Sensorless Field Oriented Control of 3-Phase Induction Motors Using F2833x

Manish Bhardwaj

ABSTRACT

This application report presents a solution to control an AC induction motor using floating point TMS320F2833x microcontrollers. TMS320F2833x devices are part of the family of C2000™ microcontrollers which enable cost-effective design of intelligent controllers for three phase motors by reducing the system components and increase efficiency. With these devices it is possible to realize far more precise digital vector control algorithms like Field Orientated Control (FOC). This algorithm's implementation is discussed in this document using TI's Digital Motor Control (DMC) Library. The FOC algorithm maintains efficiency in a wide range of speeds and takes into consideration torque changes with transient phases by processing a dynamic model of the motor. Among the solutions proposed are ways to eliminate the phase current sensors and use an observer for speed sensorless control. The DMC Library uses TI's IQ math library, which supports both fixed and floating point maths. This makes migrating from floating to fixed point devices easy. A configuration for TMS320F2803x, which is a fixed point microcontroller, is available in the project to highlight this.

This application report covers the following:

- A theoretical background on field oriented motor control principle.
- Incremental build levels based on modular software blocks
- Experimental results

Contents

List of Figures

List of Tables

Submit Documentation Feedback

1   Introduction

The motor control industry is a strong, aggressive sector. To remain competitive new products must address several design constraints including cost reduction, power consumption reduction, power factor correction, and reduced EMI radiation. In order to meet these challenges advanced control algorithms are necessary. Embedded control technology allows both a high level of performance and system cost reduction to be achieved. According to market analysis, the majority of industrial motor applications use AC induction motors. The reasons for this are higher robustness, higher reliability, lower prices and higher efficiency (up to 80%) on comparison with other motor types. However, the use of induction motors is challenging because of its complex mathematical model, its non linear behavior during saturation and the electrical parameter oscillation which depends on the physical influence of temperature. These factors make the control of induction motor complex and call for use of a high performance control algorithms such as "vector control" and a powerful microcontroller to execute this algorithm.

During the last few decades the field of controlled electrical drives has undergone rapid expansion due mainly to the benefits of microcontrollers. These technological improvements have enabled the development of very effective AC drive control with lower power dissipation hardware and more accurate control structures. The electrical drive controls become more accurate in the sense that not only are the DC quantities controlled but also the three phase AC currents and voltages are managed by so-called vector controls. This document briefly describes the implementation of the most efficient form of a vector control scheme: the Field Orientated Control (FOC) method. The FOC method is based on three major points:

- Machine current and voltage space vectors
- The transformation of a three-phase speed- and time-dependent system into a two coordinate time invariant system
- Effective Space Vector Pulse Width Modulation pattern generation

These algorithms allow the control of an AC machine to acquire every advantage of DC machine control and free itself from mechanical commutation drawbacks. Furthermore, this control structure, by achieving a very accurate steady state and transient control, leads to high dynamic performance in terms of response times and power conversion.

2   Induction Motors

Induction motors derive their name from the way the rotor magnetic field is created. The rotating stator magnetic field induces currents in the short circuited rotor. These currents produce the rotor magnetic field, which interacts with the stator magnetic field, and produces torque, which is the useful mechanical output of the machine.

The three phase squirrel cage AC induction motor is the most widely used motor. The bars forming the conductors along the rotor axis are connected by a thick metal ring at the ends, resulting in a short circuit as shown in Figure 1. The sinusoidal stator phase currents fed in the stator coils create a magnetic field rotating at the speed of the stator frequency (ωs). The changing field induces a current in the cage conductors, which results in the creation of a second magnetic field around the rotor wires. As a consequence of the forces created by the interaction of these two fields, the rotor experiences a torque and starts rotating in the direction of the stator field.
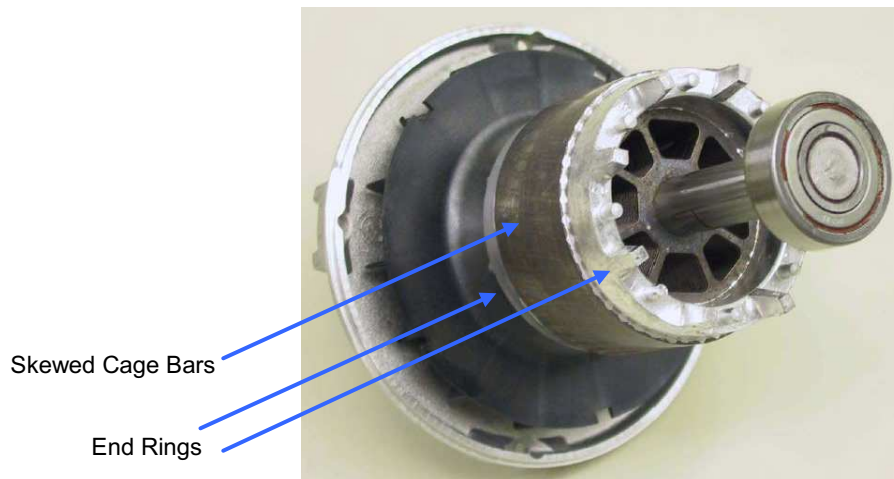

Skewed Cage Bars

End Rings

Figure 1. Induction Motor Rotor

As the rotor begins to speed up and approach the synchronous speed of the stator magnetic field, the relative speed between the rotor and the stator flux decreases, decreasing the induced voltage in the stator and reducing the energy converted to torque. This causes the torque production to drop off, and the motor will reach a steady state at a point where the load torque is matched with the motor torque. This point is an equilibrium reached depending on the instantaneous loading of the motor. In brief:

- Since the induction mechanism needs a relative difference between the motor speed and the stator flux speed, the induction motor rotates at a frequency near, but less than, that of the synchronous speed.
- This slip must be present, even when operating in a field-oriented control regime.
- The rotor in an induction motor is not externally excited. This means that there is no need for slip rings and brushes. This makes the induction motor robust, inexpensive and need less maintenance.
- Torque production is governed by the angle formed between the rotor and the stator magnetic fluxes.

In Figure 2 the rotor speed is denoted by Ω. Stator and rotor frequencies are linked by a parameter called the slip s, expressed in per unit as s = (ω_s - ω_r) / ω_s.
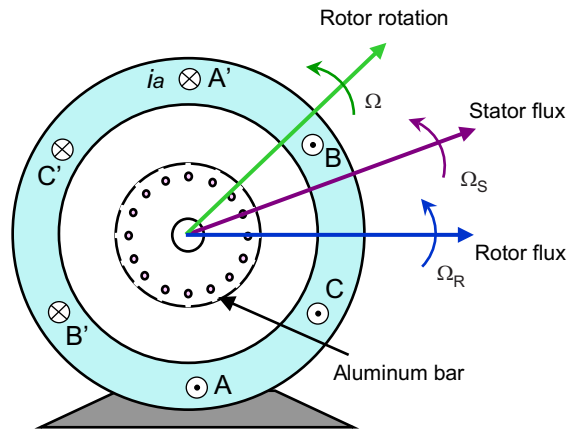
Figure 2. Squirrel Cage Rotor AC Induction Motor Cutaway View

- *Stator rotating field speed (rad / s)*: $\Omega_s = \dfrac{\omega_s}{p} \cdot \left\{ \begin{array}{l} \omega_s : \text{AC supply freq } (rad / s) \\ p : \text{stator poles pairs number} \end{array} \right\}$

- *Rotor rotating speed (rad / s)*: $\Omega = (1 - s)\Omega_s = (1 - s)\dfrac{\omega_s}{p}$

where s is called the "slip": it represents the difference between the synchronous frequency and the actual motor rotating speed.

3 Field Oriented Control

3.1 Introduction

A simple control such as the V/Hz strategy has limitations on the performance. To achieve better dynamic performance, a more complex control scheme needs to be applied, to control the induction motor. With the mathematical processing power offered by the microcontrollers, we can implement advanced control strategies, which use mathematical transformations in order to decouple the torque generation and the magnetization functions in an AC induction motor. Such decoupled torque and magnetization control is commonly called rotor flux oriented control, or simply Field Oriented Control (FOC).

3.2 The Main Philosophy Behind the FOC

In order to understand the spirit of the Field Oriented Control technique, let us start with an overview of the separately excited direct current (DC) Motor. In this type of motor, the excitation for the stator and rotor is independently controlled. An electrical study of the DC motor shows that the produced torque and the flux can be independently tuned. The strength of the field excitation (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.
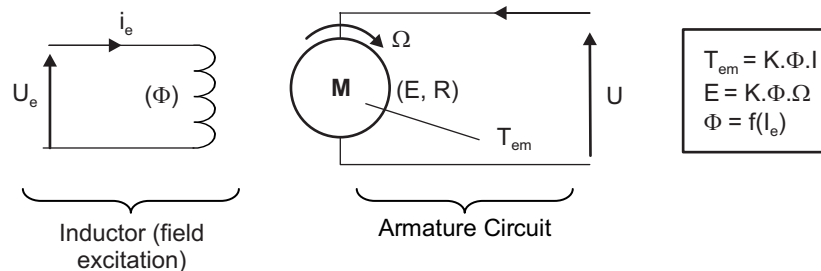

Figure 3. Separated Excitation DC Motor Model

Induction machines do not have the same key features as the DC motor. However, in both cases we have only one source that can be controlled which is the stator currents. On the synchronous machine, the rotor excitation is given by the permanent magnets mounted onto the shaft. On the synchronous motor, the only source of power and magnetic field is the stator phase voltage. Obviously, as opposed to the DC motor, flux and torque depend on each other.

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. The control technique goal is to (in a sense) imitate the DC motor's operation.

3.3 Why Field Oriented Control

Asynchronous machines have some natural limitations with a V/Hz control approach. FOC control bypasses these limitations by decoupling the effect of the torque and the magnetizing flux. With decoupled control of the magnetization, the torque-producing component of the stator flux can now be thought of as independent torque control. Now, decoupled control, at low speeds, the magnetization can be maintained at the proper level, and the torque can be controlled to regulate the speed.

To decouple the torque and flux, you must engage several mathematical transforms, wherein 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.

3.4 Technical Background

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 coordinates) 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:

$$m \propto \Psi_R \, i_{Sq}$$

By maintaining the amplitude of the rotor flux (φR) 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.

3.5 Space Vector Definition and Projection

The three-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 $\bar{i}_s$ is defined by:

$$\bar{i}_S = i_a + \alpha i_b + \alpha^2 i_c$$

where $\alpha = e^{j\frac{2}{3}\Pi}$ and $\alpha^2 = e^{j\frac{4}{3}\Pi}$, represent the spatial operators. The following diagram shows the stator current complex space vector:
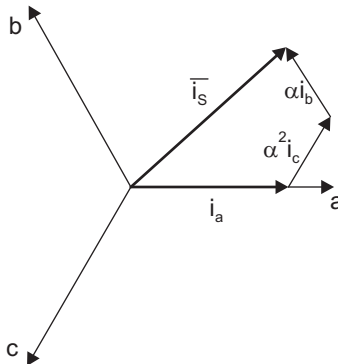

Figure 4. Stator Current Space Vector and Its Component in (a,b,c)

where (a,b,c) are the three phase system axes. This current space vector depicts the three phase sinusoidal system but must still be transformed into a two time invariant coordinate system. This transformation can be split into two steps:

- (a,b,c) → *(α, β)* (the Clarke transformation), which outputs a two coordinate time variant system
- (a, β) → (d,q) (the Park transformation), which outputs a two coordinate time invariant system

3.6 The (a,b,c) → (α, β) Projection (Clarke transformation)

The space vector can be reported in another reference frame with only two orthogonal axis called *(α, β)*. Assuming that the axis a and the axis *α* are in the same direction, you have the following vector diagram:
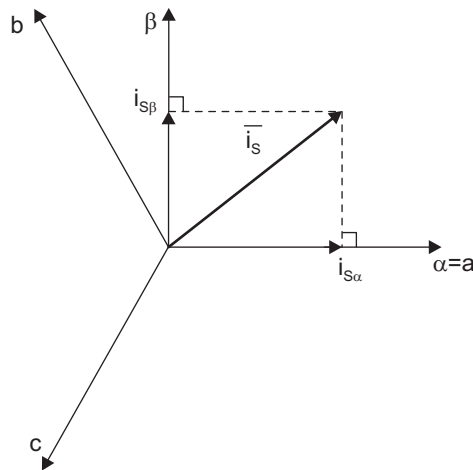

Figure 5. Stator Current Space Vector and Its Component in the Stationary Reference Frame

The projection that modifies the three phase system into the *(α, β)* two dimension orthogonal system is:

$$\begin{cases} i_{s\alpha} = i_a \\ i_{s\beta} = \dfrac{1}{\sqrt{3}} i_a + \dfrac{2}{\sqrt{3}} i_b \end{cases}$$

The two phase *(α, β)* currents still depend on time and speed.

3.7 The (α, β) → (d.q) Projection (Park Transformation)

This is the most important transformation in the FOC. In fact, this projection modifies a two phase orthogonal system *(α, β)* into the d,q rotating reference frame. If we consider the d axis aligned with the rotor flux, Figure 6 shows, for the current vector, the relationship from the two reference frame:
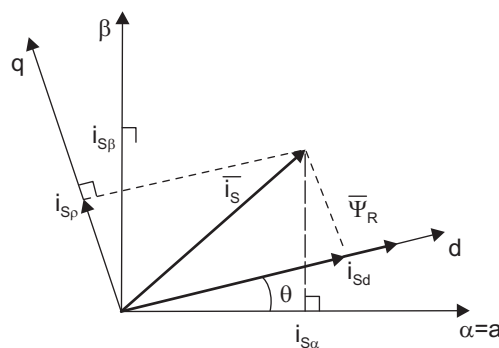

Figure 6. Stator Current Space Vector and Its Component in *(α, β)* and in the d,q Rotating Reference Frame

where θ is the rotor flux position. The flux and torque components of the current vector are determined by the following equations:

$$\begin{cases} i_{sd} = i_{s\alpha} \cos\theta + i_{s\beta} \sin\theta \\ i_{sq} = -i_{s\alpha} \sin\theta + i_{s\beta} \cos\theta \end{cases}$$

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.

3.8   The Basic Scheme for the FOC

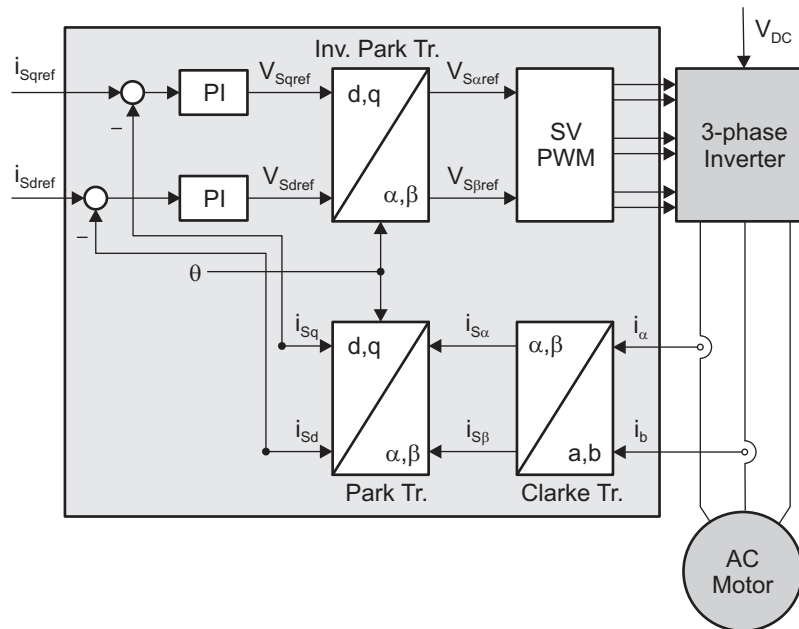Figure 7 summarizes the basic scheme of torque control with FOC:

Figure 7. Basic Scheme of FOC for ACI Motor

Two motor phase currents are measured. These measurements feed the Clarke transformation module. The outputs of this projection are designated $i_{sα}$ and $i_{sβ}$. 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) and i_{sqref} (the torque reference). At this point, this control structure shows an interesting advantage: you can control either synchronous or induction machines by simply changing the flux reference and obtaining rotor flux position. In a synchronous permanent magnet a motor, the rotor flux is fixed (determined by the magnets) and there is no need to create additional flux. Therefore, when controlling a PMSM, i_{sdref} should be set to zero. Since induction motors need a rotor flux creation in order 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 V_{sdref} and V_{sqref}. They are applied to the inverse Park transformation. The outputs of this projection are $V_{sαref}$ and $V_{sβref}$, 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 pulse width modulation (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 considerations are made in Section 3.9.

3.9 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 will not be aligned with d-axis, and i_{sd} and i_{sq}, flux and torque components of the stator current, will be incorrect. Figure 8 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.
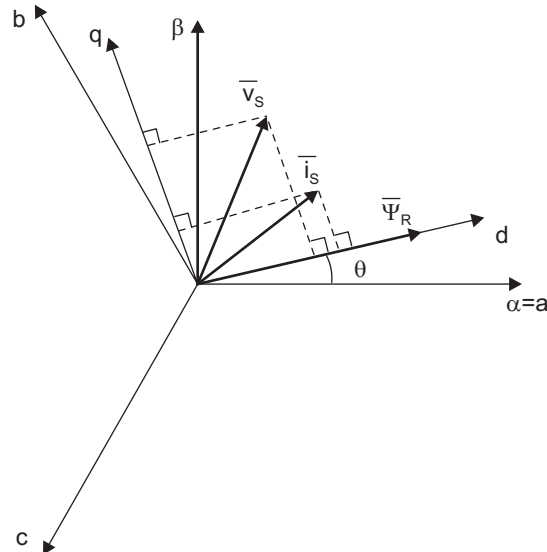

Figure 8. Current, Voltage and Rotor Flux Space Vectors in the d,q Rotating Reference Frame and Their Relationship With a,b,c and *(α, β)* Stationary Reference Frame

The measure of the rotor flux position is different if we consider synchronous or induction 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 induction 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 to use the current model which needs two equations of the motor model in d,q reference frame.

Theoretically, the field oriented control for an induction motor drive can be mainly categorized into two types: indirect and direct schemes. The field to be oriented could be a rotor, stator, or airgap flux linkage. In the indirect field oriented control, the slip estimation with measured or estimated rotor speed is required in order to compute the synchronous speed. There is no flux estimation appearing in the system. For the direct scheme, the synchronous speed is computed basing on the flux angle which is available from flux estimator or flux sensors (Hall effects). In this implementing system, the direct (rotor) flux oriented control system with flux and open-loop speed estimators is described. The key module of this system is the flux estimator. The overall block diagram of this project is depicted in Figure 9.
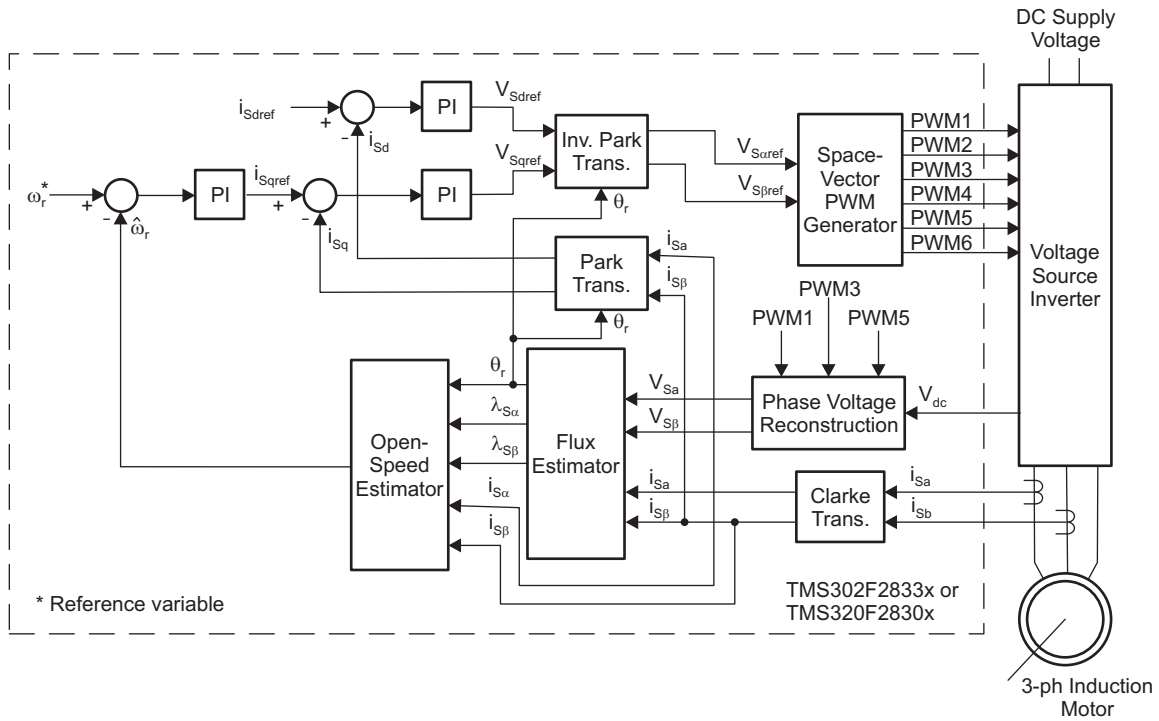
Figure 9. Overall Block Diagram

4   Benefits of 32-Bit C2000 Controllers for Digital Motor Control (DMC)

The C2000 family of devices posses the desired computation power to execute complex control algorithms along with the right mix of peripherals to interface with the various components of the DMC hardware like the ADC, ePWM, QEP, eCAP, and so forth. These peripherals have all the necessary hooks for implementing systems which meet safety requirements, like the trip zones for PWMs and comparators. Along with this the C2000 ecosystem of software (libraries and application software) and hardware (application kits) help in reducing the time and effort needed to develop a Digital Motor Control solution. The DMC Library provides configurable blocks that can be reused to implement new control strategies. IQMath Library enables easy migration from floating point algorithms to fixed point thus accelerating the development cycle.

Therefore, with the C2000 family of devices it is easy and quick to implement complex control algorithms (sensored and sensorless) for motor control. The use of C2000 devices and advanced control schemes provides the following system improvements:

- Favors system cost reduction by an efficient control in all speed range implying right dimensioning of power device circuits.
- Advanced control algorithms reduce torque ripple, thus resulting in lower vibration and longer life time of the motor.
- Advanced control algorithms reduce harmonics generated by the inverter thus reducing filter cost.
- Sensorless algorithms eliminate the need for speed or position sensor.
- Fewer look-up tables which reduces the amount of memory required.
- Real-time generation of smooth near-optimal reference profiles and move trajectories results in better-performance.
- Generation of high-resolution PWMs is possible with the use of ePWM peripheral for controlling the power switching inverters.
- Single chip control system.

For advanced controls, C2000 controllers:

- Enable control of multi-variable and complex systems using modern intelligent methods such as neural networks and fuzzy logic
- Perform adaptive control. C2000 controllers have the speed capabilities to concurrently monitor and control the system. A dynamic control algorithm adapts itself in real time to variations in system behavior.
- Perform parameter identification for sensorless control algorithms, self commissioning, online parameter estimation update
- Perform advanced torque ripple and acoustic noise reduction.
- Provide diagnostic monitoring with spectrum analysis. By observing the frequency spectrum of mechanical vibrations, failure modes can be predicted in early stages.
- Produce sharp-cut-off notch filters that eliminate narrow-band mechanical resonance. Notch filters remove energy that would otherwise excite resonant modes and possibly make the system unstable.

5   TI Literature and Digital Motor Control (DMC) Library

The Digital Motor Control (DMC) library is composed of functions represented as blocks. These blocks are categorized as:

- Transforms and Estimators (Clarke, Park, Sliding Mode Observer, Phase Voltage Calculation, and Resolver, Flux, and Speed Calculators and Estimators)
- Control (Signal Generation, PID, BEMF Commutation, Space Vector Generation)
- Peripheral Drivers (PWM abstraction for multiple topologies and techniques, ADC drivers, and motor sensor interfaces)

Each block is a modular software macro and separately documented with source code, use, and technical theory. For the source codes and explanation of macro blocks, install the controlSUITE software from http://www.ti.com/controlsuite. After installing the software, check the following folders for the source codes and explanations for macro blocks:

- C:\TI\controlSUITE\libs\app_libs\motor_control\math_blocks\v4.0
- C:\TI\controlSUITE\libs\app_libs\motor_control\drivers\f2833x_v2.0
- C:\TI\controlSUITE\libs\app_libs\motor_control\drivers\f2803x_v2.0

These modules allow users to quickly build, or customize, their own systems. The library supports the three motor types: ACI, BLDC, PMSM, and comprises both peripheral dependent (software drivers) and target dependent modules.

The DMC library components have been used by TI to provide system examples. At initialization all DMC Library variables are defined and inter-connected. At run-time the macro functions are called in order. Each system is built using an incremental build approach, which allows some sections of the code to be built at a time, so that the developer can verify each section of their application one step at a time. This is critical in real-time control applications where so many different variables can affect the system and many different motor parameters need to be tuned.

NOTE: TI DMC modules are written in form of macros for optimization purposes. For more details, see *Optimizing Digital Motor Control (DMC) Libraries* (SPRAAK2). The macros are defined in the header files. The user can open the respective header file and change the macro definition, if needed. In the macro definitions, there should be a backslash "\" at the end of each line as shown in Example 1, which means that the code continues in the next line. Any character including invisible ones like "space" or "tab" after the backslash will cause compilation error. Therefore, make sure that the backslash is the last character in the line. In terms of code development, the macros are almost identical to C function, and the user can easily convert the macro definition to a C functions.

Example 1. A Typical DMC Macro Definition

```
#define PARK_MACRO(v)                                        \
                                                             \
    v.Ds = _IQmpy(v.Alpha,v.Cosine) + _IQmpy(v.Beta,v.Sine); \
    v.Qs = _IQmpy(v.Beta,v.Cosine)  - _IQmpy(v.Alpha,v.Sine);
```

5.1   System Overview

This document describes the "C" real-time control framework used to demonstrate the sensorless field oriented control of induction motors. The "C" framework is designed to run both on TMS320C2833x- and TMS320C2803x-based controllers controllers on Code Composer Studio™ software. The framework uses the following modules.

For details and the theoretical background of each macro, see the PDF documents in C:\TI\controlSUITE\libs\app_libs\motor_control.

Macro Names	Explanation
CLARKE	Clarke Transformation
PARK / IPARK	Park and Inverse Park Transformation
PI	PI Regulators
RC	Ramp Controller (slew rate limiter)
RG	Ramp / Sawtooth Generator
QEP / CAP	QEP and CAP Drives (optional for speed loop tuning with a speed sensor)
SPEED_PR	Speed Measurement (based on sensor signal frequency)
SPEED_FR	Speed Measurement (based on sensor signal period)
ACI_SE / ACI_FE	Flux and Speed Estimators for Sensorless Applications
SVGEN	Space Vector PWM with Quadrature Control (includes IClarke Transformation)
PHASEVOLT	Phase Voltage Calculator
PWM / PWMDAC	PWM and PWMDAC Drives

In this system, the sensorless Direct Field Oriented Control (DFOC) of Induction Motor will be experimented with and will explore the performance of the speed controller. The induction motor is driven by a conventional voltage-source inverter. TMS320x2833x or TMS320x2803x control cards can be used to generate three pulse width modulation (PWM) signals. The motor is driven by an integrated power module by means of space vector PWM technique. Two phase currents of HVPM motor (ia and ib) are measured from the inverter and sent to the controller via two analog-to-digital converters (ADCs). In addition, the DC-bus voltage in the inverter is measured and sent to the TMS320x2833x or TMS320x2803x via an ADC. This DC-bus voltage is necessary to calculate the three phase voltages when the switching functions are known.

HVACI_Sensorless project has the following properties:

C Framework		
System Name	Program Memory Usage 2803x	Data Memory Usage 2803x [1]
HVACI_Sensorless	5079 words [2]	1554 words

[1] Excluding the stack size
[2] Excluding "IQmath" Look-up Tables

CPU Utilization		
	Without CLA	With CLA
Total Number of Cycles	1133 [1]	873 [1]
CPU Utilization @ 60 MHz	18.8%	14.5%
CPU Utilization @ 40 MHz	28.3%	21.8%

[1] At 10 kHz ISR frequency. Debug macros excluded (in other words, PWMDAC, Datalog and RG). IQSin and Cos tables used in macros.

System Features	
Development and Emulation	Code Composer Studio V4.1 (or above) with real-time debugging
Target Controller	TMS320F2833x or TMS320F2803x
PWM Frequency	10 kHz PWM (Default), 60 kHz PWMDAC
PWM Mode	Symmetrical with a programmable dead band
Interrupts	ADC, end of conversion – Implements 10 kHz ISR execution rate
Peripherals Used	PWM 1 / 2 / 3 for motor control
	PWM 6A, 6B, 7A and 7B for DAC outputs (7A and 7B only 2803x)
	QEP1 A,B, I or CAP1 (optional for tuning the speed loop)
	ADC A7 for DC Bus voltage sensing, A1 and B1 for phase current sensing

The overall system implementing a 3-ph induction motor control is depicted in Figure 10. The induction motor is driven by the conventional voltage-source inverter. The TMS320F2833x or TMS320F2803x can be used to generate the six pulse width modulation (PWM) signals using a space vector PWM technique, for six power switching devices in the inverter. Two input currents of the HVPM motor (ia and ib) and DC bus voltage are measured from the inverter and they are sent to the controller via two analog-to-digital converters (ADCs). The DC-bus voltage is necessary in order to calculate three phase voltages of HVPM motor when the switching functions are known.
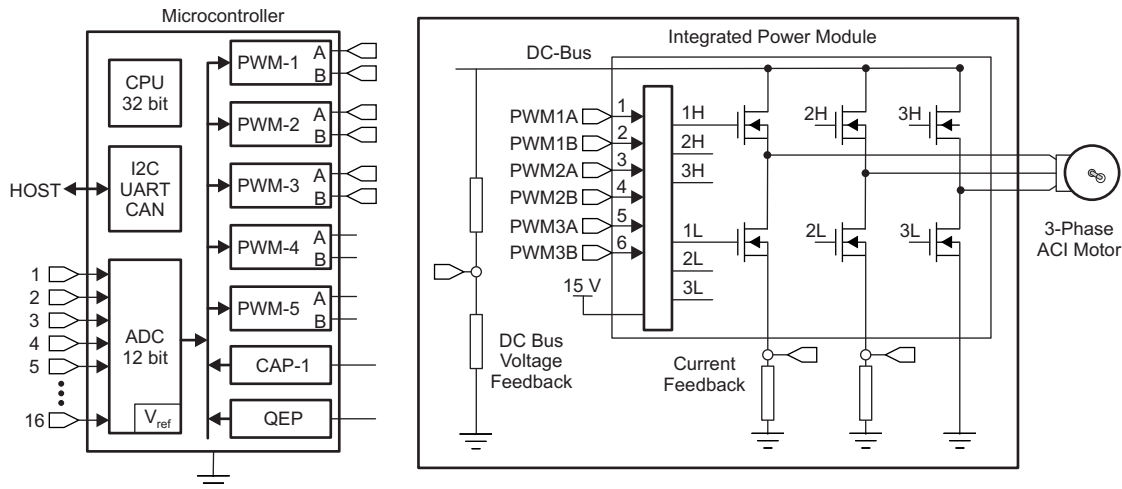
Figure 10. A 3-ph Induction Motor Drive Implementation

The software flow is described in Figure 11.
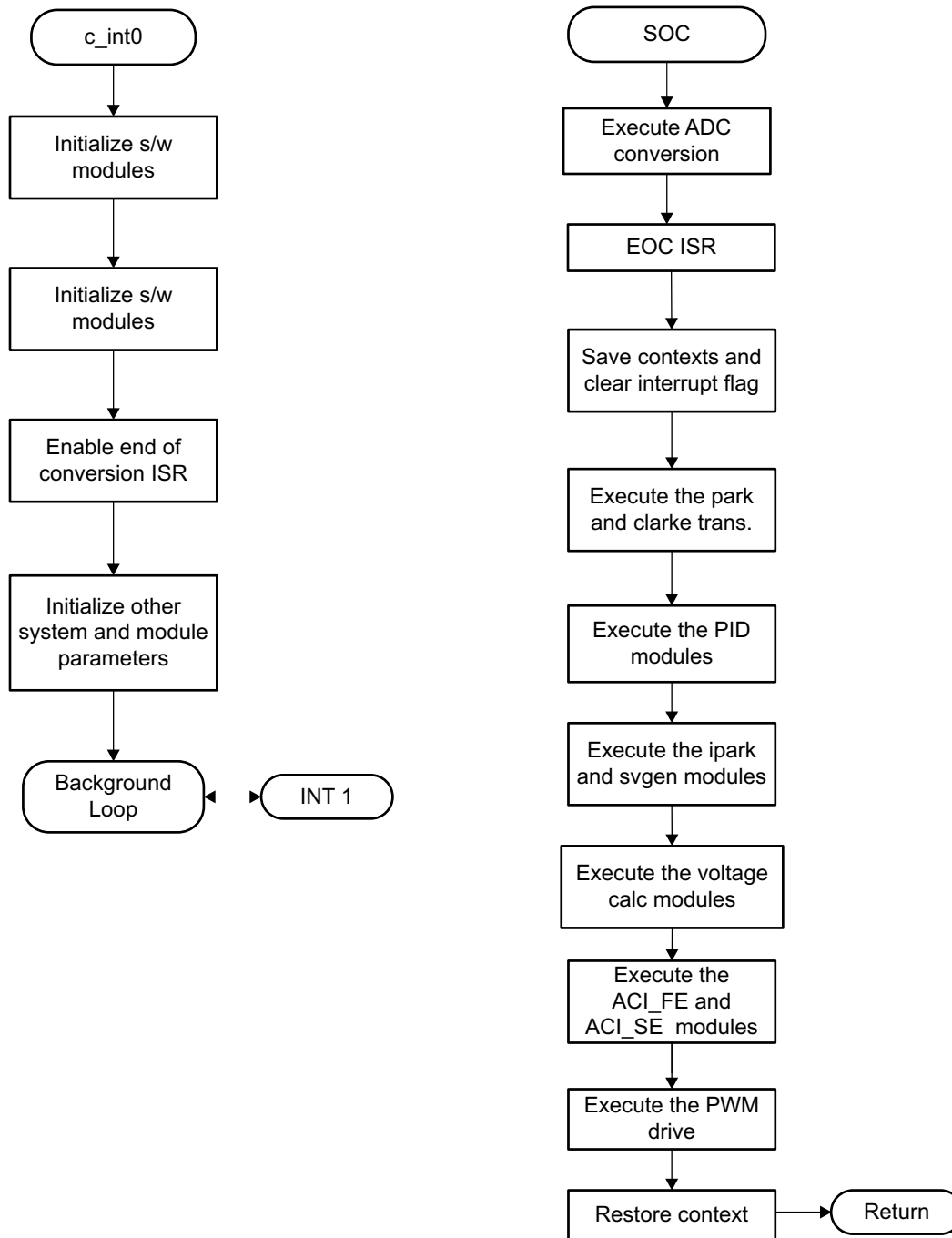
Figure 11. Software Flow

6 Hardware Configuration (HVDMC R1.1 Kit)

For an overview of the kit's hardware and steps on how to set up this kit, see the *HVMotorCtrl+PFC How to Run Guide* located at: www.ti.com/controlsuite and choose the HVMotorKit installation.

The guide will be installed at: C:\TI\controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0\~Docs.

Some of the hardware setup instructions are captured below for quick reference.

6.1 Hardware Setup Instructions

1. Open the Lid of the HV Kit
2. Install the Jumpers [Main]-J3, J4 and J5, J9 for 3.3V, 5V and 15V power rails and JTAG reset line.
3. Unpack the DIMM style controlCARD and place it in the connector slot of [Main]-J1. Push down vertically using even pressure from both ends of the card until the clips snap and lock. (to remove the card simply spread open the retaining clip with thumbs)
4. Connect a USB cable to connector [M3]-JP1.

   This step enables an isolated JTAG emulation to the C2000 device. [M3]-LD1 should turn on. Make sure [M3]-J5 is not populated. If the included Code Composer Studio is installed, the drivers for the onboard JTAG emulation will automatically be installed. If a Windows® installation window appears, try to automatically install drivers from those already on your computer. The emulation drivers are found at http://www.ftdichip.com/Drivers/D2XX.htm. The correct driver is the one listed to support the FT2232.
5. If a third party JTAG emulator is used, connect the JTAG header to [M3]-J2 and additionally [M3]-J5 needs to be populated to put the onboard JTAG chip in reset.
6. Ensure that [M6]-SW1 is in the "Off" position.
7. Connect 15-V DC power supply to [M6]-JP1.
8. Turn on [M6]-SW1. Now [M6]-LD1 should turn on. Notice the control card LED will light up as well indicating the control card is receiving power from the board.
9. Note that the motor should be connected to the [M5]-TB3 terminals after you finish with the first incremental build step.
10. Note the DC Bus power should only be applied during incremental build levels when instructed to do so. The two options to get DC Bus power are:
    - To use DC power supply, set the power supply output to zero and connect [Main]-BS5 and BS6 to DC power supply and ground respectively.
    - To use AC Main Power, Connect [Main]-BS1 and BS5 to each other using banana plug cord. Now connect one end of the AC power cord to [Main]-P1. The other end needs to be connected to output of a variac. Ensure that the variac output is set to zero and is connected to the wall supply through an isolator.

Since the motor is rated at 220 V, the motor properly runs only at a certain speed and torque range without saturating the PID regulators in the control loop when the DC bus is fed from 110-V AC entry. As an option, the user can run the PFC on HV DMC drive platform as boost converter to increase the DC bus voltage level or directly connect a DC power supply.

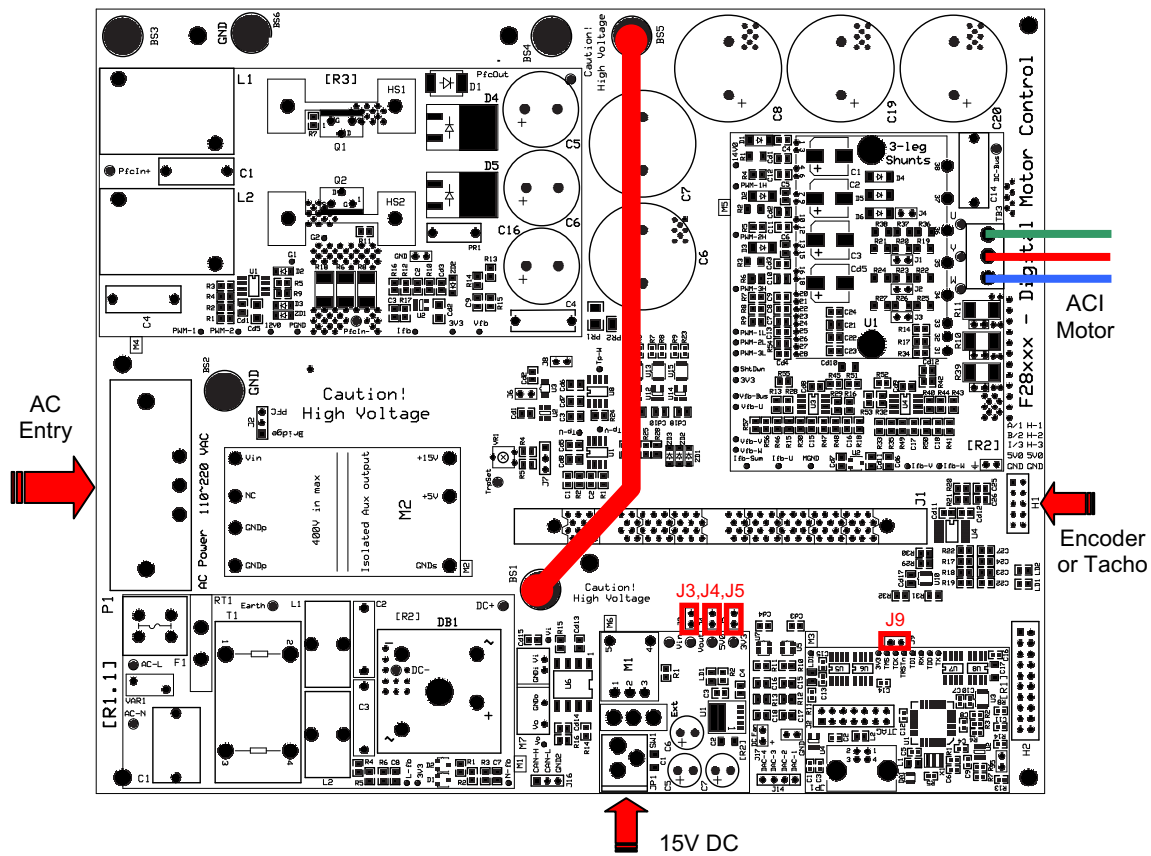For reference, the following figures show the jumper and connectors that need to be connected for this lab.

Figure 12. Using AC Power to Generate DC Bus Power

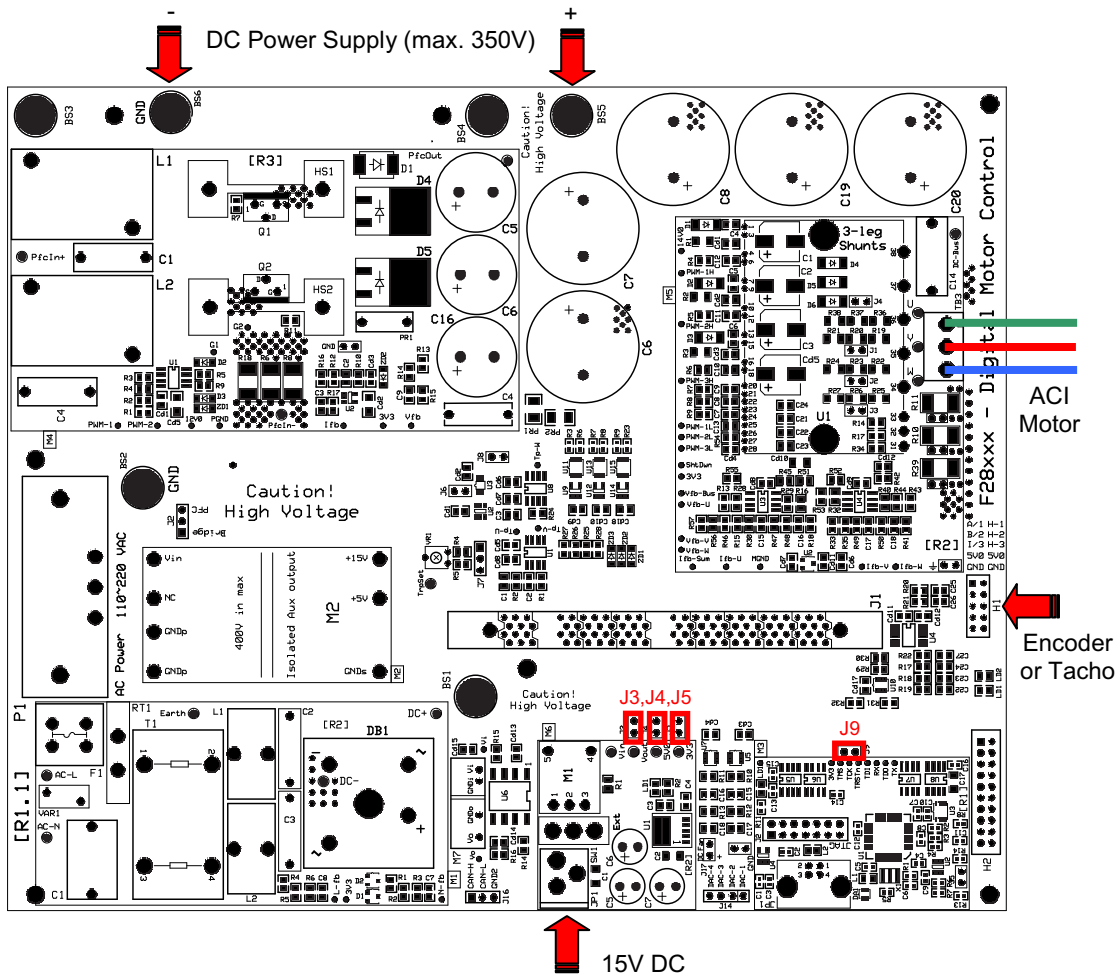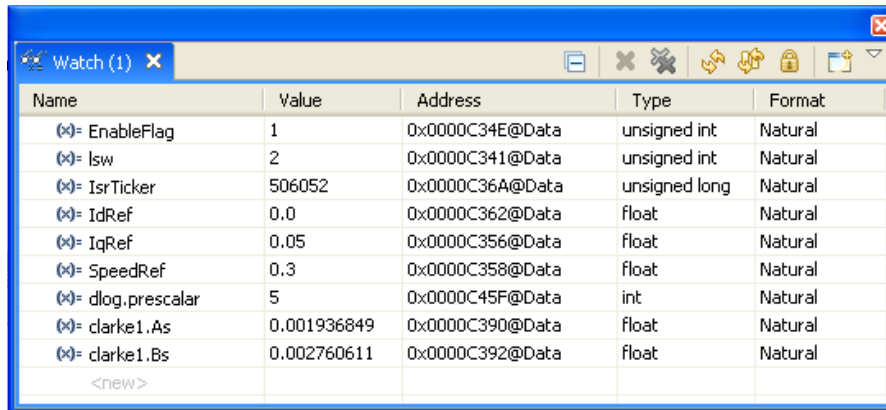| CAUTION |
| --- |
| The inverter bus capacitors remain charged for a long time after the high power line supply is switched off or disconnected. Proceed with caution! |

Figure 13. Using External DC Power Supply to Generate DC-Bus for the Inverter

---

CAUTION

The inverter bus capacitors remain charged for a long time after the high power line supply is switched off or disconnected. Proceed with caution!

6.2 Software Setup Instructions to Run HVACI_Sensorless Project

For more information, see the *Software Setup for HVMotorCtrl+PFC Kit Projects* section in the *HVMotorCtrl+PFC Kit How to Run Guide* in:

```
C:\TI\controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0\~Docs
```

1. Select the HVACI_Sensorless as the active project.
2. Verify that the build level is set to 1.
3. Right-click on the project name and select "Rebuild Project".
4. Once the build completes, launch a debug session to load the code into the controller.
5. Open a watch window and add the critical variables as shown in Figure 14 and select the appropriate Q format for them.

Figure 14. Watch Window Variables

6. Set up time graph windows by importing Graph1.graphProp and Graph2.graphProp from:
   `C:\TI\ControlSUITE\developement_kits\HVMotorCtrl+PfcKit_v2.0\ACI_Sensorless_`
   `2833x.`

7. Click on Continuous Refresh button on the top left corner of the graph tab to enable periodic
   capture of data from the microcontroller.

7 Incremental System Build for ACI Sensorless Project

The system is gradually built up so the final system can be confidently operated. Six phases of the incremental system build are designed to verify the major software modules used in the system. Table 1 summarizes the modules testing and using in each incremental system build.

Table 1. Testing Modules in Each Incremental System Build [(1)]

Software Module	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6
PWMDAC_MACRO	√	√	√	√	√	√
RC_MACRO	√	√	√	√		
RG_MACRO	√	√	√	√		
IPARK_MACRO	√√	√	√	√	√	√
SVGEN_MACRO	√√	√	√	√	√	√
PWM_MACRO	√√	√	√	√	√	√
CLARKE_MACRO		√√	√	√	√	√
PARK_MACRO		√√	√	√	√	√
PHASEVOLT_MACRO		√√	√	√	√	√
CAP_MACRO			√√	√	√	
SPEED_PR_MACRO			√√	√	√	
QEP_MACRO			√√	√	√	
PEED_FR_MACRO			√√	√	√	
PI_MACRO (IQ)			√√	√	√	√
PI_MACRO (ID)			√√	√	√	√
ACI_FE				√√	√	√
ACI_SE				√√	√	√
PI_MACRO (SPD)					√√	√√

[(1)] The symbol √ means this module is using and the symbol √√ means this module is testing in this phase.

7.1 Level 1 Incremental Build

Keep the motor disconnected at this step. Assuming the load and build steps described in the *HVMotorCtrl+PFC Kit How To Run Guide* completed successfully, this section describes the steps for a "minimum" system check-out, which confirms the operation of system interrupt, the peripheral and target independent I_PARK_MACRO (inverse park transformation) and SVGEN_MACRO (space vector generator) modules, the peripheral dependent PWM_MACRO (PWM initializations and update) modules.

1. Open HVACI_Sensorless-Settings.h.
2. Select level 1 incremental build option by setting the BUILDLEVEL to LEVEL1 (#define BUILDLEVEL LEVEL1).
3. Right-click on the project name and click Rebuild Project.
4. When the build is complete:
   (a) Click on the debug button
   (b) Reset the CPU and restart.
   (c) Enable real-time mode and run.
   (d) Set the "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will now continue to increase, as seen in the watch window, to confirm the system interrupt is properly working.

In the software, the key variables to adjust are:

- SpeedRef (Q24): for changing the rotor speed in per-unit.
- VdTesting (Q24): for changing the d-qxis voltage in per-unit.
- VqTesting (Q24): for changing the q-axis voltage in per-unit.

7.2 Level 1A (SVGEN_MACRO Test)

In this level we would test the Space Vector Generator Macro (SVGEN_MACRO). The SpeedRef value is specified to the RG_MACRO module via RC_MACRO module. The IPARK_MACRO module is generating the outputs to the SVGEN_MACRO module. Three outputs from SVGEN_MACRO module are monitored via the graph window, as shown in Figure 15, where Ta, Tb, and Tc waveform are 120° apart from each other. Specifically, Tb lags Ta by 120° and Tc leads Ta by 120°. Check the PWM test points on the board to observe PWM pulses (PWM-1H to 3H and PWM-1L to 3L) and ensure that the PWM module is running properly.
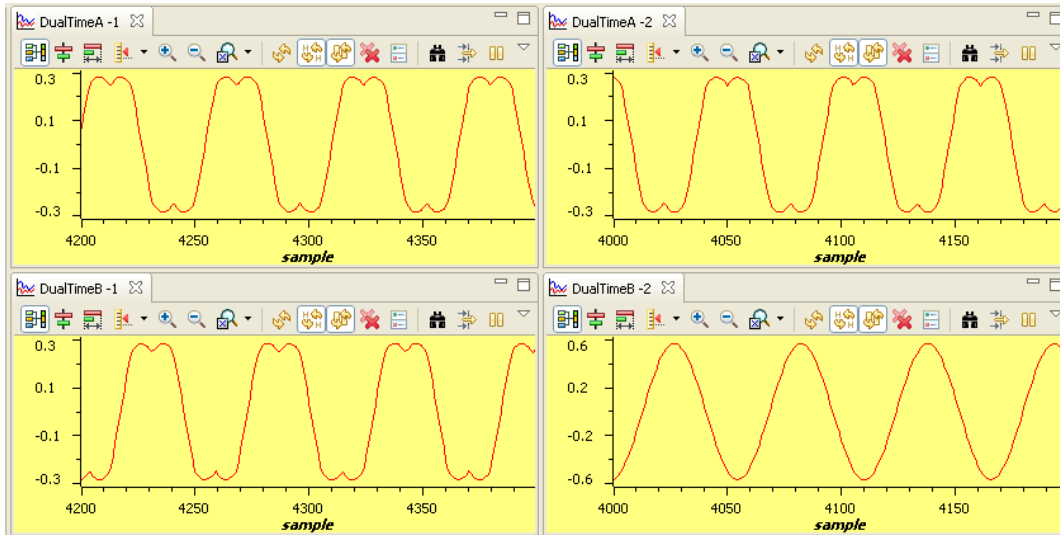
Figure 15. SVGEN Duty Cycle Outputs Ta, Tb, Tc and Tb-Tc

7.3 Level 1B (Testing The PWMDAC Macro)

To monitor internal signal values in real time, PWM DACs are very useful tools. Present on the HV DMC board are PWM DACs which use an external low pass filter to generate the waveforms ([Main]-J14, DAC-1 to 4). A simple first-order low-pass filter RC circuit filters out the high frequency components.

The selection of R and C value (or the time constant, τ) is based on the cut-off frequency (f_c), for this type of filter; the relation is as follows:

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

For example, R = 1.8 kΩ and C = 100 nF results in f_c = 884.2 Hz. This cut-off frequency must be below the PWM frequency. Use the formula above to customize low-pass filters used for the signal being monitored.

The DAC circuit low-pass filters ([Main]-R10 to13 and [Main]-C15 to18) are shipped with 2.2 kΩ and 220 nF on the board. For more details, see *Using PWM Output as a Digital-to-Analog Converter on a TMS320F280x Digital Signal Controller* (SPRAA88).
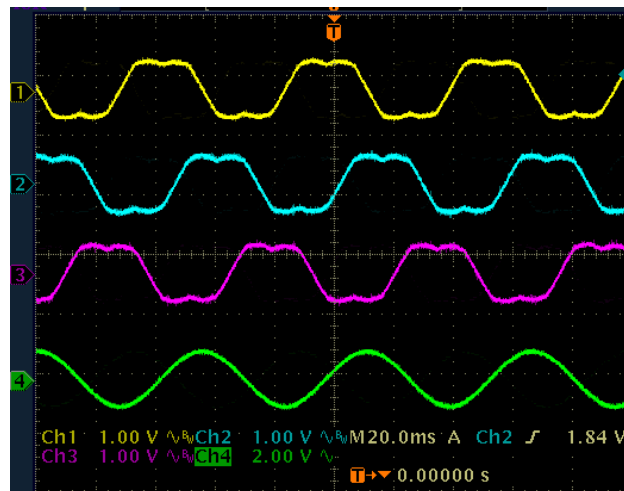
Figure 16. DAC-1-4 Outputs Showing Ta, Tb, Tc and Tb-Tc Waveforms

7.4   Level 1C (PWM_MACRO and INVERTER Test)

After verifying the SVGEN_MACRO module in Level 1A, the PWM_MACRO software module and the 3-phase inverter hardware are tested by looking at the low-pass filter outputs. For this purpose, if using the external DC power supply gradually increase the DC bus voltage and check the Vfb-U, V and W test points using an oscilloscope or if using AC power entry slowly change the variac to generate the DC bus voltage. Once the DC bus voltage is greater than 15 V to 20 V, you will start observing the inverter phase voltage dividers and waveform monitoring filters (Vfb-U, Vfb-V, Vfb-W) enable the generation of the waveform, which ensures that the inverter is working appropriately. Note that the default RC values are optimized for AC motor state observers employing phase voltages.

CAUTION

After verifying the SVGEN_CLA_MACRO module:

1. Reduce the DC bus voltage.

2. Take the controller out of real-time mode (disable).

3. Reset the processor ![icon] (for more details, see the *HVMotorCtrl+PFC Kit How To Run Guide*).
Note that after each test, this process needs to be repeated for safety purposes. Also note that improper shutdown might halt the PWMs at some certain states where high currents can be drawn, therefore, caution needs to be taken while doing these experiments.
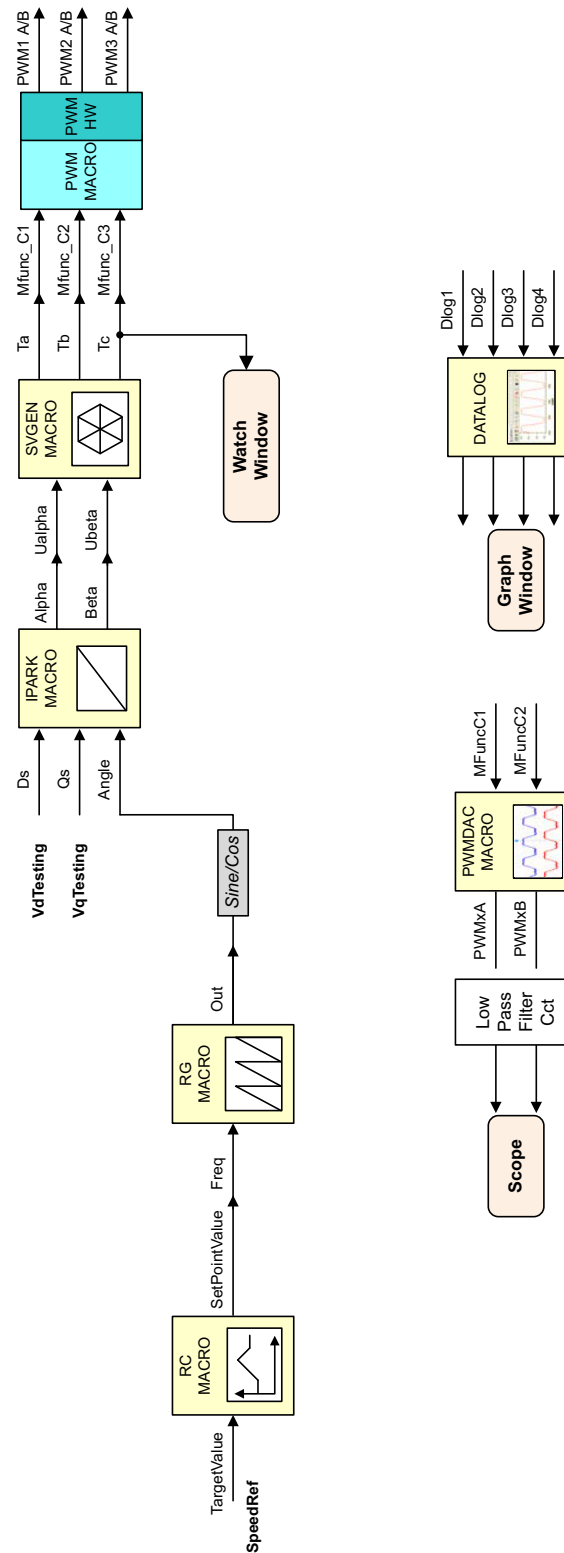

Figure 17. Level 1 — Incremental System Build Block Diagram

Level 1 verifies the target independent modules, duty cycles and PWM updates. The motor is disconnected at this level.

7.5 Level 2 - Incremental Build

Assuming the Level1 incremental build 1 is completed successfully, this section verifies the analog-to-digital conversion, Clarke/Park transformations and phase voltage calculations. Now the motor can be connected to the HVDMC board.

In the software, the key variables to adjust are:

- SpeedRef(Q24): for changing the rotor speed in per-unit.
- VdTesting(Q24): for changing the d-axis voltage in per-unit.
- VqTesting(Q24): for changing the q-axis voltage in per-unit.

1. Open HVACI_Sensorless-Settings.h and select level 2 incremental build option by setting the BUILDLEVEL to LEVEL2 (#define BUILDLEVEL LEVEL2) and save the file.
2. Right-click on the project name and click Rebuild Project.
3. When the build is complete:

    (a) Click on the debug button.

    (b) Reset the CPU and restart.

    (c) Enable real-time mode and run.

    (d) Set "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

7.6 Level 2A – Testing the Phase Voltage Module

In this part, the phase voltage calculation module, PHASEVOLT_MACRO, will be tested. Gradually increase the DC bus voltage. The waveforms in Figure 18 should be observed.
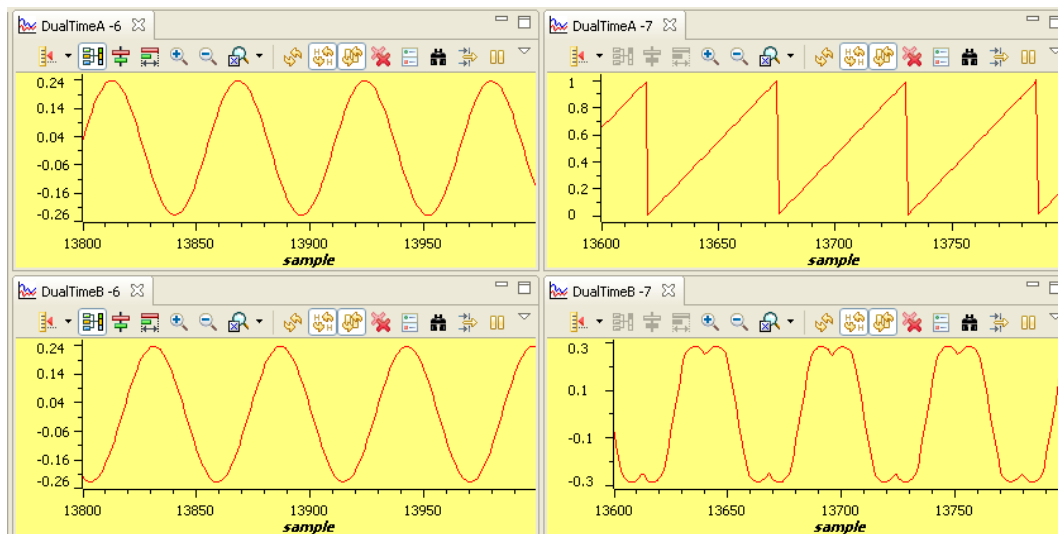
Figure 18. Calculated Phase A and B Voltages By Volt1 Module, rg1.Out, and svgen_dq1.Ta

The VphaseA, VphaseB, and VphaseC waveforms should be 120° apart from each other. Specifically, VphaseB lags VphaseA by120° (Figure 18) and VphaseC leads VphaseA by 120.

Alternatively, users can modify the datalog setting to verify the following relation between the signals:

- The Valpha waveform should be the same as the VphaseA waveform.
- The Valpha waveform should be leading the Vbeta waveform by 90° at the same magnitude.

Note that the open loop experiments are meant to test the ADCs, inverter stage, software modules, and so forth. Running the motor under load or at various operating points is not recommended.

7.7 *Phase 2B – Testing the Clarke Module*

In this part the Clarke module will be tested. The three measured line currents are transformed to two phase dq currents in a stationary reference frame. The outputs of this module can be checked from the graph window.

Verify the following:

- The clarke1.Alpha waveform leads the clarke1.Beta waveform by 90° at the same magnitude.
- The clarke1.Alpha waveform is the same as the clarke1.As waveform.

The measured line current must be lagging with the reconstructing phase voltage because of the nature of the ACI motor.

As mentioned in the previous section, three input switching functions may not be functioning correctly (see this module documentation for details). This can be easily checked as follows (Note: You may need to modify datalog settings and graph windows settings to observe these signals):

- The clarke1.Alpha waveform lags the Valpha waveform at an angle by nature of the reactive load of Induction motor.
- The clarke1.Beta waveform lags the Vbeta waveform at the same angle.
- If the clarke1.Alpha and Valpha or clarke1.Beta and Vbeta waveforms in the previous step are not truly affecting the lagging relationship, then set OutofPhase to 1 at the beginning of the PHASEVOLT_MACRO module.
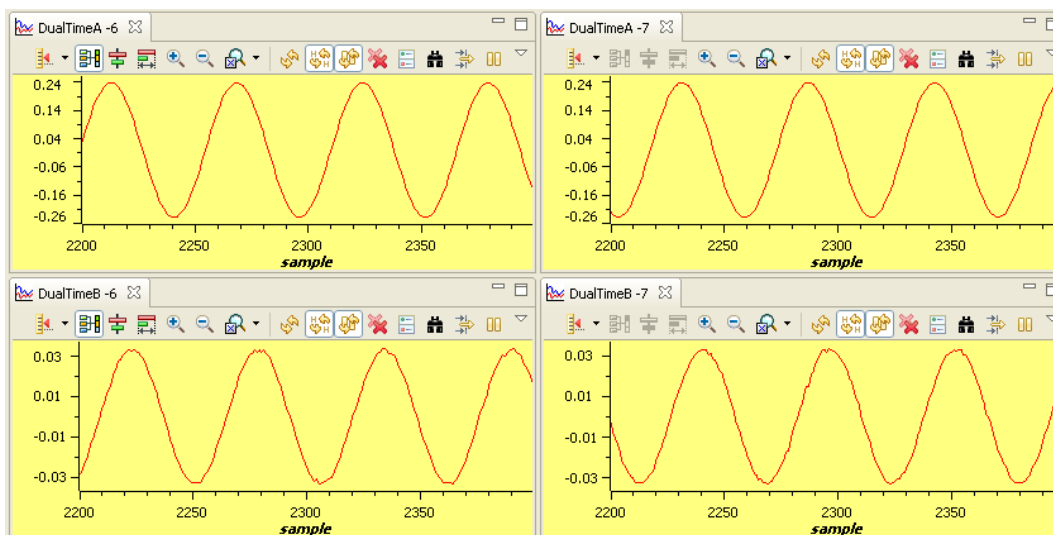
Figure 19. The Waveforms of Phase A Voltage and Current and Phase B Voltage and Current

Since the low side current measurement technique is used employing shunt resistors on inverter phase legs, the phase current waveforms observed from current test points ([M5]-Ifb-U, and [M5]-Ifb-V) are composed of pulses, as shown in Figure 20.
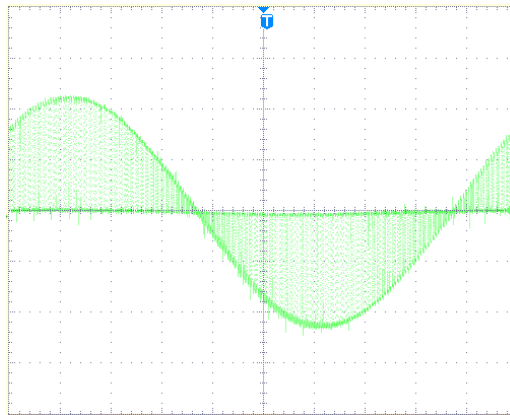
Figure 20. Amplified Phase A Current

7.8 Level 2C – Adjusting PI Limits

Note that the vectorial sum of d-q PI outputs should be less than 1.0 which refers to maximum duty cycle for SVGEN macro. Another duty cycle limiting factor is the current sense through shunt resistors which depends on hardware and software implementation. Depending on the application requirements 3, 2 or a single shunt resistor can be used for current waveform reconstruction. The higher number of shunt resistors allow higher duty cycle operation and better DC bus utilization.

Run the system with default VdTesting, VqTesting and SpeedRef and gradually increase VdTesting and VqTesting values. Meanwhile, watch the current waveforms in the graph window. Increase until you notice distorted current waveforms and write down the maximum allowed VdTesting and VqTesting values. Ensure that these values are consistent with expected d-q current component maximums while running the motor. After this build level, PI outputs will automatically generate the voltage reference and determine the PWM duty cycle depending on the d-q current demand, therefore set pi_id.Umax/min and pi_iq.Umax/min according to recorded VdTesting and VqTesting values, respectively.

Note: Running the motor without proper PI limits can yield distorted current waveforms and unstable closed loop operations which may damage the hardware.

Bring the system to a safe stop, as described at the end of build 1 (Section 7.1), by reducing the bus voltage and taking the controller out of real-time mode and reset.
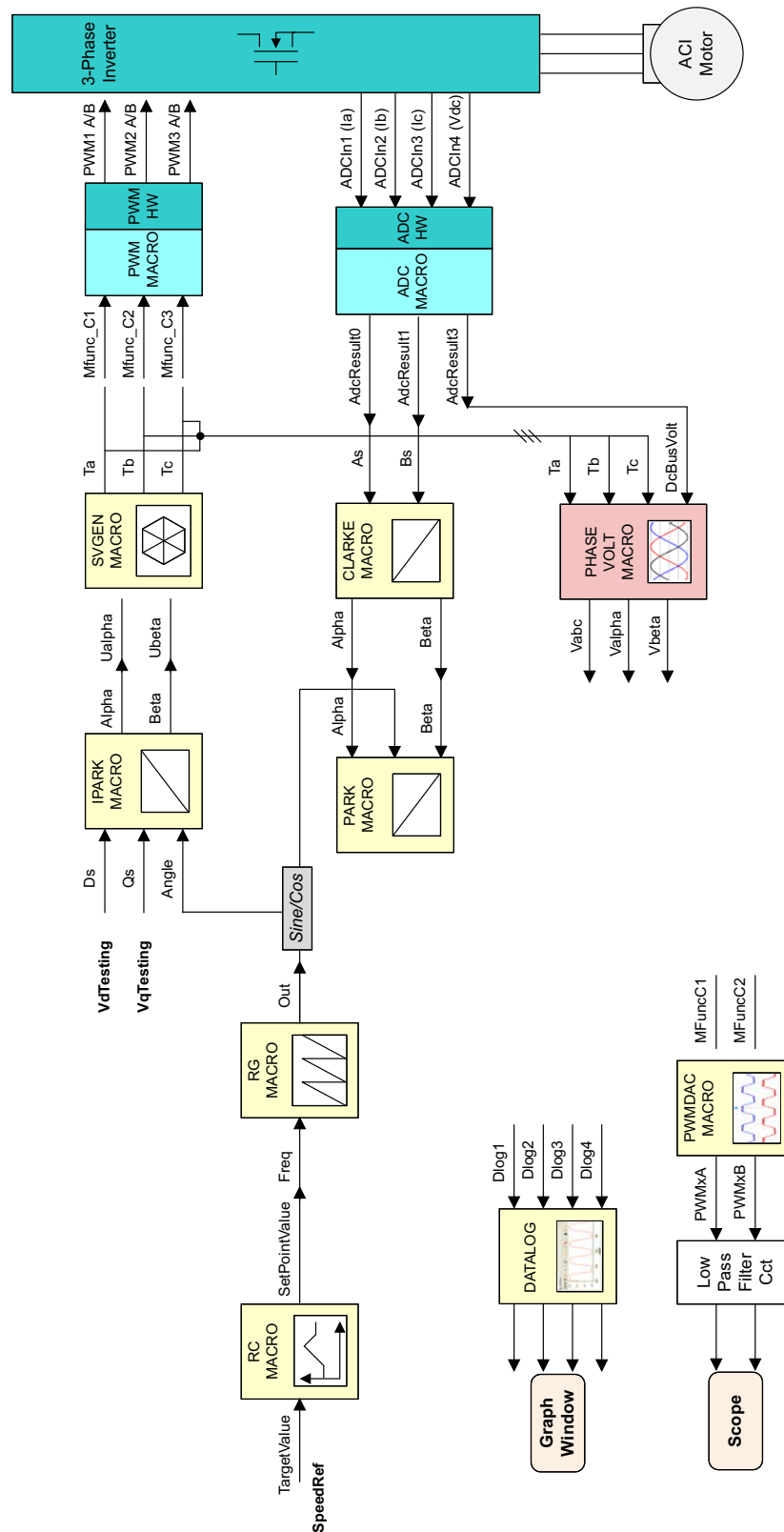
Figure 21. Level 2 — Incremental System Build Block Diagram

Level 2 verifies the analog-to-digital conversion, offset compensation, clarke / park transformations.

7.9 Level 3 Incremental Build

Assuming the previous section is completed successfully, this section verifies the dq-axis current regulation performed by PI modules and speed measurement modules (optional). To confirm the operation of current regulation, the gains of these two PI controllers are necessarily tuned for proper operation.

1. Open HVACI_Sensorless-Settings.h and select level 3 incremental build option by setting the BUILDLEVEL to LEVEL3 (#define BUILDLEVEL LEVEL3).

2. Right-click on the project name and click Rebuild Project.

3. When the build completes:

    (a) Click on the debug button.

    (b) Reset CPU and restart.

    (c) Enable Real-time mode and run.

    (d) Set "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

In the software, the key variables to adjust are:

- SpeedRef (Q24): for changing the rotor speed in per-unit.
- IdRef(Q24): for changing the d-qxis voltage in per-unit.
- IqRef(Q24): for changing the q-axis voltage in per-unit.

In this build, the motor is supplied by AC input voltage and the (AC) motor current is dynamically regulated by using PI module through the park transformation on the motor currents.

The steps are:

1. Launch a debug session, enable time mode and run the project.

2. Set SpeedRef to 0.3 pu (or another suitable value if the base speed is different), Idref to a certain value to generate rated flux.

3. Gradually increase voltage at variac / dc power supply to get an appropriate DC-bus voltage.

4. Check pi_id.Fdb in the watch windows with continuous refresh feature whether or not it should be keeping track pi_id.Ref for PI module. If not, adjust its PI gains properly.

5. To confirm these two PI modules, try different values of pi_id.Ref and pi_iq.Ref or SpeedRef.

6. For both PI controllers, the proportional, integral, derivative and integral correction gains may be re-tuned to have the satisfied responses.

7. Bring the system to a safe stop, as described at the end of build 1 (Section 7.1), by reducing the bus voltage and taking the controller out of real-time mode and reset.

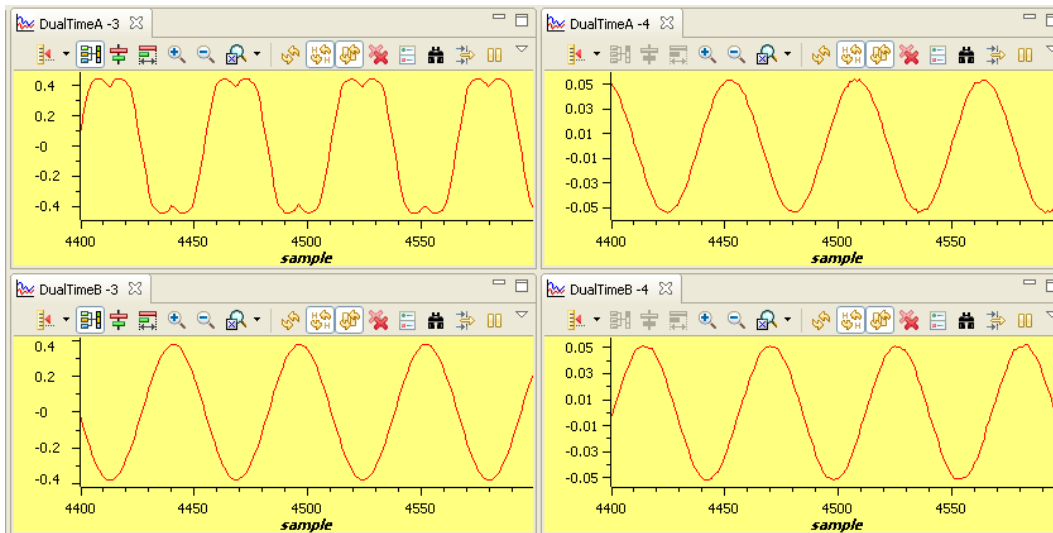While running this build, the current waveforms in the CCS graphs should appear as shown in Figure 22.

Figure 22. Svgen_dq1.Ta, Phase A Voltage, and Phase A and B Current Waveforms

7.10 Level 3B – QEP / SPEED_FR Test (Optional for Speed-Loop Tuning)

This section verifies the QEP1 driver and its speed calculation. Qep drive macro determines the rotor position and generates a direction (of rotation) signal from the shaft position encoder pulses. Make sure that the output of the incremental encoder is connected to [Main]-H1 and QEP/SPEED_FR macros are initialized properly in the HVACI_Sensorless.c file depending on the features of the speed sensor. Refer to the PDF files regarding the details of related macros in the motor control folder (C:\TI\controlSUITE\libs\app_libs\motor_control).

The steps to verify these two software modules related to the speed measurement are:

1. Set SpeedRef to 0.3 pu (or another suitable value if the base speed is different).

2. Compile, load, and run program with real-time mode and then increase voltage at variac and DC power supply to get the appropriate DC-bus voltage. Now the motor is running close to reference speed.

3. Check the "speed1.Speed" in the watch windows with continuous refresh feature whether or not the measured speed is less than SpeedRef a little bit due to a slip of motor.

4. To confirm these modules, try different values of SpeedRef to test the Speed.

5. Use oscilloscope to view the electrical angle output, ElecTheta, from QEP_MACRO module and the emulated rotor angle, Out, from RG_MACRO at PWMDAC outputs with external low-pass filters.

6. Check that both ElecTheta and Out are of saw-tooth wave shape and have the same period. If the measured angle is in opposite direction, then change the order of motor cables connected to inverter output (TB3 for HVDMC kit).

7. Check from Watch Window that qep1.IndexSyncFlag is set back to 0xF0 every time it reset to 0 by hand. Add the variable to the watch window if it is not already in the watch window.

8. Bring the system to a safe stop, as described at the end of build 1 (Section 7.1), by reducing the bus voltage and taking the controller out of real-time mode and reset.

7.11 Level 3C – CAP / SPEED_PR Test (Optional for Speed-Loop Tuning)

In this case, the CAP1 input is chosen to detect the edge. If available, make sure that the sensor output is connected to [Main]-H1 and CAP/SPEED_PR macros are initialized properly in the HVACI_Sensorless.c file depending on the features of the speed sensor. Typically the capture is used to measure speed when a simple low cost speed sensing system is available. The sensor generates pulses when detecting the teeth of a sprocket or gear and the capture drive provides the instantaneous value of the selected time base (GP Timer) captured on the occurrence of an event. Refer to the PDF files regarding the details of related macros in the motor control folder (C:\TI\controlSUITE\libs\app_libs\motor_control).

The steps to verify these two software modules related to the speed measurement are:

1. Set SpeedRef to 0.3 pu (or another suitable value if the base speed is different).

2. Compile, load, and run program with real time mode and then increase voltage at variac / dc power supply to get the appropriate DC-bus voltage. Now the motor is running reference speed.

3. Check the "speed2.Speed" in the watch windows with continuous refresh feature whether or not they should be less than SpeedRef a little bit due to a slip of motor.

4. To confirm these modules, try different values of SpeedRef to test the Speed.

5. Reduce voltage at variac and DC power supply to zero, halt program and stop real time mode. Now the motor is stopping.

An alternative to verify these two software modules without running the motor is to use a function generator. The key steps can be explained as follows:

1. Use a function generator to generate the 3.3 V (DC) square-wave with the desired frequency corresponding to the number of teeth in sprocket and the wanted speed in rpm. Then, connect only the pulse signal and ground wires from the function generator to HVDMC board. The desired frequency of the square-wave produced by function generator can be formulated as:

$$f_{square\_wave} = \frac{RPM \times TEETH}{60} \, Hz$$

where RPM is the wanted speed in rpm, and TEETH is the number of teeth in sprocket.

2. Compile, load, and run program with real-time mode and then increase voltage at variac to get the appropriate DC-bus voltage. Now the motor is running. Note that the SpeedRef could be set to any number.

3. Check the speed2.Speed and speed2.SpeedRpm in the watch windows with continuous refresh feature whether or not they should be corresponding to the wanted speed that is chosen before.

4. To confirm these modules, change different frequencies of square-wave produced by function generator with corresponding wanted (known) speed to check the Speed and SpeedRpm.

5. Bring the system to a safe stop, as described at the end of build 1 (Section 7.1), by reducing the bus voltage and taking the controller out of real-time mode and reset.
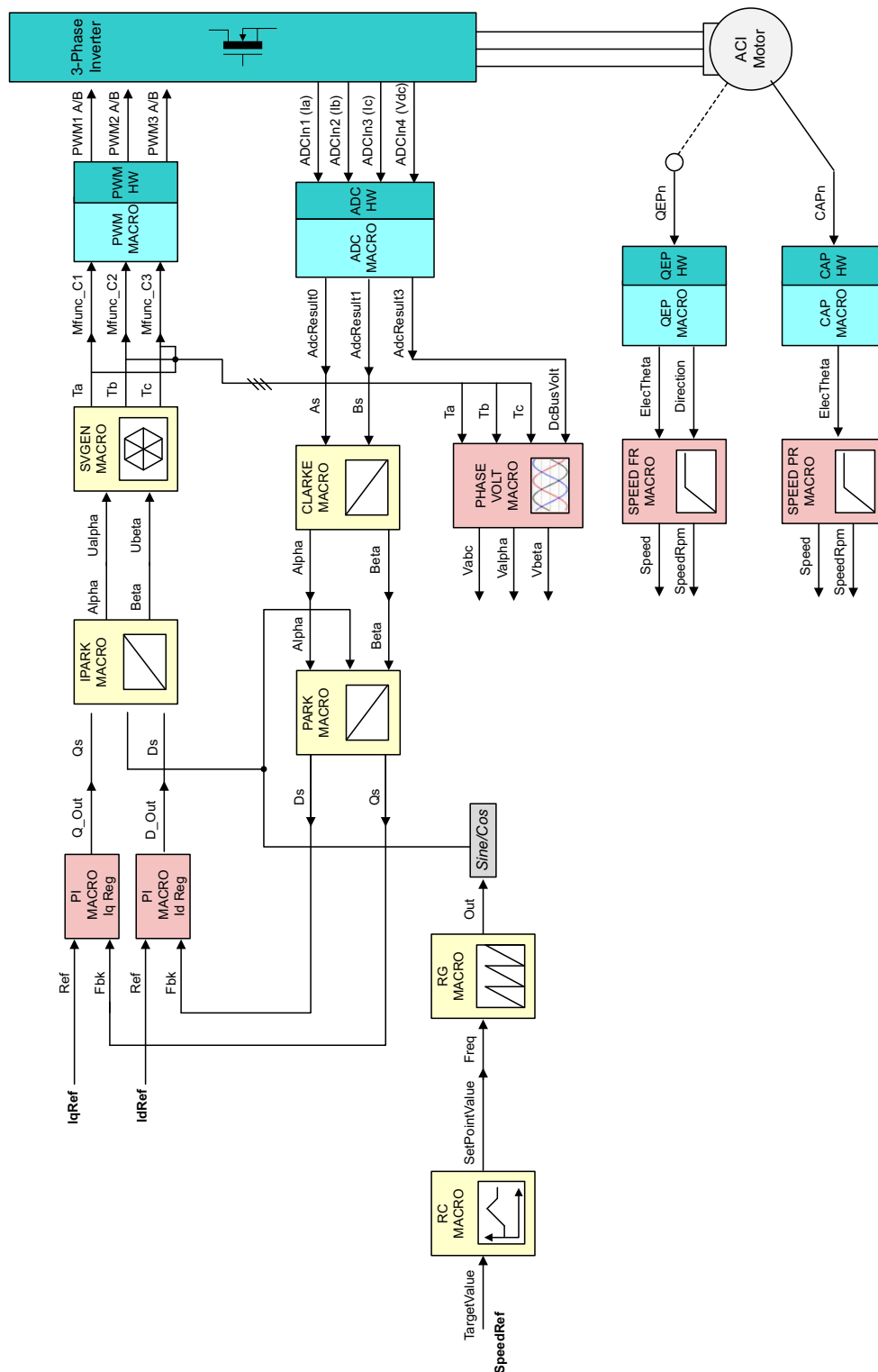
Figure 23. Level 3 — Incremental System Build Block Diagram

Level 3 verifies the dq-axis current regulation performed by PI macros and speed measurement modules.

7.12 Level 4 Incremental Build

Assuming the previous section is completed successfully; this section verifies flux estimation (ACI_FE) and open-loop speed estimation (ACI_SE), respectively.

1. Open HVACI_Sensorless-Settings.h and select level 4 incremental build option by setting the BUILDLEVEL to LEVEL4 (#define BUILDLEVEL LEVEL4).
2. Right-click on the project name and click Rebuild Project.
3. When the build completes:

   (a) Click on the debug button.

   (b) Reset CPU and restart.

   (c) Enable Real-time mode and run.

   (d) Set "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

The key variables to adjust are:

- SpeedRef (Q24): for changing the rotor speed in per-unit.
- IdRef (Q24): for changing the d-qxis voltage in per-unit.
- IqRef (Q24): for changing the q-axis voltage in per-unit.

The tuning of proportional and integral gains (Kp_fe and Ki_fe) inside the flux estimator may be critical for very low speed operation. The key steps are:

1. Set SpeedRef to 0.3 pu (or another suitable value if the base speed is different).
2. Compile, load, and run program with real time mode and then increase voltage at variac / dc power supply to get the appropriate DC-bus voltage. Now the motor is running close to reference speed.
3. Compare fe1.ThetaFlux with rg1.Out via PWMDAC with external low-pass filter and an oscilloscope. They should be identical with a small phase shift.
4. If fe1.ThetaFlux does not give the ramp waveform, the Kp and Ki inside the flux estimator are required to be re-tuned (likely to be reduced).
5. To confirm this flux estimator, try different values of SpeedRef.
6. Compare se1.WrHat with reference speed or measured speed in the watch windows with continuous refresh feature whether or not it should be nearly the same.
7. To confirm this open-loop speed estimator, try different values of SpeedRef.

While running this build, the current waveforms in the CCS graphs should appear as shown in Figure 24.
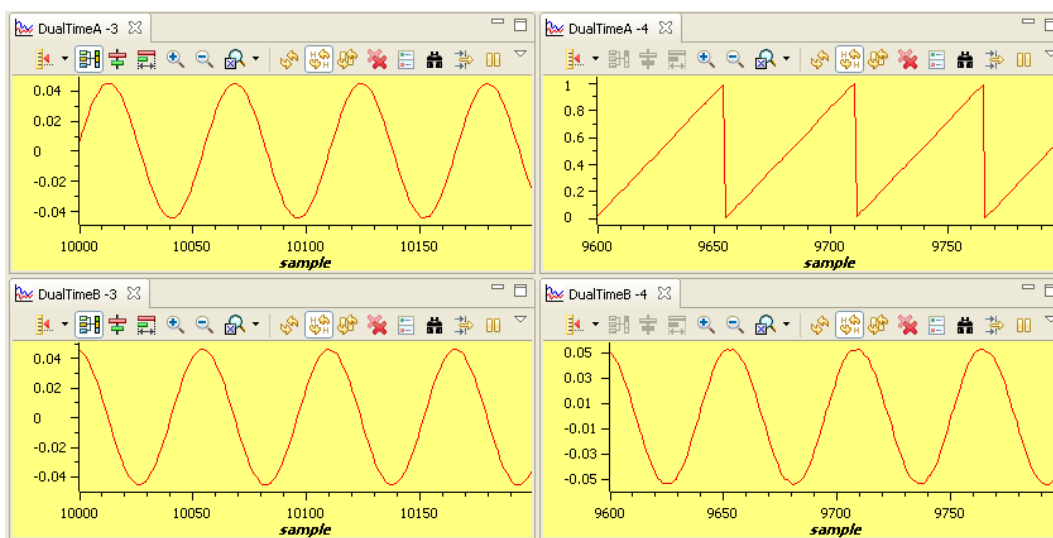
Figure 24. Estimated d and q Fluxes With Estimated Theta and Phase A Current Waveform
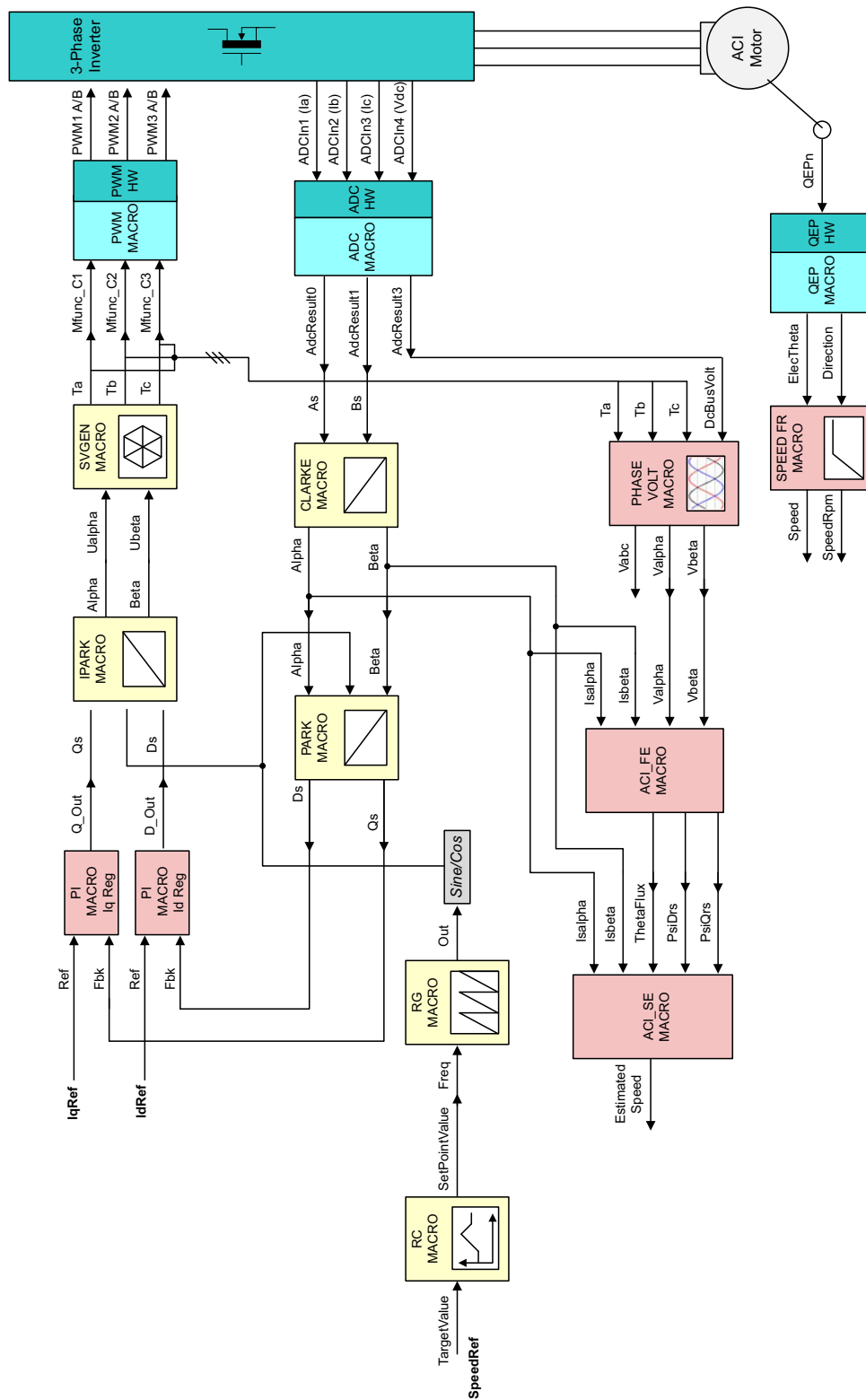
Figure 25. Level 4 — Incremental System Build Block Diagram

Level 4 verifies the flux and speed estimation performed by ACI_FE and ACI_SE.

7.13   Level 5 Incremental Build

Assuming the previous section is completed successfully, this section verifies the speed regulator performed by PI module. The system speed loop is closed by using the measured speed as a feedback.

1. Open HVACI_Sensorless-Settings.h and select level 5 incremental build option by setting the BUILDLEVEL to LEVEL5 (#define BUILDLEVEL LEVEL5).
2. Right-click on the project name and click Rebuild Project.
3. When the build is complete:

    (a) Click on the debug button.

    (b) Reset CPU and restart.

    (c) Enable Real-time mode and run.

    (d) Set "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

The key variables to adjust are:

- SpeedRef (Q24): for changing the rotor speed in per-unit.
- IdRef (Q24): for changing the d-qxis voltage in per-unit.

The speed loop is closed by using measured speed. The motor can spin only one direction when the measured speed (from capture driver) does not give information about the direction like QEP-based speed measurement. Therefore, if the speed sensor is not an incremental encoder, the SpeedRef is required to be positive. The key steps are:

7.13.1   Level 5A

1. Compile, load, and run program with real time mode.
2. Set SpeedRef to 0.3 pu (or another suitable value if the base speed is different).
3. Add the soft-switch variables "lsw" to the watch window in order to switch from current loop to speed loop. In the code lsw manages the loop setting as follows:
    - lsw=0, closes the current loop.
    - lsw=1, close the speed loop.
4. Set lsw to 1. Gradually increase voltage at variac / dc power supply to get an appropriate DC-bus voltage and now the motor is running around the reference speed (0.3 pu).
5. Gradually increase voltage at variac / dc power supply to get an appropriate DC-bus voltage and now the motor is running around the reference speed (0.3 pu).
6. Compare Speed with SpeedRef in the watch windows with continuous refresh feature whether or not it should be nearly the same.
7. To confirm this speed PI module, try different values of SpeedRef (positive only).
8. For speed PI controller, the proportional, integral, derivative and integral correction gains may be re-tuned to have the satisfied responses.
9. At very low speed range, the performance of speed response relies heavily on the good rotor flux angle computed by flux estimator.
10. Bring the system to a safe stop, as described at the end of build 1 (Section 7.1), by reducing the bus voltage and taking the controller out of real-time mode and reset.
11. Note that the IdRef is set to be constant at a certain value that is not too much for driving the motor. Practically, you can calculate IdRef from the rated flux condition.

7.13.2 Level 5B (Alternative Method)

In order to test ACI_FE only:

1. Disconnect the speed PI and Iq PI modules.

2. Apply constant "Iqref" as the reference to Iq PI as shown in Level 5B.

   If the ACI_FE runs properly, the motor should spin smoothly. If not, check the parameters in HVACI_Sensorless-Settings.h.

3. Ensure that the base (pu) quantities are set to maximum measurable current, voltage, and motor electrical parameters are correct. The DC bus voltage should be high enough in order not to saturate the PI outputs.

4. Run the same experiment again and keep tuning ACI_FE gains. Note that, in this scheme the speed is not controlled, therefore a non-zero torque (Iq) reference will spin the motor very fast unless loaded. Therefore keep the Iqref low (typically less than 0.1 pu depending on the experimental setup) and load the motor using a brake, generator, and other parts.

5. Increase Iqref or reduce the amount of load, if the motor speed is too low or the generated torque by the motor is not enough to handle the applied load. After tuning the ACI_FE, add speed PI into the system as shown in the block diagram and tune it, if needed. This method will help the user tune ACI_FE and speed PI separately.
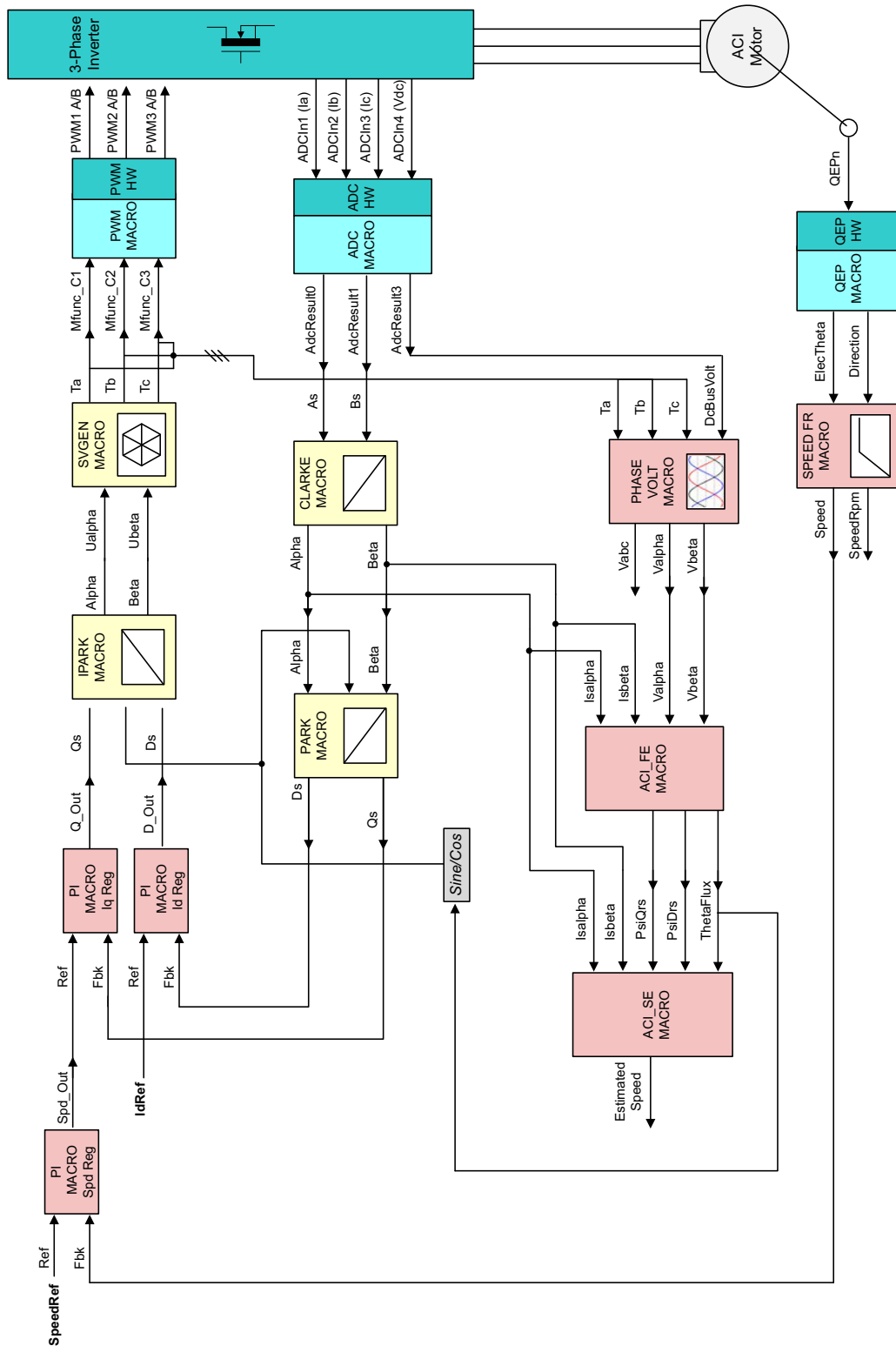
Figure 26. Level 5A — Incremental System Build Block Diagram

Level 5 verifies the speed PI module and estimated theta (lsw=1).
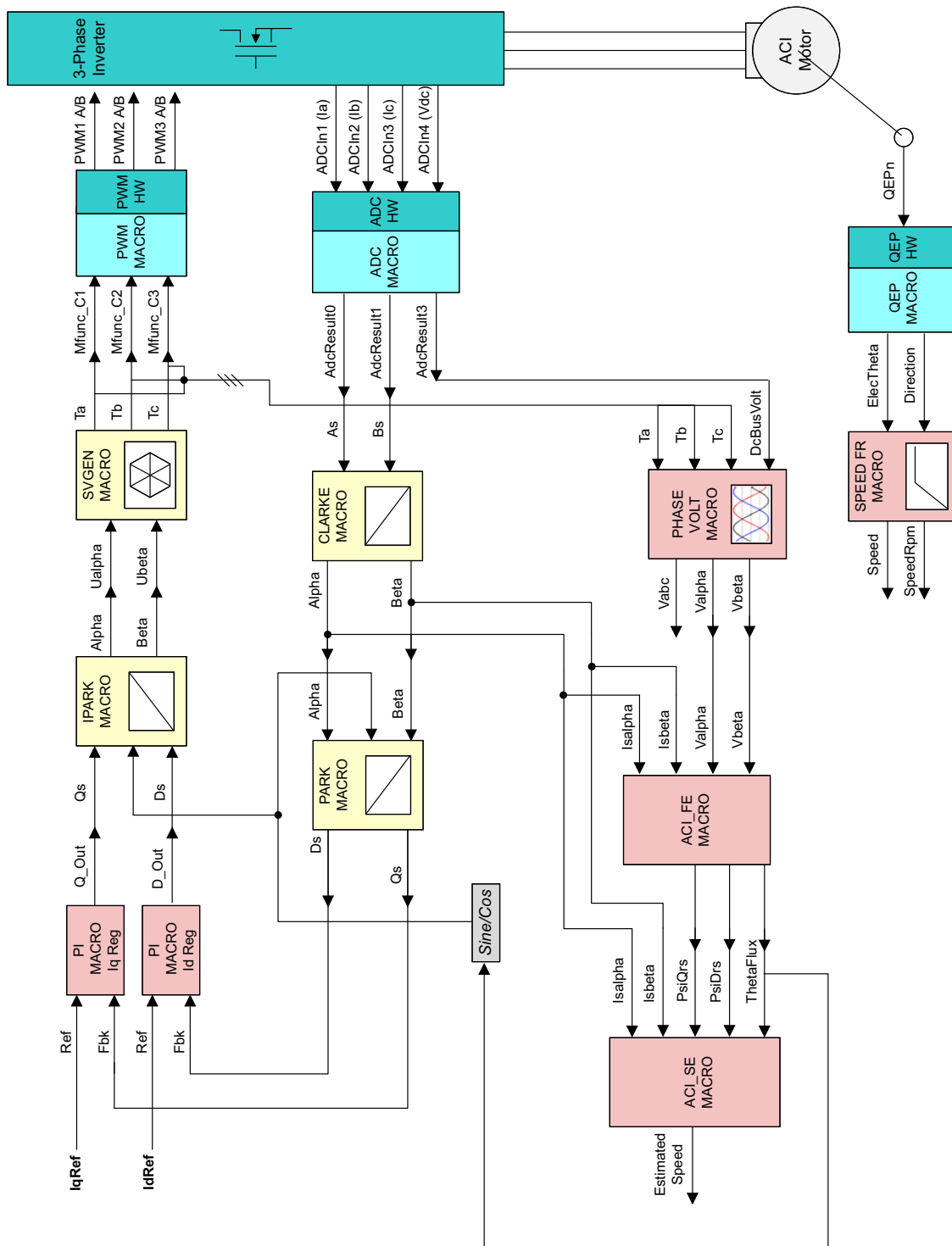
Figure 27. Level 5B — Incremental System Build Block Diagram

Level 5 verifies the estimated theta.

7.14 Level 6 Incremental Build

Assuming the previous section is completed successfully, this section verifies the speed regulator performed by PI module. The system speed loop is closed by using the estimated speed as a feedback.

1. Open HVACI_Sensorless-Settings.h and select level 6 incremental build option by setting the BUILDLEVEL to LEVEL6 (#define BUILDLEVEL LEVEL6).

2. Right-click on the project name and click Rebuild Project.

3. When the build is complete:

   (a) Click on the debug button.

   (b) Reset CPU and restart.

   (c) Enable Real-time mode and run.

   (d) Set "EnableFlag" to 1 in the watch window. The variable named "IsrTicker" will be incrementally increased as seen in watch windows to confirm the interrupt working properly.

The key variables to adjust are:

- SpeedRef (Q24): for changing the rotor speed in per-unit.
- IdRef (Q24): for changing the d-qxis voltage in per-unit.

The speed loop is closed by using estimated speed. Unlike the Capture based sensored system in previous build, the motor can spin both directions because the estimated speed (from open-loop speed estimator) gives information about the direction. Therefore, the SpeedRef can be either positive or negative value. The key steps are:

1. Compile, load, and run program with real time mode.

2. Set SpeedRef to 0.3 pu (or another suitable value if the base speed is different).

3. Set lsw to 1. Gradually increase voltage at variac / dc power supply to get an appropriate DC-bus voltage and now the motor is running with this reference speed (0.3 pu).

4. Compare se1.WrHat with SpeedRef in the watch windows with continuous refresh feature whether or not it should be nearly the same.

5. To confirm this speed PI module, try different values of SpeedRef (positive or negative).

6. For speed PI controller, the proportional, integral, derivative and integral correction gains may be re-tuned to have the satisfied responses.

7. At very low speed range, the performance of speed response relies heavily on the good rotor flux angle computed by flux estimator.

8. Bring the system to a safe stop, as described at the end of build 1 (Section 7.1), by reducing the bus voltage and taking the controller out of real-time mode and reset.

9. Note that the IdRef is set to be constant at a certain value that is not too much for driving the motor. Practically, you can calculate IdRef from the rated flux condition and follow a flux weakening profile at high speed region.

NOTE: Using scheme 6B is recommended until completing the loop-tuning process to achieve smooth start-up. This scheme employs a soft loop-switch in the code for sensorless speed loop transient. For this purpose, add the soft-switch variable "lsw" to watch window and set "lsw" to 1 in order close the speed loop. Each time close the speed loop earlier manually from watch window until achieving direct sensorless start-up.

As a mid-step in between level 5 and level 6, a soft switch can be used in level 5A to change the speed PI feedback from sensor based speed measurement to estimated speed. In this case, the initial speed estimation ambiguities will be handled by measured speed until the system is perfectly tuned.

While running this build, the current waveforms in the CCS graphs should appear as in Figure 28:
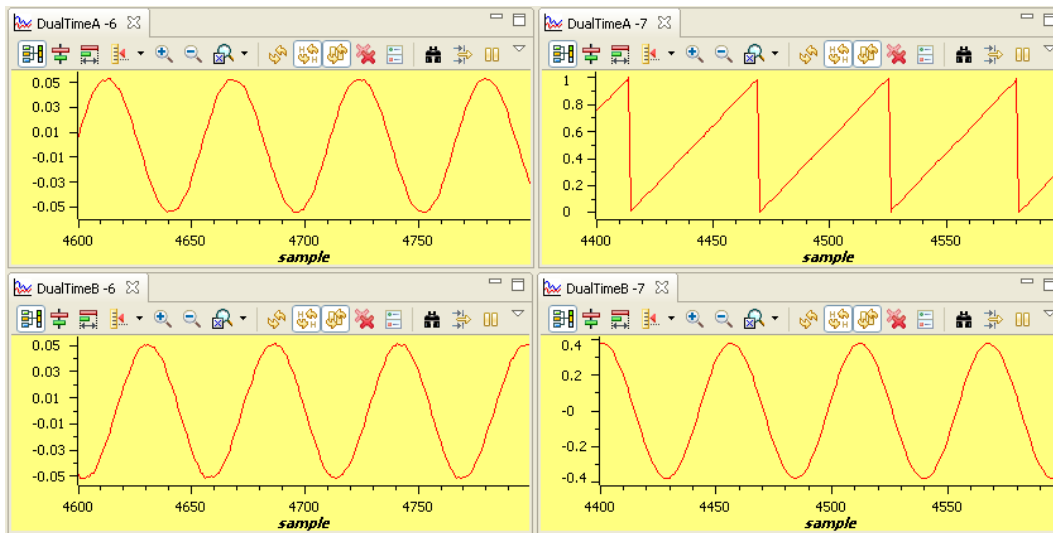
Figure 28. Phase A and B Current With Estimated Theta and Phase A Voltage Waveforms
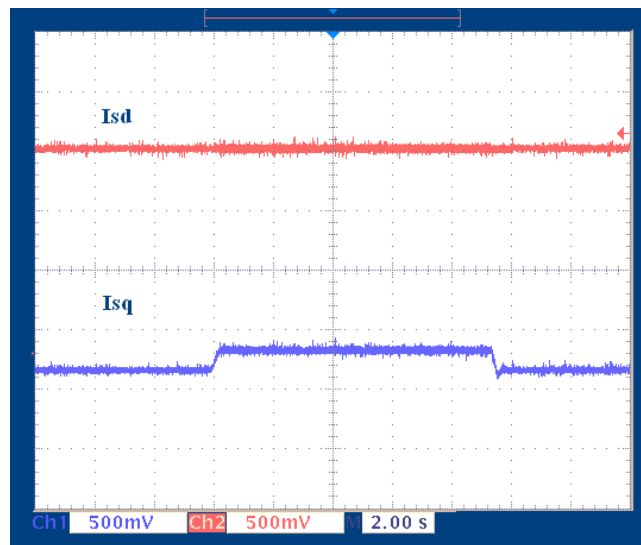
Figure 29. Flux and Torque Components of the Stator Current in the Synchronous Reference Frame

The flux and torque components of the stator current in the synchronous reference frame are under 1.0 pu step-load and 0.3 pu speed monitored from the PWMDAC outputs
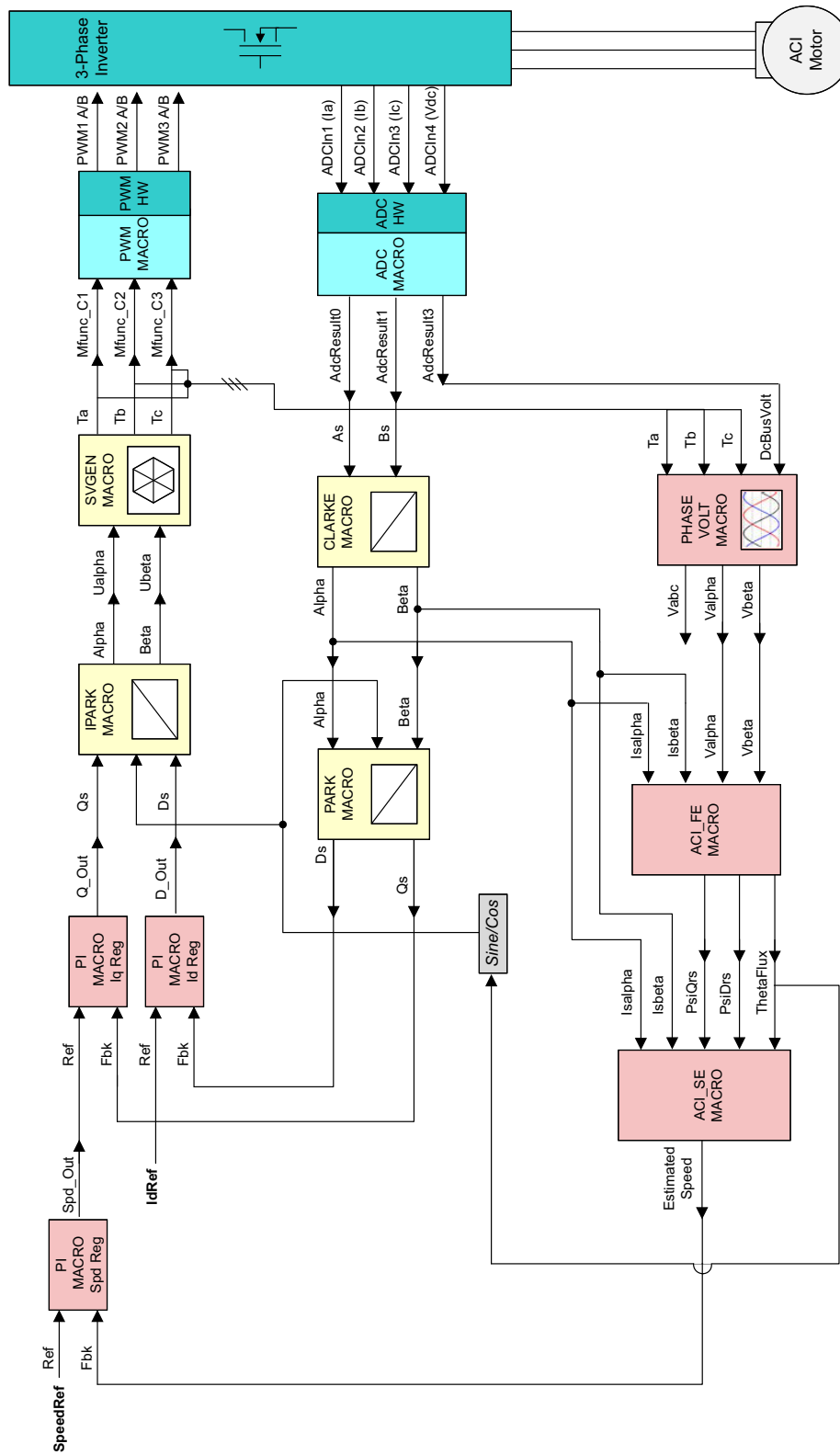
Figure 30. Level 6A — Incremental System Build Block Diagram

Level 6 verifies the complete system.

Copyright © 2013, Texas Instruments Incorporated

Figure 31. Level 6B — Incremental System Build Block Diagram

Level 6 verifies the complete system.

8 References

- *Using PWM Output as a Digital-to-Analog Converter on a TMS320F280x Digital Signal Controller* (SPRAA88)

IMPORTANT NOTICE

Products		Applications	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		