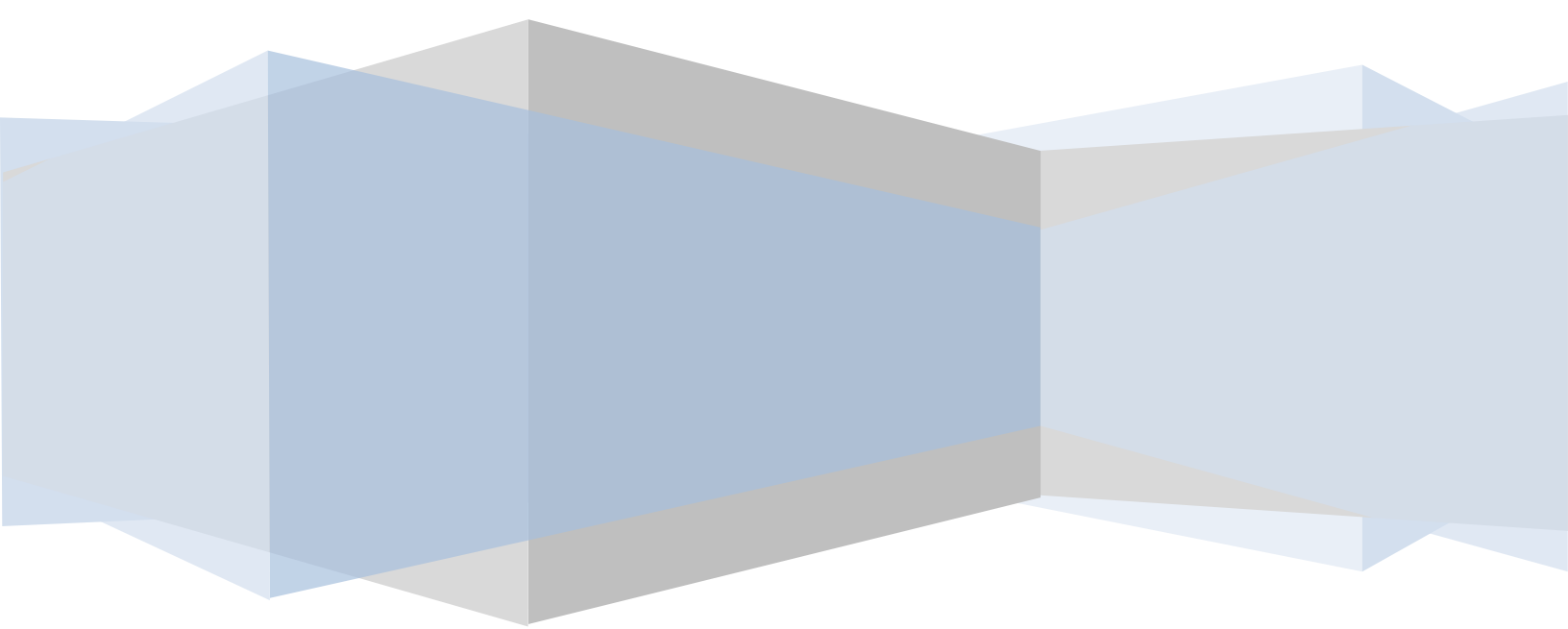


Implementing Sensorless Field Oriented Control of an AC Induction Motor with PFC front end on 2000 Microcontrollers



Contents

Abstract	2
Introduction.....	3
Reason for PFC	3
Field Oriented Control of AC Induction Motor	4
Challenge integrating PFC and motor control on a single microcontroller	4
Benefits of 32-bit C2000 Controllers	6
TI Digital Motor Control and Digital Power Libraries	7
System Overview	8
PFC implementation on TMDSHVMTRPFCKIT	9
Motor inverter implementation on TMDSHVMTRPFCKIT	9
Hardware Configuration (TMDSHVMTRPFCKIT).....	10
Software Setup for HVACI_Sensorless_PFC Project.....	12
Software Setup for HVACI_Sensorless_PFC Project.....	13
Incremental System Build for ACI Sensorless Project.....	13
Incremental Build Level 1	14
Incremental Build Level 2	16
Incremental Build Level 3	18
Level 3A	18
Level 3B	20
References.....	24

Abstract

This application note presents a solution to control a power factor correction (PFC) stage while also controlling an AC induction motor using *TMS320F2803x* microcontrollers. *TMS320F2803x* devices are part of the family of C2000 microcontrollers which enable cost-effective design of intelligent controllers for three phase motors or digital power systems by reducing system components and increasing efficiency. The solution uses on the High-Voltage Motor Control and PFC Kit (*TMDSHVMTRPFCKIT*) and a F28035 *controlCARD* (*TMDSCNCD28035*) for demonstration.

This application note covers the following:

- A brief review of Power Factor Correction
- Incremental build levels based on modular software blocks
- Experimental results

Introduction

Reason for PFC

While three phase motors are used in a wide variety of applications, they are typically operated from a single-phase AC supply. However, due to the non-linearity of the AC rectification stage, the current drawn becomes distorted and this in turn compromises the power factor. Thus, a power factor correction stage (PFC) has become an integral part of most rectifier designs. A typical motor control system consists of three major power stages: an AC rectifier, the PFC stage to boost the voltage to a DC value, and the motor inverter stage. The output of the rectifier serves as input to the PFC stage while the motor inverter acts as its load; this is illustrated in *Figure 1*.

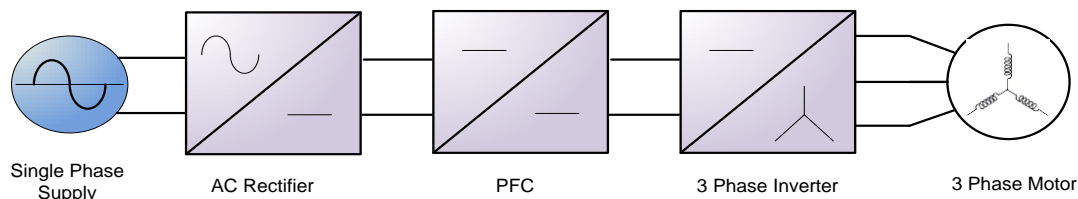


Figure 1: Typical three-phase motor control system.

To understand why a PFC stage is necessary consider a typical AC rectifier circuit as shown in *Figure 2*. A sketch of the bridge rectifier input current shows how it is dominated by pulses that correspond to the time when the output DC bus capacitor is being charged. A Fourier analysis of the waveform shows the presence of harmonics in the line current. The harmonics bring about two difficulties: Firstly, they represent reactive power, which results in losses; and secondly, such currents can distort the line voltage. To avoid these undesirable effects, PFC attempts to draw a sinusoidal current from the line that is exactly in phase with the line voltage.

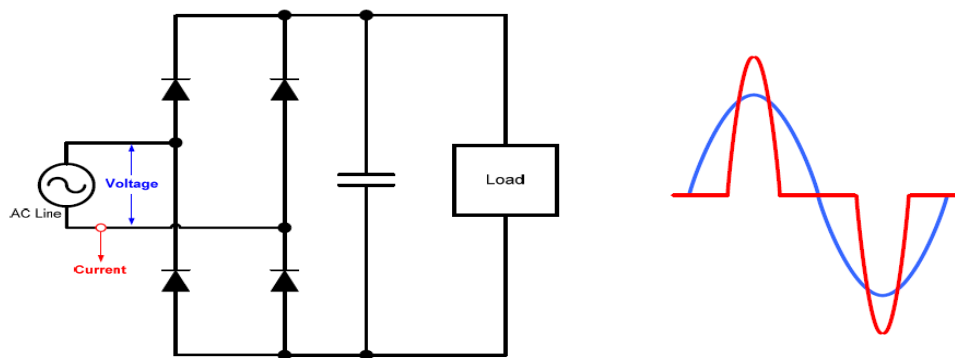


Figure 2: Typical AC Rectifier input circuit and input current waveform.

Currently, PFC solutions employ boost regulators and average current mode control techniques for wave shaping of the input current. The boost converter switches at a much higher frequency (tens/hundreds of kHz) compared to the line frequency, thus reducing the input filter and the boost inductor size drastically. Further improvements may be obtained by implementing an interleaved boost converter topology. This reduces the input current ripple, increases efficiency, reduces stress on power electronic components and also results in reduction in the overall inductor volume.

Field Oriented Control of AC Induction Motor

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; the useful mechanical output of the machine. Simple control such as the V/Hz strategy has limitations on 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 de-coupled torque and magnetization control is commonly called rotor flux oriented control, or simply Field Oriented Control (FOC).

FOC consists of controlling stator currents represented by a vector. This control is based on projections which transform a three phase time and speed dependent system into a two coordinate (d-q coordinates) time invariant system. These projections lead to a structure similar to that of a DC machine control. FOC machines need two constants as input references: the torque component (aligned with the q axis) and the flux component (aligned with d axis). 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 remainder of this document is focused on PFC operation and integration with an AC induction motor control algorithm. More information regarding FOC theory when using AC induction motors can be found in the HV-ACI project documents available within *controlSUITE*:

controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0\HVACI_Sensorless\~Docs
controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0\HVACI_Sensored\~Docs

Challenge integrating PFC and motor control on a single microcontroller

Real-time motor control using complex control algorithms such as FOC need a lot of processor bandwidth. Hence, it is challenging to accommodate the PFC algorithm, which operates at a much higher control loop rate, along with a motor control algorithm on a single microcontroller while guaranteeing cycle by cycle execution. Fortunately, the ADC and PWM peripherals on the C2000 device family have been designed to integrate multi-frequency control loops and guarantee sampling at correct instances of the PWM waveform. However, there is only one ADC present (with two sample and hold circuits) on these low cost devices. Thus if the two independent control loops need the ADC; in this case PFC control and motor control; both control loops need to guarantee that they do not conflict for the ADC resource at any instance.

To eliminate conflicts the phase shift mechanism found on the ePWM peripheral is employed. *Figure 3* illustrates the timing diagram for configuring the ePWM modules for motor control and PFC, and the synchronization mechanism used to avoid ADC conflicts. In this example a phase shift of 30 clock cycles is chosen to account for 8 cycle sampling window and 15 cycle first conversion delay

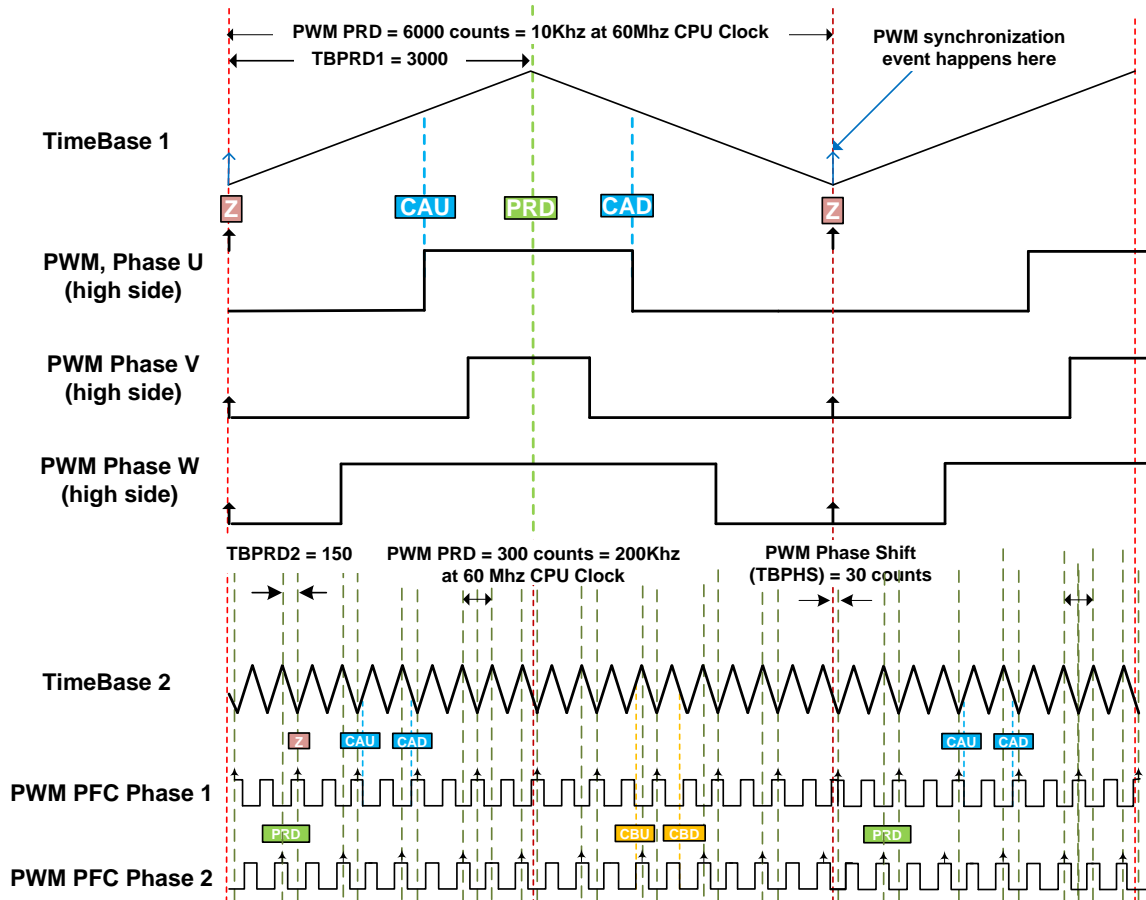


Figure 3: Timing diagram for PFC and Motor Control integration

The diagram above illustrates PWM waveform generation on a 60MHz device for a 10 kHz motor control algorithm and a 100 kHz PFC control loop. Note: the switching rate is 200 kHz; this is dependent on the inductor present on the PFC power stage. The PWM peripheral offers the flexibility to trigger ADC Start of Conversions (SOC's) every switching cycle or alternate thus avoiding any unnecessary load on the ADC.

The implementation presented in this text will have both, the PFC control loop and the motor control loop, execute on a single CPU core of a *TMS320F28035* device. The PFC control loop, running at a higher frequency, will have higher priority over the motor control loop. When the solution is implemented, the PFC control must be capable of interrupting the motor control. This requires that the PFC control loop be as efficient and as fast as possible so as not severely affect the execution of the motor control loop.

Finally, this implementation works well for applications dedicated to PFC and motor control; where little or no communication requirements exist and there are minor system diagnostics requirements. It is well suited for a typical HVAC application. However, for systems in which fault diagnosis, monitoring and communication requirements are prevalent, there is not much microcontroller bandwidth is left to implement these functions. Another PFC and motor control implementation is provided where the PFC control loop is offloaded onto the control law *Control Law Accelerator (CLA)*. The *CLA* is an independent floating point unit, present on some C2000 devices, designed to offload fast control tasks with the purpose of freeing up bandwidth on the main CPU (C28x) core. The project can be found within *controlSUITE*:

controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0\HVACI_Sensorless_PFC_CLA

Benefits of 32-bit C2000 Controllers

The C2000 family of devices not only possesses the computation power to execute complex control algorithms but also feature the right mix of peripherals to interface with various hardware components (ADC, ePWM, QEP, eCAP, etc). Additionally, 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 software ecosystem (libraries and application software) and hardware (application kits) reduce the time and effort needed to develop a Digital Motor Control (DMC) or a Digital Power (DP) solution. The DMC and DP libraries provide configurable blocks that can be reused to implement new control strategies. The IQMath library enables easy migration from floating point algorithms to fixed point thus accelerating the development cycle.

With the C2000 family of devices it is easy and quick to implement complex control algorithms. The use of C2000 devices and advanced control schemes provide the following system improvements:

- Favor system cost reduction by an efficient control in all speed range implying right dimensioning of power device circuits.
- Advanced control algorithms make it possible to reduce torque ripple, thus resulting in lower vibration and longer motor life; and reduce harmonics generated by the inverter thus reducing filter cost.
- Sensorless algorithms eliminate the need for speed or position sensors.
- Decreases the number of look-up tables further reducing the amount of memory required.
- The real-time generation of smooth near-optimal reference profiles and move trajectories, resulting in better-performance.
- Generation of high resolution PWM's is possible with the use of ePWM peripheral for controlling the power switching inverters.
- Provides single chip control system

For advanced controls, C2000 controllers can also perform the following:

- Enables control of multi-variable and complex systems using modern intelligent methods such as neural networks and fuzzy logic.
- Performs adaptive control. C2000 controllers have the speed capabilities to concurrently monitor the system and control it. A dynamic control algorithm adapts itself in real time to variations in system behavior.
- Performs parameter identification for sensorless control algorithms, self-commissioning, and online parameter estimation update.
- Performs advanced torque ripple and acoustic noise reduction.
- Provides diagnostic monitoring with spectrum analysis. By observing the frequency spectrum of mechanical vibrations, failure modes can be predicted in early stages.
- Produces 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.

TI Digital Motor Control and Digital Power Libraries

The DMC library and the DP library are composed of functions represented as blocks. These blocks are categorized as Transforms, Estimators, Controllers, and Peripheral Drivers. Each software block is separately documented with source code, use, and technical theory. These modules allow users to quickly build or customize their own systems. Library documentation and source code can be found within *controlSUITE*:

- *controlSUITE\libs\app_libs\digital_power\2803x_v3.4\Doc*
- *controlSUITE\libs\app_libs\motor_control\drivers\2803x_v2.0*
- *controlSUITE\libs\app_libs\motor_control\math_blocks\4.0*

The library components have been used by TI to provide system examples. At initialization all library variables are defined and inter-connected. During 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 parameters need to be tuned.

Note: TI modules are written in form of macros for optimization purposes (refer to application note *SPRAAK2* for more details at TI website). The macros are defined in header files; users can open the respective header file and change the macro definition if desired. While most library modules can be found written in C language, there are some macro blocks written in C28x Assembly. Assembly macro blocks are provided for applications with time/cycle constraints. In the C language macro definitions, there should be a backslash “\” at the end of each line as shown below which means that the code continue in the next line. **When using Code Composer Studio version 4 (CCSv4) or earlier: any character, including invisible ones like “space”, located after the backslash will cause compilation error.** Therefore, make sure that the backslash is the last character. With respect to code development, the macros are very similar to C functions; users can easily convert a macro definition to a C function.

```
#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);
```

Figure 4: Motor control macro example.

System Overview

This document describes the real-time control framework used to run PFC control while also controlling an AC induction motor using a sensorless field oriented control algorithm. The framework is designed to run on *TMS320C2803x* based controllers with *Code Composer Studio (CCS) v5*. *Table 1* below summarizes the system configuration; *Table 2* and *Table 3* list the framework modules to be used¹.

Table 1: System Configuration

System Configuration	
Development/Emulation	Code Composer Studio v5.2 (or higher) with Real-Time debugging
Target Controller	TMS320F2803x
PWM Frequency	100kHz PWM (PFC), 10kHz PWM (Motor control), 60kHz PWMDAC
PWM Mode	Symmetrical with programmable dead band
Interrupts	ADC End of Conversion – Triggers 10 kHz and 100kHz ISR execution.
Peripherals Used (Motor)	PWM: 1, 2, 3 – Motor control ADC: A1, B1 – Phase current sensing, A7 - DC Bus voltage sensing
Peripherals Used (PFC)	PWM: 4 – PFC control ADC: A4, A6 – PFC feedback, B0, B2 – AC input measurement
Peripherals Used (PWMDAC)	PWM 6A, 6B, 7A & 7B for PWMDAC outputs.

Table 2: Digital Power Library Modules

Macro Names	Explanation
ADCDRV_1ch	Returns normalized (0, 1) ADC conversion values.
PWMDRV_2ch_UpDwnCnt	PWM driver capable of driving two channels
CNTL_2P2Z	Second order IIR filter with programmable output saturation
PFC_InvRmsSqr	Squares the reciprocal of a scaled unipolar signal
MATH_EMAVG	Calculates exponential moving average
PFC_ICMD	Calculates current command for power factor correction
SineAnalyzer	Calculates RMS, average and frequency of a sinusoidal input wave

Table 3: Digital Motor Control Library Modules

Macro Names	Explanation
CLARKE	Clarke Transformation
PARK / IPARK	Park and Inverse Park Transformation
PI	PI Regulators
RC	Ramp Controller (slew rate limiter)
RG	Ramp/Saw-tooth 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 Inv. Clarke Trans.)
PHASEVOLT	Phase Voltage Calculator
PWM / PWMDAC	PWM and PWMDAC Drives

¹ Please refer to library documents for detailed explanation and theoretical background of each macro.

In this system the *TMS320F2803x controlCARD* is used to generate pulse width modulation (PWM) signals used to control the PFC and the motor. The following sections describe the PFC and motor control implementations found on the TMDSHVMTRPFCKIT.

PFC implementation on TMDSHVMTRPFCKIT

The TMDSHVMTRPFCKIT features a two-phase interleaved PFC topology (PFC-2PhIL). Figure 5 shows a block diagram of the PFC stage. The AC input is feed through an EMI filter followed by a bridge rectifier. The output of the rectifier is the input to the PFC circuit which is then boosts the rectified DC voltage to the desired DC bus voltage. Phase A of the boost circuit is formed by: inductor L1, MOSFET Q1, and diode D1. Likewise, Phase B is formed by inductor L2, MOSFET Q2, and diode D2. Finally a capacitor at the boost converter output acts as an energy reservoir, which reduces the output voltage ripple.

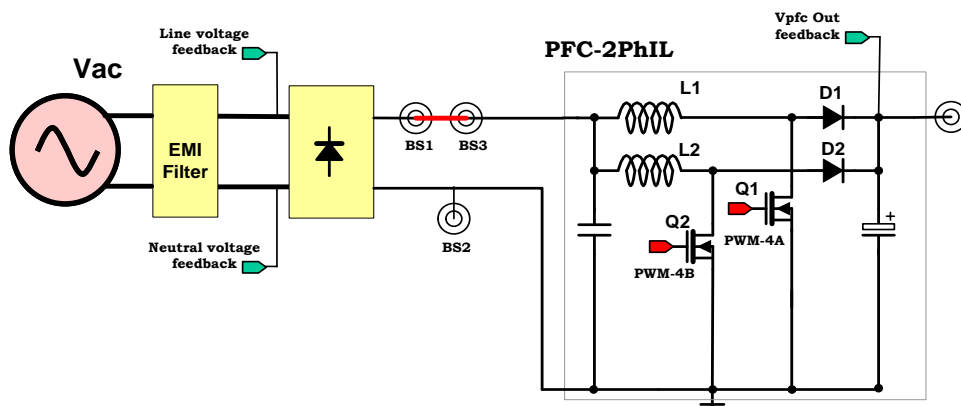


Figure 5: Two-Phase Interleaved PFC found on the TMDSHVMTRPFCKIT

The TMS320F2803x device uses feedback signals and PWM outputs to interact with the hardware. The AC input (after the EMI filter) as well as PFC output are used by the microcontroller to determine proper PWM duty cycle. The MCU achieves PFC by controlling the duty cycle in such a way that the rectified input current is made to follow the rectified input voltage while providing load and line regulation at the same time. The PWM outputs switch the MOSFETs On/Off boosting the PFC output voltage.

Motor inverter implementation on TMDSHVMTRPFCKIT

Motor control is accomplished with the help of an Integrated Power Module (IPM), a conventional voltage-source inverter. The IPM receives six pulse-width modulation (PWM) signals from the TMS320F2803x device which control the power switching devices within the IPM. The TMS320F2803x uses the integrated analog-to-digital converter (ADC) to measure two motor-phase currents (I_a and I_b) and the DC-bus voltage. These feedback measurements are the input required for the FOC motor control algorithm. The overall system implementing a 3-ph induction motor control is depicted in Figure 6. The outputs obtained from the motor control algorithm are used to change/update PWM duty cycles.

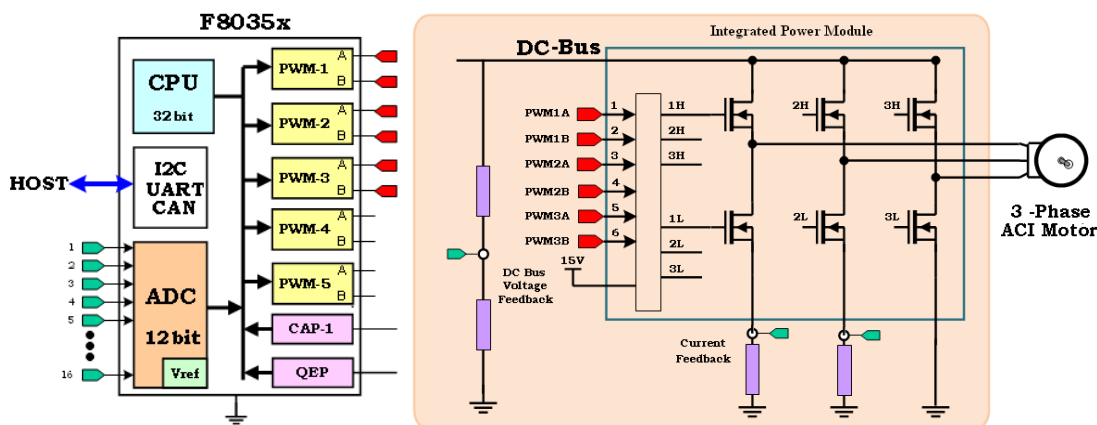


Figure 6: Motor control hardware found on TMDSHVMTRPFCKIT

Hardware Configuration (TMDSHVMTRPFCKIT)

For an overview of the kit hardware and details setup instructions please refer to the Hardware Guide found in:

controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0\~Docs

As quick reference, some brief setup instructions can be found below.

HW Setup Instructions

1. Install jumpers [Main]-J3, [Main]-J4 and [Main]-J5 for 3.3V, 5V and 15V power rails. Likewise, ensure that jumper [Main]-J9 is populated and that jumper [M3]-J5 is **NOT** populated.
2. Install jumpers [Main]-J7 between pins 2-3 (pins furthest from the DIMM socket) and [Main]-J8 to enable over-current protection.
3. Move [Main]-J2 to connect the center pin with the pin labeled “PFC” (pin furthest from block M2).
4. Connect [Main]-BS1 to [Main]-BS3 using a banana plug cord.
5. Unpack the 100-pin DIMM *controlCARD* and place it in the [Main]-J1 socket. Push vertically down using even pressure from both ends of the card until the clips snap and lock (to remove the card simply spread open the retaining clips with thumbs).
6. Connect a USB cable to connector [M3]-JP1. This will enable isolated JTAG emulation to the C2000 device. [M3]-LD1 should turn on.
7. Ensure that [M6]-SW1 is **NOT** in the “EXT” position. Connect 15V DC power supply (supplied with the kit) to [M6]-JP1.
8. Turn [M6]-SW1 to the “EXT” position. Both [M6]-LD1 and LD1 on the *controlCARD* should turn on. This indicates both are receiving power from the DC Power entry macro.

Note: Motor should **NOT** be connected to the [M5]-TB3 terminals until the third incremental build setup. The [Main]-P1 plug will be used to provide AC power; power should only be applied when instructed to do so, keep disconnected otherwise.

Important: Make sure jumper [Main]-J2 is either completely removed from the kit or set to connect the center pin with the pin labeled “PFC”. Failing to do so will add an undesired capacitive load to the [M4] PFC macro block. The capacitive load will produce undesired effects on the PFC circuitry which can result in personal harm or damage to the kit hardware.

Note: If operating from 240Vrms a step down transformer can be used. The use of an isolation transformer is recommended regardless of the AC power source. Do not connect scope probes to the board without fully understanding the isolation requirements.

For reference, hardware configuration is summarized in *Figures 7, 8* found below.

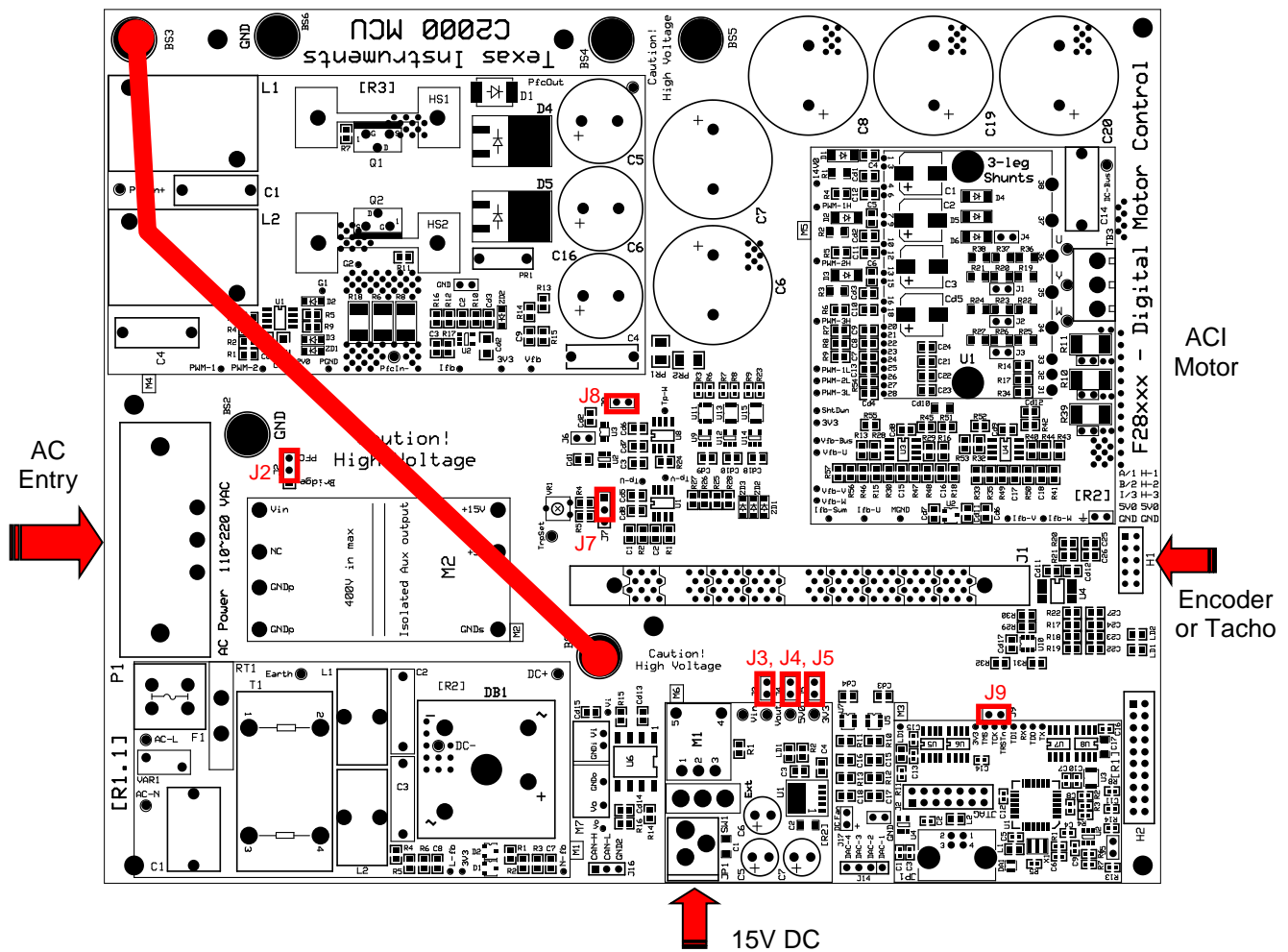


Figure 7: Hardware jumper settings

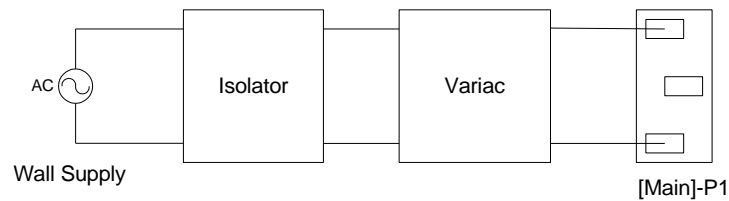


Figure 8: AC supply isolation



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

Software Setup for HVACI_Sensorless_PFC Project

As mentioned, the project will be controlling a PFC stage while also running an FOC motor control algorithm. Additionally, these control algorithms will both be executed on the main CPU (C28x core). Each algorithm will be contained within an interrupt service routine (ISR). Each ISR will operate at a different frequency: 100 kHz for PFC and 10 kHz for motor control. These ISRs will be triggered by ADC End of Conversions (EOC). A system software flowchart is provided in *Figure 9*.

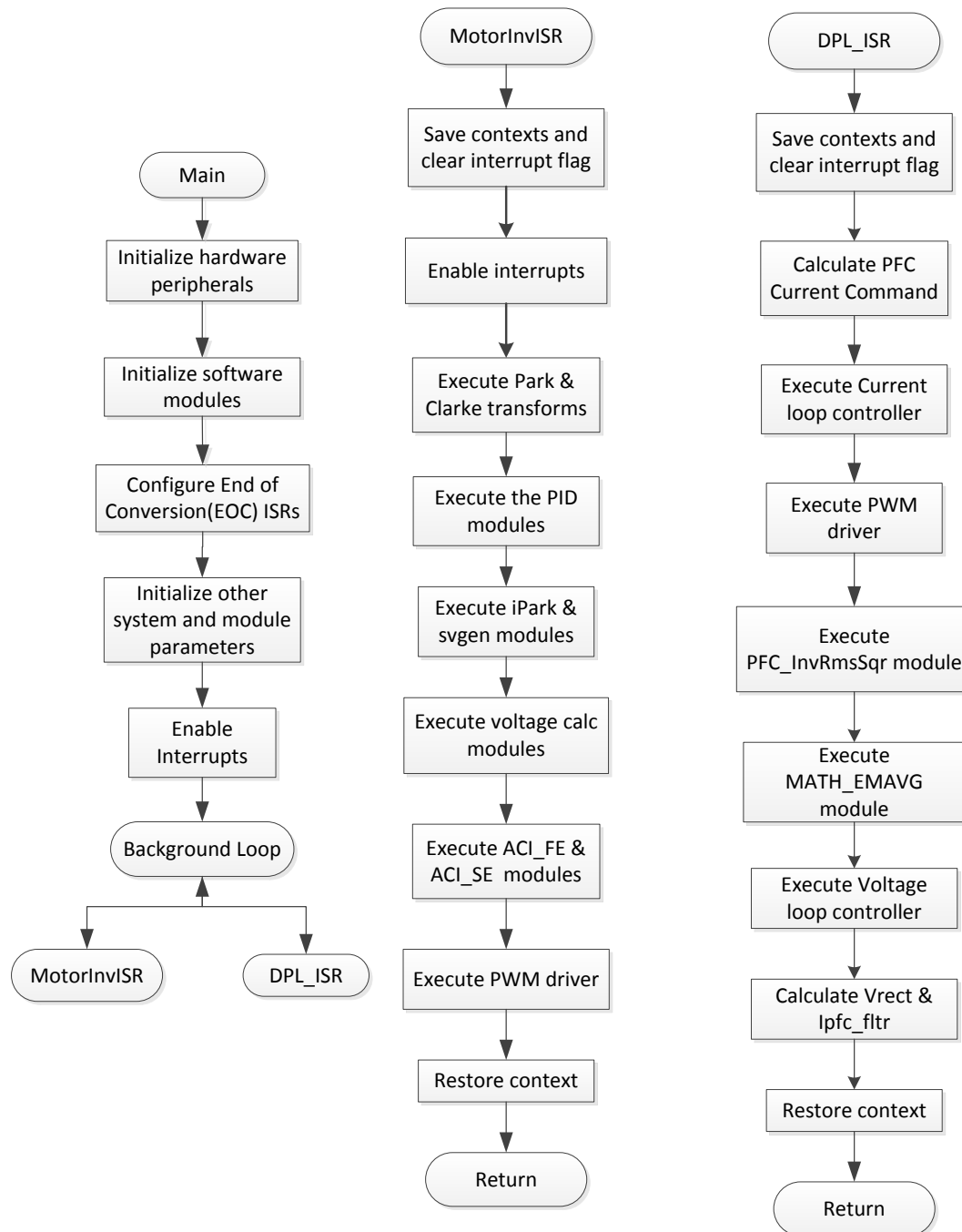



Figure 9: Motor control and PFC flowcharts

Software Setup for HVACI_Sensorless_PFC Project

For detailed setup instructions refer to the “Software Setup for HVMotorCtrl+PFC Kit Projects” section in the “How to Run Guide”:

controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0\~Docs

Import the *HVACI_Sensorless_PFC_2803x* project to the workspace and select it as the active project. Open the *Scripting Console* (Click *View ► Scripting Console*). Click on the “*Open Command File*” button  on the window. Use the pop-up window to navigate to the project folder and select the file named “*AddExpressionsWinwdownVars.js*”. The *Scripting Console* will proceed to automatically remove any variables on the *Expressions Window* and add those necessary to run this project. All variables are added in their respective correct format (float, Q-Value, etc.).


Incremental System Build for ACI Sensorless Project

The system is gradually built up in order for the final system can be confidently operated. Three phases of the incremental system build are designed to verify the major software modules used in the system. *Table 4* summarizes the modules tested in each incremental system build.

Table 4: Motor control and PFC modules used per incremental build

Software Module	Build 1	Build 2	Build 3
ADCDRV_1ch	√√	√	√
MATH_EMAVG	√√	√	√
PWMDRV_2ch_UpDwnCnt	√√	√	√
PFC_ICMD		√√	√
CNTL_2P2Z		√√	√
PFC_InvRmsSqr		√√	√
PWMDAC_MACRO	√	√	√
RC_MACRO			√
RG_MACRO			√
IPARK_MACRO			√
SVGENDQ_MACRO			√
PWM_MACRO			√
CLARKE_MACRO			√
PARK_MACRO			√
PHASEVOLT_MACRO			√
QEP_MACRO			√
SPEED_FR_MACRO			√
PI_MACRO (IQ)			√
PI_MACRO (ID)			√
ACI_FE			√
ACI_SE			√
PI_MACRO (SPD)			√
√ means this module used			
√√ means this module is being tested.			

Incremental Build Level 1

This build is used to verify the operation/configuration of the C28x, the peripherals, and some of the Digital Power library modules. Assuming the steps described in the “How to Run Guide” were completed successfully; this section describes the procedure for a “minimum” system check-out. During this process the motor must remain disconnected ([M5]-TB3) and no load should be present at the output of the PFC stage ([Main]-BS4). Open the *HVACI_Sensorless_PFC-Settings.h* and set *INCR_BUILD* to *LEVEL1* (`#define INCR_BUILD LEVEL1`); this will ensure the project is configured to run the first incremental build. Right-click on the project name and select “Rebuild Project”; once the build is complete click the debug button to load the program and reset the device. Click the Continuous Refresh button  on the top right corner of the Expressions Window tab to enable periodic capture of data from the microcontroller.

Enable real-time debugging, enable continuous refresh, and run the code. As a precaution the code is initially disabled and uses the *EnableFlag* variable to start execution; the *BackTicker* variable should be increasing. Set *EnableFlag* to 1 in the *Expressions* window and the *BackTicker* variable should stop incrementing; the *IsrTicker* variable will now start counting. This indicates the system interrupts are working properly.

To verify PWM operation on the PFC stage, the *DutyA* (IQ24) variable can be used to change the PWM duty cycle. Using an oscilloscope probe the PWM-1 and PWM-2 vias found in the [M4] macro and vary the *DutyA* value, within [0. 0, 1.0). The PWM waveforms on the oscilloscope should adjust accordingly; see *Figure 10*. This confirms the Digital Power library modules are working properly.

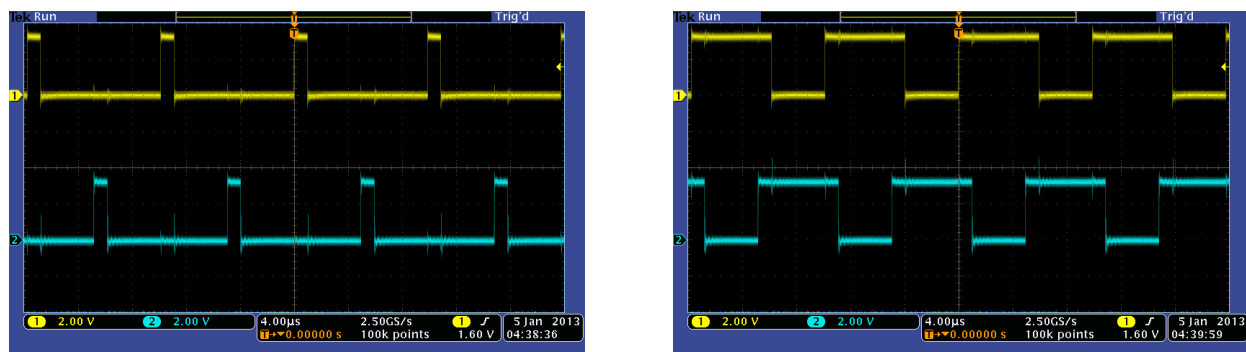


Figure 10: PWM duty cycles, 10% duty (Left), 60% duty (Right).

To verify ADC operation, set the *DutyA* variable to 0.0, this will completely stop PWM generation. View the *Ipfc* (IQ24) variable in the *Expressions* window, this value should be updating constantly. This value represents the offset present in the PFC stage; this value is subtracted from *Ipfc_fldr* (IQ24) within the ISR. *Ipfc_fldr* is used for current control in the following build levels. This confirms ADC operation.



After verification, reduce the AC input voltage, take the controller out of real-time mode, and reset the processor (refer to the “How to Run Guide” for details). This step needs to be repeated after each test for safety purposes. Improper shutdown might halt the PWMs at some certain states where high currents can be drawn; caution needs to be taken when performing these experiments.

Level 1 - Incremental System Build Block Diagram

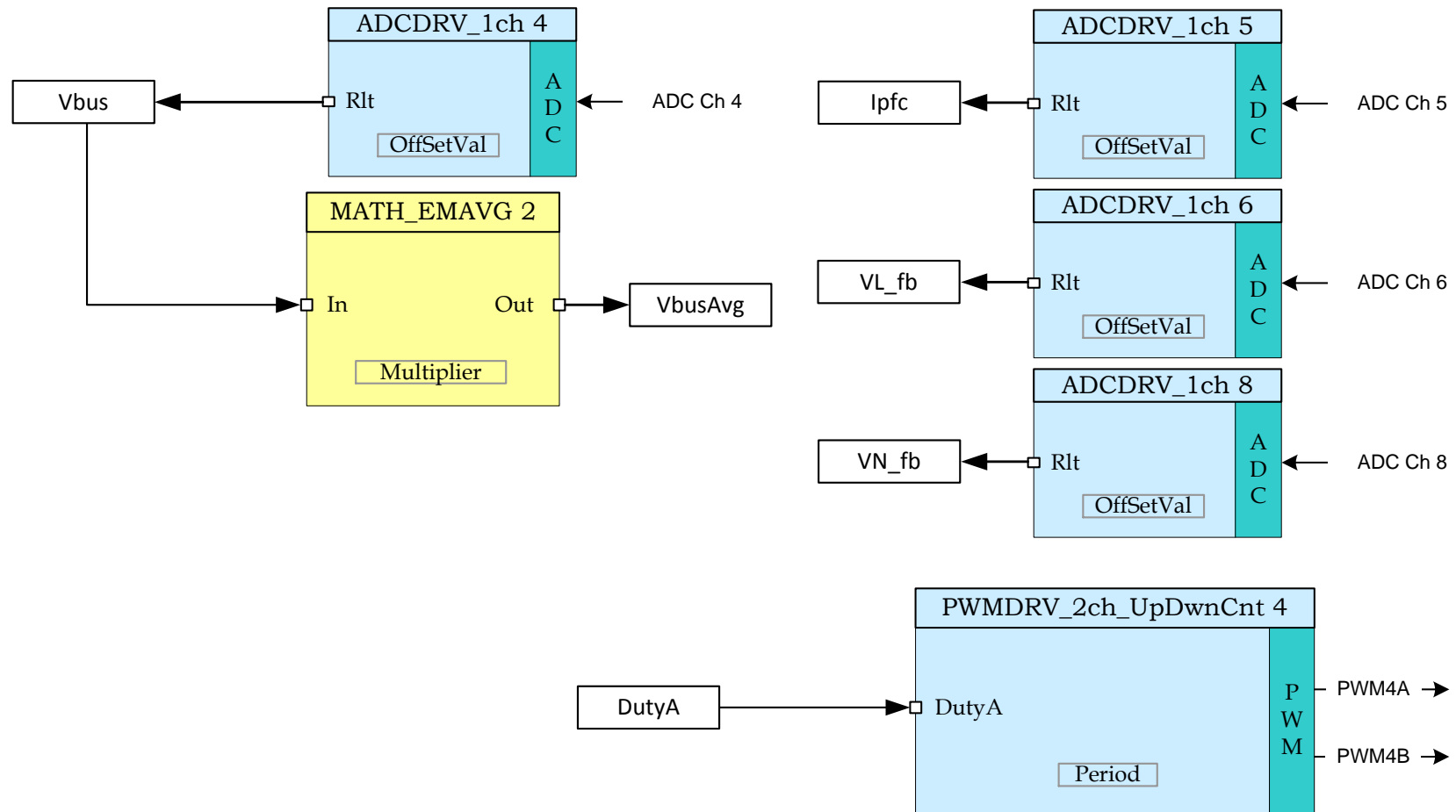



Figure 11: PFC Build Level 1 Block Diagram

Incremental Build Level 2

Assuming the previous build level was completed successfully, this build level is used to verify the operation of the PFC current control loop. During this process the motor must remain disconnected ([M5]-TB3) and a load must be applied at the output of the PFC stage. The PFC stage is rated for a maximum of 700W with a max boosting voltage of 400V. For this build an 80W load should be connected between [Main]-BS4 and [Main]-BS6. A variable AC supply will be required to perform this incremental build. For safety reasons it is recommended to use an appropriately isolated AC source. The supply should be connected to [Main]-P1 and must remain off until indicated otherwise.

Open the *HVACL_Sensorless_PFC-Settings.h* and set *INCR_BUILD* to *LEVEL2* (`#define INCR_BUILD LEVEL2`); this will ensure the project is configured to run the second incremental build. Right-click on the project name and select "Rebuild Project"; once the build is complete click the debug button to load the program and reset the device. Click the Continuous Refresh button  on the top right corner of the Expressions Window tab to enable periodic capture of data from the microcontroller.

Enable real-time debugging, enable continuous refresh, run the code, and set *EnagleFlag* to 1. Slowly increase the AC supply to a maximum of 55V AC. The output voltage should boost as you start to increase the AC supply. Once the AC input voltage exceeds 50V AC, the *SineAnalyzer* module, part of the Digital Power library, begins calculating input frequency (*Gui_Freq_Vin* (IQ06)) and rectified RMS voltage (*Gui_VrectRMS* (IQ06)).

During this test the *DutyA* variable is no longer used, the PWMs are updated by the current control loop. The bus voltage command is controlled by the *VbusVcmd* (IQ24) variable. For safety, any changes to *VbusVcmd* must be done in small increments or decrements. The *VbusVcmd* is used to calculate the current command (*PFC_ICMD* module). The current command is then used by the current controller (*CNTL_2P2Z* module) as the reference. The output of the current controller is the new duty cycle for the PWM module. Other modules are used to calculate the average rectified voltage, average bus voltage and the squared inverse of the RMS voltage. Use an oscilloscope to view the input current waveforms. New current controller coefficients can be tested at this stage.

Bring the system to a safe stop as described at the end of build level 1: reduce the AC input voltage, take the controller out of real-time mode and reset.

Level 2 - Incremental System Build Block Diagram

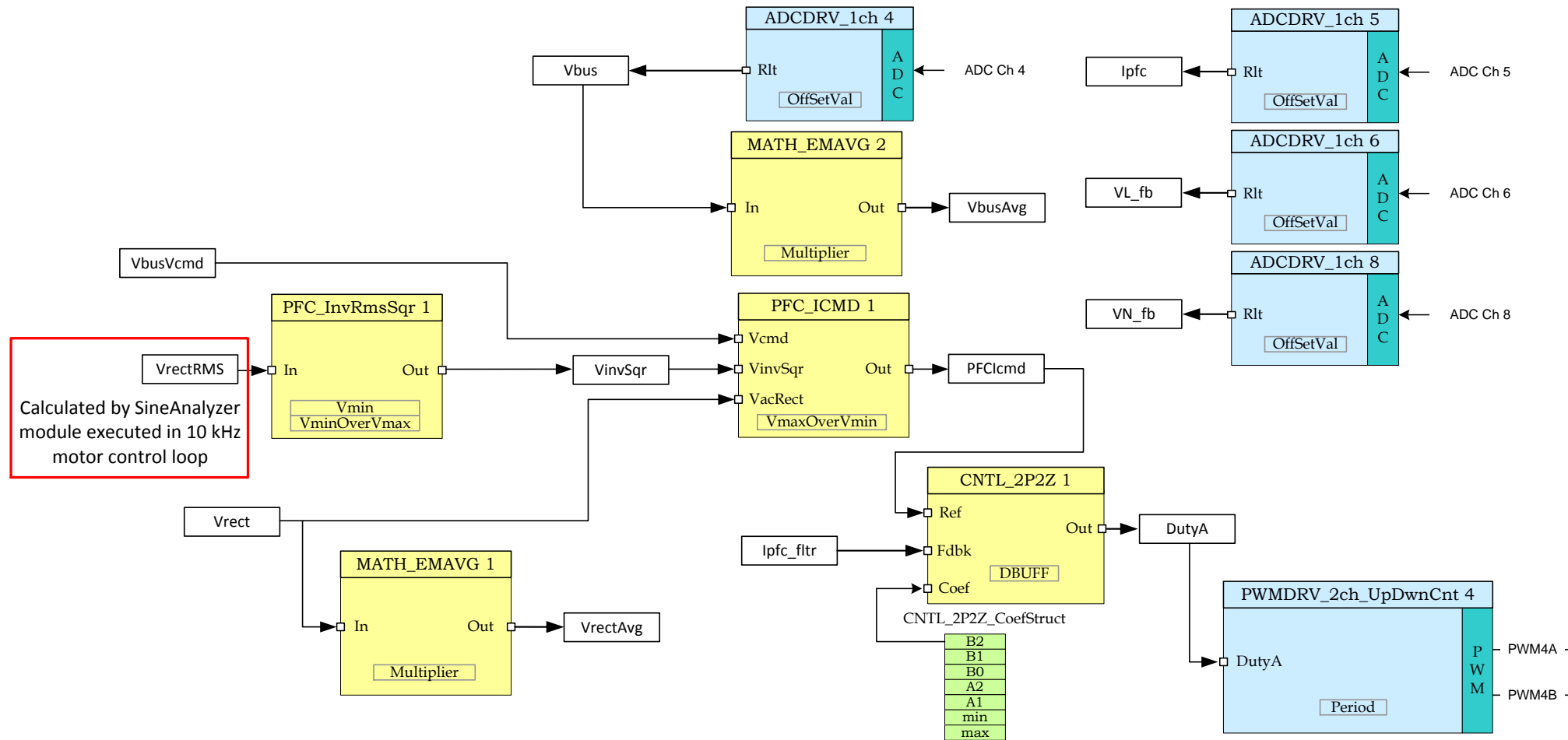



Figure 12: PFC Incremental Build Level 2 Block Diagram

Incremental Build Level 3

Level 3A

Assuming the previous build level was completed successfully, this build level is used to verify complete PFC operation (current and voltage control). During this process the motor must remain disconnected ([M5]-TB3) and a load must be applied at the output of the PFC stage (connected between [Main]-BS4 and [Main]-BS6). The PFC stage is rated for a maximum of 700W with a max boosting voltage of 400V. To test the control algorithm, the loads will be varied from light load to high load for both low line voltage (110V AC) and high line voltage (220V AC) conditions. The use of a bank of resistors is recommended. A variable AC supply will be required to perform this incremental build. For safety reasons it is recommended to use an appropriately isolated AC source. The supply must remain off until indicated otherwise.

Open the *HVACI_Sensorless_PFC-Settings.h* and set *INCR_BUILD* to *LEVEL3* (`#define INCR_BUILD LEVEL3`); this will ensure the project is configured to run the third incremental build. Right-click on the project name and select "Rebuild Project"; once the build is complete click the debug button to load the program and reset the device. Click the Continuous Refresh button  on the top right corner of the Expressions Window tab to enable periodic capture of data from the microcontroller.

Enable real-time debugging, run the code and set *EnableFlag* to 1. During this test the *DutyA* and the *VbusVcmd* variables are no longer used, PFC stage is managed by the current and voltage control loops. To start boosting PFC voltage, this stage uses a slow start method; this method requires a 100V AC minimum input voltage. Slowly increase the AC supply to 110V AC. Once the input is at 110V AC, set the *start_flag* variable to 1; the PFC will begin to boost the bus voltage to approximately 398 VDC. After the voltage is boosted, the *start_flag* will automatically clear and the *run_flag* will be set.

At this point proceed to gradually increase the load on the PFC stage while viewing the input current waveform on an oscilloscope. *Figures 13-15* show the input current under different load conditions. Be sure not to exceed PFC rated power when varying the load. Any load increases/decreases should be done gradually.

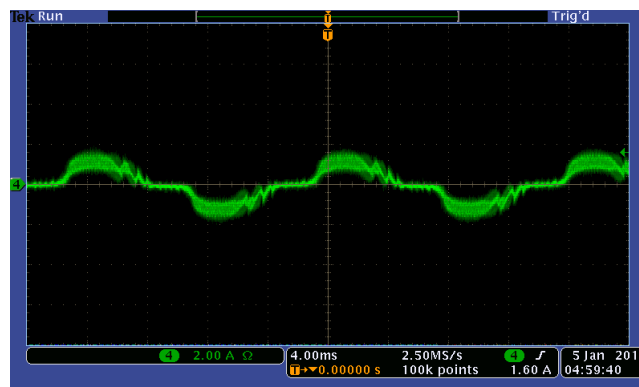


Figure 13: 80 W load, 110V AC input, 387V output, PF = 0.943, CF = 1.62, I = 0.77 A

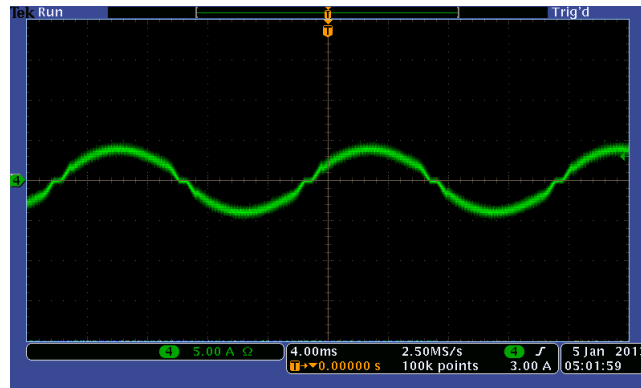


Figure 14: 320 W load, 110V AC input, 387V output, PF = 0.996, CF = 1.39, I = 2.91 A

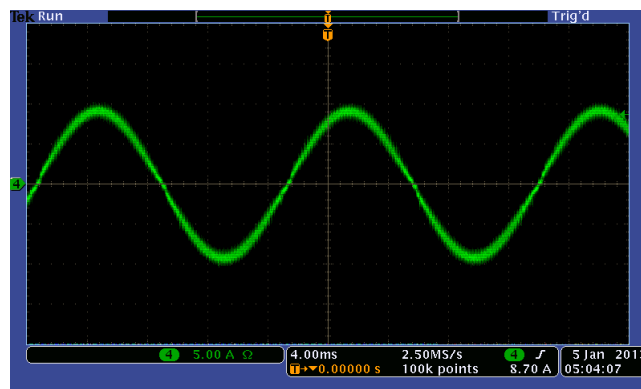


Figure 15: 685 W load, 110V AC input, 387V output, PF = 0.998, CF = 1.44, I = 6.22 A

Once complete bring the system to a safe stop as described at the end of build level 1: reduce the AC input voltage, take the controller out of real-time mode and reset. If desired, repeat the previous steps for an AC input of 220V. Figures 16-18 show current waveforms of operation with 220 VAC input.

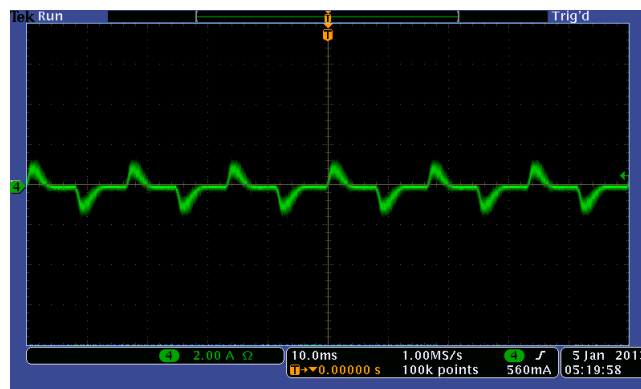


Figure 16: 77.6 W load, 2200V AC input, 387V output, PF = 0.776, CF = 2.12, I = 0.45 A

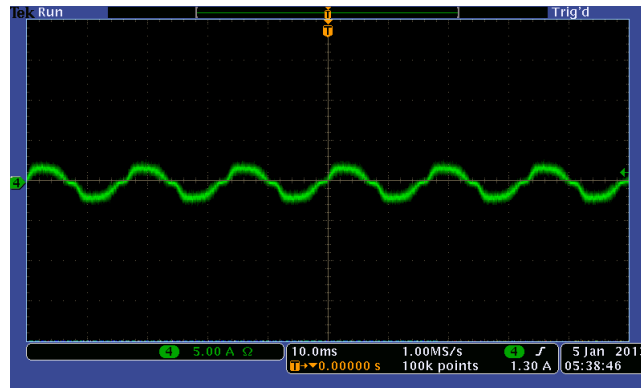


Figure 17: 310 W load, 220V AC input, 387V output, PF = 0.990, CF = 1.47, I = 1.41 A

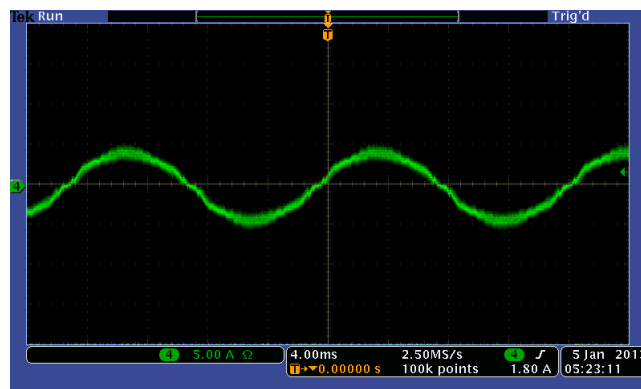


Figure 18: 681 W load, 220V AC input, 387V output, PF = 0.997, CF = 1.43, I = 3.09 A

Level 3B

Assuming the previous steps were completed successfully, the next step is to run the motor with PFC. Ensure the AC supply is completely powered off and the processor has also been reset before proceeding. Disconnect any of loads present on the PFC output. Use a banana cable to connect the PFC output to the motor inverter bus voltage (connect [Main]-BS4 to [Main]-BS5). Finally connect the AC induction motor to the [M5]-TB3 connector.

As previously explained, raise the AC input supply to 110V or 220V AC and set the *start_flag* to 1. Once the PFC voltage has boosted (*run_flag* set to 1), set the *run_motor* flag to 1 to start motor control. The motor will not run if the bus voltage is less than 350V DC. At this point, the motor control algorithm is running simultaneously with the PFC control algorithm. The motor speed can be changed by using the *SpeedRef* variable. *Figure 19* below show the input current waveform for an unloaded motor.

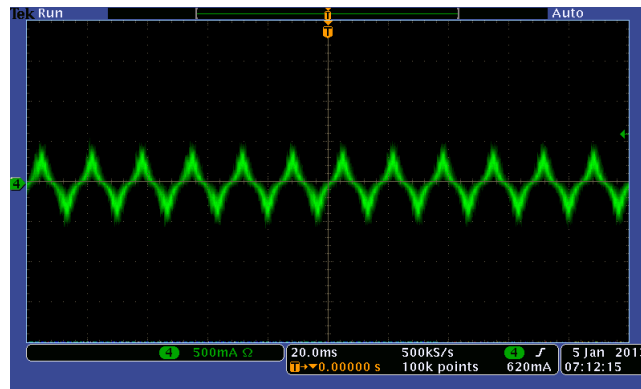
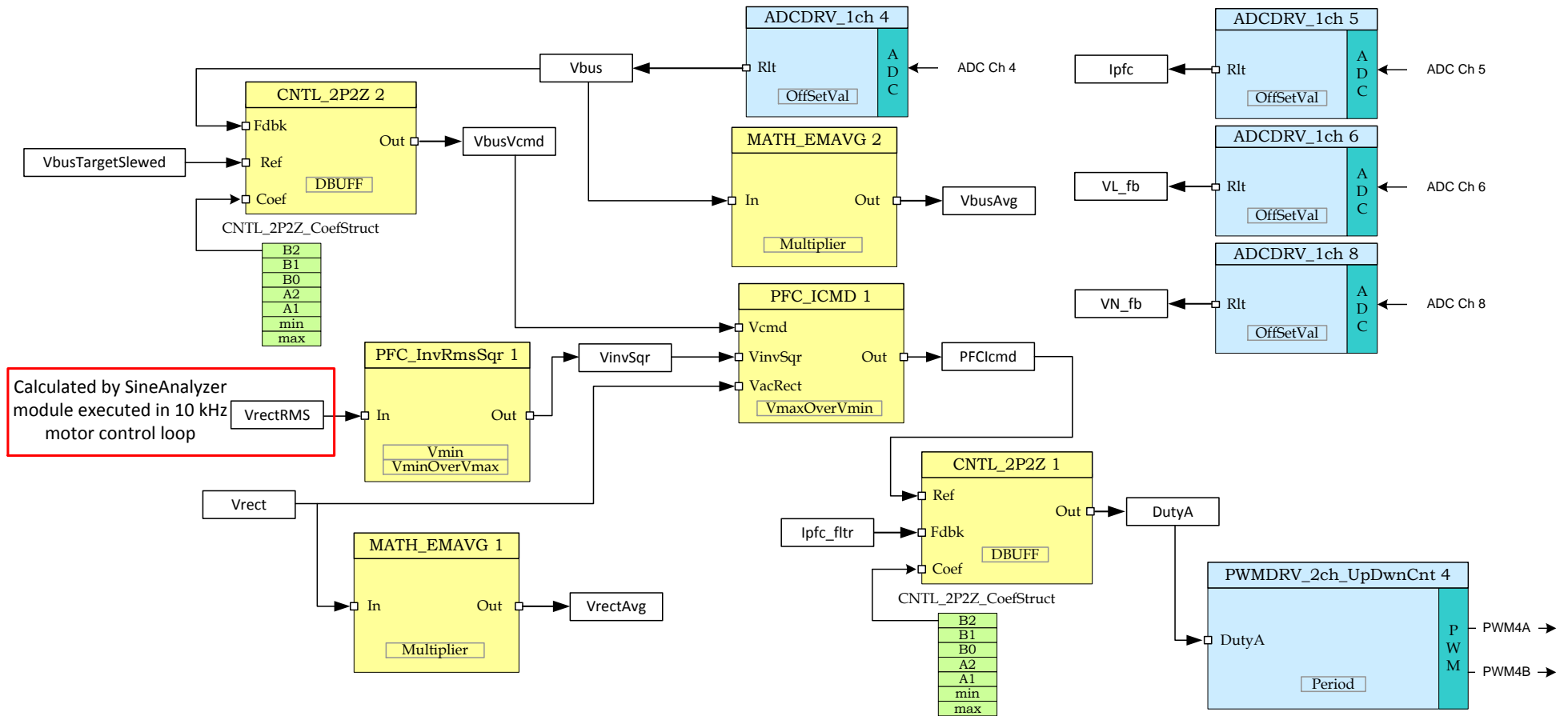


Figure 19: Unloaded motor - 40 W load, 110V AC input, 387V output, PF = 0.979, CF = 1.65, I = 0.36 A

To further test motor control with PFC, a dynamometer can be used to load the motor. Once complete, lower the motor speed ($SpeedRef = 0.05$) and bring the system to a safe stop as described at the end of build level 1: reduce the AC input voltage, take the controller out of real-time mode and reset. If desired, repeat the previous steps for an AC input of 220V.

Level 3 - Incremental System Build Block Diagram



Level 6A – AC Induction Motor Sensorless FOC Algorithm

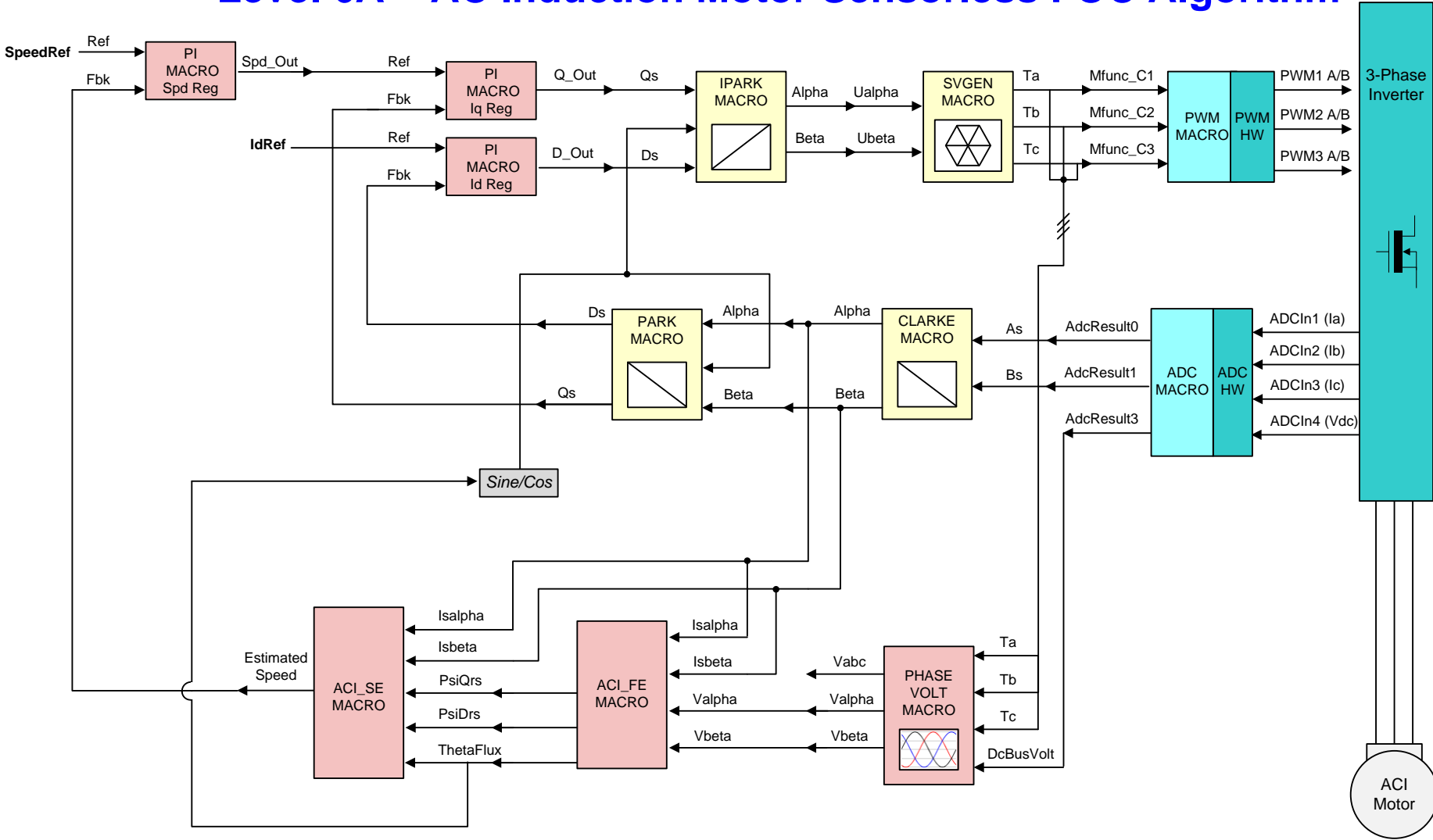


Figure 10: Sensorless FOC Motor Control Algorithm Block Diagram

References

For more information refer to the following:

1. C2000 Devices:
<http://www.ti.com/C2000>
2. controlSUITE for C2000 software:
http://www.ti.com/lscs/ti/microcontroller/32-bit_c2000/tools.page
3. Code Composer Studio:
<http://www.ti.com/tool/ccstudio>
http://processors.wiki.ti.com/index.php/Download_CCS
4. C2000 Digital Power Library:
controlSUITE\libs\app_libs\digital_power
5. Digital Motor Control Library:
controlSUITE\libs\app_libs\motor_control
6. High Voltage Motor Control + PFC kit:
controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0
<http://www.ti.com/tool/tmdshvmtrpfckit>