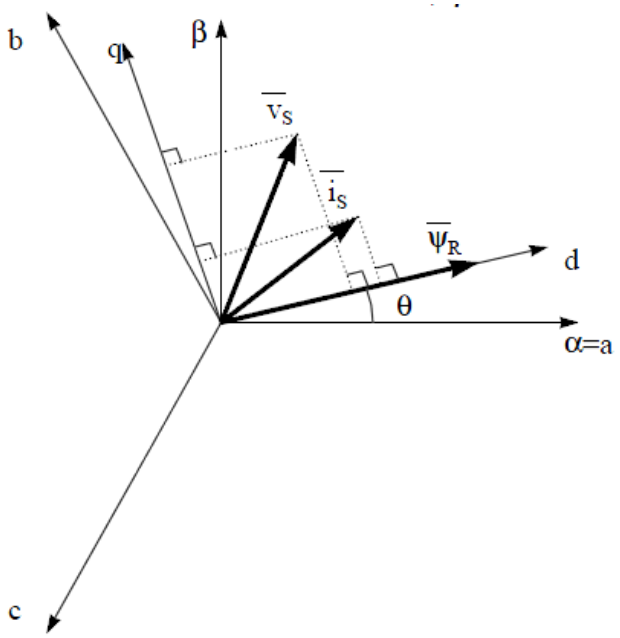


Figure 2-11. Current, Voltage and Rotor Flux Space Vectors in the (d, q) Rotating Reference Frame

The measure of the rotor flux position is different if we consider 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 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 it needs a particular method 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 field oriented control for the PMSM drive allows the motor torque 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 will be controlling torque while d-axis current is forced to zero. Therefore, the key module of this system is the estimation of rotor position using enhance Sliding-Mode Observer (eSMO) or FAST estimator.

Figure 2-12 shows the overall block diagram of sensorless FOC of fan PMSM using eSMO with flying start in this reference design.

Figure 2-13 shows the overall block diagram of sensorless FOC of compressor PMSM using eSMO with field weakening control (FWC) and maximum torque per ampere (MTPA) in this reference design.

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

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

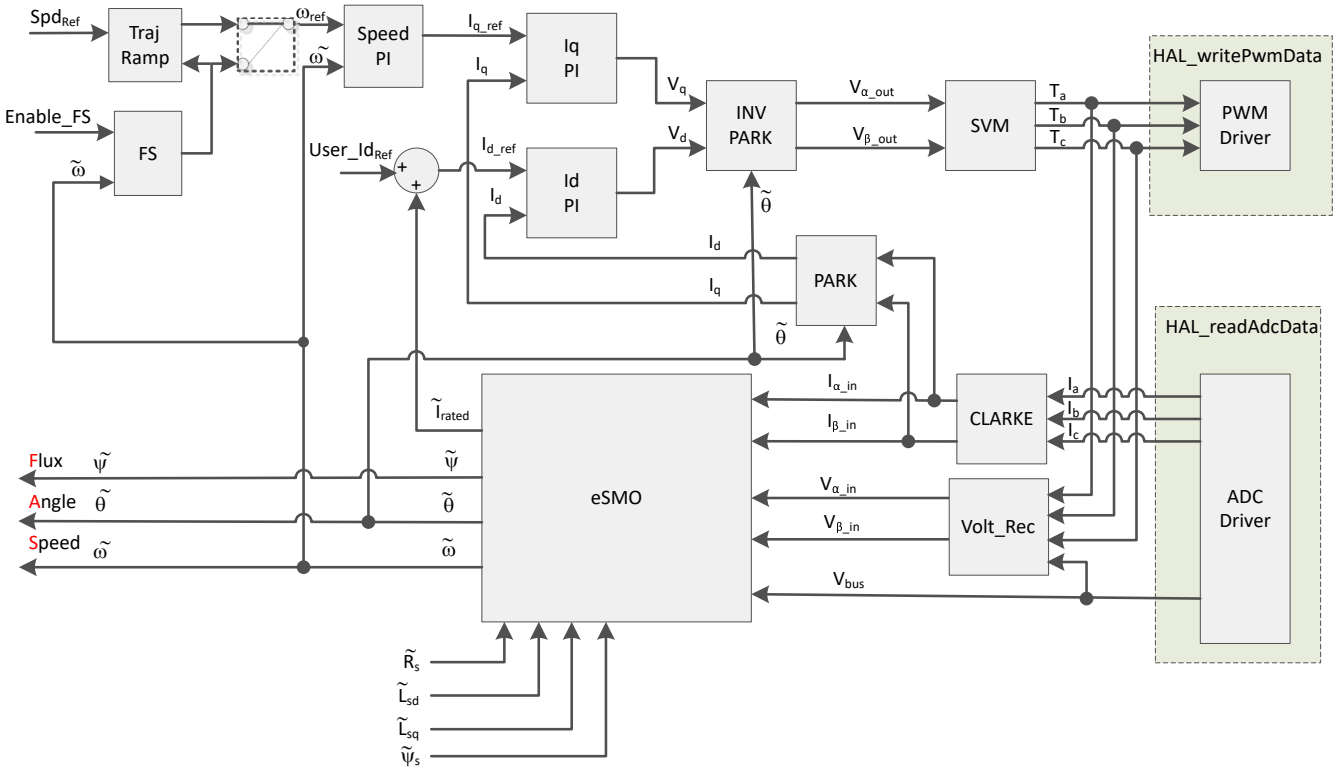


Figure 2-12. Sensorless FOC of Fan PMSM using eSMO with Flying Start (FS)

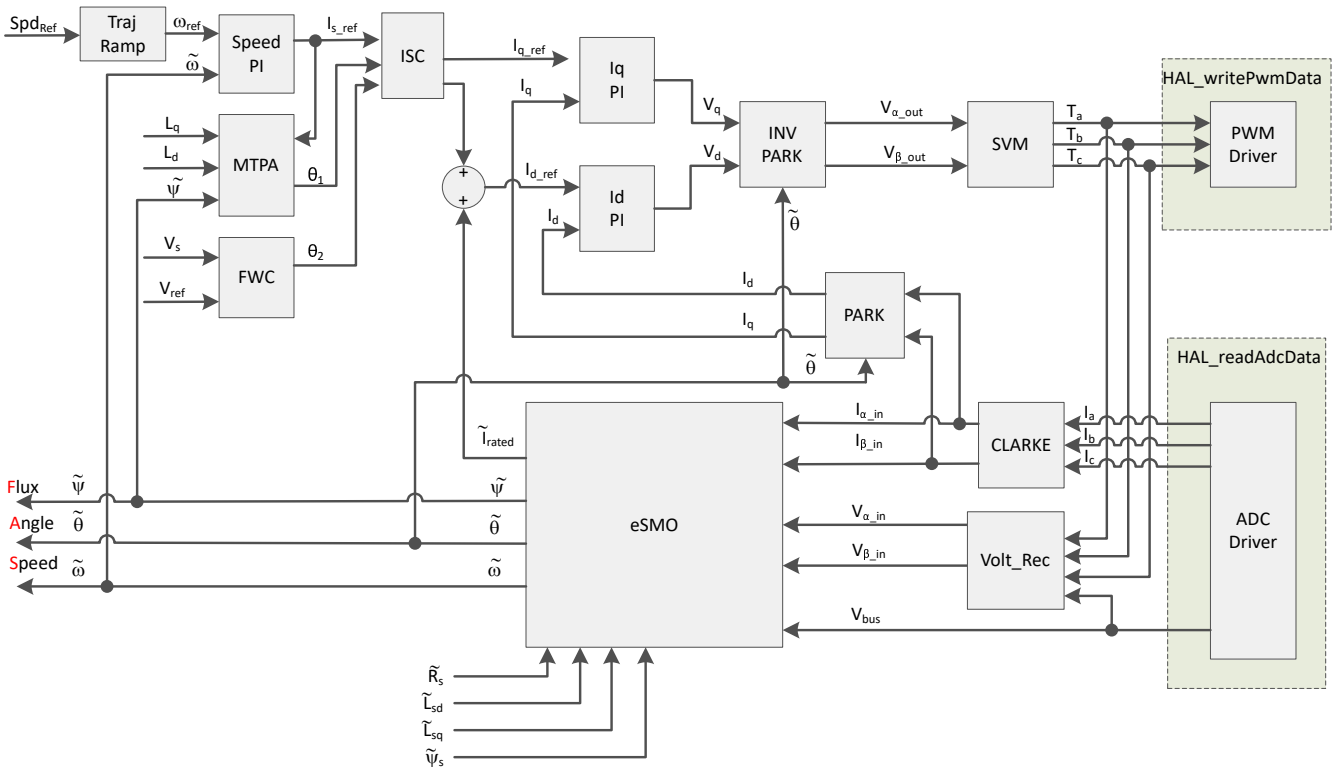


Figure 2-13. Sensorless FOC of Compressor PMSM using eSMO with FWC and MTPA

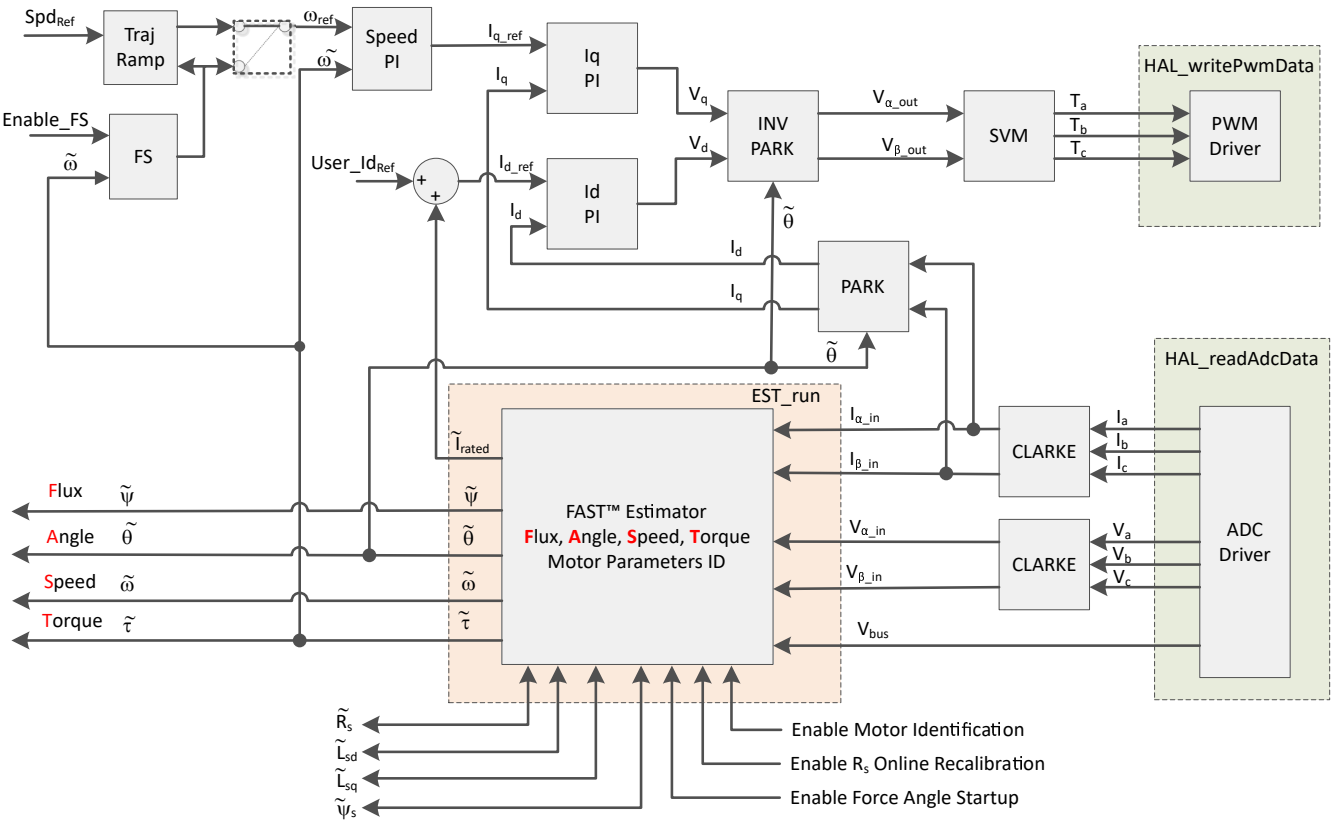


Figure 2-14. Sensorless FOC of Fan PMSM using FAST with Flying Start (FS)

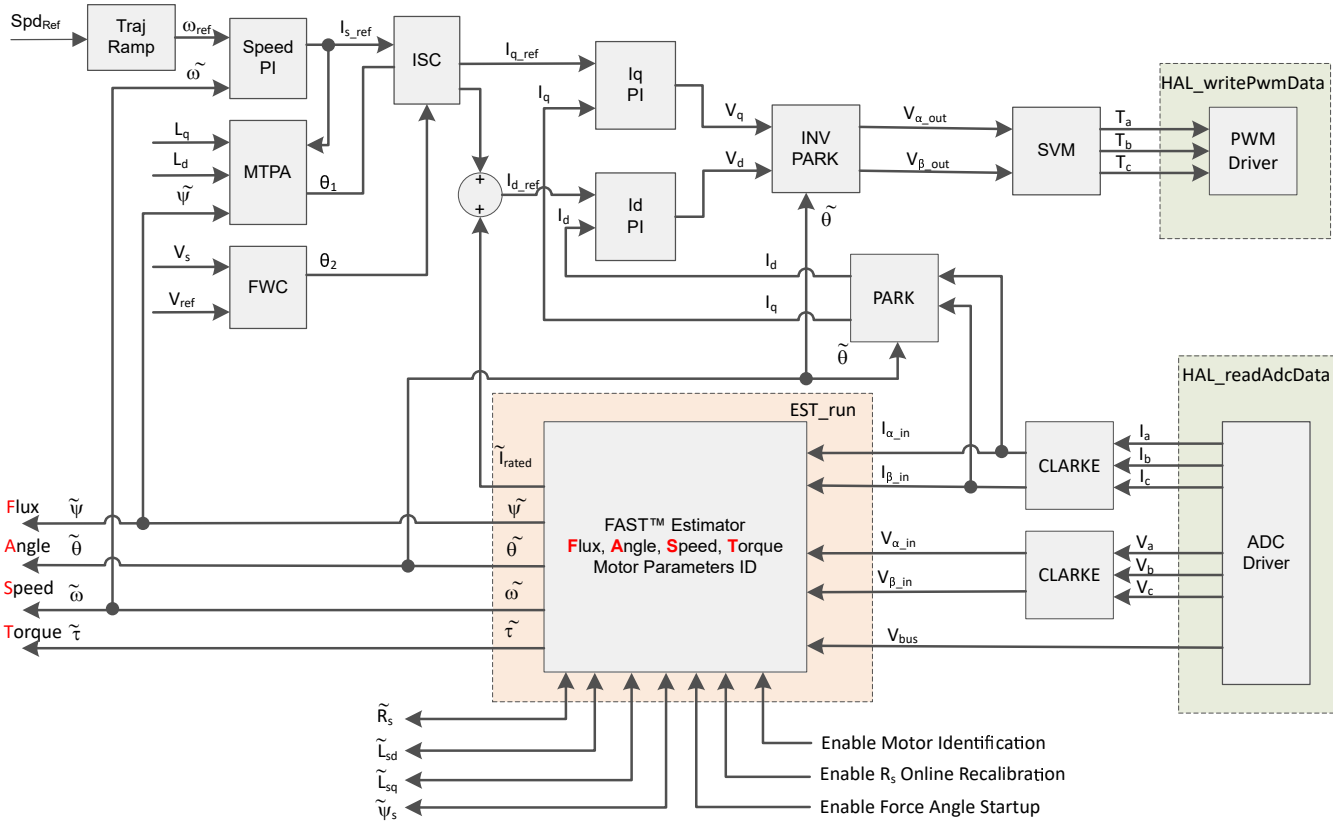


Figure 2-15. Sensorless FOC of Compressor PMSM using FAST with FWC and MTPA

2.4.2.2 Sensorless Control of PM Synchronous Motor

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

2.4.2.2.1 Enhanced Sliding Mode Observer with Phase Locked Loop

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. Its advantages include fast response, strong robustness, and insensitivity to both parameter changes and disturbances.

2.4.2.2.1.1 Mathematical Model and FOC Structure of an IPMSM

The sensorless FOC structure for an IPMSM is illustrated in Figure 2-16. In this system, the eSMO is used for achieving the sensorless control an IPMSM system, and the eSMO model is designed by utilizing the back EMF model together with a PLL model for estimating the rotor position and speed.

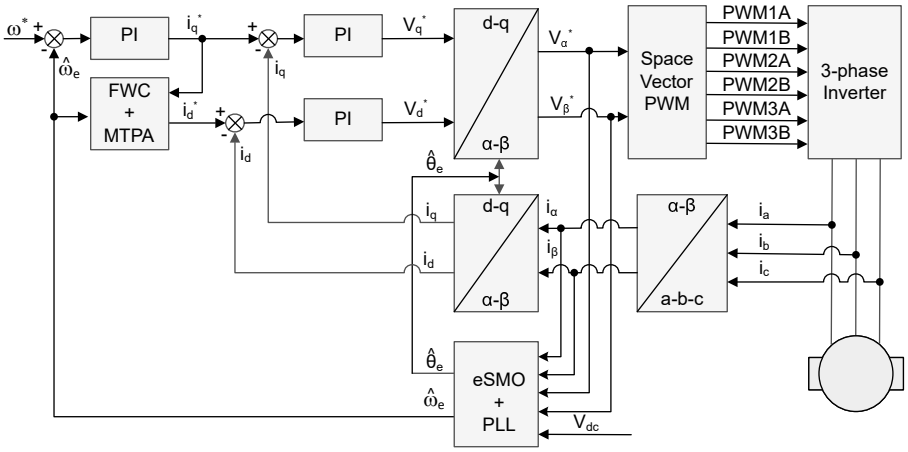


Figure 2-16. Sensorless FOC Structure of an IPMSM System

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 46. The dynamic model of a generic PMSM can be written in the d-q rotor reference frame as:

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} R_s + pL_d & -\omega_e L_q \\ \omega_e L_d & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_e \lambda_{pm} \end{bmatrix} \quad (46)$$

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; and ω_e is the electrical angular velocity of the rotor.

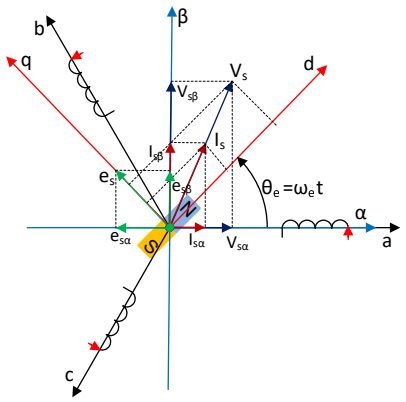


Figure 2-17. Definitions of Coordinate Reference Frames for PMSM Modeling

By using the inverse Park transformation as shown in Figure 2-17, the dynamics of the PMSM can be modeled in the α-β stationary reference frame as:

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} R_s + pL_d & \omega_e\left(L_d - L_q\right) \\ -\omega_e\left(L_d - L_q\right) & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} \tag{47}$$

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

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \left(\lambda_{pm} + \left(L_d - L_q\right) i_d\right) \omega_e \begin{bmatrix} -\sin(\theta_e) \\ \cos(\theta_e) \end{bmatrix} \tag{48}$$

According to Equation 47 and Equation 48, 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 47 as a state equation using the stator current as a state variable:

$$\begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} = \frac{1}{L_d} \begin{bmatrix} -R_s & -\omega_e\left(L_d - L_q\right) \\ \omega_e\left(L_d - L_q\right) & -R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} V_\alpha - e_\alpha \\ V_\beta - e_\beta \end{bmatrix} \tag{49}$$

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

$$S(x) = \begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix} = \begin{bmatrix} \tilde{i}_\alpha \\ \tilde{i}_\beta \end{bmatrix} \tag{50}$$

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

2.4.2.2.1.2 Design of ESMO for the IPMSM

The conventional PLL integrated into the SMO is shown in Figure 2-18.

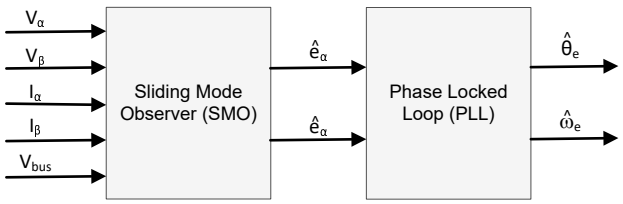


Figure 2-18. Block Diagram of eSMO with PLL for a PMSM

The traditional reduced-order sliding mode observer is constructed, which mathematical model is shown in Equation 51 and the block diagram is shown in Figure 2-19.

$$\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} = \frac{1}{L_d}\begin{bmatrix} -R_s & -\hat{\omega}_e\left(L_d - L_q\right) \\ \hat{\omega}_e\left(L_d - L_q\right) & -R_s \end{bmatrix}\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} + \frac{1}{L_d}\begin{bmatrix} V_\alpha - \hat{e}_\alpha + z_\alpha \\ V_\beta - \hat{e}_\beta + z_\beta \end{bmatrix} \tag{51}$$

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

$$\begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} = \begin{bmatrix} k_\alpha sign\left(\hat{i}_\alpha - i_\alpha\right) \\ k_\beta sign\left(\hat{i}_\beta - i_\beta\right) \end{bmatrix} \tag{52}$$

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 guarantee the stable operation of the SMO, the k_α and k_β should be large enough to hold $k_\alpha > \max(|e_\alpha|)$ and $k_\beta > \max(|e_\beta|)$.

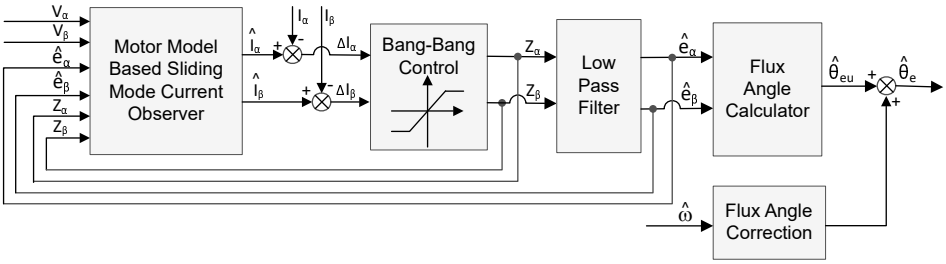


Figure 2-19. Block Diagram of Traditional Sliding Mode Observer

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

$$\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = \frac{\omega_c}{s + \omega_c}\begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} \tag{53}$$

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, defined as follow

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) \tag{54}$$

Low pass filter removes the high-frequency term of the sliding mode function, which leads to occur phase delay resulting. It can be compensated by the relationship between the cut-off frequency ω_c and back EMF frequency ω_e, which is defined as:

$$\Delta\theta_e = -\tan^{-1}\left(\frac{\omega_e}{\omega_c}\right) \tag{55}$$

And then the estimated rotor position by using SMO method is:

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) + \Delta\theta_e \tag{56}$$

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 51 in α-β coordinates is given by Equation 57 as:

$$\begin{bmatrix}\hat{i}_\alpha(n+1)\\ \hat{i}_\beta(n+1)\end{bmatrix} = \begin{bmatrix}F_\alpha\\ F_\beta\end{bmatrix}\begin{bmatrix}\hat{i}_\alpha(n)\\ \hat{i}_\beta(n)\end{bmatrix} + \begin{bmatrix}G_\alpha\\ G_\beta\end{bmatrix}\begin{bmatrix}V^*_\alpha(n) - \hat{e}_\alpha(n) + z_\alpha(n)\\ V^*_\beta(n) - \hat{e}_\beta(n) + z_\beta(n)\end{bmatrix} \tag{57}$$

Where the matrix $[F]$ and $[G]$ are given by Equation 58 and Equation 59 as:

$$\begin{bmatrix}F_\alpha\\ F_\beta\end{bmatrix} = \begin{bmatrix}e^{-\frac{R_S}{L_d}}\\ e^{-\frac{R_S}{L_q}}\end{bmatrix} \tag{58}$$

$$\begin{bmatrix}G_\alpha\\ G_\beta\end{bmatrix} = \frac{1}{R_S}\begin{bmatrix}1 - e^{-\frac{R_S}{L_d}}\\ 1 - e^{-\frac{R_S}{L_q}}\end{bmatrix} \tag{59}$$

The time discrete form of Equation 53 is given by Equation 60 as:

$$\begin{bmatrix}\hat{e}_\alpha(n+1)\\ \hat{e}_\beta(n+1)\end{bmatrix} = \begin{bmatrix}\hat{e}_\alpha(n)\\ \hat{e}_\beta(n)\end{bmatrix} + 2\pi f_c\begin{bmatrix}z_\alpha(n) - \hat{e}_\alpha(n)\\ z_\beta(n) - \hat{e}_\beta(n)\end{bmatrix} \tag{60}$$

2.4.2.2.1.3 Rotor Position and Speed Estimation with PLL

With the arc tangent method, the accuracy of the position and velocity estimations are affected due to the existence of noise and harmonic components. To eliminate this issue, the PLL model can be used for velocity and position estimations in the sensorless control structure of the IPMSM. The PLL structure used with SMO is illustrated in Section 2.4.2.2.1.2. The back-EMF estimations $\hat{e}_\alpha$ and $\hat{e}_\beta$ can be used with a PLL model to estimate the motor angular velocity and position as shown in Figure 2-20.

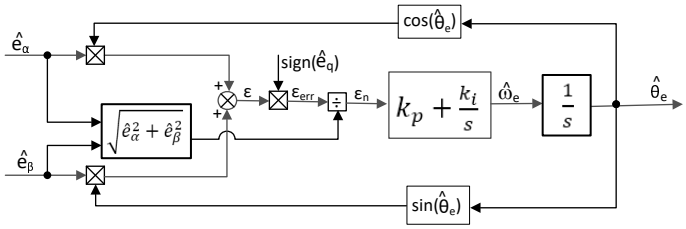


Figure 2-20. Block Diagram of Phase Locked Loop Position Tracker

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:

$$\varepsilon = \hat{e}_\beta\cos\left(\hat{\theta}_e\right) - \hat{e}_\alpha\sin\left(\hat{\theta}_e\right) = E\sin(\theta_e)\cos\left(\hat{\theta}_e\right) - E\cos(\theta_e)\sin\left(\hat{\theta}_e\right) = E\sin\left(\theta_e - \hat{\theta}_e\right) \tag{61}$$

Where E is the magnitude of the EEMF, which is proportional to the motor speed ω_e. When $\left(\theta_e - \hat{\theta}_e\right) < \frac{\pi}{2}$, the Equation 61could be simplified as

$$\varepsilon = E\left(\theta_e - \hat{\theta}_e\right) \tag{62}$$

Further, the position error after the normalization of the EEMF can be obtained:

$$\varepsilon_n = \theta_e - \hat{\theta}_e \tag{63}$$

According to the analysis, the simplified block diagram of the quadrature phase locked loop position tracker can be obtained as shown in Figure 2-21. The closed-loop transfer functions of the PLL can be expressed as:

$$\frac{\hat{\theta}_e}{\theta_e} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} = \frac{2\xi\omega_n s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{64}$$

where the k_p and k_i are the proportional and the integral gains of the standard PI regulator, its natural frequency ω_n and the damping ratio ξ is given:

$$k_p = 2\xi\omega_n, \qquad k_i = \omega_n^2 \tag{65}$$

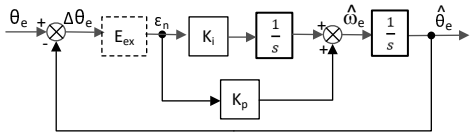


Figure 2-21. Simplified Block Diagram of Phase Locked Loop Position Tracker

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

Permanent magnet synchronous motor (PMSM) is widely used in home appliance applications due to its 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 their 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 its base speed, expanding its operating limits to reach speeds higher than rated speed and allow optimal 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 66 and Equation 67.

$$v_d = L_d\frac{di_d}{dt} + R_s i_d - p\omega_m L_q i_q \tag{66}$$

$$v_q = L_q\frac{di_q}{dt} + R_s i_q + p\omega_m L_d i_d + p\omega_m \psi_m \tag{67}$$

The dynamic equivalent circuit of an IPM synchronous motor is shown in Figure 2-22.

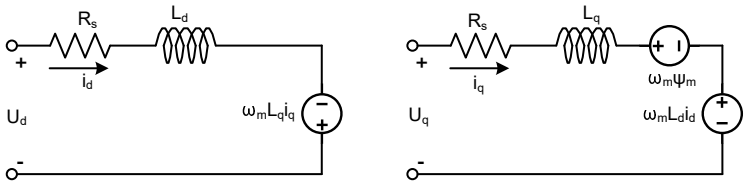


Figure 2-22. Equivalent Circuit of an IPM Synchronous Motor

The total electromagnetic torque generated by the IPMSM can be expressed as Equation 68 that the produced torque is composed of two distinct terms. The first term corresponds to the mutual reaction torque occurring between torque current i_q and the permanent magnet ψ_m, while the second term corresponds to the reluctance torque due to the differences in d-axis and q-axis inductance.

$$T_e = \frac{3}{2}p\left[\psi_m i_q + \left(L_d - L_q\right)i_d i_q\right] \tag{68}$$

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 69 and Equation 70.

$$I_a = \sqrt{i_d^2 + i_q^2} \le I_{max} \tag{69}$$

$$V_a = \sqrt{v_d^2 + v_q^2} \le V_{max} \tag{70}$$

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 71 if Space Vector Modulation (SVPWM) is adopted.

$$\sqrt{v_d^2 + v_q^2} \le v_{max} = \frac{v_{dc}}{\sqrt{3}} \tag{71}$$

Usually the stator resistance R_s is negligible at high speed operation and the derivate of the currents is zero in steady state, thus Equation 72 is obtained as shown.

$$\sqrt{L_d^2\left(i_d + \frac{\psi_{pm}}{L_d}\right)^2 + L_q^2 i_q^2} \le \frac{V_{max}}{\omega_m} \tag{72}$$

The current limitation of Equation 69 produces a circle of radius I_{max} in the d-q plane, and the voltage limitation of Equation 71 produces an ellipse whose radius V_{max} decreases as speed increases. The resultant d-q plane current vector must be controlled to obey the current and voltage constraints simultaneously. According to these constraints, three operation regions for the IPMSM can be distinguished as shown in Figure 2-23.

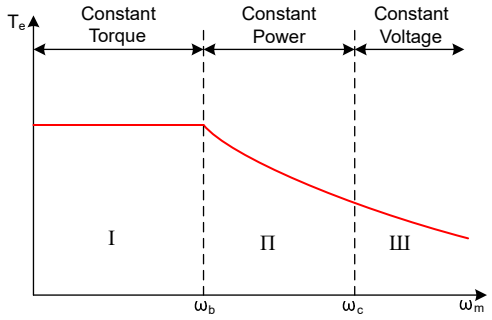


Figure 2-23. IPMSM Control Operation Regions

1. Constant Torque Region: MTPA can be implemented in this operation region to ensure 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 68, 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 a SPM motors only utilizes electromagnetic torque by setting the commanded i_d to zero for non-field weakening modes. But an IPMSM will utilize the reluctance torque of the motor, d-axis current should be controlled as well. The aim of the MTPA control is to calculate the reference currents i_d and i_q to maximize the ratio between produced electromagnetic torque and reluctance torque. The relationship between i_d and i_q, and the vectorial sum of the stator current I_s is shown in the following equations.

$$I_s = \sqrt{i_d^2 + i_q^2} \tag{73}$$

$$I_d = I_s \cos\beta \tag{74}$$

$$I_q = I_s \sin\beta \tag{75}$$

Where β is the stator current angle in the synchronous (d-q) reference frame. Equation 68 can be expressed as Equation 76 where I_s substituted for i_d and i_q.

Equation 76 shows that motor torque depends on the angle of the stator current vector; as such

$$T_e = \frac{3}{2} p I_s \sin\beta \left[\psi_m + (L_d - L_q) I_s \cos\beta \right] \tag{76}$$

, 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 77.

$$\frac{dT_e}{d\beta} = \frac{3}{2} p \left[\psi_m I_s \cos\beta + (L_d - L_q) I_s^2 \cos 2\beta \right] = 0 \tag{77}$$

Following, the current angle of the MTPA control can be derived as in Equation 78.

$$\beta_{mtpa} = \cos^{-1} \frac{-\psi_m + \sqrt{\psi_m^2 + 8*(L_d - L_q)^2 * I_s^2}}{4*(L_d - L_q)*I_s} \tag{78}$$

Thus, the effective d-axis and q-axis reference currents can be expressed by Equation 79 and Equation 80 using the current angle of the MTPA control.

$$I_d = I_s * \cos\beta_{mtpa} \tag{79}$$

$$I_q = I_s * \sin\beta_{mtpa} \tag{80}$$

However, as shown in Equation 78, the angle of the MTPA control, β_{mtpa} is related to d-axis and q-axis inductance. This means that the variation of inductance will impede the ability to find the optimal MTPA point. To improve the efficiency of a motor drive, the d-axis and q-axis inductance should be estimated 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 ensures 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, assumes 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 their d-q currents respectively, as shown in Equation 81 and Equation 82.

$$L_d = f_1(i_d, i_q) = f_1(i_d) \tag{81}$$

$$L_q = f_2(i_q, i_d) = f_2(i_q) \tag{82}$$

To reduce the ISR calculation burden by simplifying Equation 78. The motor-parameter-based constant, K_{mtpa} is expressed instead as Equation 84, where K_{mtpa} is computed in the background loop using the updated L_d and L_q.

$$K_{mtpa} = \frac{\psi_m}{4*(L_q - L_d)} = 0.25 * \frac{\psi_m}{(L_q - L_d)} \tag{83}$$

$$\beta_{mtpa} = cos^{-1}\left(K_{mtpa}/I_s - \sqrt{(K_{mtpa}/I_s)^2 + 0.5} \right) \tag{84}$$

A second intermediate variable, G_{mtpa} described in Equation 85, is defined to further simplify the calculation. Using G_{mtpa}, the angle of the MTPA control, β_{mtpa} can be calculated as Equation 86. These two calculations are performed in the ISR to achieve a real current angle β_{mtpa}.

$$G_{mtpa} = K_{mtpa}/I_s \tag{85}$$

$$\beta_{mtpa} = cos^{-1}\left(G_{mtpa} - \sqrt{G_{mtpa}^2 + 0.5}\right) \tag{86}$$

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 69 and Equation 70. 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 72, the IPMSM has two factors: one is a permanent magnet value and the other is made by inductance and current of flux.

Figure 2-24 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 its limit, the input of the PI controller of FW is always positive and therefore the output is always saturated at 0.

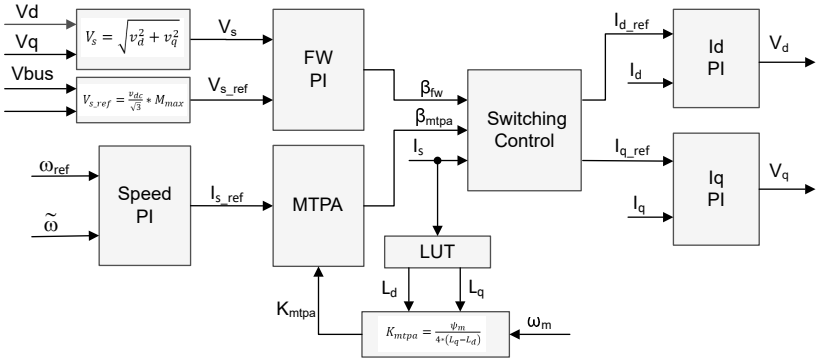


Figure 2-24. Block Diagram of Field-Weakening and Maximum Torque per Ampere Control

Figure 2-13 and Figure 2-15 show the implementation of FAST or eSMO based FOC block diagram. The block diagrams provide an overview of the FOC system's functions and variables. 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 show in Figure 2-25.

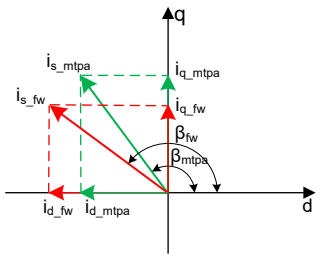


Figure 2-25. Current Phasor Diagram of an IPMSM During FW and MTPA

The switching control module is used to decide which angle should be applied, and then calculate the reference $i_d$and i_q as shown in Equation 74 and Equation 75. The current angle is chosen as following Equation 87 and Equation 88.

$$\beta = \beta_{fw} \, if \, \beta_{fw} > \beta_{mtpa} \quad (87)$$

$$\beta = \beta_{mpta} \, if \, \beta_{fw} < \beta_{mtpa} \quad (88)$$

Figure 2-26 is the flowchart that shows the steps required to run InstaSPIN-FOC with FW and MPTA in the main loop and interrupt.

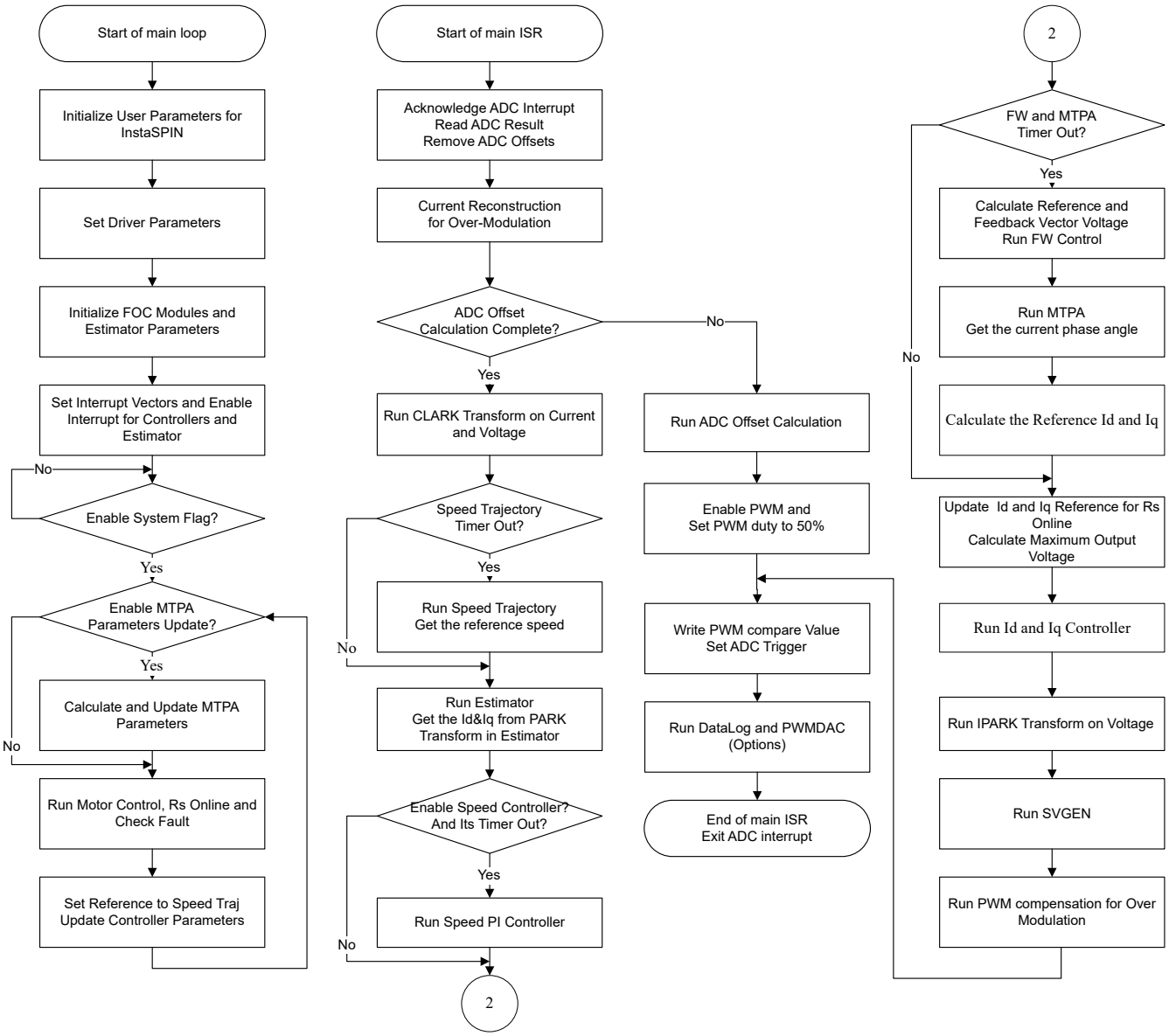


Figure 2-26. Flowchart for an InstaSPIN-FOC Project with FW and MTPA

2.4.2.4 Compressor Drive with Automatic Vibration Compensation

Vibration and noise can become a problem in air conditioning compressors applications since they cause an undesirable end user experience, as well as mechanical failures due to stress. The compressor applications contain pulsating loads, which is dependent on the mechanical angle as shown in Figure 2-27, can cause motor vibration and audible noise. There are several causes of vibration and noise. The main cause is the vibration produced by the load characteristic. A new dynamic and adaptive compensation method will also be covered, showing the details on how it operates and the minimal tuning required.

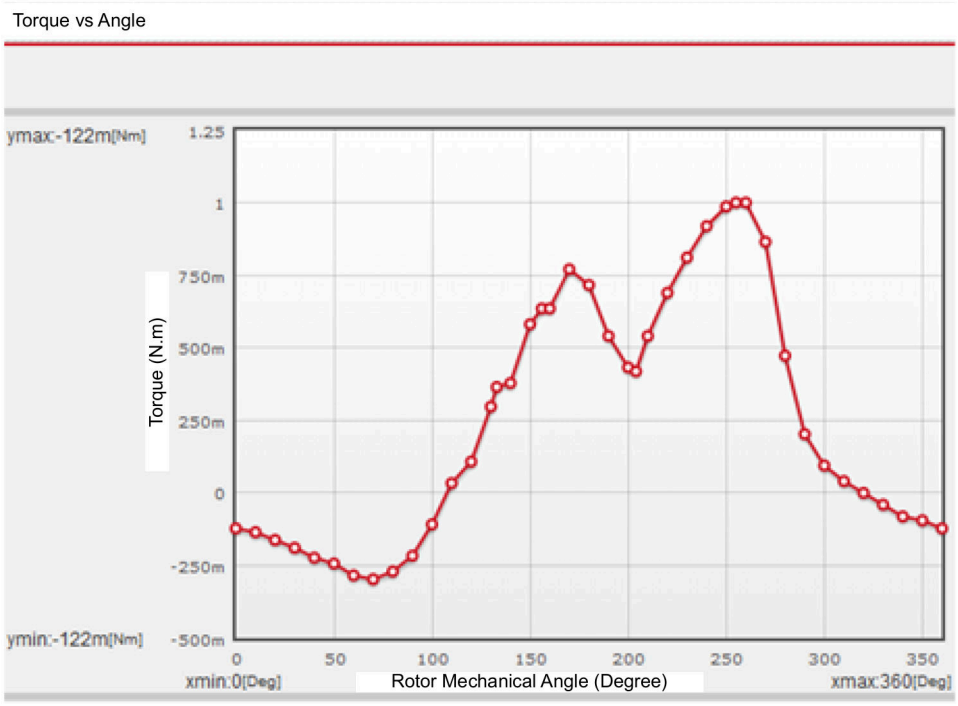


Figure 2-27. Waveform of Load Torque

The vibration compensation algorithm learns the load profile as the motor runs, and as the speed controller tries to correct for these load changes, and once the load is learned, the algorithm is used to extract load information relative to the mechanical angle, and uses that information as a feed forward in the speed controller. As shown in Figure 2-28, a new block was added, called *Dynamic Vibration Compensation* is added to the FOC system, is used to learn the torque load, to allow adding a feedforward term to the speed controller, in the form of a summing point to the output generated by the speed controller.

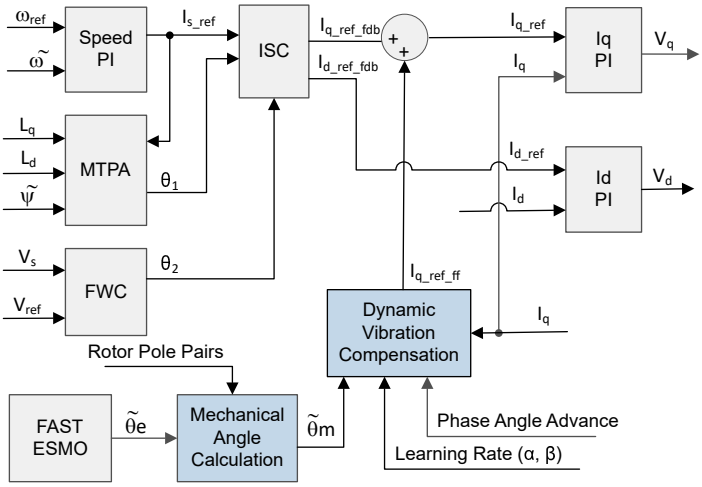


Figure 2-28. Block Diagram of FOC based Motor Drive with Vibration Compensation

This algorithm requires four main blocks to be able to work:

1. A speed controller with feed forward input
2. A table to hold a learning curve
3. A way of extracting a specific member of that table based on an index
4. Calculation of an index to update the learning curve and an advanced index to extract a value from that table

The algorithm is able to dynamically learn a load profile based on two inputs:

1. Electrical angle information. Starting from the lower left as shown in Figure 2-28, the first input that is required by the vibration compensation module is the mechanical angle. This is calculated based on the electrical angle and the number of pole pairs. It is not required for the mechanical angle to be synchronized with the electrical angle. For example, the zero of the mechanical angle, physically, does not need to be the zero of the electrical angle. This is because the vibration compensation module will learn the load according to the mechanical angle provided, independently of what the mechanical angle is compared to the physical position of the shaft.
2. A measured current value, in the case of FOC, Iq, which is responsible for the torque of the motor.

Then the vibration compensation module is implemented. The module needs four parameters in this implementation.

1. Phase Advance. which is how the learned load will be accessed with respect to the mechanical angle. If zero phase advance, the loaded value on IqRef_ff will correspond to the provided mechanical angle. If phase advance is 10, the loaded value on IqRef_ff will correspond to the mechanical angle plus 10 mechanical degrees (in a scale from 0 to 360 degrees).
2. Learning Rate. This parameter is a value from 0 to 1, which is essentially how fast (less noise immune) or how slow (more noise immune) the load learning happens
3. Points. This is the number of points of the compensation table to be used for the learning curve.
4. Pole Pairs. This is the number of poles in the motor.

Then the summation point in between the speed controller and the Iq controller. This is where the output of the vibration compensation module is used, to help the speed controller with this term. This technique is also known as feedforward, since the load is known in advance, according to the mechanical angle provided.

Once the load has been learned by the vibration compensation module, the speed controller will correct for transients in load change, that don't relate to the natural mechanical load vs. mechanical angle, which is already compensated by the vibration compensation module. To illustrate how the vibration compensation module helps, let's take a look at the following plot, where we show the output of the speed controller with vibration compensation disabled. It is obvious that the speed controller gains need to be high to track the load changes as the motor spins every cycle.

Tuning the learning rate

The learning rate can be adjusted depending on two factors. One is how quickly the user wants to learn the curve, and the second consideration is how noisy is the input to the learning curve. The second consideration is important, because the noise is not only coming from the current sensing method itself, but minor mechanical perturbations in the system, that are not periodic, and that we would like to filter out and not have it in our compensation table. If the learning rate is too low, the learning time could be too long for a specific application, so a trade-off needs to be made.

Tuning the phase angle

In a discrete system there are several delays when it comes to outputting a voltage to the motor, and also delays related to sensing currents. For example, when implementing an FOC system in a processor, usually the output voltage goes through a Pulse Width Modulator (PWM) which has delays. This phase advance parameter allows fine tuning of that delay, by providing a number in units of table positions, from the learned table, so that the appropriate output can be applied to the speed controller's feed forward input. The easiest way to tune this parameter is by looking at the speed variation after applying dynamic compensation, and tuning the value so that a minimum amount of speed variation is achieved.

2.4.2.5 Fan Drive with Flying Start

Flying start is a feature that allows the drive to determine the speed and direction of a spinning motor and begin the output voltage and frequency at that speed and direction. Without flying start, the drive will begin its output at zero volts and zero speed and attempt to ramp to the commanded speed. If the inertia or direction of rotation of a load requires the motor to produce a large amount of torque, excess current can result and overcurrent trips might occur on the drive. These problems can be eliminated with flying start.

Flying start is the capacity to start control at any speed other than **ZERO**, which is an important function in air-condition application for fan drive.

When a motor is started in its normal mode, the control initially applies a frequency of 0 Hz and ramps to the desired frequency. If the drive is started in this mode with the motor already spinning with non-zero frequency, large currents are generated. An over current trip can result if the current limiter cannot react quickly enough. Even if the current limiter is fast enough to prevent an over current trip, it can take an unacceptable amount of time for synchronization to occur and for the motor to reach its desired frequency. In addition, larger mechanical stress is placed on the application.

In flying start mode, the drive's response to a start command is to synchronize with the motor's speed (frequency and phase) and voltage. The motor then accelerates to the commanded frequency. This process prevents an over current trip and significantly reduces the time for the motor to reach its commanded frequency. Because the drive synchronizes with the motor at its rotating speed and ramps to the proper speed, little or no mechanical stress are present.

The flying start function implements an algorithm that searches for the rotor speed. The algorithm searches for a motor voltage that corresponds with the excitation current applied to the motor

When the motor is spinning, the speed and position information can be estimated from the BEMF voltages. Since the stator voltage is measured in InstaSPIN drive, the speed and position are easily obtained by switching the inverter. A zero torque current is applied to the motor and the generated current and stator voltage is measured, then InstaSPIN-FOC module uses these signals to estimate rotor position and speed.

The block diagram of FOC with flying start is shown in Figure 2-29, the flying start module outputs a flag to enable or disable speed close loop control. A zero reference torque current is set and the speed PI controller output is disabled while flying start is operating.

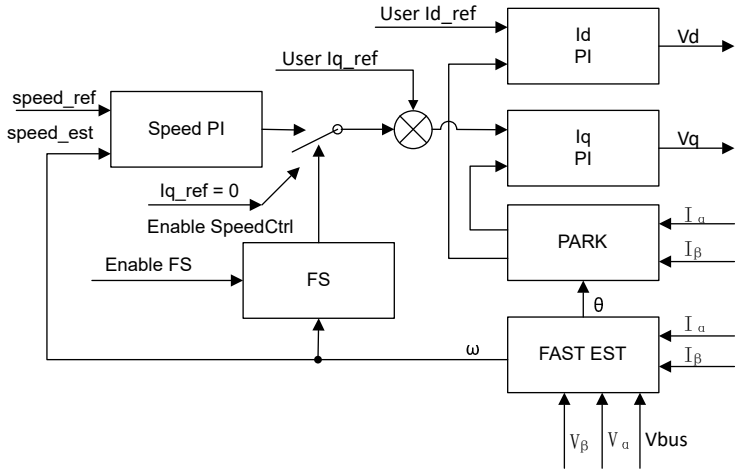


Figure 2-29. Flying Start Control Block Diagram

As shown in Figure 2-30, the module routine disables speed close loop control, sets the reference Iq to zero, and enables the FOC module during starting run the motor. After the phase currents and voltages are measured, the routine runs InstaSPIN-FOC and the real motor speed can be estimated. The program re-enables speed closed loop control and sets the speed reference value after flying start is completed. The current waveform during restart is as shown in Figure 3-32.

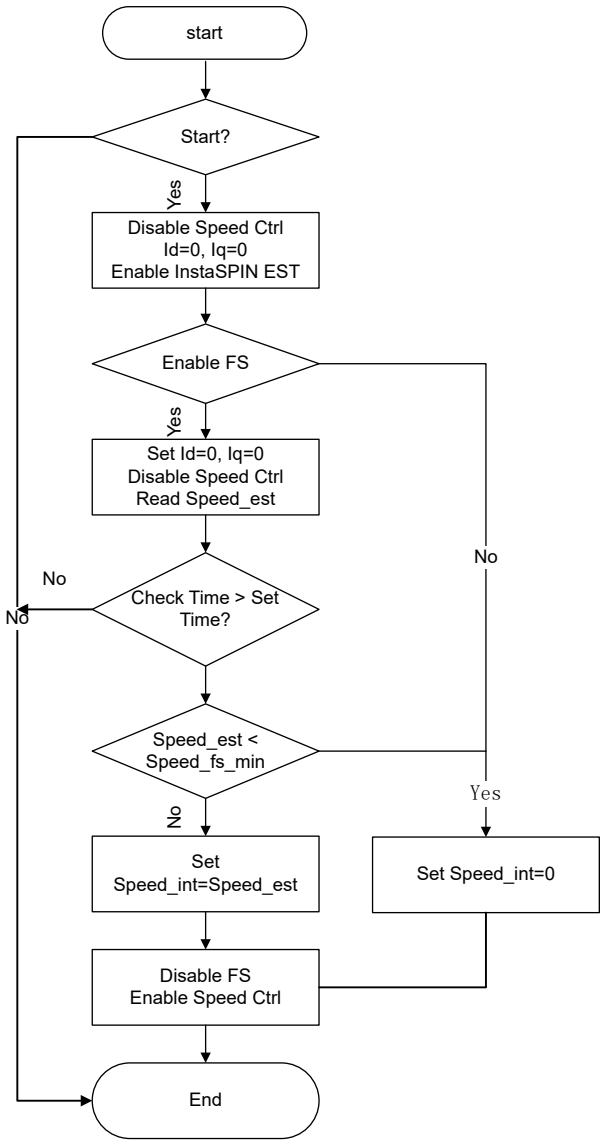


Figure 2-30. Flying Start Module Program Flowchart

2.4.2.6 Hardware Prerequisites for Motor Drive

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

2.4.2.6.1 Motor Current Feedback

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

- Single shunt current sensing
- Three shunt current sensing

Either one of these two current sensing techniques can be selected in the build configuration of the project. The "HVAC_REV3P2_3SC_LIB" build configuration supports three-shunt current sensing method, the "HVAC_REV3P2_1SC_LIB" supports single-shunt current sensing method as described in Section 3.2.2.

2.4.2.6.1.1 Current Sensing with Three-Shunt

The current through the motor is sampled by the microcontroller as part of the motor control algorithm during every one (Compressor) or three (Fan) PWM cycle. To measure bidirectional currents of the motor phase, that is, positive and negative currents, the circuits below require a reference voltage of 1.65-V. This offset reference voltage is created by a voltage follower as shown in Figure 2-31. This 1.65-V reference voltage is used to both fan and compressor motors phases current and PFC AC voltage feedback sensing circuit. In this revision hardware, compressor (Motor 1), Fan (Motor 2) and PFC have the offset reference separately, but these three sampling circuit can share the same offset reference in a design to save the cost.

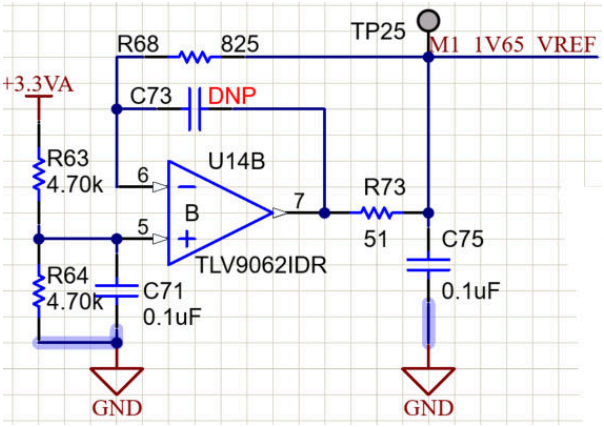


Figure 2-31. 1.65-V Reference from 3.3-V Input Circuit

Figure 2-32 shows how the motor current is represented as a voltage signal, with filtering, amplification, and offset to the center of the ADC input range. This circuit is used for each phase of the 3-phase PMSM of compressor and fan. The transfer function of this circuit is given by Equation 89.

$$V_{OUT} = V_{OFFSET} + (I_{IN} \times R_{SHUNT} \times G_i) \quad (89)$$

Where R_{shunt}=0.01(OHM) and V_{offset}=1.65(V)

The calculated resistance values lead to the sensing circuit shown in Figure 2-35, the G_i is given by Equation 90.

$$G_i = \frac{R_{fb}}{R_{in}} = \frac{R45}{(R54 + R39)} = \frac{7.5K}{(20 + 825)} = 8.876 \quad (90)$$

The maximum peak to peak current measurable by the microcontroller is given by Equation 91.

$$I_{scale\_max} = \frac{V_{ADC\_max}}{R_{SHUNT} \times G_i} = \frac{3.3}{0.01 \times 8.876} = 37.18A \quad (91)$$

which is the peak to peak value of ±18.59A. 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         (37.18f)
```

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 non-inverting pin of the operational amplifier. The highlighted sign is required to be configured to have correct polarity for the current feedback in software as shown in the following code snippet in motor1_drive.c:

```
// define the sign of current feedback based on hardware board
adcData[MTR_1].current_sf = -userParams[MTR_1].current_sf;
```

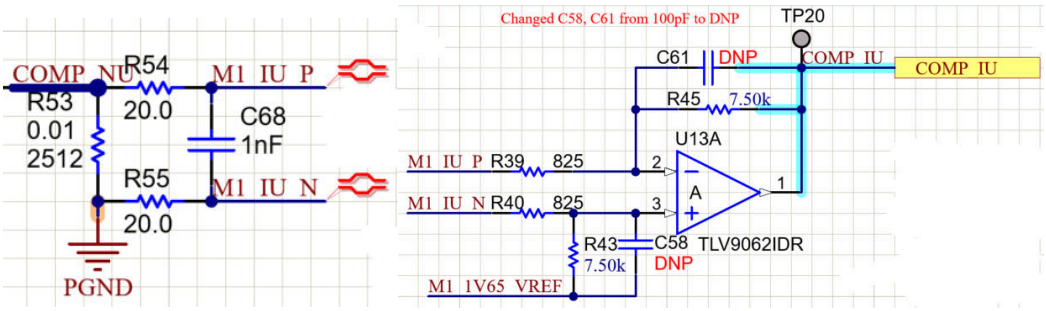


Figure 2-32. Motor Current Sensing Circuit with Three-Shunt

Implementing the same calculation steps for Fan motor, and set the scale values in user_mtr2.h file.

2.4.2.6.1.2 Current Sensing with Single-Shunt

The single shunt current sensing technique measures the dc-link bus current and, with knowledge of the power FET switching states, reconstructs the three phase current of the motor. The detailed description of the single shunt technique is described in the application note Sensorless-FOC for PMSM With Single DC-Link Shunt (spract7).

On this reference board, to implement 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 2-33.

1. Remove current shunt resistors R53, and R65, just keep a single shunt resistor R56 to sense the DC_link current for compressor drive.
2. Use a thick wire to connect the NU, NV, and NW pins together.

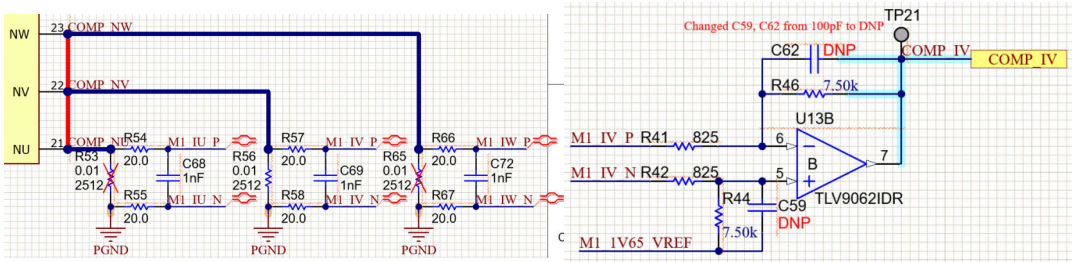


Figure 2-33. Motor Current Sensing Circuit with Single-Shunt

The DC-Link current is not bi-directional 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 2-34. Change the R64 from 4.70k (OHM) to 47.5K (OHM)/1% resistor for the reference voltage.

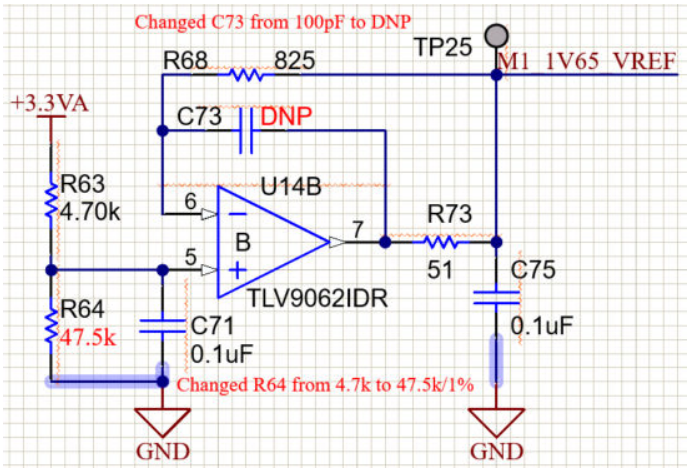


Figure 2-34. DC Offset Reference from 3.3-V Input Circuit

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

2.4.2.6.2 Motor Voltage Feedback

Voltage feedback is needed in the FAST estimator to allow the best performance at the widest speed range, the phase voltages are measured directly from the motor phases instead of a software estimate. The eSMO relies on software estimation values to represent the voltage phases without using the motor phase voltage sensing circuit. This software value (USER_ADC_FULL_SCALE_VOLTAGE_V) depends on the circuit that senses the voltage feedback from the motor phases. Figure 2-35 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 92, considering the maximum voltage for the ADC input is 3.3 V.

$$V_{FS} = V_{ADC\_FS} \times G_v = 3.3V \times 122.46 = 404.13V \tag{92}$$

Where G_v is attenuation factor can be calculated from

$$G_V = \frac{(R59 + R60 + R61 + R62)}{R62} = \frac{(332K + 332K + 332K + 8.2K)}{8.2K} = 122.46 \tag{93}$$

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         (404.13f)
```

The voltage filter pole is needed by the FAST estimator to allow an accurate detection of the voltage feedback. The filter should be 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. The hardware filter should only be changed when ultra-high-speed motors are run, which generate phase-voltage frequencies in the order of a few kHz.

The filter pole setting can be calculated as the following Equation 94 in this reference design:

$$f_{filter\_pole} = \frac{1}{(2 \times \pi \times R_{Parallel} \times C)} = 405.15 \quad Hz \tag{94}$$

$$where,$$

$$C = 47nF$$

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

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           (405.15f)
```

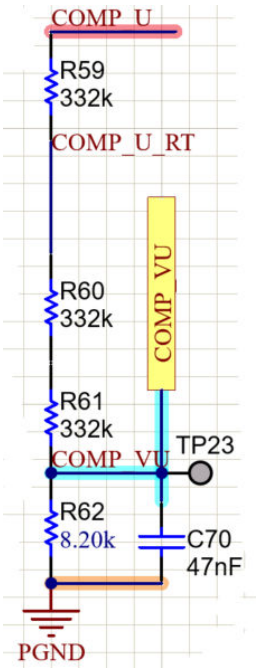


Figure 2-35. Motor Voltage Sensing Circuit

3 Hardware, Software, Testing Requirements, and Test Results

3.1 Getting Started Hardware

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

3.1.1 Hardware Board Overview

Figure 3-1 shows an overview of a typical dual motor control with PFC system running from AC power. The PFC stage enables wave shaping of the input AC current and provides the adjustable DC power for 3-phase inverter for compressor and fan motors drive.

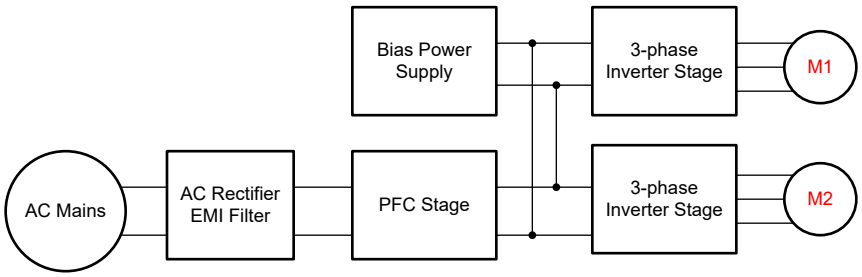


Figure 3-1. Hardware Board Block Diagram of TIDM-02010

The motor control board has functional groups that enable a complete motor drive system. The following is a list of the blocks on the board and their functions, Figure 3-2 shows the top view of the board and different blocks of the TIDM-02010 PCB.

- Power line input filter
  - Dual common mode filter with cut-off frequency is 2.2 kHz.
- Digital interleaved PFC
  - The maximum power is up to 1.5 kW.
  - Two phase interleaved boost PFC with 72 kHz switching frequency
  - Power MOSFET with UCC27517A high-speed gate drivers
- 3-phase inverter for motor_1 (Compressor)

  - Up to 1.5 kW 3-phase inverter supports PMSM or IPM
  - 6 kHz switching frequency
  - 3-shunt for current sensing
- 3-phase inverter for motor_2 (Fan)
  - Up to 150 W 3-phase inverter supports PMSM or IPM
  - 12 kHz or 18 kHz switching frequency
  - 3-shunt for current sensing
- Control
  - Single TMS320F280025C or TMS320F28039C series MCU in 64-pin LQFP package
  - 100 MHz 32-bit CPU with FPU and TMU
  - Amplify and input filters for the analog signals
- Valve and relay drive for system function
  - Relay drive for AC valve
  - Stepper motor drive for electronic expansion valve
- Auxiliary power supply
  - On-board power supply +3.3 V, +12 V, and +15 V

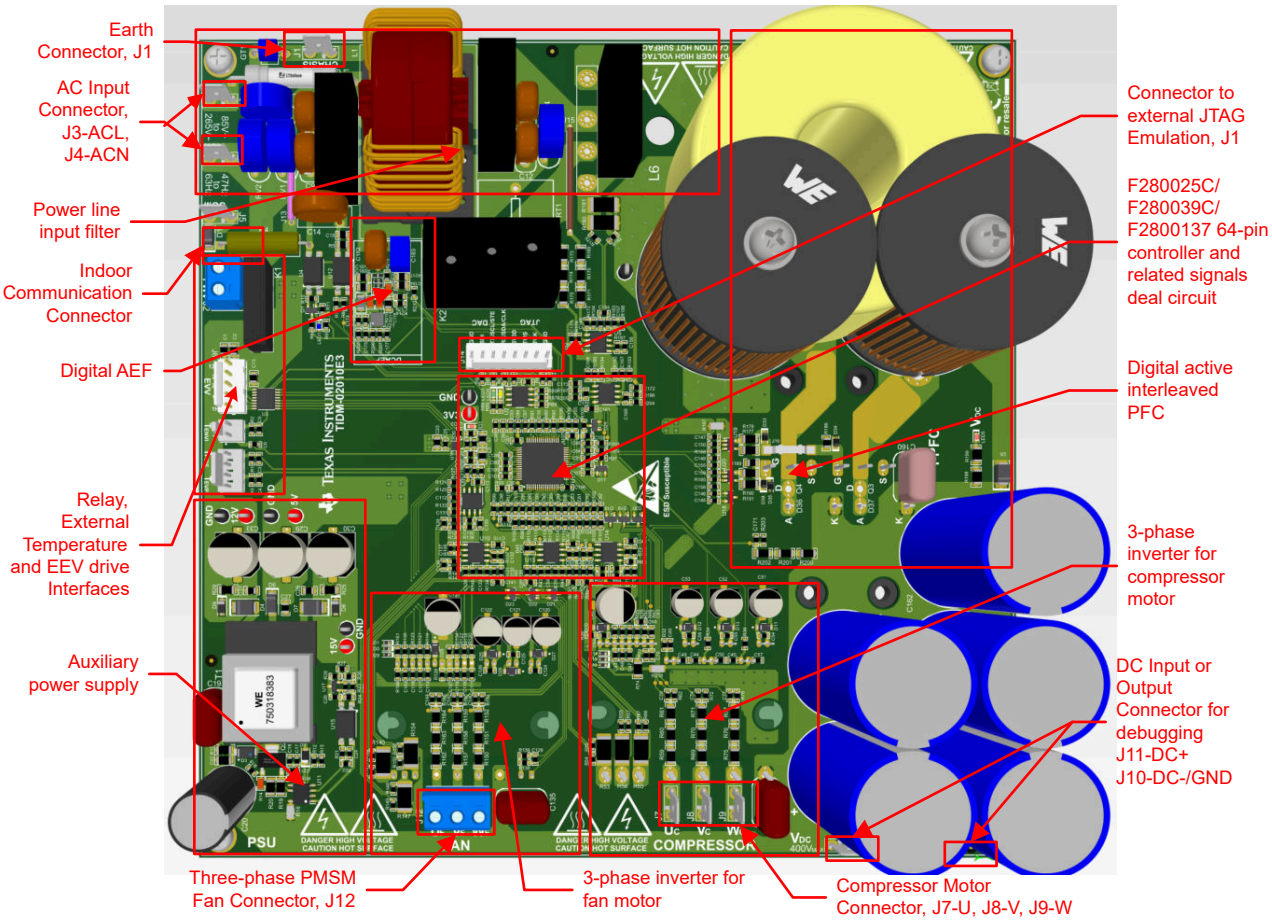


Figure 3-2. The Layout of Dual Motor Control with PFC Reference Design Board

Note

F28002x, F28003x, and F280013x are fully pin to pin compatible on this 64-pin package, so the same PCB board can be used for one of these three MCUs.

TI recommends taking the following precautions when using the board:

- Do not touch any part of the board or components connected to the board when the board is energized.
- Use the AC Mains/wall power supply to power the kit. TI recommends an isolation AC source.

- Do not touch any part of the board, the kit or its assembly when energized. (Though the power module heat sink is isolated from the board, high-voltage switching generates some capacitive coupled voltages over the heat sink body.)
- Control Ground can be hot.

3.1.2 Test Conditions

For the user to test the reference design software

- PFC Operation
  - For input, the power supply source must range from 165-VAC to 265-V AC. Set the input current limit of input AC source to 10A for full power test, but start with a lower current limit during initial board bring-up.
  - For output, use an electronic variable load or a variable resistive load, which must be rated for ≥ 400 V and must vary the load current from 0 mA to 5 A.
- Motor 1 (Compressor) Drive Operation
  - For input, the power supply source must range from 165-V to 265-V AC if using the AC source, or must range from 100 V to 400 V if using the DC power supply. Set the input current limit of input AC source to 10 A or DC power supply to 6.5 A, but start with a lower current limit during initial board bring-up.
  - For output, a 3-phase PMSM with dynamo meters
- Motor 2 (Fan) Drive Operation
  - For input, the power supply source (VIN) must range from 165-V to 265-V AC if using the AC source, or must range from 100 V to 400 V if using the DC power supply. Set the input current limit of input AC source to 3 A or DC power supply to 1.5 A, but start with a lower current limit during initial board bring-up.
  - For output, a 3-phase PMSM with dynamo meters

3.1.3 Test Equipment Required for Board Validation

- Isolated AC source
- Single-phase power analyzer
- Digital oscilloscope
- Multimeters
- Electronic or resistive load
- DC power supply
- 1.5 Kw and 150 W 3-phase PM synchronous motors
- Dynamo meters
- Three-phase power analyzer

3.1.4 Test Setup

Figure 3-2 shows the position of these blocks and the connectors on the board. Setting up the hardware following the steps below.

1. Connect a JTAG emulator to connector J14 to debug or program C2000 device.

   In this HVAC hardware board, cJTAG mode with 2-pin (TMS and TCK) must be used for the external emulator.
2. Connect the compressor motor wires to the terminals J7, J8 and J9 after completing the first incremental build step.
3. Connect the fan motor wires to the terminals J12 after completing the first incremental build step.
4. Apply a DC bus power, AC power supply or AC mains power to the inverter by connecting the power to J3 and J4 when instructed in the step-by-step procedure in Section 3.3.
   a. The maximum output of the DC power supply is 380 VDC.
   b. The maximum output of the AC power supply is 265 VAC, 50/60Hz.
   c. AC main power is 220 VAC, 50/60Hz.
5. Connect output terminals J10 and J11 to the electronic load maintaining correct polarity, or resistive load as described in the step-by-step procedure in Section 3.3. J11 is the VDC output and J10 is the GND terminal.
6. Connect multimeter, oscilloscope probes, and other measurement equipment to probe or analyze various signals and parameters as desired. Only use appropriately rated equipment and follow proper isolation and safety practices.

Note

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

WARNING

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

3.2 Getting Started Firmware

Download and install C2000WARE-MOTORCONTROL-SDK v4.01.00.00 or newer software from the link provided by TI, install this Motor Control SDK software in its default folder.

The software of this reference design is available in C2000WARE-MOTORCONTROL-SDK.
The software project would then reside inside C2000Ware Motor Control SDK folder at `<install_location>\solutions\tidm_02010_dmpfc\`. Follow these steps to build and run this code with different incremental builds.

3.2.1 Download and Install Software Required for Board Test

1. Download and install Code Composer Studio from the Code Composer Studio (CCS) Integrated Development Environment (IDE) tools folder. Version 12.0 or newer is recommended.
2. Install C2000WARE-MOTORCONTROL-SDK in one of two ways:
   - Download the software through the C2000Ware MotorControl SDK tools folder
   - Go to CCS and under View → Resource Explorer. Under the TI Resource Explorer, go to Software→C2000Ware_MotorControl_SDK, and click the install button.
3. Once installation complete, close CCS, and create a new workspace for importing the project.

Note

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

3.2.2 Opening Project Inside CCS

There are three projects for F28002x, F28003x and F280013x separately, the projectspec file for F28002x based reference design is in the directory below`<install_location>\solutions\tidm_02010_dmpfc\f28002x\ccs\sensorless_foc`, the projectspec file for F28003x based reference design is in the directory below`<install_location>\solutions\tidm_02010_dmpfc\f28003x\ccs\sensorless_foc`, projectspec file for F280013x based reference design is in the directory below`<install_location>\solutions\tidm_02010_dmpfc\f280013x\ccs\sensorless_foc`

Import the project within CCS and select the right build configurations by right-clicking on project name as shown in Figure 3-3. Select the right build configuration for the HVAC reference design. The "HVAC_REV3P2_3SC_LIB" build configuration supports three-shunt current sensing method, the "HVAC_REV3P2_1SC_LIB" supports single-shunt current sensing method.

Configure the project to select the supporting functions in project by using either one of the following two ways:

- Navigate to `<install_location>\solutions\tidm_02010_dmpfc\f28002x\ccs\sensorless_foc` for F28002x, `<install_location>\solutions\tidm_02010_dmpfc\f28003x\ccs\sensorless_foc` for F28003x, or

<install_location>\solutions\tidm_02010_dmpfc\f280013x\ccs\sensorless_foc for F280013x. Open the project specs file with a text editor utility, select the right predefined symbols for the system as shown in Figure 3-4. The pre-define symbol is active or disabled by removing or adding the "_N" in the name. For example, enables field weakening control by removing the "_N" in "MOTOR1_FWC_N" to "MOTOR1_FWC", disables field weakening control functions for motor 1 (Compressor) by changing "MOTOR1_FWC" symbol name to "MOTOR1_FWC_N". After set the pre-define symbols in "tidm_02010_dmpfc_002x/003x/0013x.projectspec", import the project by clicking "Project → Import CCS Projects" to select the project based the hardware board in the related folder.

- Click "Project → Import CCS Projects", and browse to <install_location>\solutions\tidm_02010_dmpfc\f28002x\ccs\sensorless_foc to import the project into CCS for F28002x based reference design or <install_location>\solutions\tidm_02010_dmpfc\f28003x\ccs\sensorless_foc for F28003x based reference design, or <install_location>\solutions\tidm_02010_dmpfc\f280013x\ccs\sensorless_foc for F280013x based reference design, and then right-click on the imported project name, click "Properties" command to set the pre-define symbols for the project as shown in Figure 3-5.

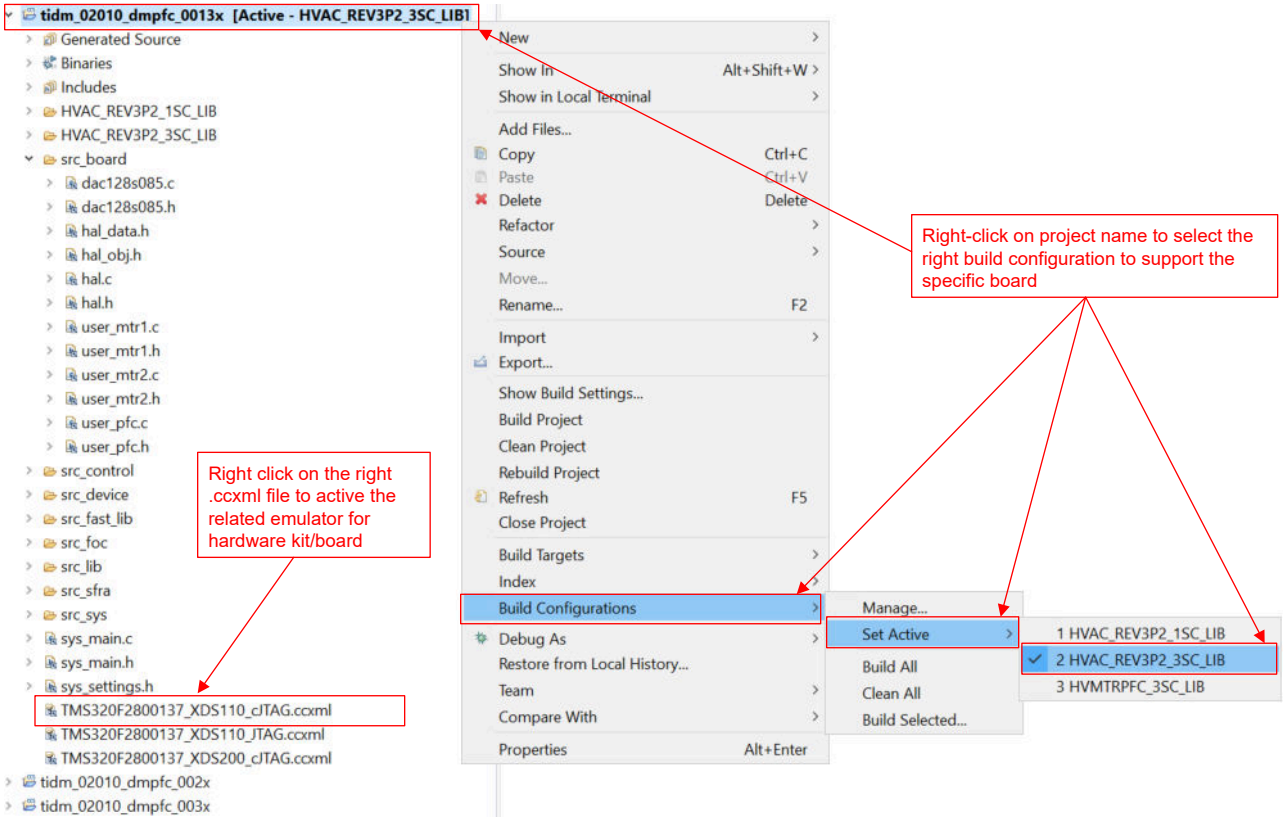


Figure 3-3. Select the Correct Build Configurations

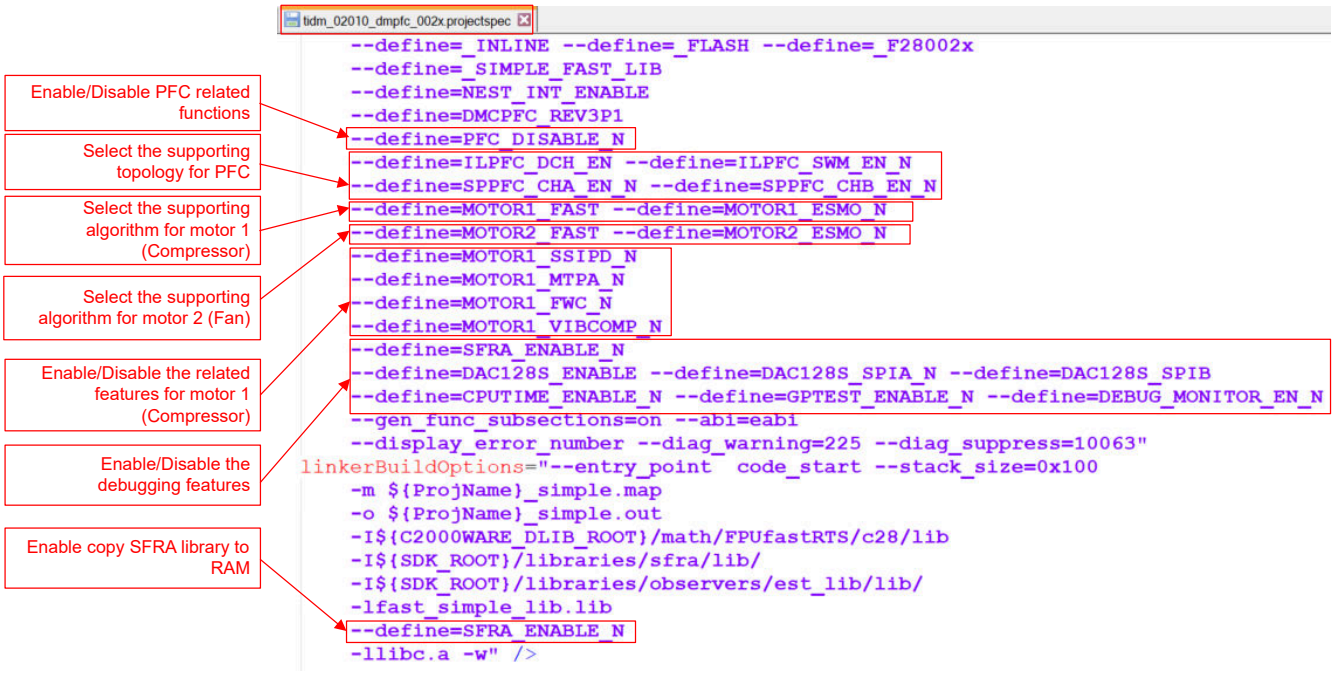


Figure 3-4. Select the Correct Pre-define Symbols in ProjectSpec File

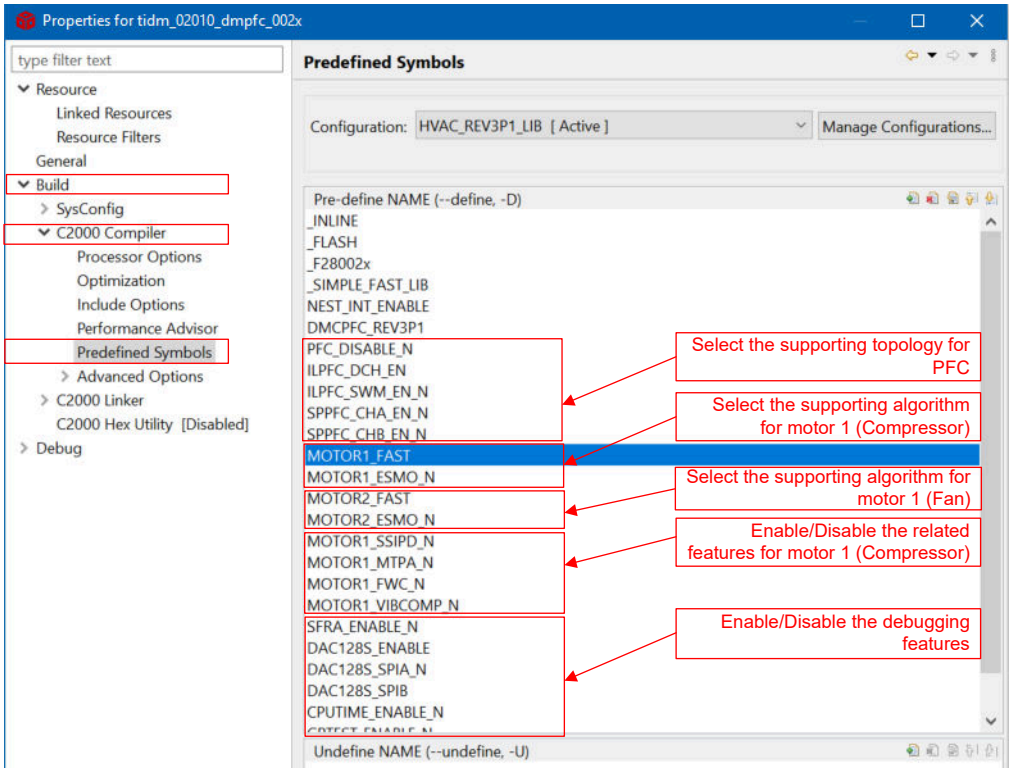


Figure 3-5. Select the Correct Pre-define Symbols in Project Properties

3.2.3 Project Structure

The general structure of the project is shown in Figure 3-6. The device peripherals configuration is based on C2000Ware driverlib. The users only need to change the codes and definitions in *hal.c* and *hal.h*, and the parameters in *user_mtr1/mtr2/pfc.h* if they want to migrate the reference design software to their own board.

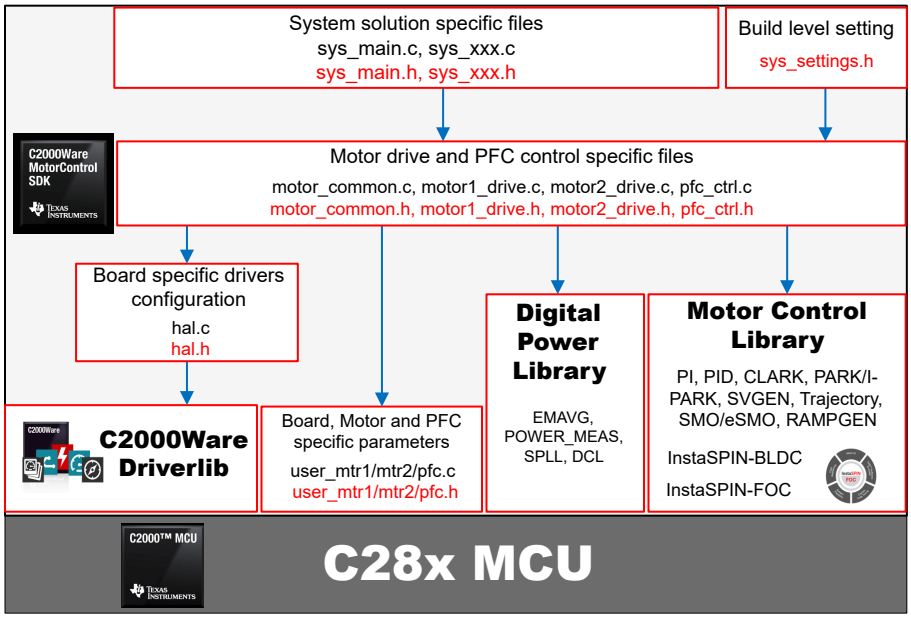


Figure 3-6. Project Structure Overview

Once the project is imported, the project explore will appear inside CCS as shown in Figure 3-7.

The folder *src_foc* includes the typical FOC modules include transform, PID and estimators that consist of the motor drive algorithm are independent on specific device and board.

The folder *src_lib* includes the digital power control modules for PFC, InstaSPIN-FOC library and math library that is independent of device and board.

The folder *src_sys* includes some files reserved for system control that are independent on specific device and board.

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

The F28002x, F28003x and F280013x projects share all of the control algorithm files. Board-specific, motor-specific and device-specific files are in the folder of *src_board*, these files consist of device specific drivers to run the solution. The F28002x, F28003x and F280013x have a dedicated *hal.c*, *hal.h*, *user_mtr1.h*, *user_mtr2.h*, and *user_pfc.h* separately. So the user only needs to make some changes in these header files, *hal.c*, *hal.h*, *user_mtr1.h*, *user_mtr2.h*, and *user_pfc.h* based on the usage of device peripherals for the board if the user wants to migrate the project into a different board or device.

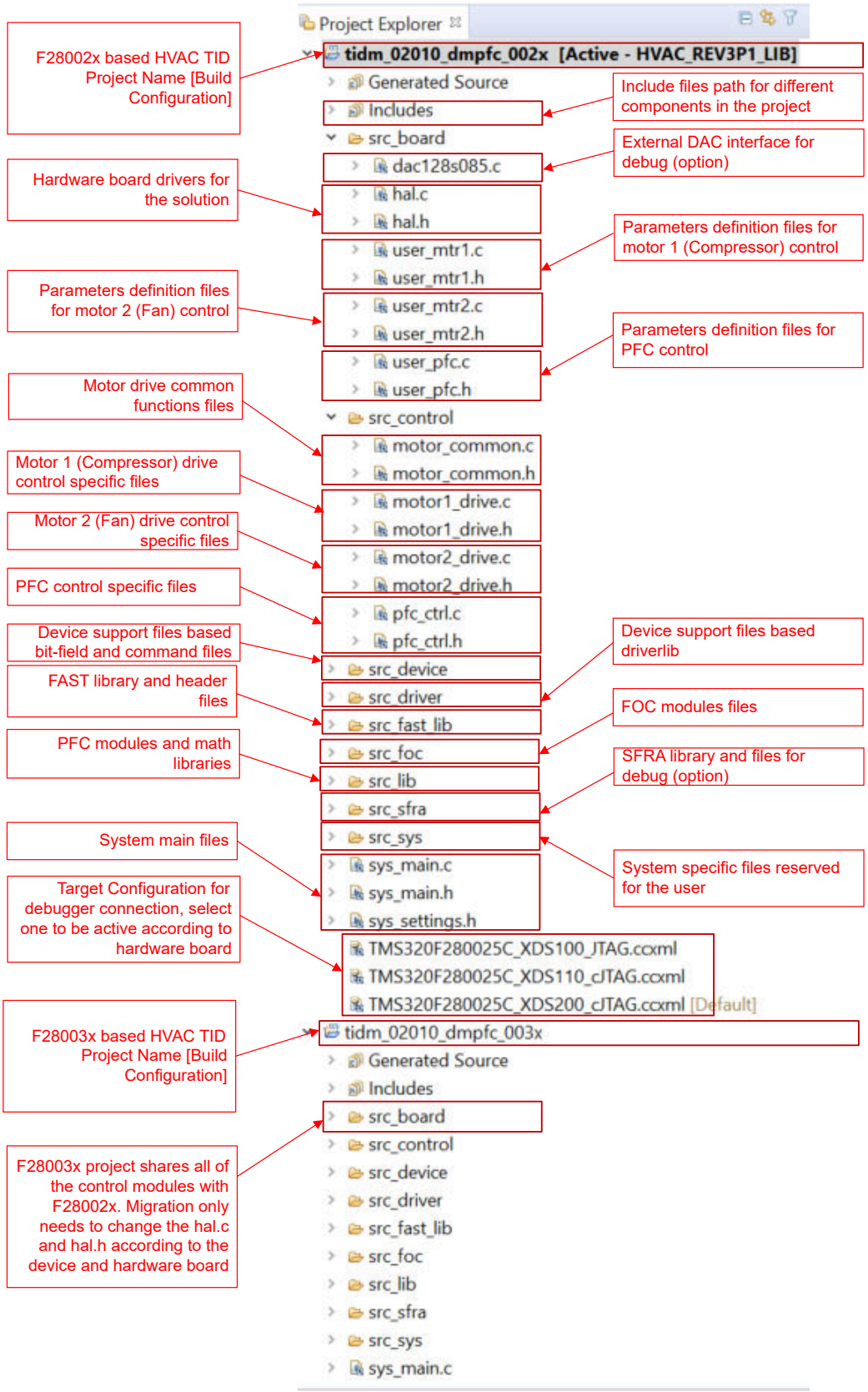


Figure 3-7. Project Explorer View of the Dual Motor Control and PFC Project

Figure 3-8 shows the project software flow diagram of the firmware that includes three ISRs for PFC and motor control, a main loop for PFC and motor control parameters update in background loop.

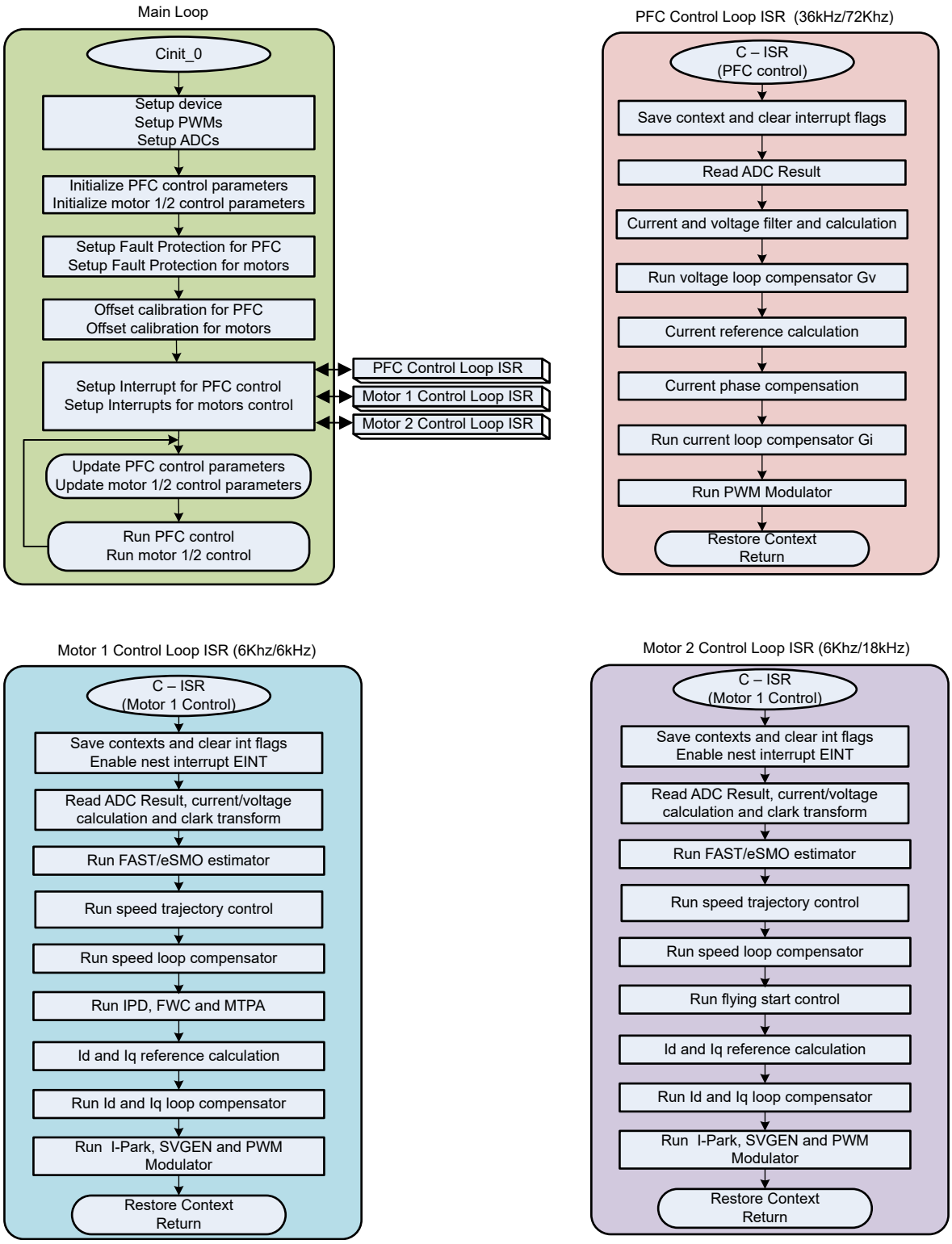


Figure 3-8. Project Flow Diagram of the Firmware

The project consists of three interrupt service routines, which are called every PWM cycle or every two PWM cycles. A few background tasks are called in a polling fashion and can be used to run slow tasks for which absolute timing accuracy is not required, such as PFC control parameters update, 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 (Compressor) that is periodically triggered at USER_M1_ISR_FREQ_Hz.

motor2CtrlISR is reserved for calling the motor drive control algorithms to spin the motor 2 (Fan) that is periodically triggered at USER_M2_ISR_FREQ_Hz.

pfcCtrlISR is reserved for calling the digital power control algorithms to implement PFC that is periodically triggered at USER_PFC_ISR_FREQ_Hz.

To simplify the system bring up and design the software of this reference design is organized in four labs with incremental builds (PFC_BUILDLEVEL and DMC_BUILDLEVEL), which makes learning and getting familiar with the board and software easier. This approach is also good for debugging and testing boards. The detailed incremental build options is shown as in Table 3-1. 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 *rebuild all* compiler option. Section 3.3 provides more details to run each of the build level options.

Table 3-1. Incremental Build Options

OPERATION	BUILD OPTION	DESCRIPTION
PFC	PFC_LEVEL_1	50% PWM duty output, verify ADC offset calibration, PWM output and phase shift
	PFC_LEVEL_2	Open loop control to check current and voltage sensing signals for PFC
	PFC_LEVEL_3	Closed current loop
	PFC_LEVEL_4	Closed current and voltage loop
MOTOR DRIVE	DMC_LEVEL_1	50% PWM duty, verify ADC offset calibration, PWM output and phase shift
	DMC_LEVEL_2	Open loop control to check current and voltage sensing signals for motor 1 and 2
	DMC_LEVEL_3	Closed current loop to check the hardware settings
	DMC_LEVEL_4	Motor parameters identify and run with InstaSPIN-FOC/eSMO

3.3 Test Procedure

CAUTION

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

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

3.3.1 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 setup 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 motor 1, motor 2, PFC. This build level verifies the sensing of feedback values from the power stage and also operation of the PWM gate driver and ensures there are no hardware issues. Additionally calibration of input and output voltage sensing can be performed in this build level. The software flow for this build level is shown in Figure 3-9.

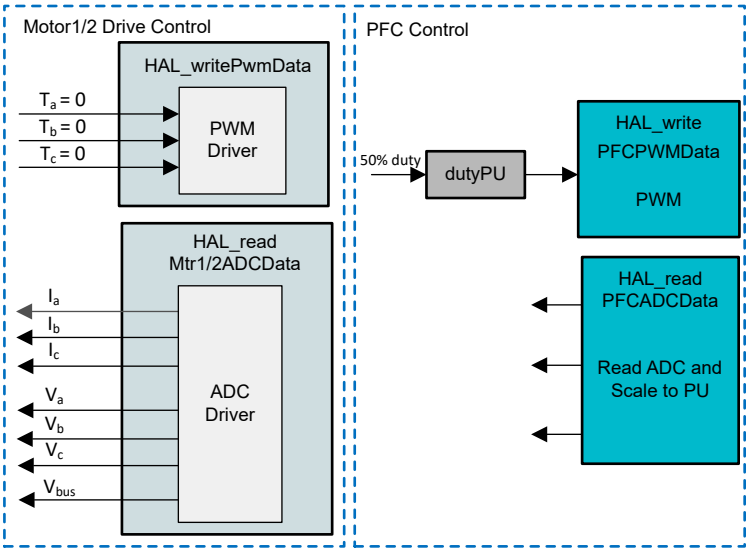


Figure 3-9. Control Software Block Diagram: Build Level 1 – Offset Validation

3.3.1.1 Start CCS and Open Project

1. Connect +5 V, +15 V and +12 V power supply to the related test points, TP7 for +5 V, TP8 for +15 V, TP6 for +12 V, TP5, TP9, TP11 for GND on board to provide the power supplies for gate drivers and controller is as shown in Figure 3-10.
2. Open CCSv11.0 (or newer). A project contains all the files and build options needed to generate an executable output file (.out), which can be run on the C2000 controller based hardware. On the menu bar click *Project* → *Import CCS Projects*. Below *Select search-directory:*, browse to the C2000Ware Motor Control SDK folder and select `<install_location>\solutions\ tidm_02010_dmpfc`, click *Finish* to import the related project into CCS. This project invokes all the necessary tools (compiler, assembler, linker) to build the project.
3. In the project window on the left, click the plus sign (+) to the left of Project. An example project window is shown in Figure 3-7.

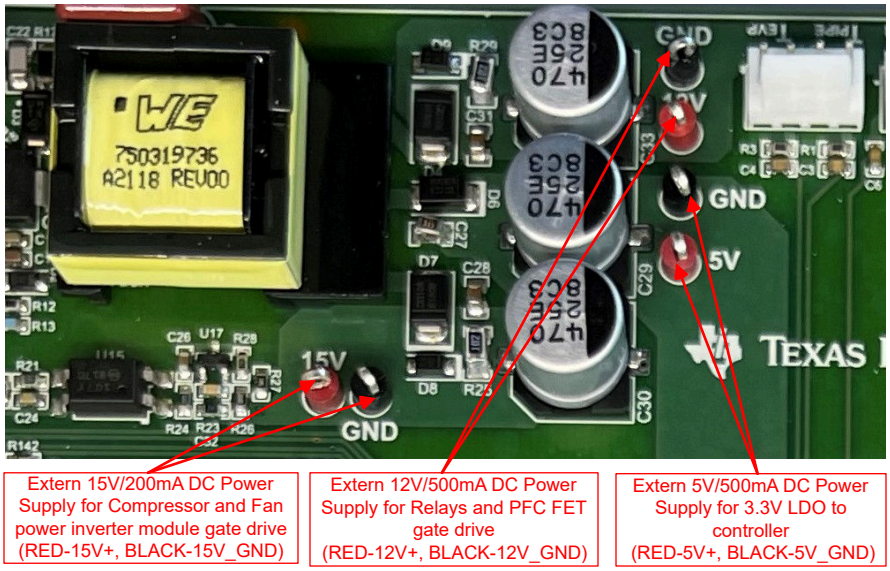


Figure 3-10. Connect the External DC Power Supply to Verify the Hardware

3.3.1.2 Build and Load Project

1. Open the sys_settings.h file, there are two incremental build options (PFC_BUILDLEVEL, DMC_BUILDLEVEL), set PFC_BUILDLEVEL to PFC_LEVEL_1, and set DMC_BUILDLEVEL to DMC_LEVEL_1.
2. If another build option was built previously, right click on the project name and click on *Clean Project*, and then click on *Build Project*. Watch the tools run in the build window. The project builds successfully.
3. In the Project Explorer make sure the correct target configuration file is set as Active as shown in Figure 3-7.
4. Turn on the bench DC power supply to provide the +5 V, +12 V and +15 V to the LDO for controller and gate driver. Click on the Debug button  or click *Run* → *Debug*. The build level 1 code should be compiled and loaded on the C2000 device. Notice the CCS Debug icon in the upper right-hand corner, indicating that the user is now in the Debug Perspective view. The program should be stopped at the start of main().

3.3.1.3 Setup Debug Environment Windows

It is standard debug practice to watch local and global variables while debugging code. 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 graph tool.

1. Click *View* → *Expressions* on the menu bar to open an Expressions watch window . Move the mouse to the Expressions window to view the variables being used in the project. Add variables to the Expressions window as shown in Figure 3-11, it uses the number format associated with variables during declaration, shows an example of the Expressions window. You can select a desired number format for the variable by right clicking on it and choosing.
2. Alternately, a group of variables can be imported into the Expressions window by right clicking within Expressions window and clicking Import, and browse to the directory of the project at `<install_location>\solutions\tidm_02010_hvac_dmpfc\common\debug` and pick *BuildLevel1.txt* and click OK to import the variables shown in Figure 3-11.

   Note that some of the variables have not been initialized at this point in the main code and might contain some useless values.
3. The structure variables *motorVars[0]*, *motorVars[1]* and *pfcVars* have references to most variables that are related to controlling dual motor with PFC. By expanding this variable, you can see them all and edit as needed.
4. Click on the Continuous Refresh button  in the expressions window. This enables the window to run with real-time mode. By clicking the down arrow in this expressions window, you can select *Customize Continuous Refresh Interval* and edit the refresh rate of the expressions window. Note that choosing too fast an interval might affect performance.

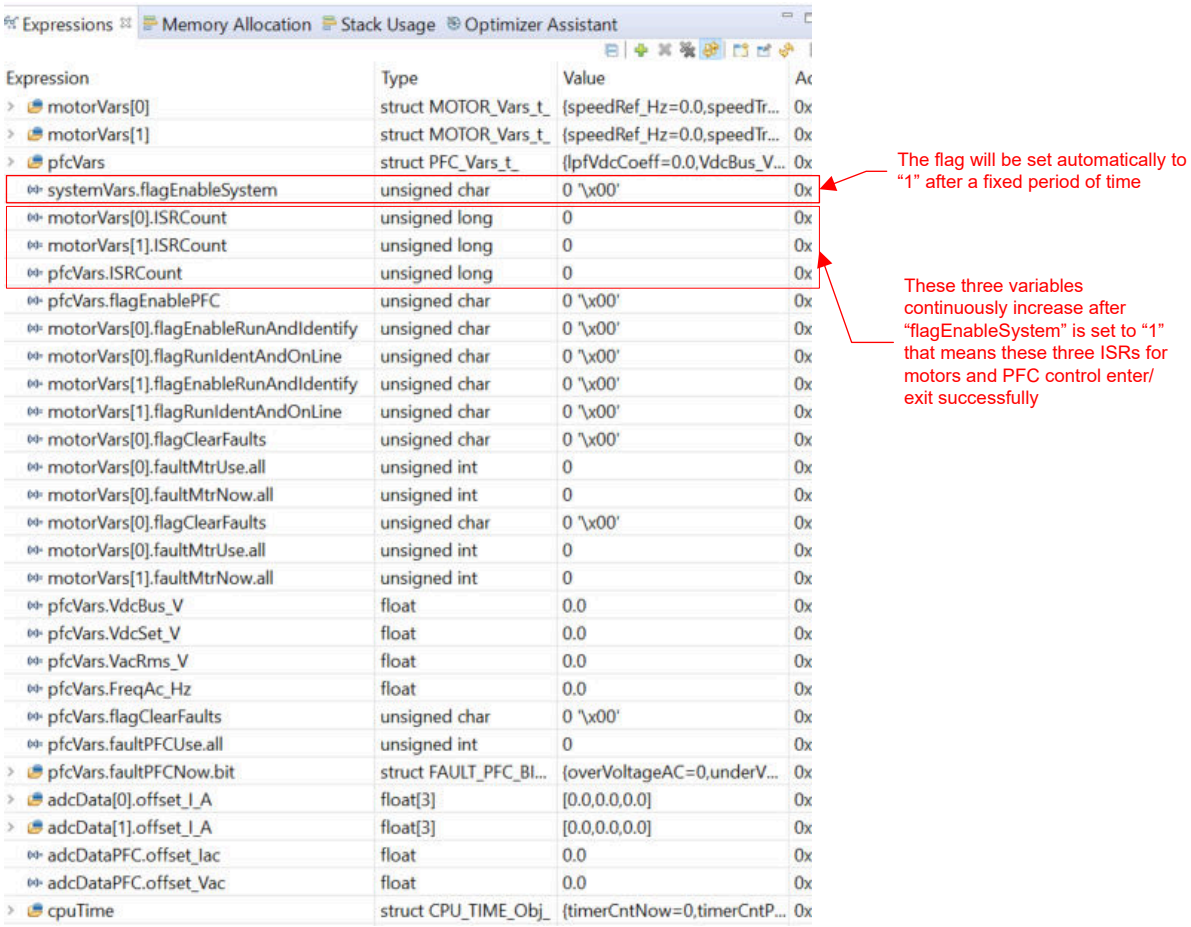


Figure 3-11. Build Level 1: Expressions Watch Window at Reset

3.3.1.4 Run the Code

1. Run the project by clicking the button , or click *Run* → *Resume* in the Debug tab.
2. In the Expressions window, set the variables *pfcVars.flagEnablePFC*, *motorVars[0].flagEnableRunAndIdentify*, and *motorVars[1].flagEnableRunAndIdentify* to 1 after *systemVars.flagEnableSystem* was automatically set to 1 in the watch window.
3. The project should now run, and the values in the graphs and expressions window should continuously update as shown in Figure 3-12 while using this project. You might want to resize the windows according to your preference.
4. In the watch view, the variables *motorVars[0].flagRunIdentAndOnLine* and motorVars[1].flagRunIdentAndOnLine should be set to 1 automatically. The *ISRCount* should be increasing continuously.
5. Check calibration offsets of dual motor and PFC, the offset value of the motor phase current sensing should be equal to approximately half of the scale current of ADC as shown in Figure 3-12.
6. Probe the PWM output for dual motor and PFC drive control with an oscilloscope. All of the PWM duty are set to 50% in this build level, the PWM output waveforms are as shown in Figure 3-13. The PWM switching frequency of motor_1 is 6 kHz, the PWM frequency of motor_2 and PFC have an integral multiple of motor_1 with 18 kHz and 72 kHz. A fixed degree lags are between motor_1, motor_2 and PFC for avoiding the ADC modules are triggered and the ISRs occupy the CPU at the same time.
7. The controller can now be halted, and the debug connection terminated. Fully halting the controller by first clicking the Halt button on the toolbar or by clicking *Target* → *Halt*. Finally, reset the controller by clicking on or clicking *Run* → *Reset*.
8. Erase the code in controller for next build level by clicking *Tools* → On-Chip Flash, and click *Erase Flash* in On-Chip Flash tab (make sure that all of the flash banks are checked) as shown in Figure 3-14. This

operation will erase all of the program code stored in flash. (This step is option, the user can ignore this step to load the new program code in next build level)

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

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

Figure 3-12. Build Level 1: Expressions Window at Run Time

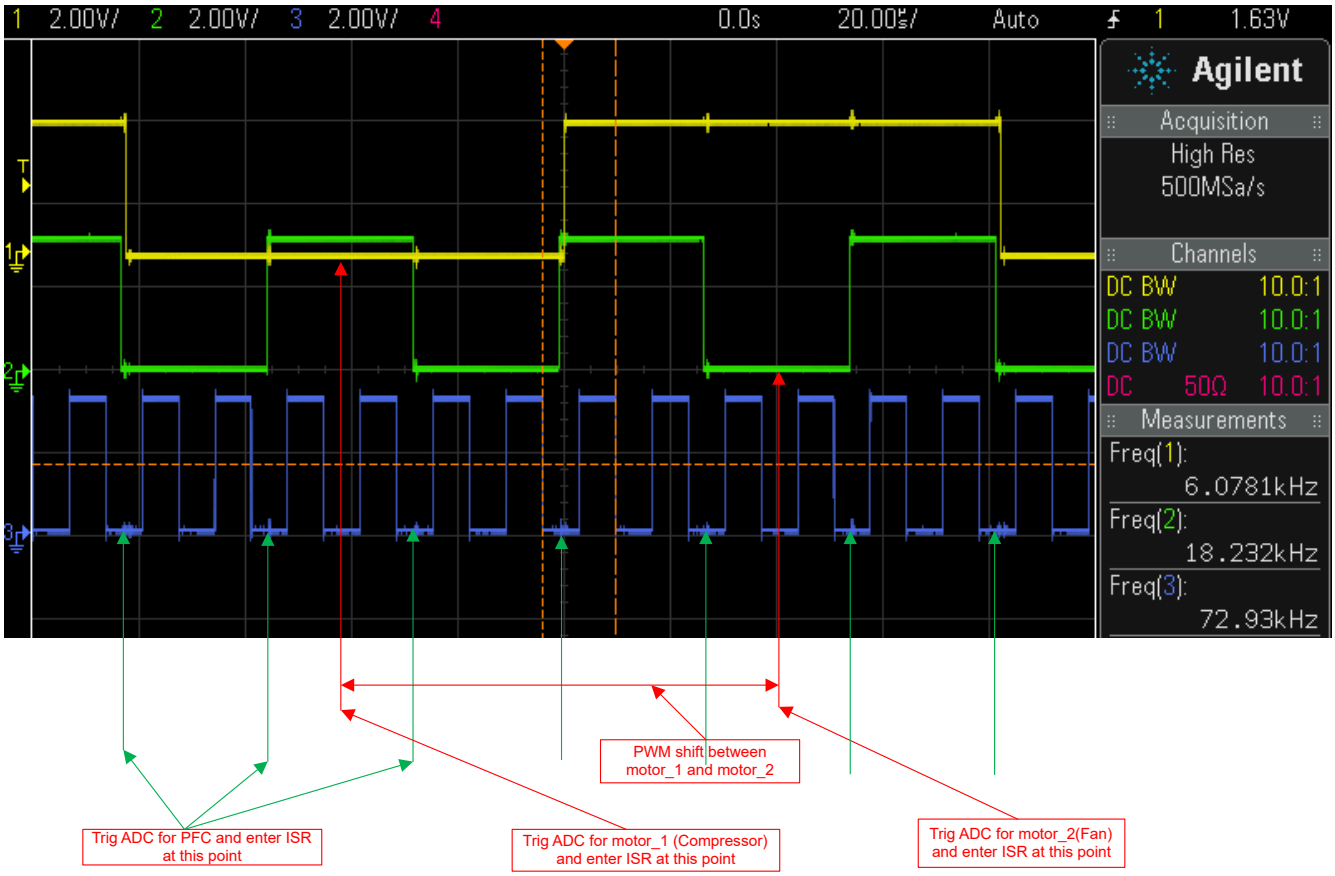


Figure 3-13. Build Level 1: Dual Motor and PFC PWM Output

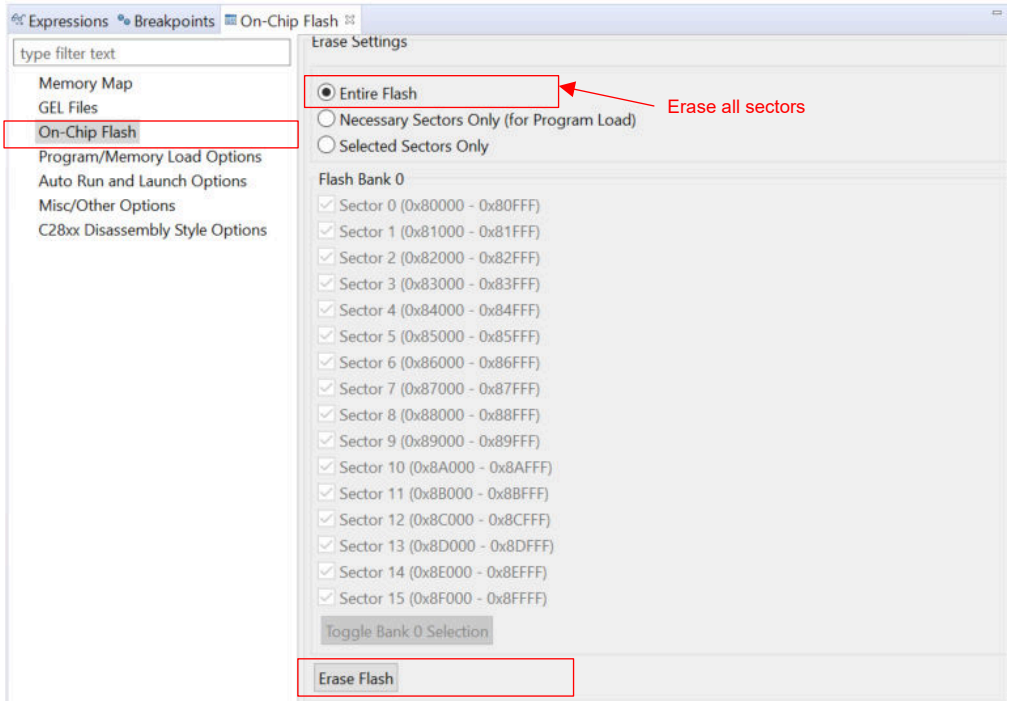


Figure 3-14. Build Level 1: Erase Program Code in Flash for Next Build Level

3.3.2 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 dual motor for validating current and voltage sensing circuit, and gate driver circuit.
- Implements open-loop control of PFC for validating the AC voltage, DC voltage, and AC current sensing circuit.
- Test InstaSPIN-FOC FAST or eSMO modules for motor control.
- Test DC voltage and current measurement module for PFC 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 MCU controller and gate drivers is provided by the auxiliary power supply module. The software flow for this build level is shown in Figure 3-15.

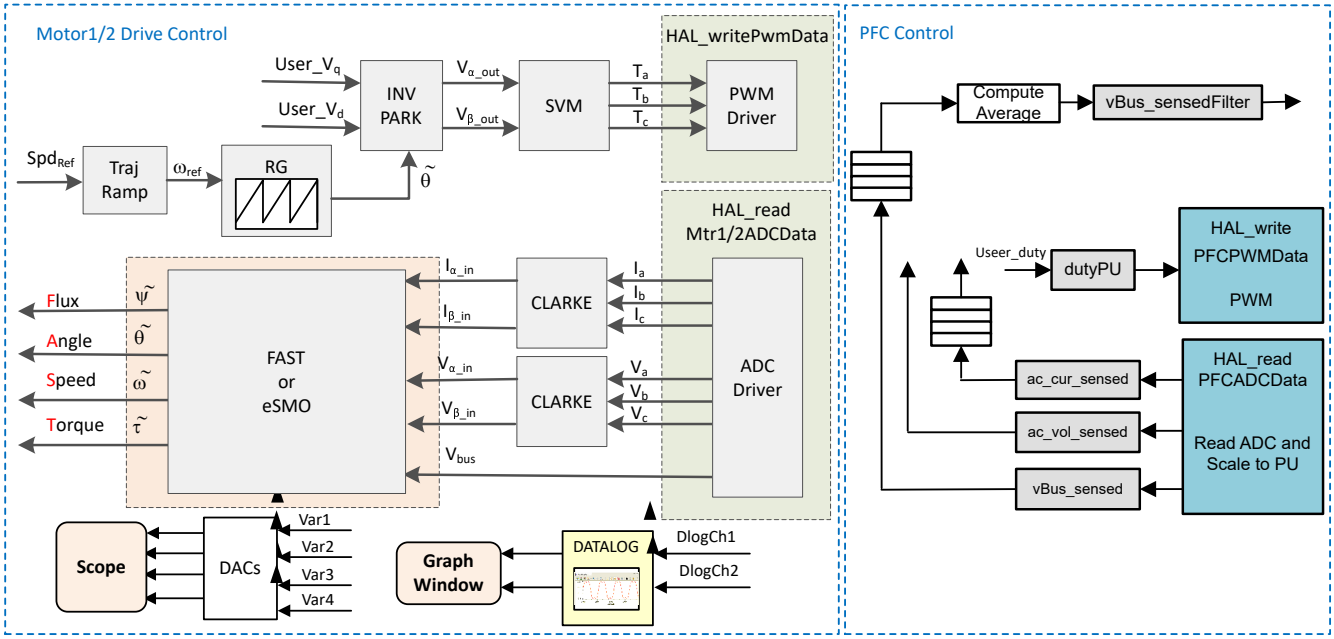


Figure 3-15. Control Software Block Diagram: Build Level 2 – Open Loop Control

3.3.2.1 Start CCS and Open Project

1. Connect a programmable, isolated AC power source capable of providing universal input up to 1.6 kW to the input terminals (connector J3 and J4) of the reference board as shown in Figure 3-2. Set the power supply current limit to 2 A. Do not turn the power supply on at this time.
2. Connect a 400-V resistive load of about 500Ω at the output (connector J11/Vdc+ and J10/Vdc-).
3. Connect a motor (compressor) to J7/J8/J9, and a motor (fan) to J12.
4. Follow Steps 2 and 3 of Section 3.3.1.1 to open the project.

3.3.2.2 Build and Load Project

1. Open the sys_settings.h file, there are two incremental build (PFC_BUILDLEVEL, DMC_BUILDLEVEL) options, set PFC_BUILDLEVEL to PFC_LEVEL_2, and set DMC_BUILDLEVEL to DMC_LEVEL_2.
2. Follow Steps 2 to 4 of Section 3.3.1.2 to build the project and load code into controller.

3.3.2.3 Setup Debug Environment Windows

1. Follow Steps 1 to 4 of Section 3.3.1.3 to import the variables into the Expressions window by picking *BuildLevel2.txt*. The Expressions window appears as shown in Figure 3-16.

3.3.2.4 Run the Code

1. Set the AC power source output to 0 V, turn on the AC power source, slowly increase the output voltage from 0-V to 110-VAC.
2. Run the project by clicking on button , or click *Run* → *Resume* in the Debug tab. The *systemVars.flagEnableSystem* should be set to *1* after a fixed time, that means the offsets calibration

have been done and the power relay for inrush is turned on. The fault flags for dual motor and PFC (*motorVars[0].faultMtrUse.all*, *motorVars[1].faultMtrUse.all*, and *pfcVars.faultPFCUse.all*) should be equal to *0* , if not, the user have to check the current and voltage sensing circuit as described in Section 3.3.1.

3. To verify current and voltage sensing circuit of the inverter for motor_1, set the variable *motorVars[0].flagEnableRunAndIdentify* to *1* in the Expressions window as shown in Figure 3-16. The motor_1 should 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 doesn't spin smoothly.

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

4. This should now spin the motor_1 with a setting speed in the variable *motorVars[0].speedRef_Hz,* check the value of *motorVars[0].speed_Hz* in Expressions window, the values of *motorVars[0].speedRef_Hz* and *motorVars[0].speed_Hz* should be very close as shown in Figure 3-16.
5. Connect oscilloscope voltage and current probes to the output of the PWMDAC or DAC128S board, the motor phase current to probe the angle, current signals, the current and angle waveforms on the oscilloscope as shown in Figure 3-17. Notice that the angle of the force angle generator is very similar as the estimated rotor angle of the FAST or eSMO estimator, a little bit shift error could be between these two angles. The sampling current waveform by using a DAC to output on oscilloscope should be the same as the phase current waveform capture by a current probe, that means the current sensing circuit is good for motor control.
6. Verify the over current fault protection by decreasing the value of the variable *motorVars[0].overCurrent_A*, the overcurrent protection is implemented by the CMPSS modules. The overcurrent fault will be trigger if the *motorVars[0].overCurrent_A* is set to a value less than the actual current, the PWM output will be disabled, the *motorVars[0].flagEnableRunAndIdentify* is cleared to *0,* and the *motorVars[0].faultMtrUse.all* will be set to *0x10.*
7. Follow the steps 3, 4 and 5 using the same approach to test the hardware for motor_2 by setting the variable *motorVars[1].flagEnableRunAndIdentify* to *1* and tune the v/f parameters in *user_mtr1.h* to spin the motor smoothly.
8. To verify the current and dc_link voltage sensing circuit for PFC, connect the probes to the output of the PWMDAC or DAC128S to the sampling current and voltage, use a high voltage probe and current probe to detect the dc_link voltage and current, the voltage and current waveform are shown in Figure 3-18.
9. Check the variables *pfcVars.VdcBus_V* in Expressions window, the values of these variables should be the same as the setting value of the AC source or measured by a multimeter. Increase the pfcVars.dutyOut from 0.0 to 0.2 very slowly, the dc_link voltage should increase simultaneously.
10. Verify the overcurrent protection by decreasing the *pfcVars.overCurrent_A*, the PWM output of PFC will be disabled if the overcurrent fault is triggered.
11. The controller can now be halted, and the debug connection terminated. Fully halting the controller by first clicking the Halt button on the toolbar  or by clicking *Target* → *Halt*. Finally, reset the controller by clicking on  or clicking *Run* → *Reset*.
12. Close CCS debug session by clicking on Terminate Debug Session  or clicking *Run* → *Terminate*.

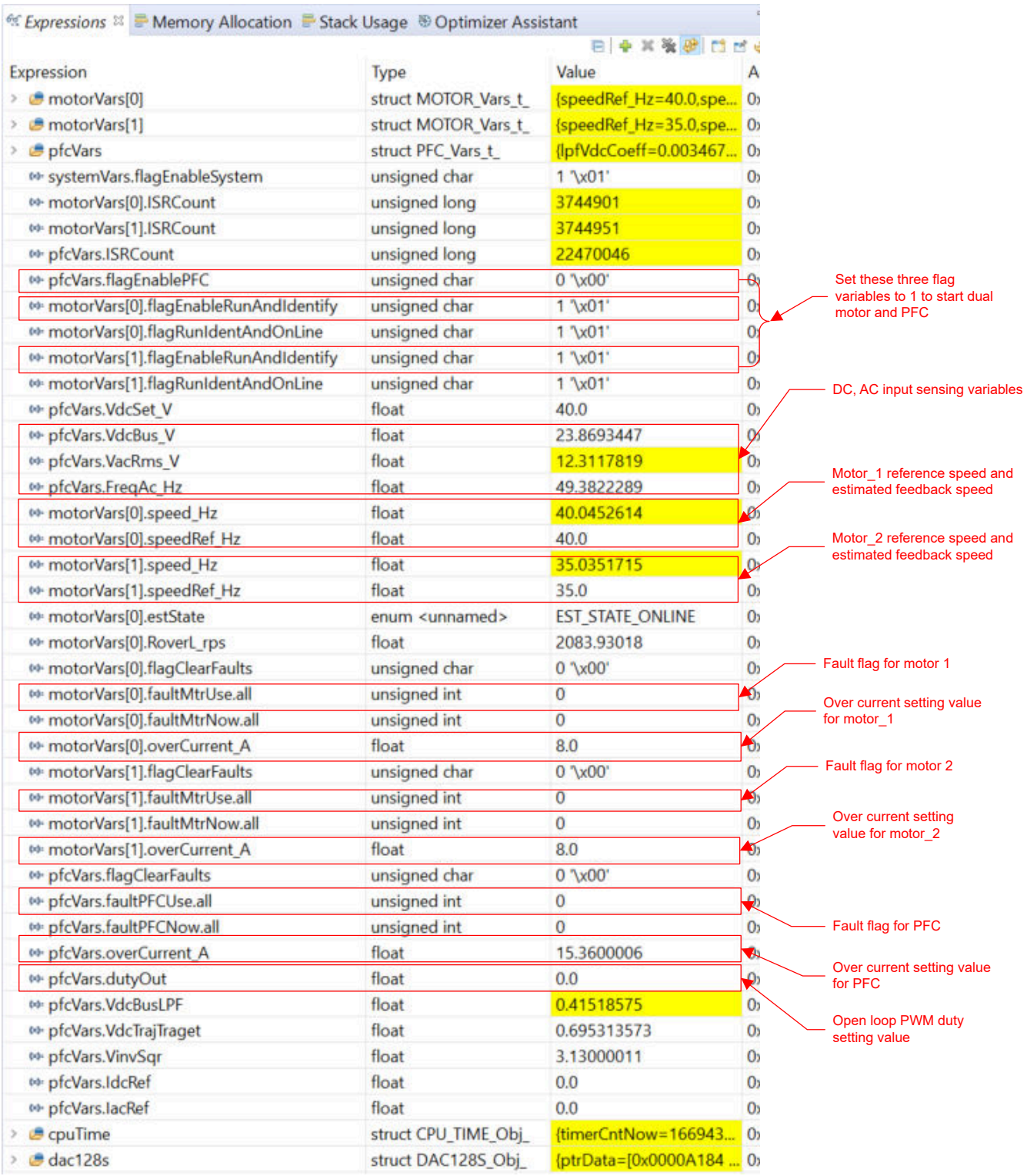


Figure 3-16. Build Level 2: Expressions Window at Run Time

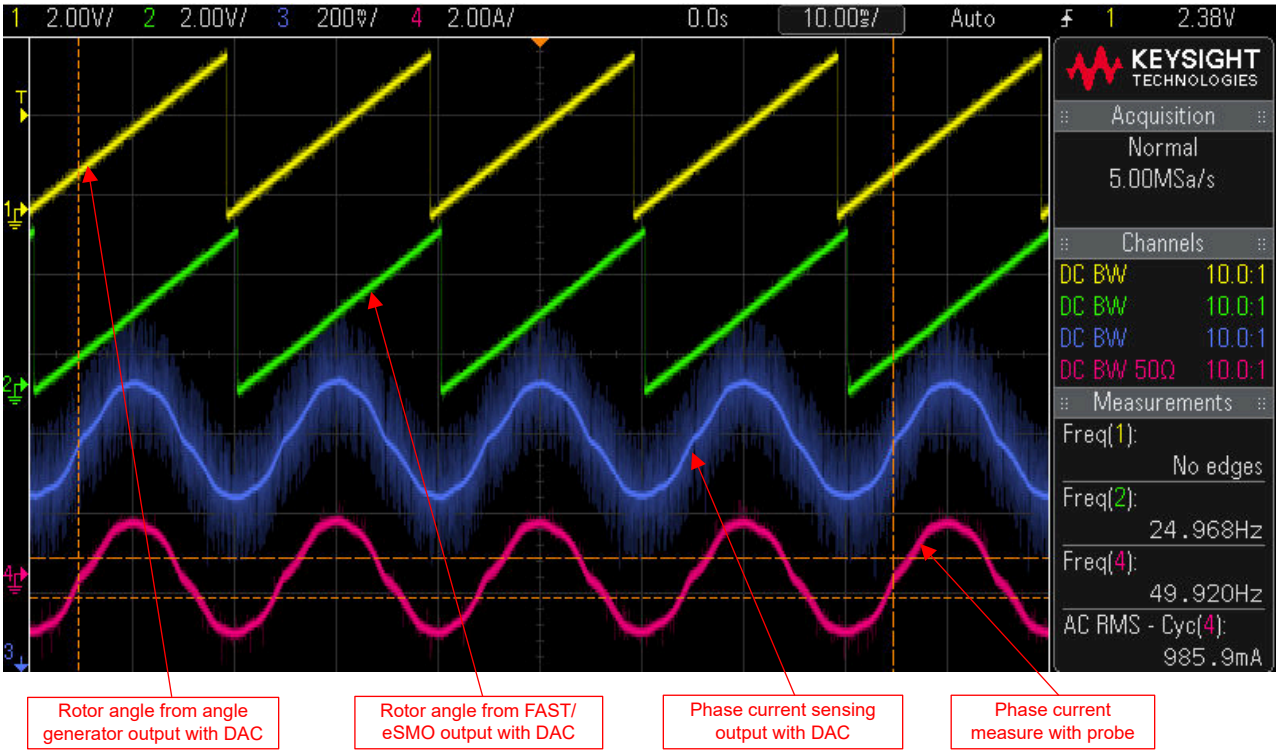


Figure 3-17. Build Level 2: Rotor Angle, Phase Current of Motor

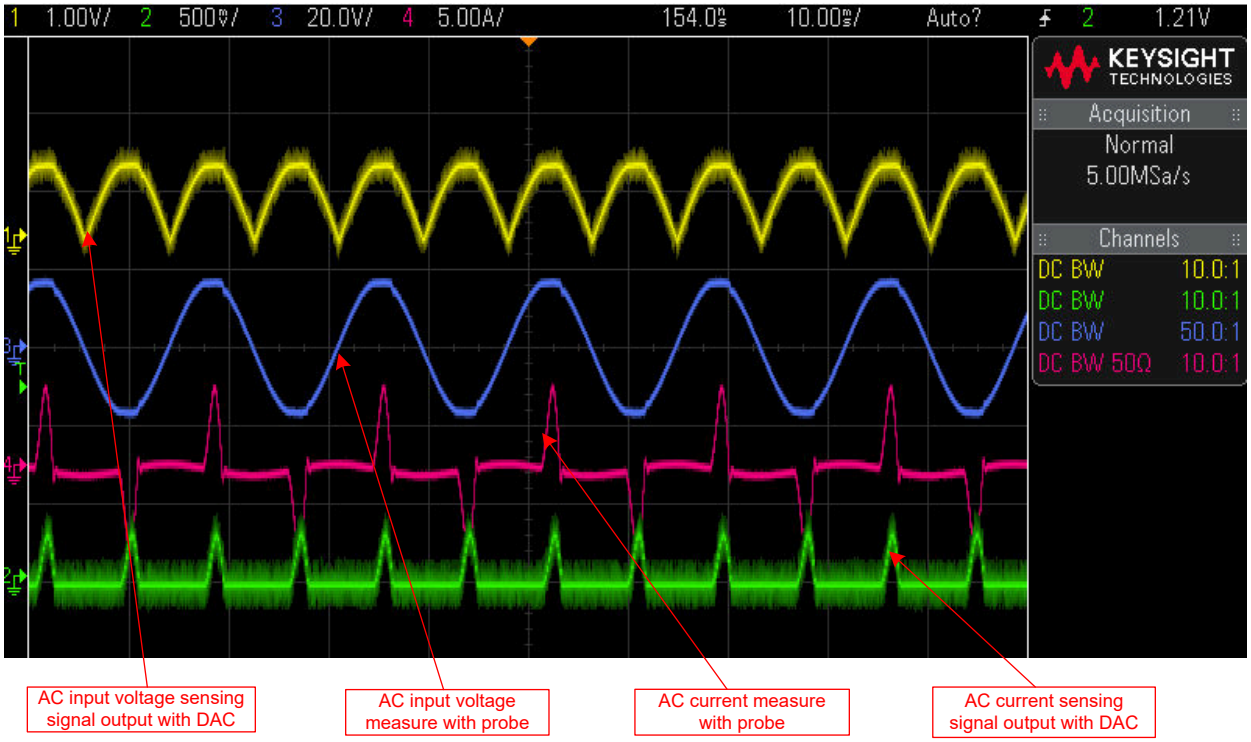


Figure 3-18. Build Level 2: DC Voltage, Current, PWM Output of PFC

3.3.3 Build Level 3: Closed Current Loop Check

Objectives learned in this build level:

- Evaluate the closed current loop operation of the PFC.
- Evaluate the closed current loop operation of dual motor.

In this build level, the inner current loop of PFC is closed that is the inductor current is controlled using a current compensator, the motor is controlled using i/f control that the rotor angle is generated from ramp generator module. The software flow for this build level is shown in Figure 3-19.

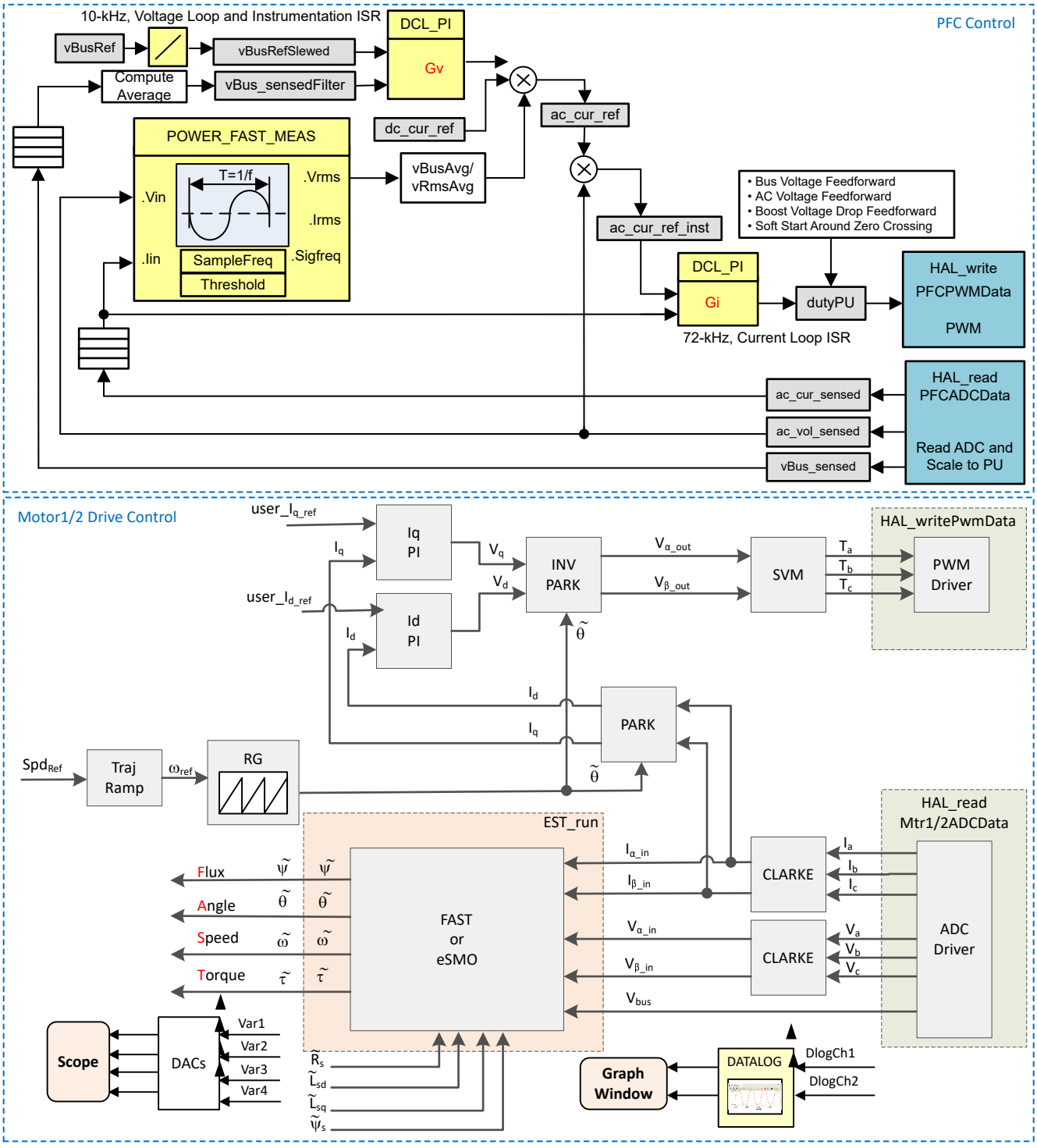


Figure 3-19. Control Software Block Diagram: Build Level 3 – Current Close Loop Control

3.3.3.1 Start CCS and Open Project

1. Connect a programmable, an isolated AC power supply capable of providing universal AC input up to 1.6 kW to the input terminals (connector J3 and J4) of the reference board as shown in Figure 3-2. Set the power supply current limit to 3 A and the output frequency to 50/60 Hz. Do not turn the power supply on at this time.

2. Connect a 400-V resistive load of about 500Ω at the output (connector J11/Vdc+ and J10/Vdc-).
3. Connect a motor (compressor) to J7/J8/J9, and a motor (fan) to J12.
4. Follow Steps 2 and 3 of Section 3.3.1.1 to open the project.

3.3.3.2 Build and Load Project

1. Open the sys_settings.h file, there are two incremental build (PFC_BUILDLEVEL, DMC_BUILDLEVEL) options, set PFC_BUILDLEVEL to PFC_LEVEL_3, and set DMC_BUILDLEVEL to DMC_LEVEL_3.
2. Follow Steps 2 to 4 of Section 3.3.1.2 to build the project and load code into controller.

3.3.3.3 Setup Debug Environment Windows

1. Follow Steps 1 to 4 of Section 3.3.1.3 to import the variables into the Expressions window by picking *BuildLevel3.txt*. The Expressions window appears as shown in Figure 3-16.

3.3.3.4 Run the Code

1. Set the AC source output to 0 V at 50/60Hz, turn on the AC power supply, slowly increase the input voltage from 0-V to 110-V AC.
2. Run the project by clicking on button , or click *Run* → *Resume* in the Debug tab. The *systemVars.flagEnableSystem* should be set to '*1*' after a fixed time, that means the offsets calibration have been done and the power relay for inrush is turned on. The fault flags for dual motor and PFC (*motorVars[0].faultMtrUse.all*, *motorVars[1].faultMtrUse.all*, and *pfcVars.faultPFCUse.all*) should be equal to '*0*' , if not, the user have to check the current and voltage sensing circuit as described in Section 3.3.1.
3. To verify current closed-loop control for motor_1, set the variable *motorVars[0].flagEnableRunAndIdentify* to '*1'* in the Expressions window as shown in Figure 3-20. The motor_1 should run with a closed-loop control using the angle from the angle generator at a setting speed in the variable *motorVars[0].speedRef_Hz,* check the value of *motorVars[0].speed_Hz* in Expressions window, both variables value should be very close.
4. Connect oscilloscope probes to the DAC output and motor phase line to probe the angle, current signals, the current and angle waveforms on the oscilloscope appear as shown in Figure 3-21. Change the *Idq_set_A[0].value[1]* in the Expressions window, the motor phase current should be increasing accordingly.
5. If the motor can not run with current-closed loop and appear a overcurrent fault, check if the sign of *adcData[0].current_sf* and the value of *userParams[0].current_sf* are set correctly according to the hardware board.
6. Follow the steps 3, 4 and 5 using the same approach to test the hardware for motor_2 by setting the variable *motorVars[1].flagEnableRunAndIdentify* to '*1'* and tune *Idq_set_A[1].value[1]* to spin the motor.
7. To verify the current closed-loop for PFC by setting the *pfcVars.flagEnablePFC* to "1", connect the probes to the output of the DAC to the sampling current and voltage, use a high voltage probe and current probe to detect the AC input voltage and current, the voltage and current waveform are shown in Figure 3-22.
8. Check the variables *pfcVars.VdcBus_V*, *pfcVars.VacRms_V*, and *pfcVars.FreqAc_Hz* in Expressions window, the values of these variables should be the same as the setting value of the AC source or measured by a multimeter.
9. Increase *pfcVars.IdcRef* from 0.0 to 0.05 very slowly, the output voltage should increase accordingly. Keep increasing *pfcVars.IdcRef* in increments of 0.01 until *pfcVars.IdcRef* is increased to a value of 0.1. The sensing current value *pfcVars.IacSen* can change rapidly and is not the same as the *pfcVars.IdcRef* value. This change is because *pfcVars.IacSen* is the instantaneous input current value while *pfcVars.IdcRef* is the amplitude reference for the current command.
10. The controller can now be halted before setting the *motorVars[0].flagEnableRunAndIdentify*, *motorVars[1].flagEnableRunAndIdentify* and *pfcVars.flagEnablePFC* to "0", and the debug connection terminated. Fully halting the controller by first clicking the Halt button on the toolbar or by clicking *Target* → *Halt*. Finally, reset the controller by clicking on or clicking *Run* → *Reset*.
11. Close CCS debug session by clicking on Terminate Debug Session or clicking *Run* → *Terminate*.

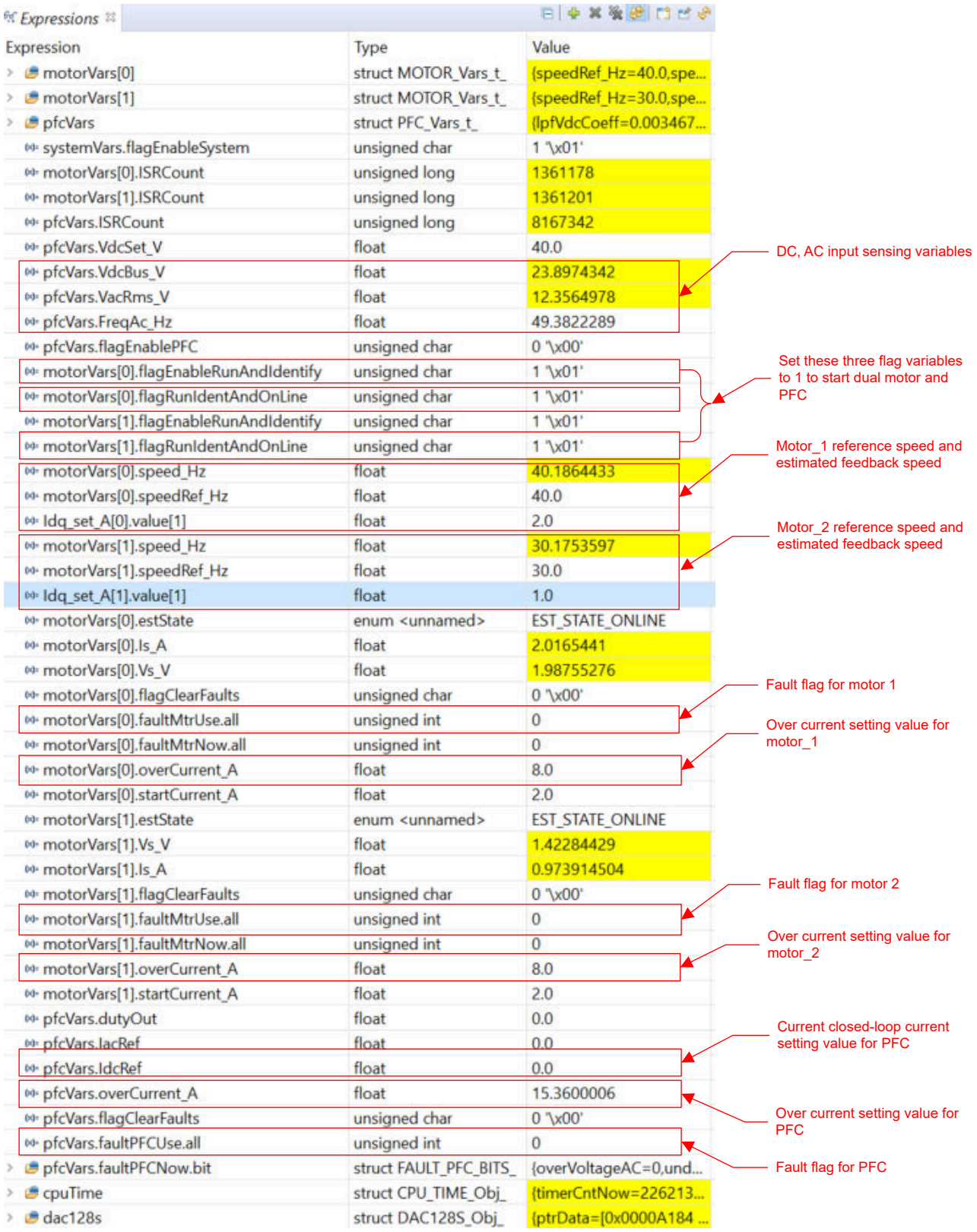


Figure 3-20. Build Level 3: Expressions Window at Run Time

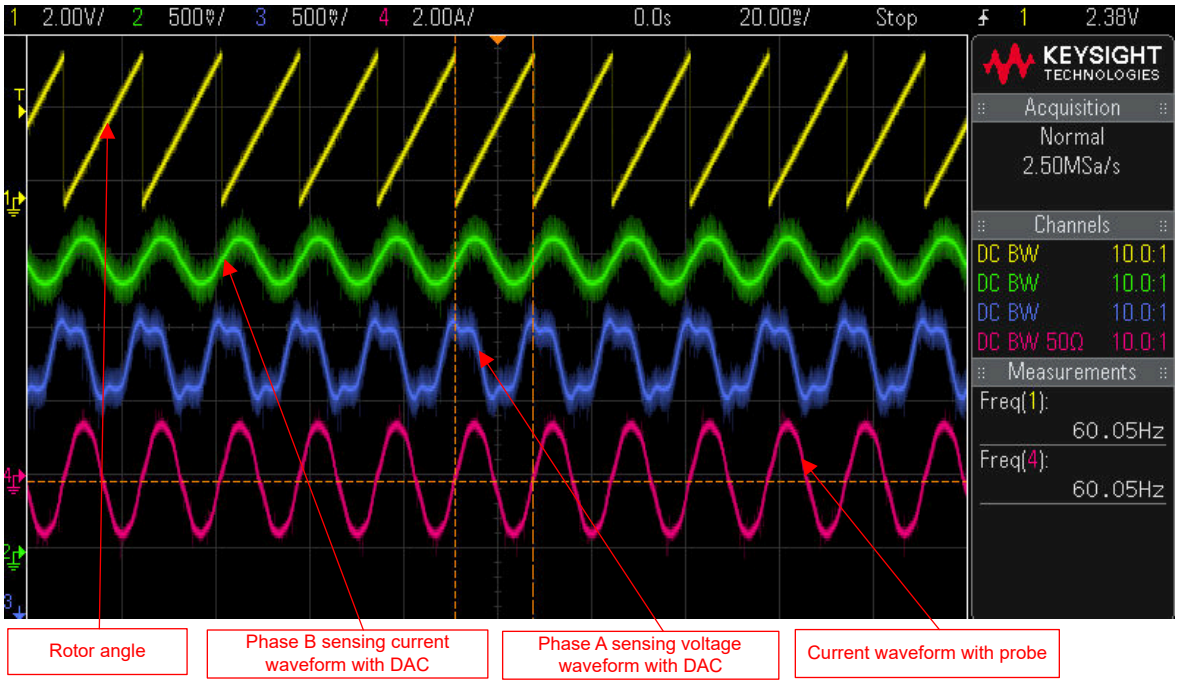


Figure 3-21. Build Level 3: Rotor Angle, Phase Current of Motor

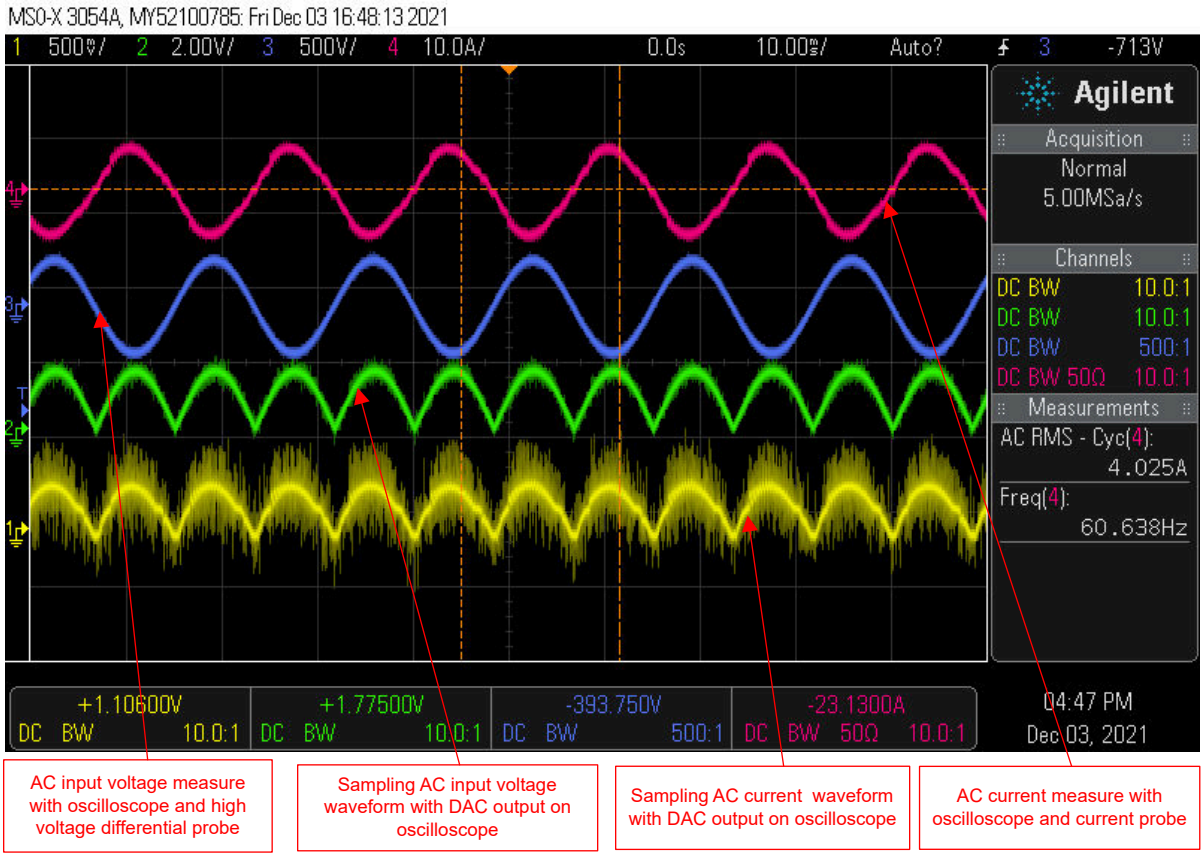


Figure 3-22. Build Level 3: AC Voltage, Current of PFC

3.3.4 Build Level 4: Full PFC and Motor Drive Control

Objectives learned in this build level:

- Evaluate the complete PFC control

- Evaluate the complete motor drive for compressor and fan
- Evaluate the additional features, field weakening control, torque compensation for compressor, the flying start for fan.
- Evaluate the completed system

In this build level, the outer voltage loop is closed with the inner current loop for PFC. The outer speed loop is closed with the inner current loop for motor 1 and motor 2 that the rotor angle is from FAST or eSMO estimator module. The software flow for this build level is shown in Figure 3-23.

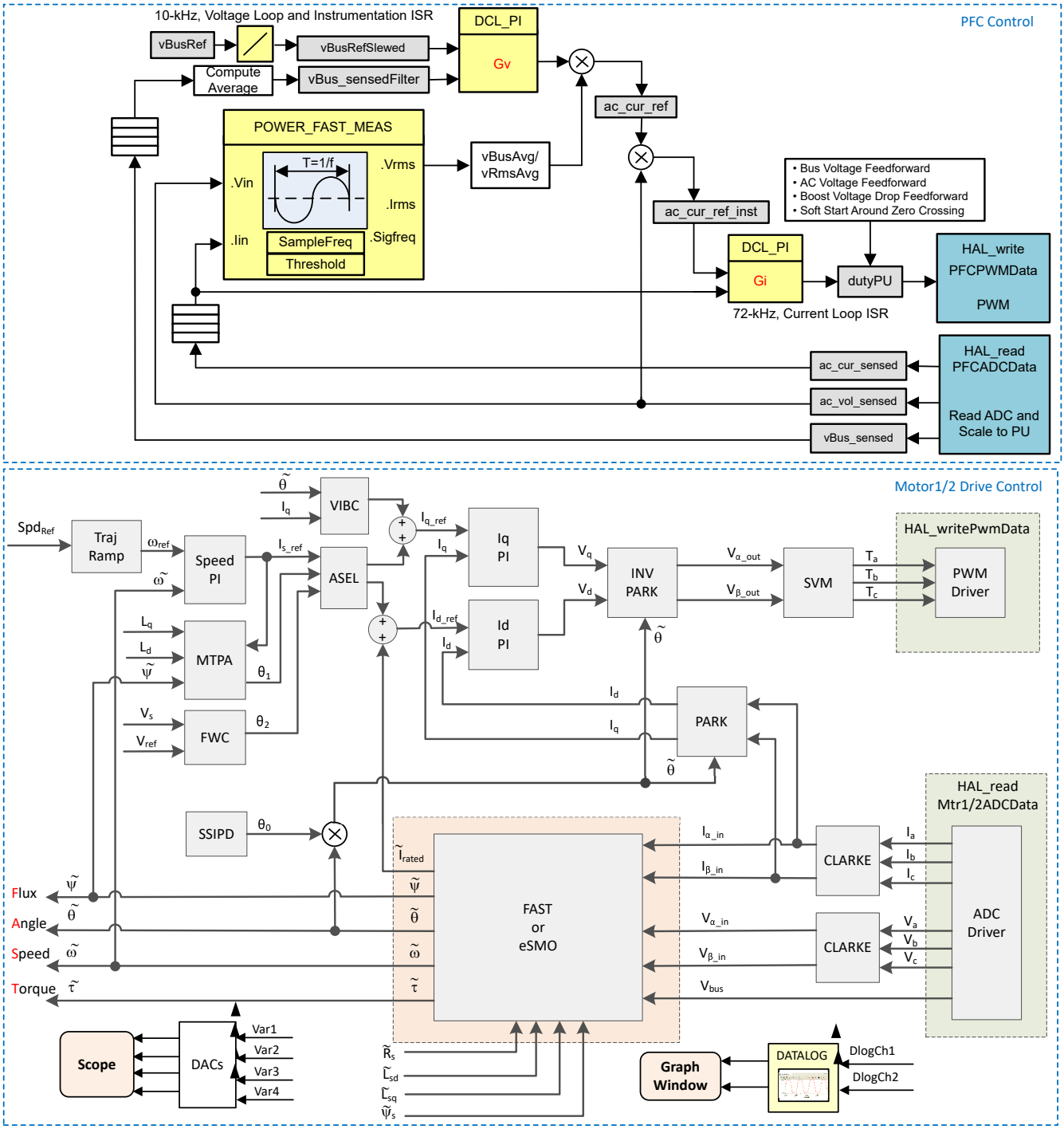


Figure 3-23. Control Software Block Diagram: Build Level 4 – Speed and Current Close Loop Control

3.3.4.1 Start CCS and Open Project

1. Connect a programmable, isolated AC source capable of providing universal AC input up to 2.0 kW to the input terminals (connector J3 and J4) of the reference board as shown in Figure 3-2. Set the AC source current limit to 10 A. Do not turn the power supply on at this time.
2. Connect a motor (compressor) to J7/J8/J9, and a motor (fan) to J12.
3. Follow Steps 2 and 3 of Section 3.3.1.1 to open the project.

3.3.4.2 Build and Load Project

1. Open the sys_settings.h file, there are two incremental build (PFC_BUILDLEVEL, DMC_BUILDLEVEL) options, set PFC_BUILDLEVEL to PFC_LEVEL_4, and set DMC_BUILDLEVEL to DMC_LEVEL_4.
2. Follow Steps 2 to 4 of Section 3.3.1.2 to build the project and load code into controller.

3.3.4.3 Setup Debug Environment Windows

1. Follow Steps 1 to 4 of Section 3.3.1.3 to import the variables into the Expressions window by picking *BuildLevel4.txt*. The Expressions window appears as shown in Figure 3-16.

3.3.4.4 Run the Code

1. Set the AC source output to 0 V at 50/60Hz, turn on the AC power supply, slowly increase the input voltage from 0-V to 220-V AC.
2. The required motor parameters must be recorded in the header files (*user_mtr1.h* and *user_mtr2.h*) as shown in the following example codes. If the motor parameters are not well know by the user, the motor identification can be used to achieve the motor parameters if the FAST estimator is implemented in the reference design.

```
#define USER_MOTOR1_Rs_Ohm                  (2.66273594f)
#define USER_MOTOR1_Ls_d_H                  (0.00943629723f)
#define USER_MOTOR1_Ls_q_H                  (0.00943629723f)
#define USER_MOTOR1_RATED_FLUX_VpHz         (0.390171647f)
```

Note

The motor identification is not supported by F28002x based reference design project, so the user must get the motor parameters from the motor manufacturer and set the parameters in user_mtr1/2.h files. The F28003x based reference design can support the motor parameters identification feature.

3. Changes the *userParams[MTR_1].flag_bypassMotorId* value to "false" to enable the motor identification as the following example code for motor 1 (compressor).

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

4. Set the right identification variables value in the *user_mtr1.h* according to the specification of the motor 1 (compressor).

```
 #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           (4.5f)      // A - 30~150% of rated current of the
motor
#define USER_MOTOR1_FLUX_EXC_FREQ_Hz        (20.0f)     // Hz - 10~30% of rated frequency of the
motor
```

5. Rebuild the project and load the code into the controller, run the project by clicking on button , or click *Run → Resume* in the Debug tab. The *systemVars.flagEnableSystem* should be set to *1* after a fixed time, that means the offsets calibration have been done and the power relay for inrush is turned on. The fault flags for dual motor and PFC (*motorVars[0].faultMtrUse.all*, *motorVars[1].faultMtrUse.all*, and *pfcVars.faultPFCUse.all*) should be equal to '*0*' , if not, the user should check the current and voltage sensing circuit as described in Section 3.3.1.
6. Set the variable *motorVars[0].flagEnableRunAndIdentify* to *1* in the Expressions window as shown in Figure 3-24, the motor identification will be executed, the whole process will take about 150s. Once

motorVars[0].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[0].Rs_Ohm's value
- USER_MOTOR1_Ls_d = motorVars[0].Ls_d_H's value
- USER_MOTOR1_Ls_q = motorVars[0].Ls_q_H's value
- USER_MOTOR_RATED_FLUX = motorVars[0].flux_VpHz's value

7. Implement the same process for motor 2 (fan) if the motor parameters are not known and need to be identified.
8. Set both *userParams[MTR_1].flag_bypassMotorId* and *userParams[MTR_2].flag_bypassMotorId* values to "true" after successfully identify the motors parameters, rebuild the project and load the code into the controller.
   - Set the variables *motorVars[0].flagEnableRunAndIdentify* equal to *1* again for starting to run the motor_1 (compressor), and *motorVars[1].flagEnableRunAndIdentify* equal to *1* again for starting to run the motor_2 (fan).
   - Set the variables *motorVars[0].speedRef_Hz*, *motorVars[1].speedRef_Hz* to a different value and watch how the motor shaft speed will follow.
   - To change the acceleration, enter a different acceleration value for the variable *motorVars[0].accelerationMax_Hzps*, *motorVars[0].accelerationMax_Hzps*.
9. To enable PFC control, set the *pfcVars.flagEnablePFC* equal to *1*, and set the variable *pfcVars.VdcSet_V* to a different value and watch how the dc bus voltage will follow the setting voltage.
10. The controller can now be halted before setting the *motorVars[0].flagEnableRunAndIdentify*, *motorVars[1].flagEnableRunAndIdentify* and *pfcVars.flagEnablePFC* to *0*, and the debug connection terminated. Fully halting the controller by first clicking the Halt button  on the toolbar or by clicking *Target* → *Halt*. Finally, reset the controller by clicking on  or clicking *Run* → *Reset*.
11. Close CCS debug session by clicking on Terminate Debug Session  or clicking *Run* → *Terminate*.

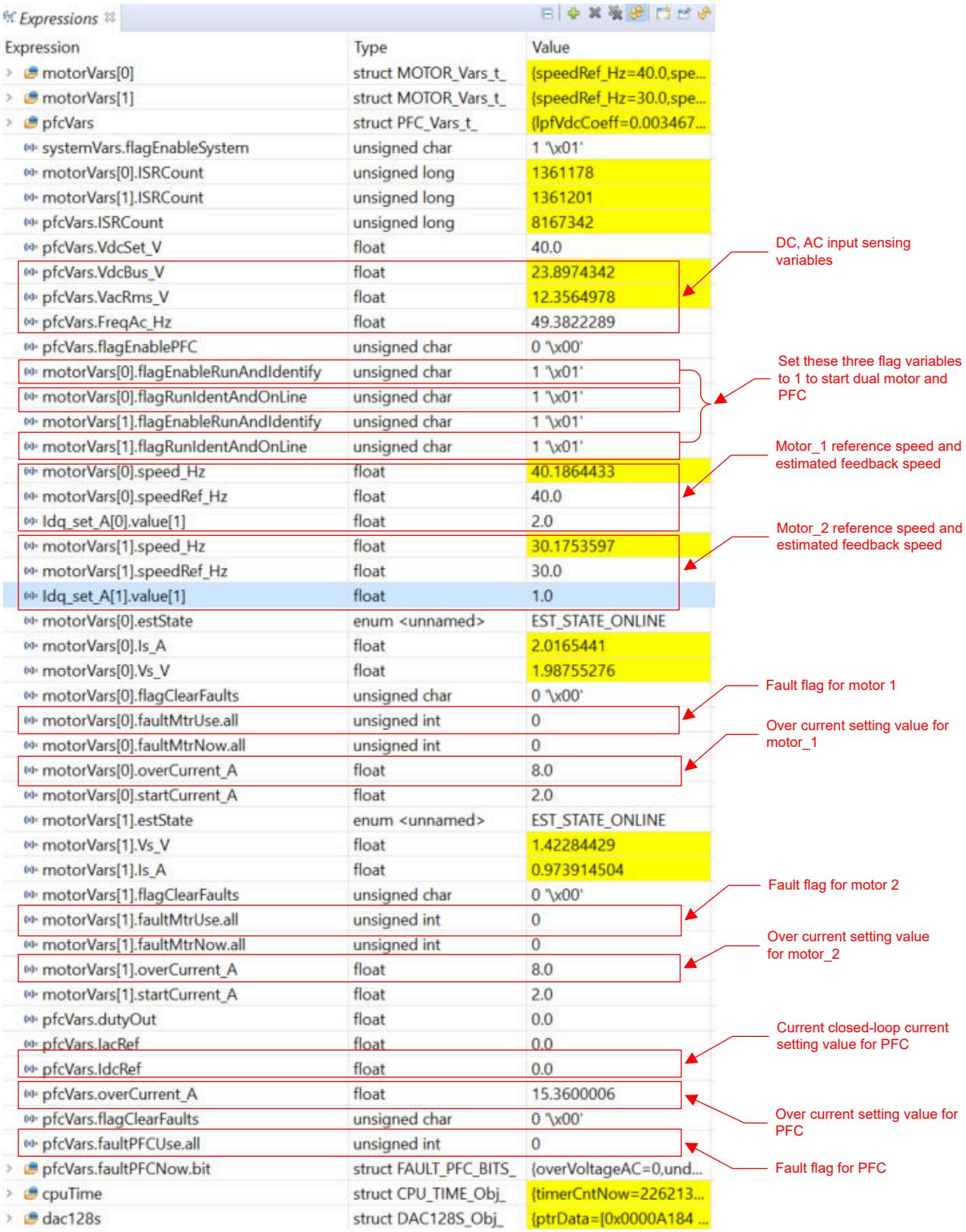


Figure 3-24. Build Level 4: Expressions Window at Run Time

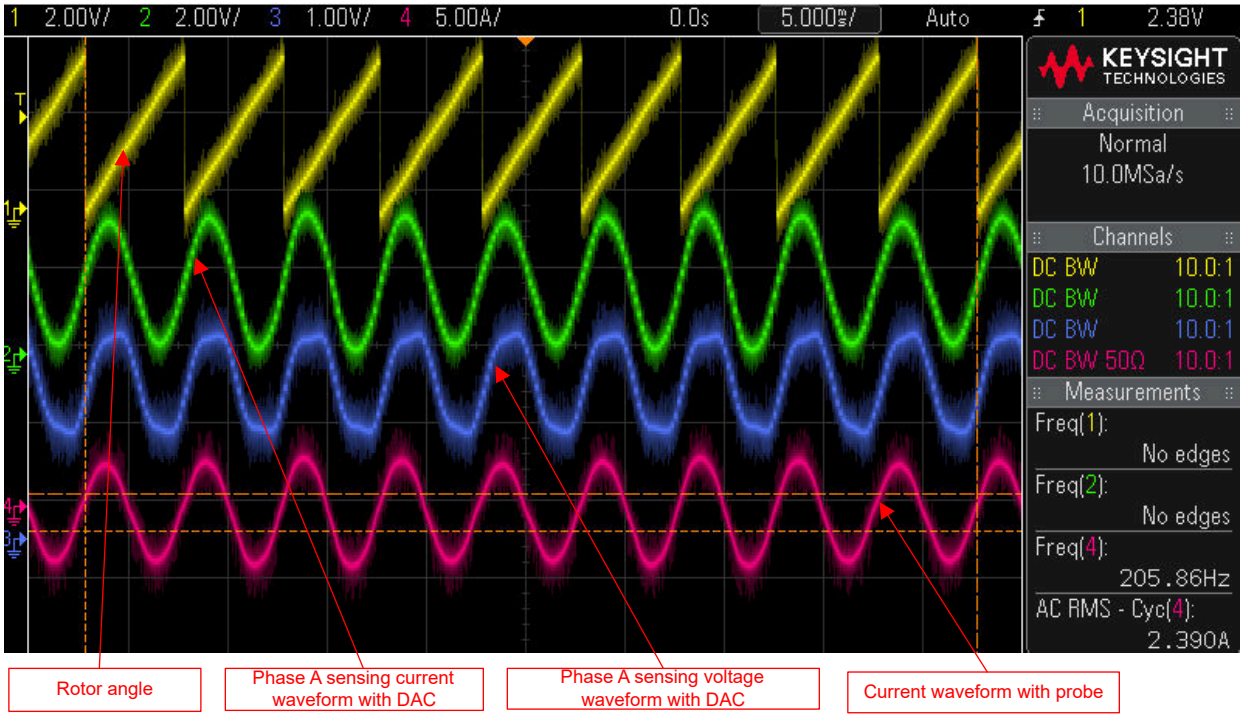


Figure 3-25. Build Level 4: Rotor Angle, Phase Current of Motor

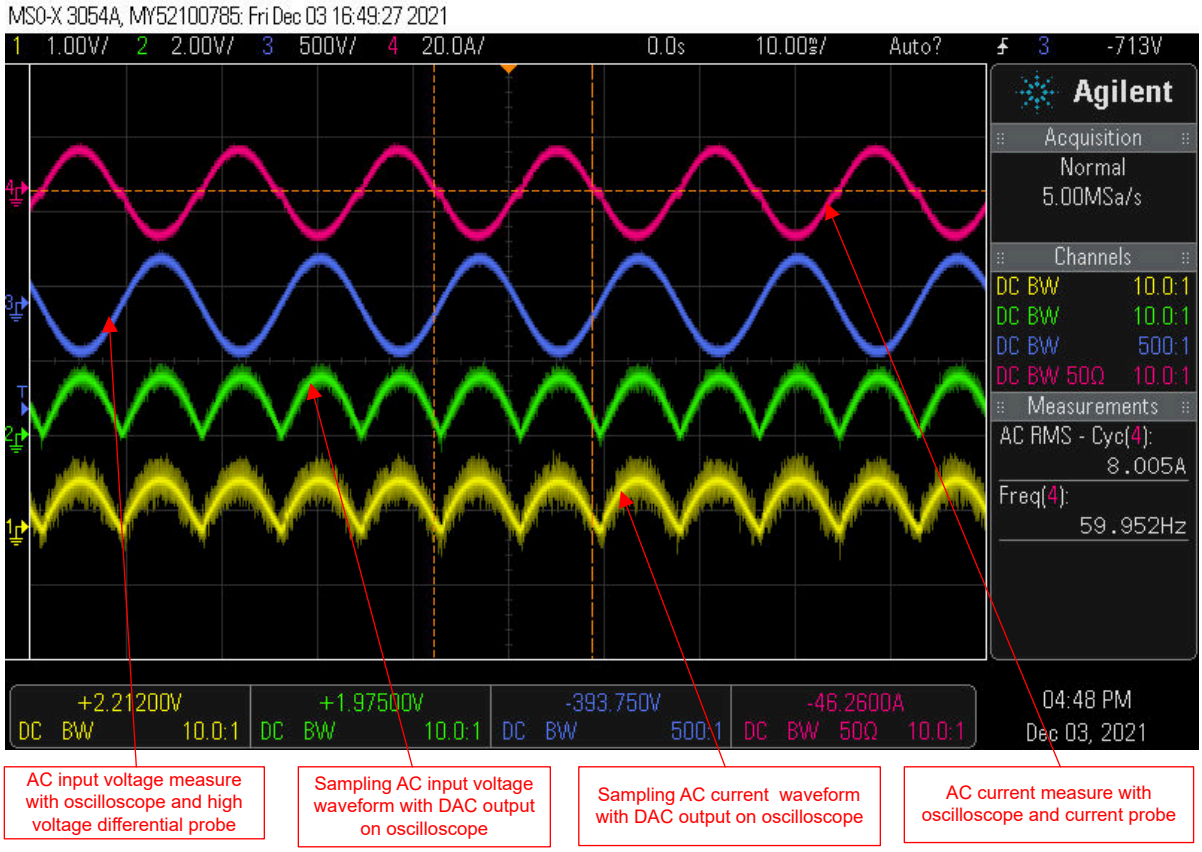


Figure 3-26. Build Level 4: Voltage and Current of PFC

3.3.4.5 Run the System

1. Set the AC source output to 220 V at 50/60Hz, turn on the AC power supply.

2. Right click on the project name, and click *Rebuild Project*. The project builds successfully. Click *Run* → *Debug*, which launches a debugging session.
3. Run the project by clicking on button  , or click *Run* → *Resume* in the Debug tab.
4. Set the variable *systemVars.flagEnableSynCtrl* equal to *true* to enable control the system with changing the PFC stetting voltage and the motors setting speed.
5. Set the variable systemVars.flagEnableRun equal to "true". The system will start the PFC, motor_2(fan), and motor_1(compressor) in sequence.
   - Set the variables *systemVars.pfcVdcbusTarget_V*, *systemVars.speedFan_Hz*, and *systemVars.speedComp_Hz* to a different value and watch how the motors speed and dc bus voltage will follow.
   - To change the acceleration, enter a different acceleration value for the variables *systemVars.accelerationFan_Hzps* and*systemVars.accelerationComp_Hzps*.
6. Set the variable systemVars.flagEnableRun equal to *false*. The system will stop the PFC, motor_2 (fan), and motor_1 (compressor) in sequence.

3.3.4.6 Tuning Motor Drive FOC Parameters

The sliding mode current observer consists of a model based current observer and a bang-bang control generator driven by error between estimated motor currents and actual motor currents. The F and G parameters are calculated based on the motor parameters Rs, and Ls as described in Section 2.4.2.2.1.2. The observer gain k for bang-bang control, the cutoff frequency for LPF, and the K_p and K_i for PLL angle tracker must be tuned according to the testing state, and try to get the optimal parameters. The user can run the FAST estimator and eSMO in parallel to validate the angle from the eSMO for tuning its parameters. The initial parameters are defined in the user-mtr1/2.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 regulators gains are calculated according to the motor parameters, the user can tune these gains online to optimize the control performance of the system.

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

3.3.4.7 Tuning PFC Parameters

The TI SFRA library is designed to enable frequency response analysis on digitally-controlled power converters using software alone. This feature enables performing frequency response analysis of the power converter with relative ease as no external connections or equipment is required. The optimized library can be used in high-frequency power conversion applications to identify the plant and the open loop characteristics of a closed-loop power converter, which can be used to get stability information such as bandwidth, gain margin, and phase margin to evaluate the control loop performance. For more information, see the C2000™ SFRA Library and Compensation Designer User's Guide.

In addition to SFRA, this PFC design supports the use of another powerSUITE tool called the Compensation Designer. The Compensation Designer tool allows the PFC design of different styles of compensator to achieve the desired closed loop performance, which can be done using the measured power stage or plant data from the SFRA Tool. The coefficients that must be programmed on the device are generated by the Compensation

Designer and can be copied into the code directly. These tools help users evaluate the complete system, adapt it for their end application, and tune it for improved performance.

3.3.4.8 Tuning Field Weakening and MTPA Control Parameters

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

1. Adding the pre-define symbols *MOTOR1_FWC* and *MOTOR1_MTPA* in build configuration of the project as described in Section 3.2.2 for enabling the FW and MTPA respectively.
2. In 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 according to the specification of a motor.
3. Adding the variables VsRef_pu, Kp_fwc, and Ki_fwc to Expressions window in CCS Debug Perspective, and tuning these parameters to achieve the expectation performance for the field weakening control according to the motor and its system.
4. After tuned and fixed these variables value, 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 is no any additional parameters need to be tuned online.

3.3.4.9 Tuning Flying Start Control Parameters

The flying start is enabled for fan drive by default. Adding and Tuning the variables, flyingStartSpeed_Hz and flyingStartTimeDelay to Expressions window in CCS Debug Perspective, tune and record the watch window values with the newly defined parameters in user_mtr2.h file.

USER_MOTOR2_SPEED_FLYST_Hz = flyingStartSpeed_Hz's value

USER_MOTOR2_DELAY_FLYST_s = flyingStartTimeDelay's value

3.3.4.10 Tuning Vibration Compensation Parameters

The automatic vibration compensation function has been added and called in motor drive ISR to calculate feedforward torque current.

1. Adding the pre-define symbols *MOTOR1_VIBCOMPA* in build configuration of the project as described in Section 3.2.2 for enabling the vibration compensation.
2. Adding the variables vibCompAlpha, vibCompGain, and vibCompIndexDelta to Expressions window in CCS Debug Perspective, and tuning these parameters to achieve the expectation performance for the vibration compensation according to the compressor and air-conditioner system.
   - vibCompAlpha is used as the learning speed. The higher this value (with a maximum of 1.0) the slowest it learns the algorithm. A high value is desirable though, since it provides noise immunity.
   - vibCompIndexDelta is to advance the output waveform into the future by a little bit so that the resulting current will be very close to the desired value by the time the mechanical angle reaches that point. A typical value of 10 is recommended, but ultimately needs to be fine-tuned by the user.
   - vibCompGain is the gain factor of the feedforward torque reference torque current value (with a maximum of 1.0).
3. Change speed reference (motorVars[0].speedRef_Hz) and speed controller gains (motorVars[0].Kp_spd and motorVars[0].Ki_spd). This step is used to take the motor and load to where the motor vibrates due to the pulsating load. For vibration compensation to work better, increase the values of the speed controller gains. Make sure the speed controller is still stable though.
4. Adding the pre-define symbols *DEBUG_MONITOR_EN* in build configuration of the project as described in Section 3.2.2 for enabling the motor running speed vibration. Now enable the vibration compensation output by setting this flag, vibCompFlagEnable = 1. Then let it run for a 5 to 10 seconds, and then get the new speed variation by setting this bit: motorVars[0].flagClearRecord = 1. Record that the speed variation is about 2Hz.

5. If the vibration was not reduced, try increasing the speed controller gains. Also try increasing the learning speed of the vibration compensation algorithm by decreasing the value of vibCompAlpha in decrements of 0.02, so try: 0.99, 0.97, 0.95, etc. each time you change vibCompAlpha, let it run for a few seconds and get a reading of the speed variation by resetting that calculation: motorVars[0].flagClearRecord = 1.
6. After tuned and fixed these variables value, record the watch window values with the newly defined parameters in user_mtr1.h file.

   USER_MOTOR1_VIBCOMPA_ALPHA = vibCompAlpha's value. The learning rate of the vibration compensation module from 0.0 to 1.0.

   USER_MOTOR1_VIBCOMPA_GAIN = vibCompGain's value. The gain of the vibration compensation module from 0.0 to 1.0.

   USER_MOTOR1_VIBCOMPA_INDEX_DELTA = vibCompIndexDelta's value. The phase advance of the vibration compensation module from 0 to 360.

Controlling motor current based on compressor torque vs angle is an alternate technique to counter the speed ripple variations. Adding the pre-define symbols *MOTOR1_VIBCOMPT* in build configuration of the project as described in Section 3.2.2 for enabling this vibration compensation method.

Depending upon the rolling piston angle, an additional torque current component can be either added or subtracted from the speed PI controller output. The current needs to be added during compression stage and subtracted during exhaustion (where it aids the movement of piston leading to increase in speed) and its magnitude can be computed empirically to match the torque profile of the compressor. Algorithm has split the 360 mechanical deg into 3 sector and compensation current can be added separately through variables VibCompAlpha0, 120 and 240. With rough tuning of the compensation, the speed ripple is reduced from 200Hz to under 100Hz @ 1200rpm. Further reduction in speed ripple is possible, when tuning matches the torque profile to the closest. Usually vibration compensation is enabled for compressor speeds between 1200 - 2000rpm (100Hz), post which its impact tends to be smaller.

3.3.4.11 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 related parameters below.

- 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 below 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      225           // 2.25us

//! \brief Defines the PWM deadband rising edge delay count (system clocks)
#define MTR1_PWM_DBRED_CNT      245           // 2.45us
```

- Minimum duration of pulse width PWM, it speifies the must be grater 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   (480U)
```

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

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

#define USER_M1_DCLINKSS_SAMPLE_DELAY   (455U)
```

3.4 Test Results

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

3.4.1 Performance Data and Curves

Table 3-2 shows the Fan motor testing data at various speed with different load, the motor is coupled to the HD-705-6N dynamometer to add the load.

Table 3-2. Fan Motor and Inverter Power, Efficiency at Different Speed with Load

$SPEED_{ref}$ (rpm)	$SPEED_{error}$ (rpm)	T_{load} (N.m)	P_{o_motor} (W)	$P_{o\_inverter}$ (W)	η_{motor} (%)	$P_{in\_inverter}$ (W)	$\eta_{inverter}$ (%)
750	3	0.7863	61.8041	89.2126	69.28	95.5376	93.38
1500	3	0.8386	131.8089	168.3758	78.28	182.5367	92.24
2250	4	0.8907	210.0507	256.1420	82.01	276.9516	92.49
1500	3	0.8456	132.8883	168.9123	78.67	183.0125	92.30
2250	4	0.8952	211.1273	256.1164	82.44	275.9552	92.81
3000	3	0.9482	298.1318	352.3146	84.62	368.9679	95.49
2250	4	0.9037	213.0801	257.8850	82.63	277.8515	92.81
1500	4	0.8542	134.2593	169.7001	79.12	183.6163	92.42
750	2	0.8035	63.1298	90.2685	69.94	98.2091	91.91

Table 3-3 shows the Compressor motor testing data at various speed with different load, the motor is coupled to the HD-815-6N dynamometer to add the load.

Table 3-3. Compressor Motor and Inverter Power, Efficient at Different Speed with Load

$Speed_{ref}$ (rpm)	$Speed_{error}$ (rpm)	T_{load} (N.m)	P_{o_motor} (W)	$P_{o\_inverter}$ (W)	η_{motor} (%)	$P_{in\_inverter}$ (W)	$\eta_{inverter}$ (%)
750	2	1.9845	155.8340	229.9068	67.74	251.6578	91.36
1500	4	2.3945	376.1113	521.2635	72.15	569.9387	0.91
2250	5	4.5485	1071.9087	1346.4348	79.61	1372.200	98.12
1500	5	4.2020	660.0950	864.5136	76.36	903.3708	95.70
750	3	5.3235	417.9971	586.1813	71.31	642.0582	91.30
1500	6	5.6984	895.2029	1153.2485	77.63	1212.664	95.10
750	2	5.2779	414.3357	582.2262	71.17	636.5326	91.47

Figure 3-27 shows the measured fan motor drive efficiency at different speed in the system with loading torque variation.

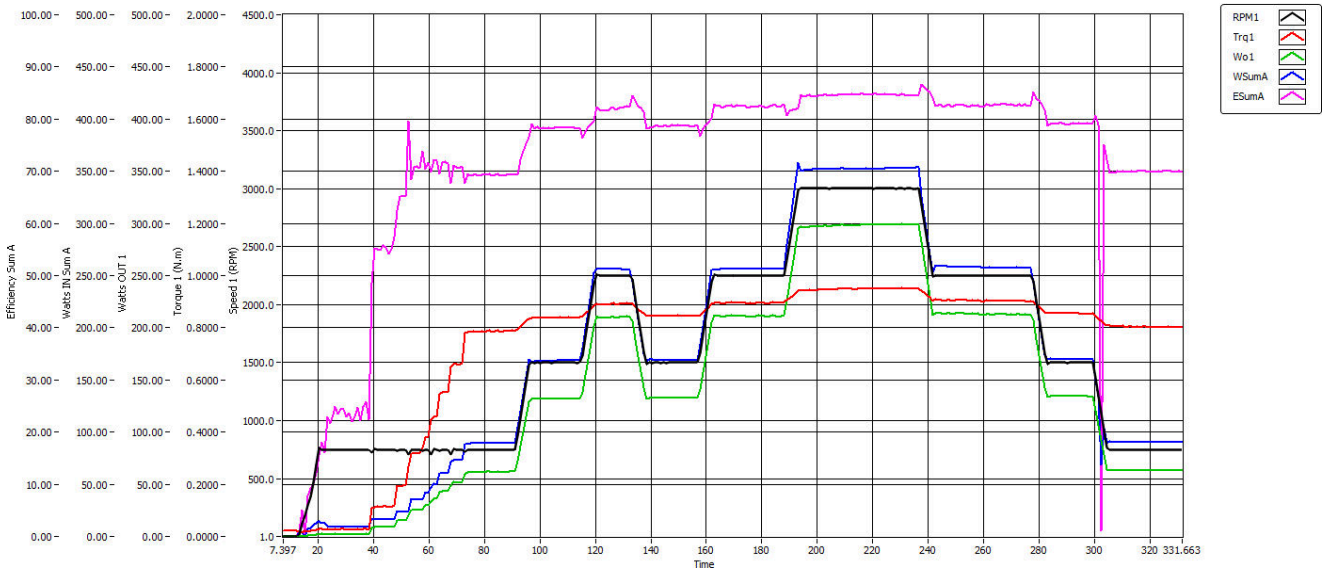


Figure 3-27. Fan Motor Output Power and Efficiency at Various Speed and Load

Figure 3-28 shows the measured fan motor drive efficiency at different speed in the system with loading torque variation.

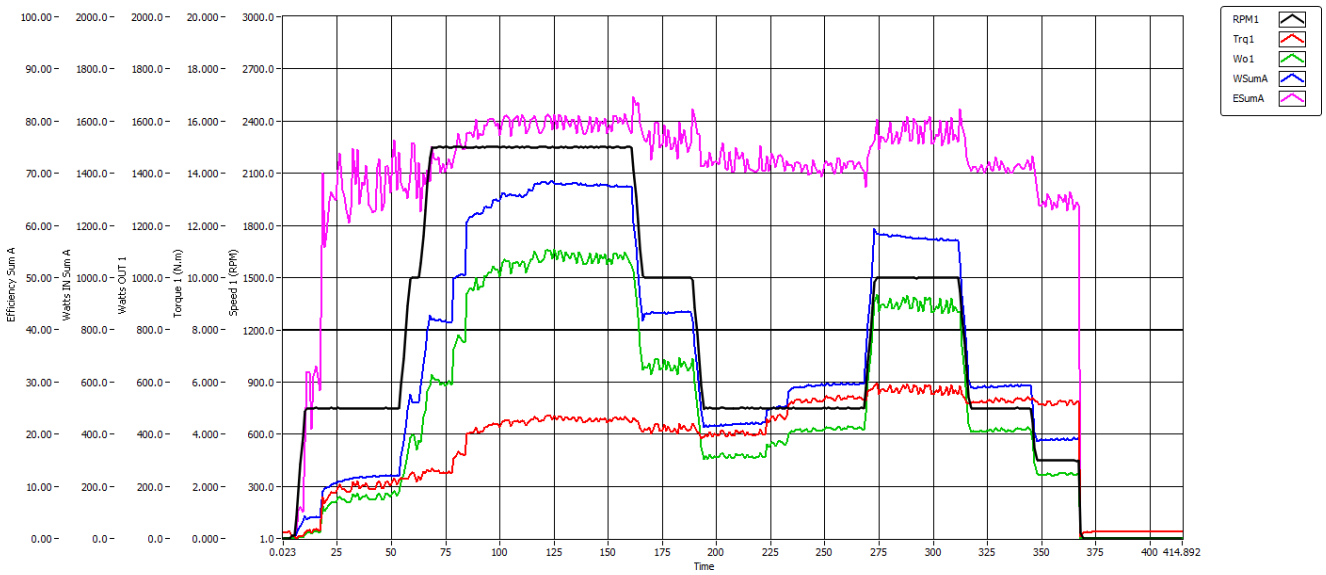


Figure 3-28. Compressor Motor Output Power and Efficiency at Various Speed and Load

Table 3-4, Table 3-5, and Table 3-6 show the PFC performance data at inputs of 165-VAC, 220-VAC, and 265-VAC input with different load.

Table 3-4. PFC Performance Data Under 165 VAC Input

V_{INAC} (V)	I_{INAC} (A)	P_{INAC} (W)	PF	THDi (%)	V_{OUT} (V)	I_{OUT} (A)	P_{OUT} (W)	EFFICIENCY (%)
165.0	0.89	145.6	0.982	13.29	376.8	0.363	137	94.1
165.0	1.75	287.6	0.987	11.69	376.5	0.729	275	95.6
165.0	2.62	430.9	0.992	8.57	376.1	1.101	415	96.3
165.0	3.49	572.9	0.991	9.65	376.4	1.465	552	96.4
165.0	4.38	715.9	0.989	9.15	376.5	2.194	827	96.5
165.0	5.22	857.3	0.991	7.76	376.5	2.194	827	96.5
165.0	6.11	1002.5	0.992	6.39	376.8	2.559	966	96.4
165.0	6.99	1146.1	0.993	5.49	377.3	2.926	1106	96.5
165.0	7.87	1295.5	0.994	4.97	377.5	3.301	1248	96.3
165.0	8.96	1465.5	0.995	3.75	380.2	3.715	1457	99.4
165.0	9.98	1642.4	0.996	3.39	380.3	4.097	1629	99.18
165.0	10.67	1757.6	0.997	3.12	380.4	4.427	1747	99.39

Table 3-5. PFC Performance Data Under 220 VAC Input

V_{INAC} (V)	I_{INAC} (A)	P_{INAC} (W)	PF	THDi (%)	V_{OUT} (V)	I_{OUT} (A)	P_{OUT} (W)	EFFICIENCY (%)
220.1	0.686	149.2	0.982	14.9	376.9	0.368	139	93.2
220.1	1.329	287.5	0.982	14.29	377.2	0.731	276	96.0
220.1	1.94	427.0	0.982	9.25	377.1	1.096	414	96.9
220.1	2.626	572.8	0.992	8.01	377	1.47	555	97.1
220.1	3.24	709.5	0.989	7.285	376.9	1.827	689	97.4
220.1	3.86	848.3	0.991	5.958	376.8	2.191	827	97.3
220.1	4.52	990.5	0.993	6.79	376.7	2.556	965	97.4
220.1	5.21	1138.5	0.992	6.4	376.6	2.938	1108	97.4
220.1	5.8	1271.5	0.993	5.98	376.6	3.283	1238	97.4
220.1	6.45	1416.5	0.994	5.35	376.6	3.656	1379	97.4
220.1	7.07	1549.5	0.994	4.93	376.6	4.003	1510	97.5
220.1	7.77	1709.8	0.995	4.29	376.6	4.404	1661	97.22

Table 3-6. PFC Performance Data Under 265 VAC Input

V_{INAC} (V)	I_{INAC} (A)	P_{INAC} (W)	PF	THDi (%)	V_{OUT} (V)	I_{OUT} (A)	P_{OUT} (W)	EFFICIENCY (%)
250.0	0.61	148.7	0.965	12.6	385.9	0.368	142	95.5
250.0	1.24	298.9	0.959	8.65	385.9	0.751	290	97.0
250.0	1.81	447.5	0.977	9.97	385.8	1.124	434	97.0
250.0	2.42	595.8	0.979	8.82	385.7	1.499	579	97.2
250.0	3.02	744.3	0.981	11.65	385.7	1.876	724	97.3
250.0	3.62	889.6	0.979	12.03	385.7	2.246	868	97.6
250.0	4.21	1034.9	0.982	10.95	385.6	2.619	1012	97.8
250.0	4.79	1183.7	0.985	9.75	385.6	2.993	1156	97.7
250.0	5.39	1335.2	0.988	7.39	385.6	3.374	1303	97.6
250.0	5.99	1488.9	0.991	5.93	385.8	3.759	1452	97.5
250.0	6.59	1640.1	0.992	5.12	385.9	4.137	1600	97.6
250.0	7.09	1768.6	0.993	4.79	384.9	4.478	1727	97.6

3.4.2 Functional Waveforms

Figure 3-29 is a screen shot when the Fan motor is running at 150Hz with 1 N.m load torque. The following displays:

- CH1 (Yellow): Rotor angle output by a DAC
- CH2 (Green): Sensing phase current output by a DAC
- CH3 (Purple): Sensing phase voltage output by a DAC
- CH4 (Pink): Measured phase current with a current probe

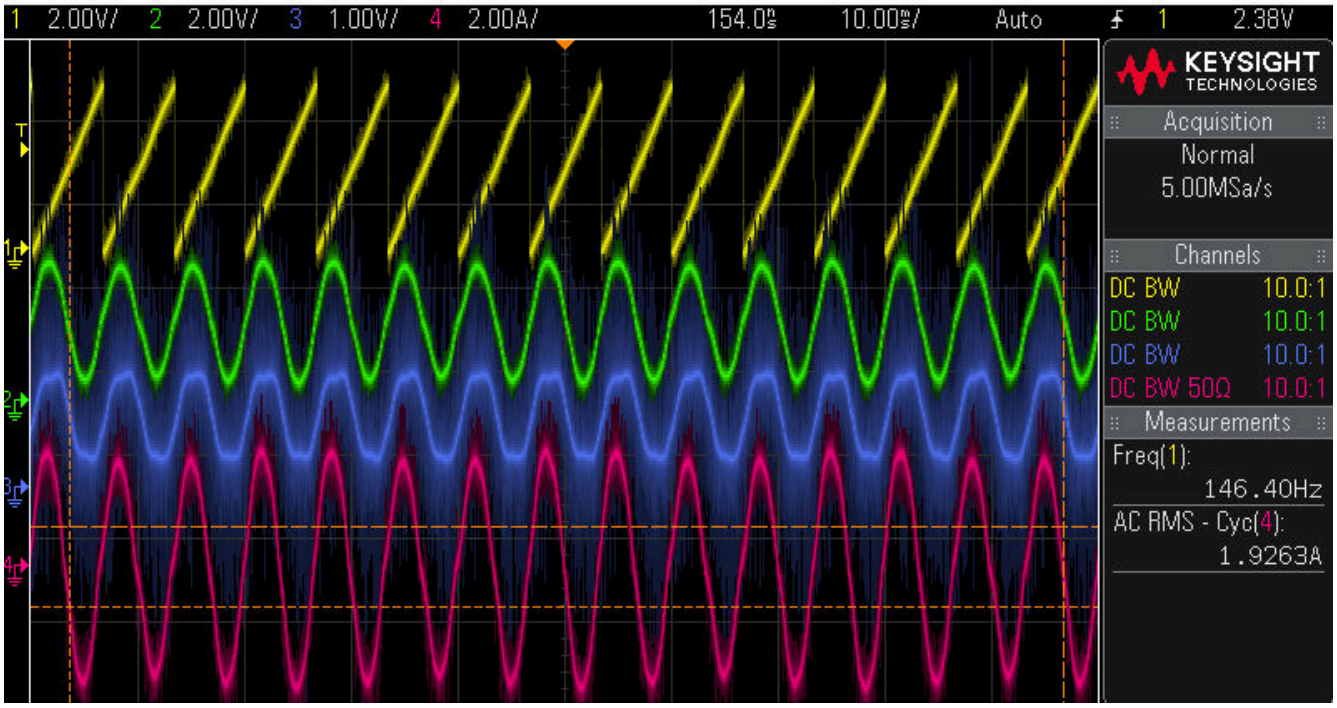


Figure 3-29. Phase Current and Voltage Waveforms of Fan Motor at 150Hz

Figure 3-29 is a screen shot when the compressor motor is running at 100Hz with 5 N.m load torque. It shows

- CH1 (Yellow): Rotor angle output by a DAC
- CH2 (Green): Sensing phase current output by a DAC
- CH3 (Purple): Sensing phase voltage output by a DAC
- CH4 (Pink): Measured phase current with a current probe

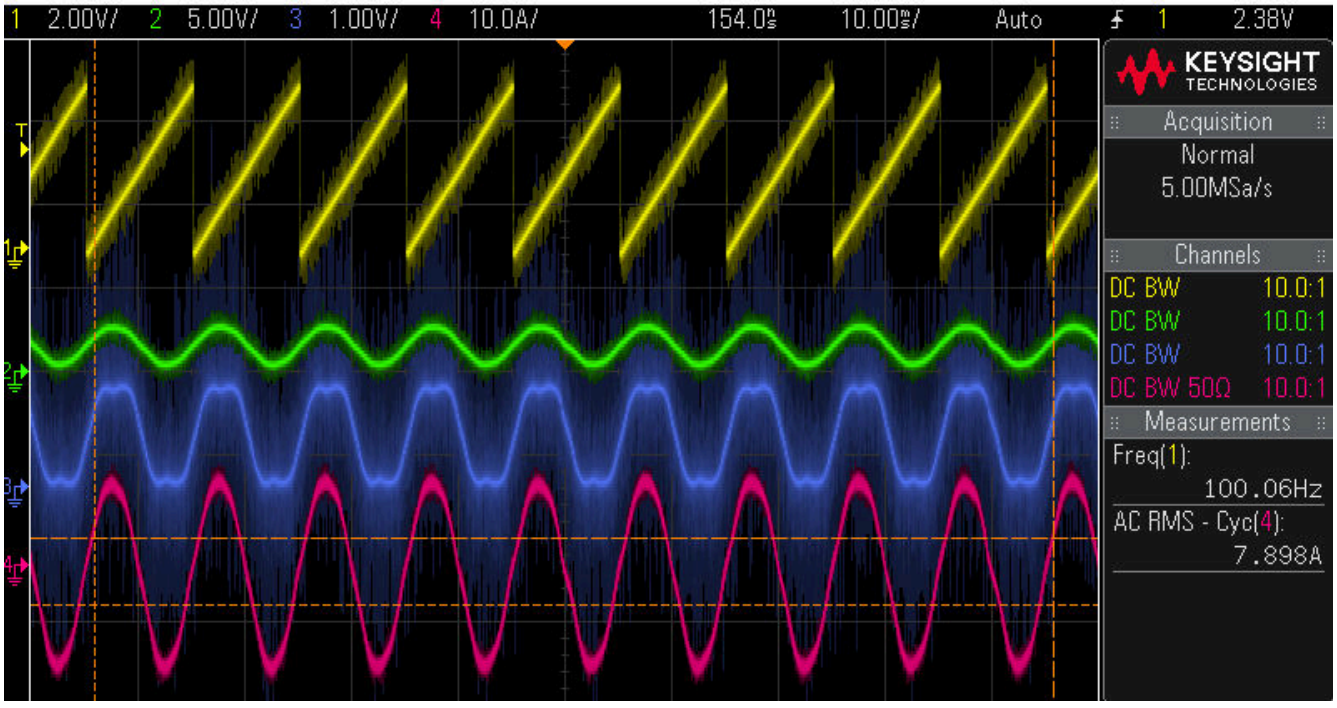


Figure 3-30. Phase Current and Voltage Waveforms of Compressor Motor at 100Hz

The flying start is implemented in fan control which is able to detect if the motor is running before the startup and skip the acceleration phase if not necessary. The fan motor is run in sensorless FOC from the begin without need to stop it before the start.

Figure 3-29 is a screen shot when the fan motor is running without the flying start feature. It shows

- CH1 (Yellow): Rotor angle output by a DAC
- CH2 (Green): Sensing phase current output by a DAC
- CH4 (Pink): Measured phase current with a current probe

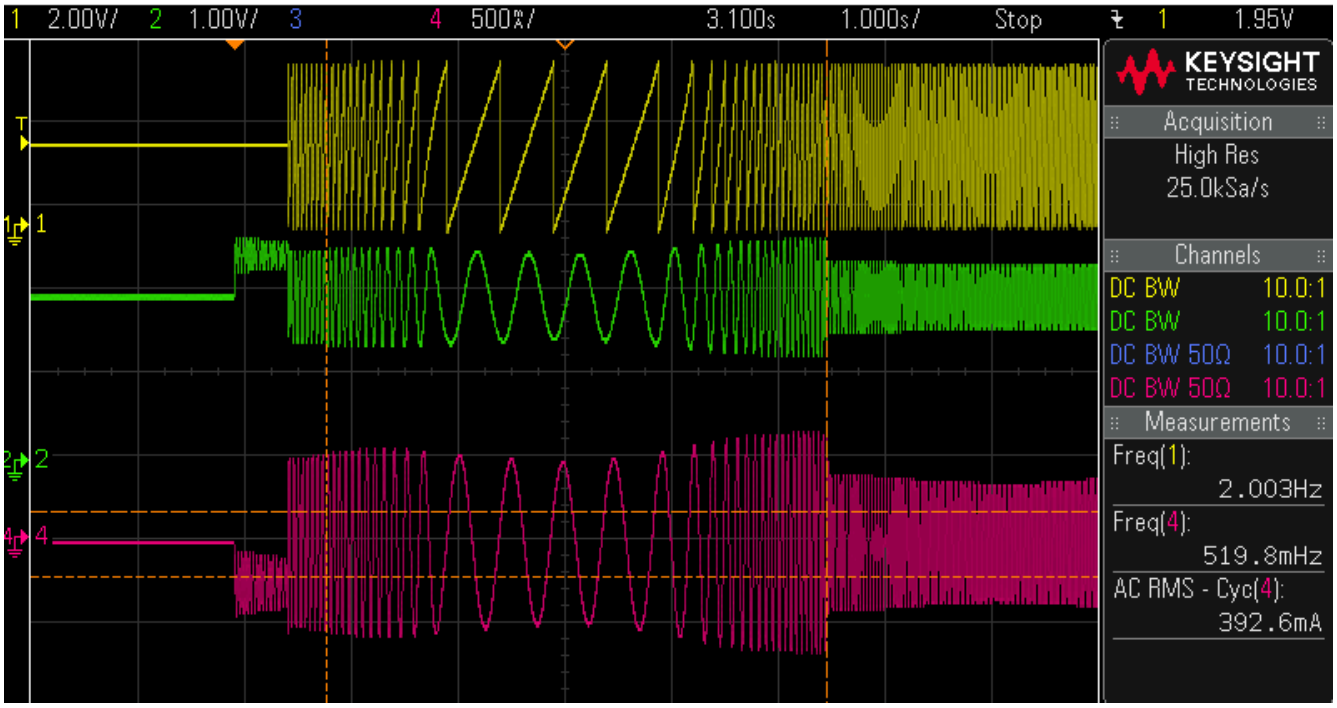


Figure 3-31. Phase Current and Rotor Angle of Fan Motor without Flying Start

Figure 3-32 is a screen shot when the fan motor is running with the flying start feature. It shows

- CH1 (Yellow): Rotor angle output by a DAC
- CH2 (Green): Sensing phase current output by a DAC
- CH4 (Pink): Measured phase current with a current probe

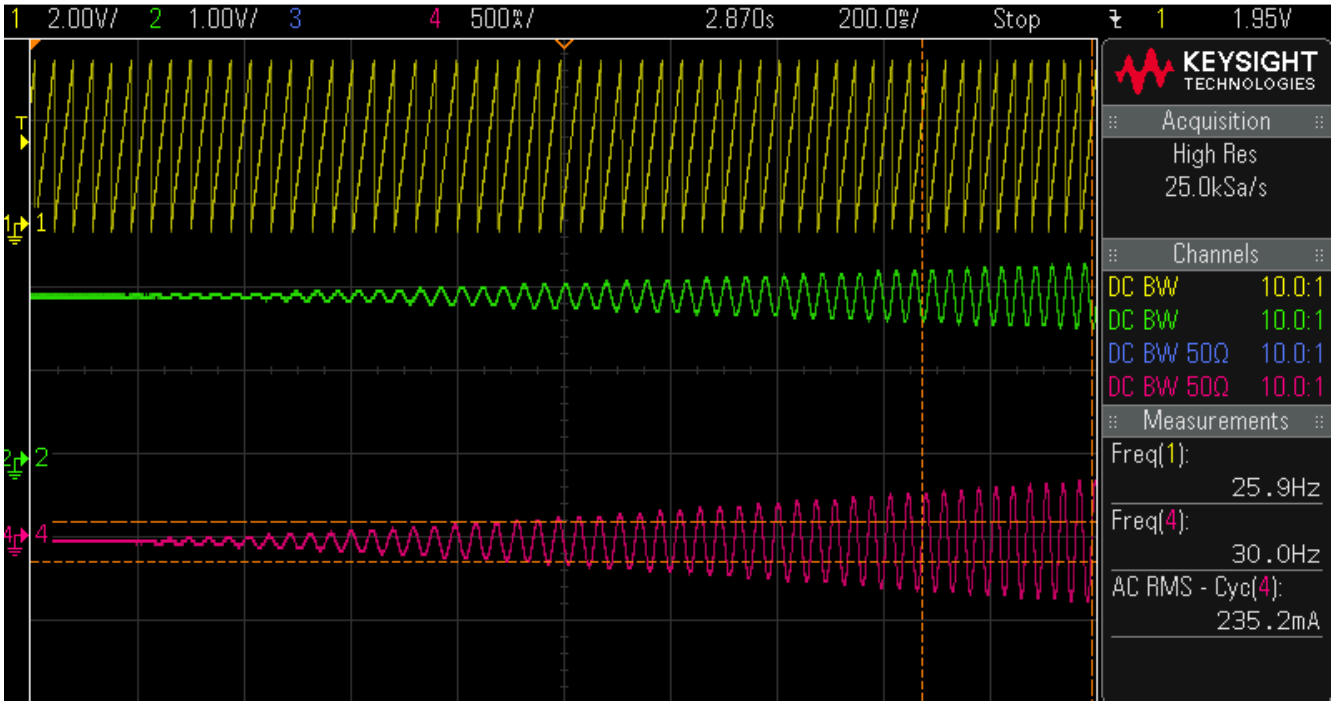


Figure 3-32. Phase Current and Rotor Angle of Fan Motor with Flying Start

Figure 3-33 is a screen shot of the system when the fan and compressor motors are running at 100Hz with a high load. It shows

- CH1 (Yellow): Fan motor rotor angle output by a DAC
- CH2 (Green): Compressor motor rotor angle output by a DAC
- CH3 (Purple): Measured fan motor phase current with a current probe
- CH4 (Pink): Measured compressor phase current with a current probe

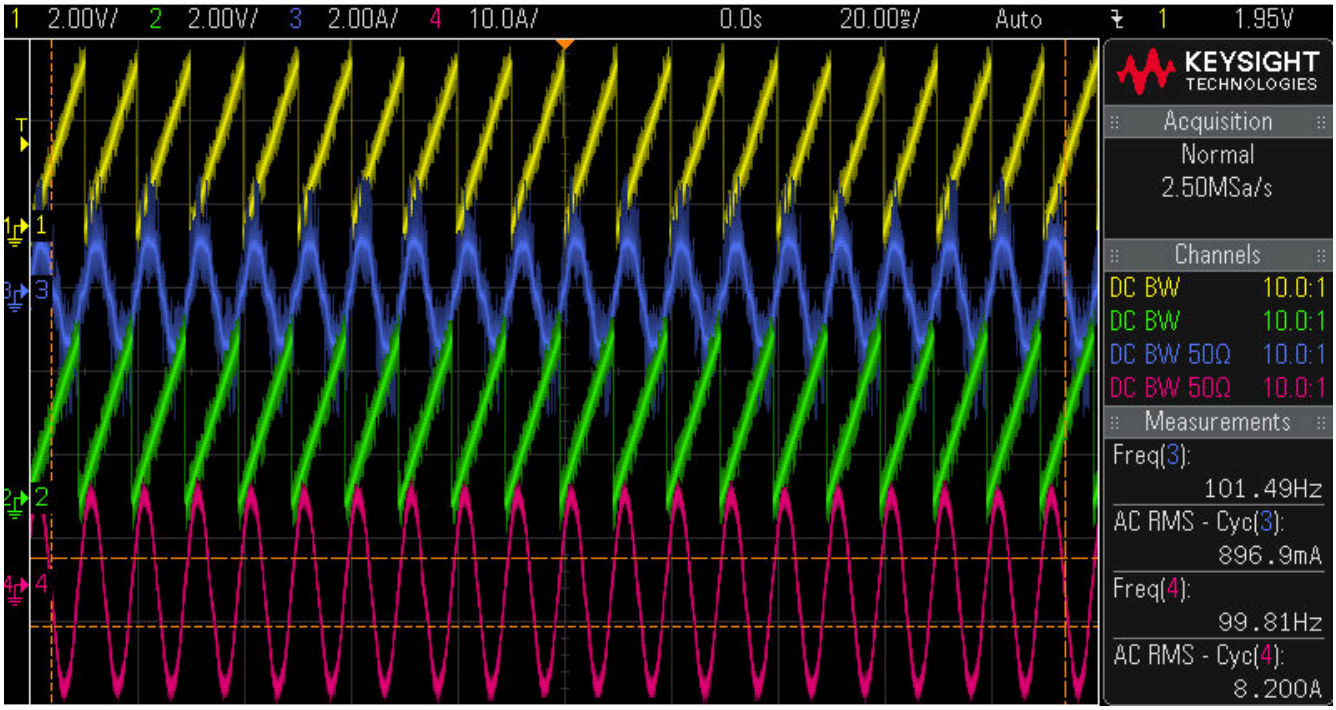


Figure 3-33. Phase Current and Rotor Angle of Fan and Compressor Motors

3.4.3 Transient Waveforms

The HD-705-6N dynamometer was used for the fan motor, the dynamometer was controller with DSP6001 controller and M-TEST software.

Figure 3-34 shows the fan motor output power, inverter input and output power at different speed with constant load.

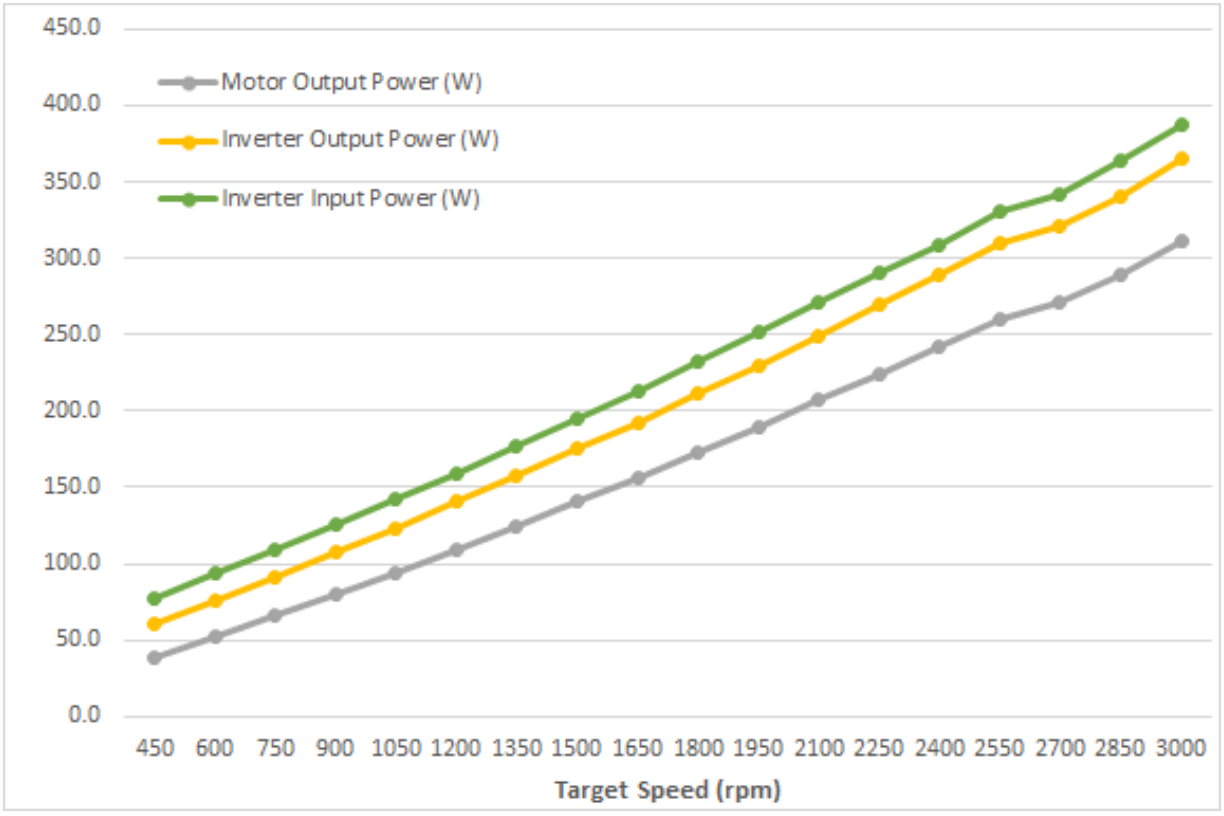


Figure 3-34. Fan Motor Output Power, Inverter Input and Output Power at Different Speed with Constant Load

Figure 3-35 shows the fan motor output and inverter efficiency at different speed with constant load.

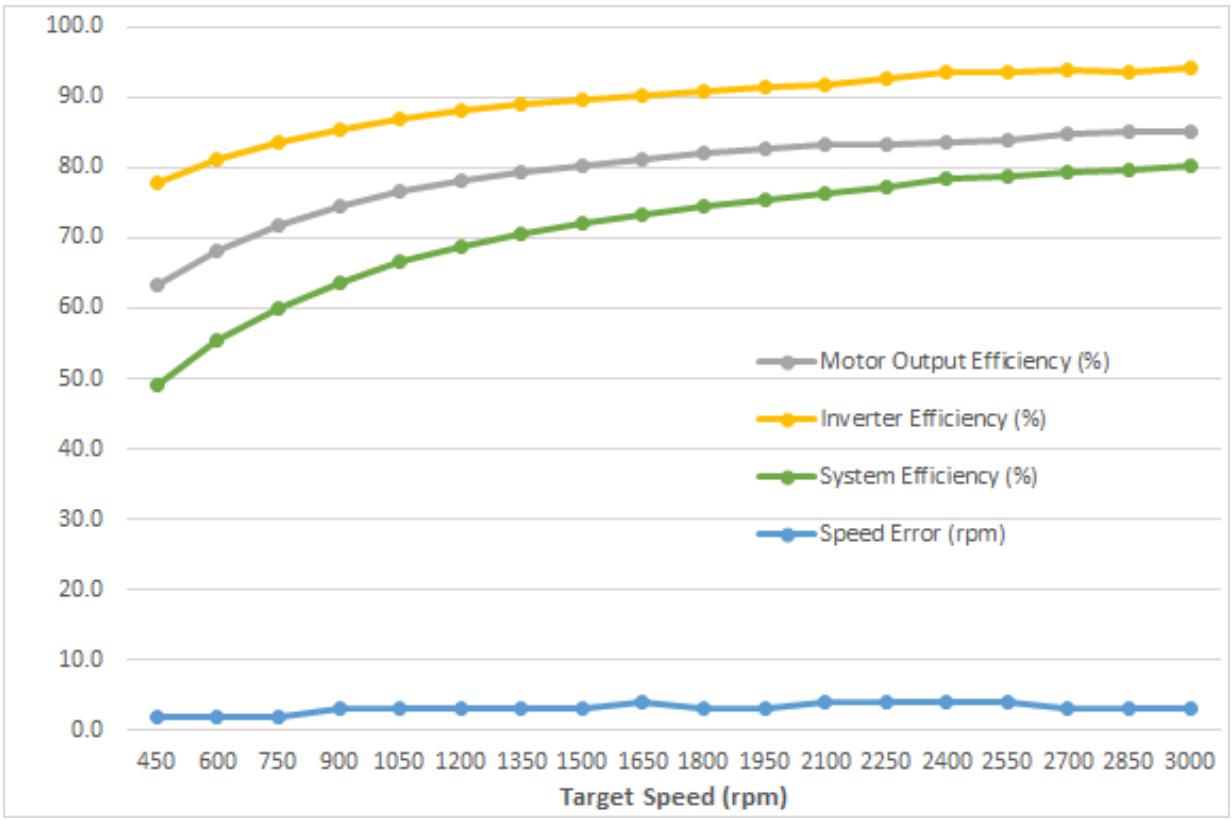


Figure 3-35. Fan Motor Output and Inverter Efficiency at Different Speed with Constant Load

Figure 3-36 shows the compressor motor output power, inverter input and output power at different speed with constant load.

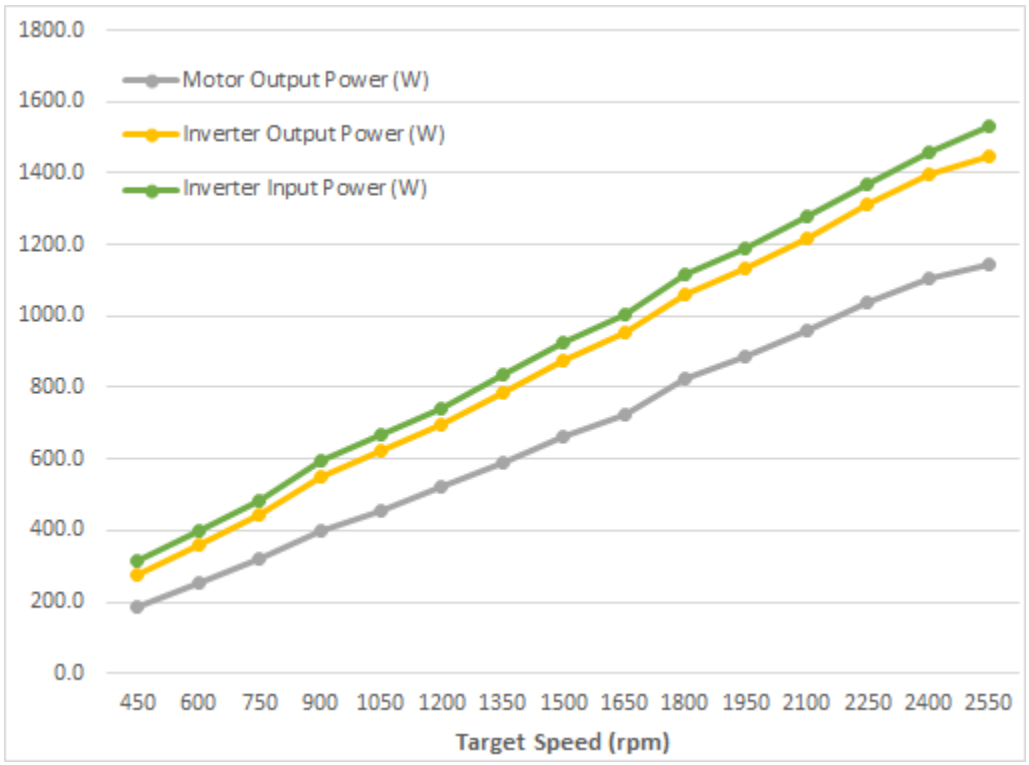


Figure 3-36. Compressor Motor Output Power, Inverter Input and Output Power at Different Speed with Constant Load

Figure 3-37 shows the compressor motor output and inverter efficiency at different speed with constant load.

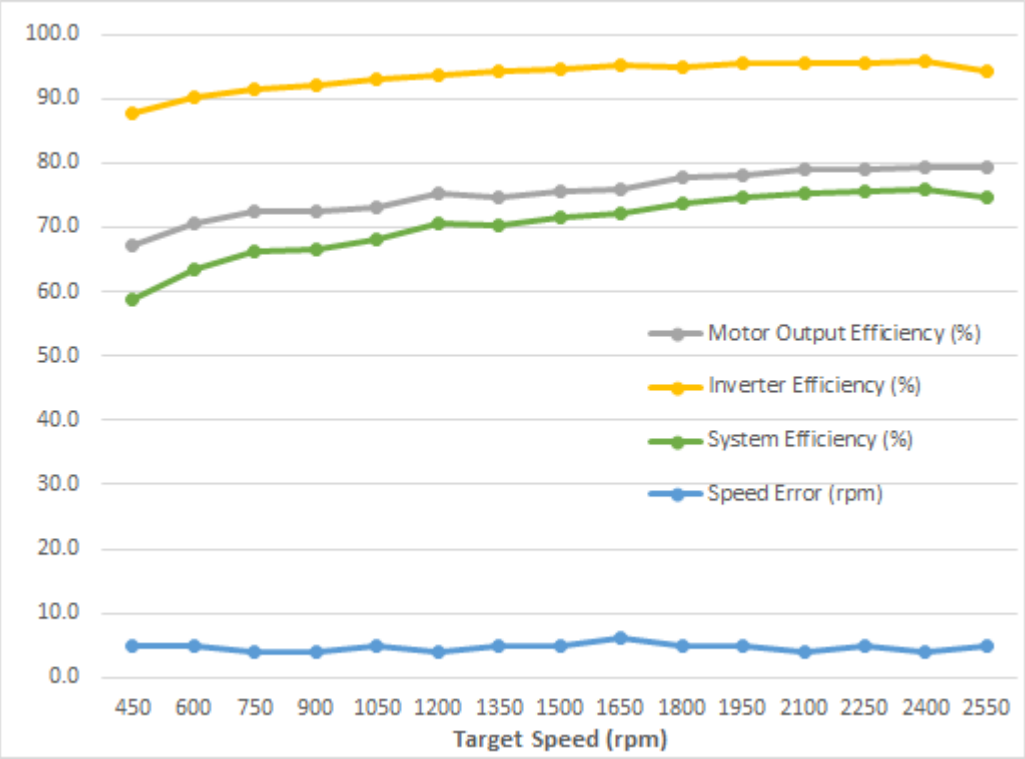


Figure 3-37. Compressor Motor Output and Inverter Efficiency at Different Speed with Constant Load

Figure 3-38 shows the fan motor output efficiency using FAST or eSMO control technique at different speed with constant load.

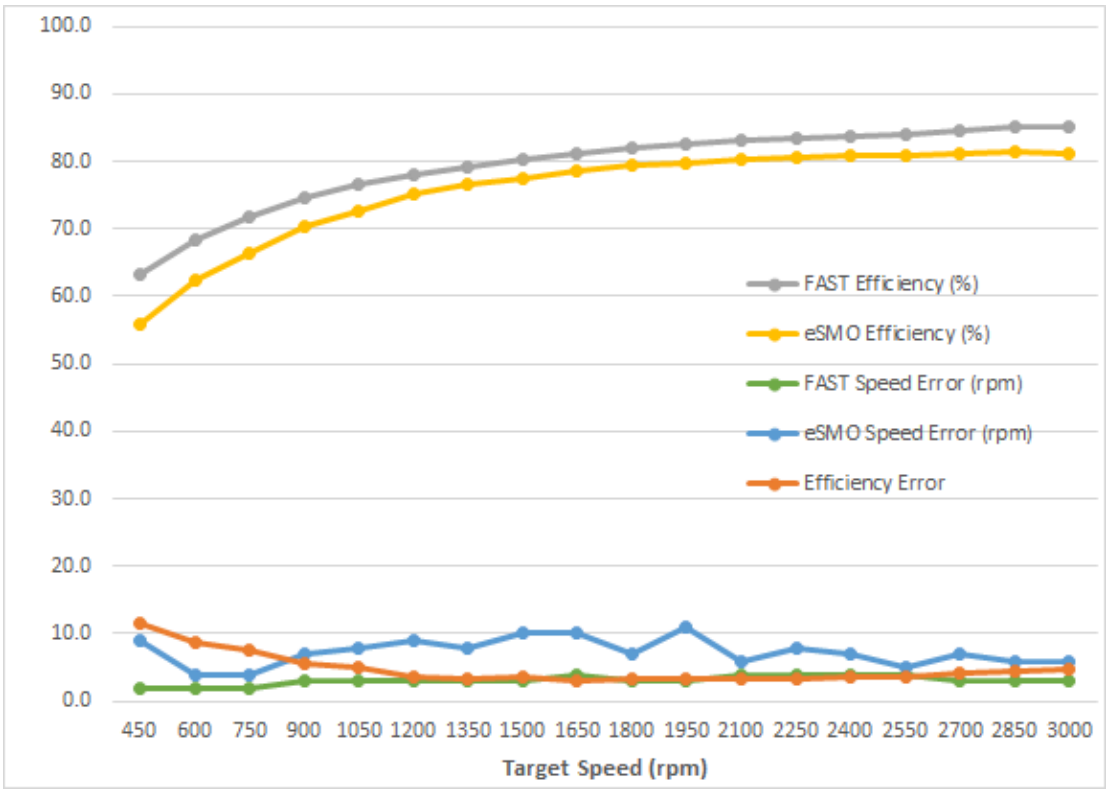


Figure 3-38. Fan Motor Output Efficiency Using FAST or eSMO Control Technique at Different Speed with Constant Load

Figure 3-39 shows the variable transient, power and efficiency of the motor at different speed with load.

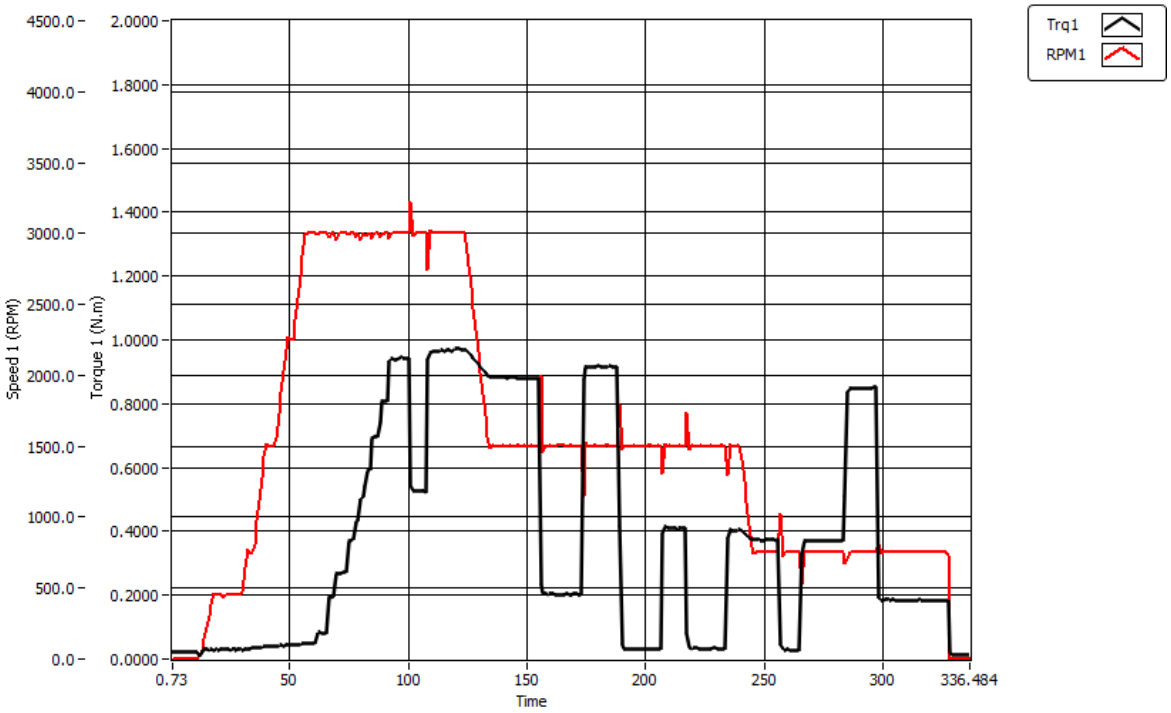


Figure 3-39. Load Transient Measurement at Different Speed and Load

3.4.4 MCU CPU Load, Memory and Peripherals Usage

3.4.4.1 CPU Load for Full Implementation

Table 3-7 shows the CPU cycles used, the CPU loading and available MIPs when a full implementation of InstaSPIN-FOC is used for dual motor control with PFC on F280025C with 100 MHz CPU, as well as users' code is loaded to FLASH, and the ISR code is copied to RAM for execution.

Table 3-7. F28002x CPU Loading for Dual Motor Control with FAST

CPU=100MHz	Maximum CPU Cycles For ISR	Maximum CPU Utilization [%]	Maximum MIPS Used [MIPS]
PFC at PWM=72kHz, ISR= 36kHz	465	16.74	16.74
Motor 1 at PWM=6kHz, ISR=6kHz	2372	14.23	14.23
Motor 2 at PWM=18kHz, ISR=6kHz	2226	13.36	13.36
Total Used CPU Utilization		**44.33**	**44.33**
Available CPU MIPS			**55.67**

Table 3-8 shows the CPU cycles used, the CPU loading and available MIPs when a full implementation of eSMO is used for dual motor control with PFC on F280025C with 100 MHz CPU, as well as users' code is loaded to FLASH, and the ISR code is copied to RAM for execution.

Table 3-8. F28002x CPU Loading for Dual Motor Control with eSMO

F280025C CPU = 100 MHz Available MIPS = 100 MIPS	Maximum CPU Cycles For ISR	Maximum CPU Utilization [%]	Maximum MIPS Used [MIPS]
PFC at PWM = 72 kHz, ISR = 36 kHz	465	16.74	16.74
Motor 1 at PWM = 6 kHz, ISR = 6 kHz	1745	10.47	10.47
Motor 2 at PWM = 18 kHz, ISR = 6 kHz	1568	9.41	9.41
Total Used CPU Utilization [%]		**36.62**	
Available CPU MIPS [MIPS]			**63.38**

Table 3-9 shows the CPU cycles used, the CPU loading and available MIPs when a full implementation of InstaSPIN-FOC is used for dual motor control with PFC on F280039C/F2800137 with 120 MHz CPU, as well as users' code is loaded to FLASH, and the ISR code is copied to RAM for execution.

Table 3-9. F28003x/F280013x CPU Loading for Dual Motor Control with FAST

CPU=120MHz	Maximum CPU Cycles For ISR	Maximum CPU Utilization [%]	Maximum MIPS Used [MIPS]
PFC at PWM=72kHz, ISR=	465	13.95	16.74
Motor 1 at PWM=6kHz, ISR=6kHz	2372	11.86	14.23
Motor 2 at PWM=18kHz, ISR=6kHz	2226	11.13	13.36
Total Used CPU Utilization [%]		**36.94**	
Available CPU MIPS [MIPS]			**75.67**

Table 3-10 shows the CPU cycles used, the CPU loading and available MIPs when a full implementation of eSMO is used for dual motor control with PFC on F280039C/F2800137 with 120MHz CPU, as well as users' code is loaded to FLASH, and the ISR code is copied to RAM for execution.

Table 3-10. F28003x/F280013x CPU Loading for Dual Motor Control with eSMO

F280039C CPU = 120 MHz Available MIPS = 120 MIPS	Maximum CPU Cycles For ISR	Maximum CPU Utilization [%]	Maximum MIPS Used [MIPS]
PFC at PWM = 72 kHz, ISR = 32 kHz	465	13.95	16.74
Motor 1 at PWM = 6kHz, ISR = 6kHz	1745	8.72	10.47
Motor 2 at PWM = 18kHz, ISR = 6kHz	1568	7.84	9,41
Total Used CPU Utilization [%]		**30.52**	
Available CPU MIPS [MIPS]			**83.38**

3.4.4.2 Memory Usage

shows how much memory is needed to run the application on TMS320F280025C, TMS320F280039C, or TMS320F2800137 MCU. A significant part of the microcontroller memory is still available for other tasks.

Table 3-11. Memory Usage for Dual Motor Control with FAST

Memory Type	Used Memory On F280025C	Available Memory on F280025C	F280025C Memory Utilization	Used Memory On F280039C	Available Memory on F280039C	F280039C Memory Utilization	Used Memory On F2800137	Available Memory on F2800137	F2800137 Memory Utilization
FLASH	**45.4 KB**	128 KB	35.5%	52.8 KB	384 KB	13.8%	50.3 KB	256 KB	19.6%
RAM	**17.7 KB**	24 KB	73.7%	21.3 KB	69 KB	30.8%	20.7 KB	36 KB	57.6%

Table 3-12. Memory Usage for Dual Motor Control with eSMO

Memory Type	Used Memory On F280025C	Available Memory on F280025C	F280025C Memory Utilization	Used Memory On F280039C	Available Memory on F280025C	F280039C Memory Utilization	Used Memory On F2800137	Available Memory on F2800137	F2800137 Memory Utilization
FLASH	**38.0 KB**	128 KB	13.8%	39.4 kB	384 KB	12.3%	37.0 KB	256 KB	14.4%
RAM	**15.3 KB**	24 KB	30.9%	15.7 KB	69 KB	22.8%	15.6 KB	36 KB	43.3%

3.4.4.3 Peripherals Usage

Table 3-13 lists the peripherals and pins of F28002x or F28003x used by this reference design. It is not allowed to use these peripherals for any other purposes, but the user can modify the pins assignment according to their hardware design.

Table 3-13. F28002x/003x/0013x MCU Peripherals Usage

Module	Purpose	Notes
EPWM1, EPWM3, EPWM5	Motor 1 3-phase PWM	Total 6 PWM channels
EPWM2, EPWM4, EPWM6	Motor 2 3-phase PWM	Total 6 PWM channels
EPWM7	PFC 2-phase PWM	Total 2PWM channels
ADCA, ADCC	Motor 1 and 2 currents	3 ADC channels for current sensing each motor
ADCA, ADCC	Motor 1 and 2 voltages	Only need for InstaSPIN-FOC, 3 ADC channels for voltage sensing each motor
ADCA (F28002x, F280013x), ADCB(F28003x)	PFC current, AC input voltage and DC output voltage	
CMPSS1, CMPSS3, CMPSS4	Motor 1 and 2 3-phase overcurrent fault	Three-phase current over current fault protection each motor
CMPSS2	PFC overcurrent and DC bus overvoltage faults	
X-Bar	CPIO and output of CMPSS to PWM trip	
SCIA	Communication with indoor control	The other UART or LIN interface can be assigned for this also.
CPU Timer 0	Virtual timer for motor and system control in background loop	CPU timer 1 or 2 can be assigned for this also.

3.5 Migrate Firmware to a New Hardware Board

If the users want to migrate the reference design to their own hardware board, the users need to change the motor and PFC control related PWM, CMPSS, ADC peripherals configuration, hardware parameters and motor parameters accordingly in the files of hal.c, hal.h, user_mtr1.h, user_mtr2.h, and user_pfc.h as described in the following sections.

3.5.1 Configure the PWM, CMPSS, and ADC Modules

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

Configure PWM and CMPSS base address for compressor motor drive

```
// EPWM
#define MTR1_PWM_U_BASE          EPWM3_BASE
#define MTR1_PWM_V_BASE          EPWM5_BASE
#define MTR1_PWM_W_BASE          EPWM6_BASE

// CMPSS
#define MTR1_CMPSS_U_BASE        CMPSS1_BASE
#define MTR1_CMPSS_V_BASE        CMPSS3_BASE
#define MTR1_CMPSS_W_BASE        CMPSS3_BASE
```

Configure ADC base address and channels for compressor motor drive

```
// ADC
#define MTR1_ADC_TRIGGER_SOC     ADC_TRIGGER_EPWM1_SOCA  // EPWM1_SOCA
#define MTR1_ADC_SAMPLE_WINDOW   14

#define MTR1_IU_ADC_BASE         ADCA_BASE               // ADCA-A11*/C0
#define MTR1_IV_ADC_BASE         ADCA_BASE               // ADCA-A5*/C2
#define MTR1_IW_ADC_BASE         ADCA_BASE               // ADCA-A0*/C15
#define MTR1_VU_ADC_BASE         ADCA_BASE               // ADCA-A7*/C3
#define MTR1_VV_ADC_BASE         ADCA_BASE               // ADCA-A12*/C1
#define MTR1_VW_ADC_BASE         ADCA_BASE               // ADCA-A1*
#define MTR1_VDC_ADC_BASE        PFC_VDC_ADC_BASE        // ADCC-A4/C14*

#define MTR1_IU_ADCRES_BASE      ADCARESULT_BASE         // ADCA-A11*/C0
#define MTR1_IV_ADCRES_BASE      ADCARESULT_BASE         // ADCA-A5*/C2
#define MTR1_IW_ADCRES_BASE      ADCARESULT_BASE         // ADCA-A0*/C15
#define MTR1_VU_ADCRES_BASE      ADCARESULT_BASE         // ADCA-A7*/C3
#define MTR1_VV_ADCRES_BASE      ADCARESULT_BASE         // ADCA-A12*/C1
#define MTR1_VW_ADCRES_BASE      ADCARESULT_BASE         // ADCA-A1*
#define MTR1_VDC_ADCRES_BASE     PFC_VDC_ADCRES_BASE     // ADCC-A4/C14*

#define MTR1_IU_ADC_CH_NUM       ADC_CH_ADCIN11          // ADCA-A11*/C0
#define MTR1_IV_ADC_CH_NUM       ADC_CH_ADCIN5           // ADCA-A5*/C2
#define MTR1_IW_ADC_CH_NUM       ADC_CH_ADCIN0           // ADCA-A0*/C15
#define MTR1_VU_ADC_CH_NUM       ADC_CH_ADCIN7           // ADCA-A7*/C3
#define MTR1_VV_ADC_CH_NUM       ADC_CH_ADCIN12          // ADCA-A12*/C1
#define MTR1_VW_ADC_CH_NUM       ADC_CH_ADCIN1           // ADCA-A1*
#define MTR1_VDC_ADC_CH_NUM      PFC_VDC_ADC_CH_NUM      // ADCC-A4/C14*

#define MTR1_IU_ADC_SOC_NUM      ADC_SOC_NUMBER0         // ADCA-A11*/C0-SOC0
#define MTR1_IV_ADC_SOC_NUM      ADC_SOC_NUMBER1         // ADCA-A5*/C2 -SOC1
#define MTR1_IW_ADC_SOC_NUM      ADC_SOC_NUMBER2         // ADCA-A0*/C15-SOC2
#define MTR1_VU_ADC_SOC_NUM      ADC_SOC_NUMBER3         // ADCA-A7*/C3 -SOC3
#define MTR1_VV_ADC_SOC_NUM      ADC_SOC_NUMBER4         // ADCA-A12*/C1-SOC4
#define MTR1_VW_ADC_SOC_NUM      ADC_SOC_NUMBER5         // ADCA-A1*    -SOC5
#define MTR1_VDC_ADC_SOC_NUM     PFC_VDC_ADC_SOC_NUM     // ADCC-A4/C14*
```

Configure peripheral interrupt for compressor motor drive control

```
// Interrupt
#define MTR1_ADC_INT_BASE        ADCA_BASE               // ADCA-A1  -SOC5
#define MTR1_ADC_INT_NUM         ADC_INT_NUMBER2         // ADCA_INT2-SOC5
#define MTR1_ADC_INT_SOC         ADC_SOC_NUMBER6         // ADCA_INT2-SOC5
#define MTR1_PIE_INT_NUM         INT_ADCA2               // ADCA_INT2-SOC5
#define MTR1_CPU_INT_NUM         INTERRUPT_CPU_INT10     // ADCA_INT2-CPU_INT10
#define MTR1_INT_ACK_GROUP       INTERRUPT_ACK_GROUP10   // ADCA_INT2-CPU_INT10
```

Configurethe connections between ADC pin and CMPSS modules in *hal.h* based on the hardware, the details refer to the Table, Analog Pins and Internal Connections in TMS320F28002x Real-Time Microcontrollers Technical Reference Manual, TMS320F28003x Real-Time Microcontrollers Technical Reference Manual, or TMS320F280013x Real-Time Microcontrollers Technical Reference Manual

```
// ADC pins connection to CMPSS
#define MTR1_IU_CMPHP_SEL        ASYSCTL_CMPHPMUX_SELECT_1    //A11*/C0, CMPSS1-HP
#define MTR1_IU_CMPLP_SEL        ASYSCTL_CMPLPMUX_SELECT_1    //A11*/C0, CMPSS1-LP, N/A

#define MTR1_IV_CMPLP_SEL        ASYSCTL_CMPLPMUX_SELECT_3    //A5*/C2,  CMPSS3-LP

#define MTR1_IW_CMPLP_SEL        ASYSCTL_CMPLPMUX_SELECT_3    //A0*/C15, CMPSS3-LP, N/A
```

```
#define MTR1_IU_CMPHP_MUX       1                           //A11*/C0, CMPSS1-HP
#define MTR1_IU_CMPLP_MUX       1                           //A11*/C0, CMPSS1-LP, N/A

#define MTR1_IV_CMPLP_MUX       1                           //A5*/C2,  CMPSS3-LP, N/A

#define MTR1_IW_CMPLP_MUX       2                           //A0*/C15, CMPSS3-LP, N/A
```

Configure the trip signals from CMPSS to be passed to EPWM and GPIO output in *hal.h* based on the hardware, the details refer to Table, ePWM X-BAR Mux Configuration Table and Table, OUTPUT X-BAR Mux Configuration Table in TMS320F28002x Real-Time Microcontrollers Technical Reference Manual, TMS320F28003x Real-Time Microcontrollers Technical Reference Manual , or TMS320F280013x Real-Time Microcontrollers Technical Reference Manual.

```
// XBARINTPUT to EPWM
#define MTR1_XBAR_TRIP_ADDRL    XBAR_O_TRIP7MUX0TO15CFG
#define MTR1_XBAR_TRIP_ADDRH    XBAR_O_TRIP7MUX16TO31CFG

#define MTR1_IU_XBAR_EPWM_MUX   XBAR_EPWM_MUX00_CMPSS1_CTRIPH       // CMPSS1-HP
#define MTR1_IV_XBAR_EPWM_MUX   XBAR_EPWM_MUX05_CMPSS3_CTRIPL       // CMPSS3-LP
#define MTR1_IW_XBAR_EPWM_MUX   XBAR_EPWM_MUX05_CMPSS3_CTRIPL       // CMPSS3-LP, N/A

#define MTR1_IU_XBAR_MUX        XBAR_MUX00          // CMPSS1-HP
#define MTR1_IV_XBAR_MUX        XBAR_MUX05          // CMPSS3-LP
#define MTR1_IW_XBAR_MUX        XBAR_MUX05          // CMPSS3-LP

#define MTR1_XBAR_INPUT         XBAR_INPUT1
#define MTR1_TZ_OSHT            EPWM_TZ_SIGNAL_OSHT1
#define MTR1_XBAR_TRIP          XBAR_TRIP7
#define MTR1_DCTRIPIN           EPWM_DC_COMBINATIONAL_TRIPIN7
```

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

```
// HAL_setupCMPSSsMTR
void HAL_setupCMPSSsMTR(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;
    uint16_t cmpsaDACH;
    uint16_t cmpsaDACL;

    // Refer to the Table 9-2 in Chapter 9 of TMS320F28004x
    // Technical Reference Manual (SPRUI33B), to configure the ePWM X-Bar
    if(obj->motorNum == MTR_1)
    {
        cmpsaDACH = MTR1_CMPSS_DACH_VALUE;
        cmpsaDACL = MTR1_CMPSS_DACL_VALUE;

        ASysCtl_selectCMPHPMux(MTR1_IU_CMPHP_SEL, MTR1_IU_CMPHP_MUX);

        ASysCtl_selectCMPLPMux(MTR1_IV_CMPLP_SEL, MTR1_IV_CMPLP_MUX);

        // --------------- U-Phase ---------------------------------------------
        // Enable CMPSS and configure the negative input signal to come from the DAC
        CMPSS_enableModule(obj->cmpssHandle[0]);
}
```

The CMPSS-generated signals go to the X-Bar, where they can be combined in different and unique fashions to flag unique trip events from multiple sources to implement the fault protection. The faults include the over-current signals from the CMPSS and the fault indicator output from the power module. Configure the XBAR registers in HAL_setupMtrFaults() function in the file of *hal.c* as the following codes.

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

    uint16_t tzSignal;
```

```
    uint16_t dcTripIn;
    if(obj->motorNum == MTR_1)
    {
    }
    else if(obj->motorNum == MTR_2)
    {
    }
}
```

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

```
void HAL_setupGPIOs(HAL_Handle handle)
{
    // GPIO0->EPWM1A->M2/FAN_UH
    GPIO_setPinConfig(GPIO_0_EPWM1_A);
    GPIO_setDirectionMode(0, GPIO_DIR_MODE_OUT);
    GPIO_setPadConfig(0, GPIO_PIN_TYPE_STD);
    return;
}  // end of HAL_setupGPIOs() function
```

As described above, to configure the PWM, ADC, CMPSS and fault protection for fan motor and PFC control based on the hardware. The PFC uses HAL_setupCMPSSsPFC(), and HAL_setupPFCFaults().

Also need to change the configuration codes in HAL_enableMtrPWM(), HAL_enablePFCPWM(), HAL_clearMtrFaultStatus(), and HAL_clearPFCFaultStatus() in hal.h file as below marked **bold** according to the used CMPSS for motor and PFC control.

```
static inline void HAL_enableMtrPWM(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;
    if(obj->motorNum == MTR_1)
    {
        // Clear any comparator digital filter output latch
        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[0]);
        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[1]);
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[2]);
    }
    else if(obj->motorNum == MTR_2)
    {
        // Clear any comparator digital filter output latch
        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[0]);
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[1]);
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[2]);
    }
... ...
    return;
} // end of HAL_enableMtrPWM() function
```

```
static inline void HAL_clearMtrFaultStatus(HAL_MTR_Handle handle)
{
    if(obj->motorNum == MTR_1)
    {
        // Clear any comparator digital filter output latch
        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[0]);
        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[1]);
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[2]);
    }
    else if(obj->motorNum == MTR_2)
    {
        // Clear any comparator digital filter output latch
        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[0]);

        CMPSS_clearFilterLatchLow(obj->cmpssHandle[1]);
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[2]);
    }
... ...
    return;
} // end of HAL_clearMtrFaultStatus() function
```

3.5.2 Setup Hardware Board Parameters

The *user_mtr<1/2>.h* is where all user parameters are stored for motor control. The maximum phase current and phase voltage at the input to the AD converter, these values are hardware dependent and should be based

on the current and voltage sensing and scaling to the ADC input. The number of current sensors and voltage (phase) sensors used are defined in user_mtr<1/2>.h that are hardware dependent.

The "*user_pfc.h*" is where all user parameters are stored for PFC control. The maximum current, AC voltage and DC voltage at the input to the AD converter, these values are hardware dependent and should be based on the current and voltage sensing and scaling to the ADC input.

All of the configurable parameters are defined in the *user_mtr<1/2>.h* and *user_pfc.h* files. These parameters can be calculated using the *TIDM_02010_DMPFC_Hardware Parameters Calculation.xlsx* Excel® spreadsheet. This file is included with the TIDM-02010 archive file at the folder *..\solutions\tidm_02010\docs* to calculate these values and copy these parameters marked ***bold*** to *user_mtr<1/2>.h* and *user_pfc.h* as shown in the following codes.

```
//! \brief Defines the maximum voltage at the AD converter
//  Full scale voltage of AD converter, not the current voltage
#define USER_M1_ADC_FULL_SCALE_VOLTAGE_V         (441.54f)

//! \brief Defines the analog voltage filter pole location, Hz
#define USER_M1_VOLTAGE_FILTER_POLE_Hz           (448.6576819f)

//! \brief Defines the maximum current at the AD converter
// High Voltage motor control kit
#define USER_M1_ADC_FULL_SCALE_CURRENT_A         (39.6f)
```

```
//! \brief Defines the maximum voltage at the AD converter
#define USER_PFC_ADC_FULL_SCALE_DC_VOLTAGE_V    (441.54f)

//! \brief Defines the maximum voltage at the AD converter
#define USER_PFC_ADC_FULL_SCALE_AC_VOLTAGE_V    (441.54f)

//! \brief Defines the maximum current at the AD converter
#define USER_PFC_ADC_FULL_SCALE_CURRENT_A       (49.5f)
```

3.5.3 Configure Faults Protection Parameters

Fault management is implemented in this system that includes over current, over voltage, under voltage, stall, over load, startup failed. The faults protection parameters are defined in *user_mtr<1/2>.h* and *user_pfc.h* as shown in the following codes, which are hardware board, motors, PFC and system dependent.

```
//! \brief motor over current threshold
#define USER_MOTOR1_OVER_CURRENT_A          (8.0)           //

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

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

```
//! \brief AC bus over voltage threshold
#define USER_OVER_VOLTAGE_FAULT_ACV             (280.0f)

//! \brief AC bus over voltage threshold
#define USER_OVER_VOLTAGE_NORM_ACV              (270.0f)

//! \brief AC bus under voltage threshold
#define USER_UNDER_VOLTAGE_FAULT_ACV            (90.0f)

//! \brief AC bus under voltage threshold
#define USER_UNDER_VOLTAGE_NORM_ACV             (100.0f)

//! \brief DC bus over voltage threshold
#define USER_OVER_VOLTAGE_FAULT_DCV             (410.0f)

//! \brief DC bus over voltage threshold
#define USER_OVER_VOLTAGE_NORM_DCV              (400.0f)

//! \brief DC bus under voltage threshold
#define USER_UNDER_VOLTAGE_FAULT_DCV            (15.0f)

//! \brief DC bus under voltage threshold
#define USER_UNDER_VOLTAGE_NORM_DCV             (20.0f)
```

```
//! \brief DC bus over voltage threshold
#define USER_OVER_VOLTAGE_SHUTDOWN_DCV          (420.0f)

//! \brief DC bus shut down voltage threshold
#define USER_SHUTDOWN_VOLTAGE_DCV               (420.0f)
```

3.5.4 Setup Motor Electrical Parameters

The parameters provided in user_mtr<1/2>.h for PMSM/BLDC motors are listed as shown in the following codes. The motor parameters can be identified if the FAST technique is implemented on this motor or getting from the motor data sheet.

```
#define USER_MOTOR1_TYPE                    MOTOR_TYPE_PM
#define USER_MOTOR1_NUM_POLE_PAIRS          (4)
#define USER_MOTOR1_Rr_Ohm                  (0.0)
#define USER_MOTOR1_Rs_Ohm                  (2.62655902f)
#define USER_MOTOR1_Ls_d_H                  (0.00860825367f)
#define USER_MOTOR1_Ls_q_H                  (0.00860825367f)
#define USER_MOTOR1_RATED_FLUX_VpHz         (0.377903223f)
```

3.5.5 Setup PFC Control Parameters

The parameters provided in *pfc_ctrl.h* for PFC control are listed as shown in the following codes.

```
#define CNTL_2p2z_B0_1  0.4104341549f
#define CNTL_2p2z_B1_1 -0.3822445616f
#define CNTL_2p2z_B2_1  0.0712511235f
#define CNTL_2p2z_A1_1 -1.1159959670f
#define CNTL_2p2z_A2_1  0.1159959670f
```

4 Design and Documentation Support

4.1 Design Files

4.1.1 Schematics

To download the schematics, see the design files at TIDM-02010.

4.1.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDM-02010.

4.1.3 Altium Project

To download the Altium project files, see the design files at TIDM-02010.

4.1.4 Gerber Files

To download the Gerber files, see the design files at TIDM-02010.

4.1.5 PCB Layout Guidelines

- This reference design has been implemented using PCB with two layer, 1 oz copper with single side SMD component placement considering cost sensitivity of the application. There are several important aspects that need to keep in mind while designing PCB. In the following, system level placement and layout of each block has been explained.
- Components in the high power path has been kept on outer edges of PCB with minimum distance possible. Microcontroller is placed at center considering optimum distance from all the power blocks that need to be controlled. Pin assignment has been set to minimize the control/feedback signal trace distance and the crossing between analog and digital signals.
- AC Line Protection and EMI Filter
  - AC line protection component are closely placed with minimum distance of connection path. Earth connection guarding has been provided around protection and EMI filter circuit.
  - Active EMI filter is placed at optimum distance so as to stay closer to switching and to have minimum distance to connect to EARTH terminal.
- IPFC Drive

  In IPFC drive, three current paths are very critical for PCB layout - High Power AC Loop, DC loop and Gate Drive loop. These paths need to be short with maximum width possible to reduce parasitic loop inductance.

  - AC loop – Consists of diode bridge (source), inductor and MOSFET drain and MOSFET Source (return). On this loop, especially connection between inductor, MOSFET Drain and Diode Anode handles high frequency and high power. Special care has been taken while connecting this node to minimize parasitic inductance by reducing distance and increasing copper area.
  - DC loop – Consists of diode bridge (source), inductor, diode, capacitor, load (return). To distribute the rms current stress evenly, bank of electrolyte capacitor should be placed optimally such electrical distance of each one from diode cathode approximately remains same. This design uses copper plane for V_{DC} and PGND connection. To suppress high frequency component metal film cap has been placed just next to Kathode of diode. It minimizes the loop inductance significantly.
  - Gate Drive loop – Consists of driver power supply (source), gate driver IC, MOSFET Gate and MOSFET source pin(return). This design uses parallel arrangement for two phases of IPFC for minimizing other two AC/DC loops. Because of this parallel arrangement, outer phase MOSFET Gate is inaccessible to gate driver. An SMD insulated thick jumper has been used to connect the gate driver signal to MOSFET gate.
- Compressor and Fan Drive
  - With highest ripple requirement, compressor drive has been placed closest to the DC bus capacitor bank of IPFC drive and FAN is placed next to compressor.
  - Low side shunt resistor method with 4 wire sensing is implemented for current sensing. Differential pair with impedance matching resistor has been used to connect sensing signal from shunt resistors to OPAMP circuit. Shunt resistors has been placed near the module with immediate ground copper plane connection.
- Auxiliary Power Supply

– With lowest power and ripple requirement, auxiliary power is placed after FAN drive. Dedicated copper plane is used to connect the APS ground to DC bus capacitor bank. This arrangement minimizes interference between high frequency / high power motor current and control circuit.

4.2 Software Files

To download the Code Composer Studio Integrated Development Environment at CCSTUDIO.

To download the TIDM-02010 hardware-specific software design files, see the design files at C2000WARE-MOTORCONTROL-SDK.

4.3 Documentation Support

1. Texas Instruments, *TMS320F28002x Real-Time Microcontrollers* data sheet.
2. Texas Instruments, *TMS320F28002x Real-Time Microcontrollers* technical reference manual.
3. Texas Instruments, *TMS320F28003x Microcontrollers* data sheet.
4. Texas Instruments, *TMS320F28003x Real-Time Microcontrollers* technical reference manual.
5. Texas Instruments, *TMS320F280013x Microcontrollers* data sheet.
6. Texas Instruments, *TMS320F280013x Real-Time Microcontrollers* technical reference manual.
7. Texas Instruments, *InstaSPIN-FOC and InstaSPIN-MOTION* user's guide.
8. Texas Instruments, *Motor Control SDK Universal Project and Lab* user's guide.
9. Texas Instruments, *C2000™ Software Frequency Response Analyzer (SFRA) Library and Compensation Designer* user's guide.
10. Texas Instruments, *Two-Phase Interleaved PFC Converter w/ Power Metering Test Results* reference design
11. Texas Instruments, *Sensorless-FOC for PMSM With Single DC-Link Shunt* application Note
12. Texas Instruments, *TIDM-1022 Valley Switching Boost Power Factor Correction (PFC)*reference design
13. Texas Instruments, *C2000 SysConfig* application note

4.4 Support Resources

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

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

4.5 Trademarks

C2000™, TI E2E™ are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

5 Terminology

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

6 Revision History

Changes from Revision * (January 2022) to Revision A (October 2022)	Page
• Updated the numbering format for tables, figures, and cross-references throughout the document	1
• Updated block diagram	5
• Added TMS320F2800137 topic	5
• Updated from *v4.00.0.00* to *v4.01.00.00*	41
• Updated Version 11 to Version 12	41
• Added note	41
• Added *F280013x* throughout publication	41
• Updated block diagram	56
• Updated block diagram	60
• Updated documentation support	88

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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

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

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