Iowa State University
Electrical and Computer Engineering
E E 452. Electric Machines and Power Electronic Drives

Laboratory #14
Squirrel-cage Induction Motor – Field Oriented Control (Part 2)

Summary

This lab continues to build on the work done in Lab 13. A closed-loop speed controller using Field Oriented Control (FOC) will be implemented. The TMDSHVMTRPFCKIT comes with a set of CCS example projects, including a FOC application pre-tuned for the Marathon Electric 5K33GN2A squirrel-cage induction motor. This project will be incrementally built using CCSv5, and will run on the F28035 MCU. The machine's response to a load step change will be analyzed.

Learning objectives

- Learn about Field Oriented Control through a CCS example project.
- Use CCS to incrementally build the project, and view system variables in real time.
- Analyze the dynamic response of the machine for varying load.

Background material (should be read before coming to the lab)

- *TMDSHVMTRPFCKIT How to Run Guide* (*HVMotorCtrl+PFCKit_HowToRunGuide.pdf*)
- *Sensored Field Oriented Control of 3-Phase Induction Motors* (*Sensored FOC of ACI.pdf*)
- Krause chapter 6.11 (*Introduction to Field-Oriented Control*)

Exercises and Questions

Instructions: every student should deliver his/her own report at the end of the lab session, even though the experiments are conducted in groups. You may want to answer the questions as you go along the exercises. Time yourselves according to the recommendations below.

1. Pre-lab assignment

This lab exercise follows the example project described in *Sensored FOC of ACI.pdf*. Please refer to that document, and any other documents referenced there, for further details regarding the material contained in this laboratory manual. All documents and project files can be found in ControlSUITE.

Note: All figures and page numbers referenced in this laboratory manual, refer to those figures and pages in *Sensored FOC of ACI.pdf*.

Review the work you did in Lab 13. You should have already completed the first two, maybe three, build steps depending on your pace. The basic FOC scheme is shown in Figure 7, on page 8. Build level 1 verified the *Inverse Park* transformation*, Space Vector Modulator,* and *IGBT PWM* output*.* Build level 2 verified the *current sensors* and the *Clarke* and *Park* transformations. Build level 3 verified the *eQEP speed and position* feedback.

Build level 4 will verify the *Current Model*, and build level 5 will bring all of the components together to form a closed loop controller.

DELIVERABLE 1: Explain your progress from Lab 13, and explain what you will do in Lab 14.

2. Hardware Setup [15 minutes]

Follow the instructions of page 16, *Hardware Setup Instructions,* to configure the TMDSHVMTRPFCKIT for running this project. Make sure the jumper from *BS1* to *BS5* is NOT installed. Generate the DC-bus from the Lab-Volt 3-ph variable power supply. Connect a 1.2 kΩ resistor in parallel with the DC-bus, to discharge the bus upon powering down the circuit.

Warning! Install the plastic cover before applying the DC-bus voltage.

Warning! The DC-bus ground is not connected to Earth ground. Do not connect the oscilloscope ground to the DC-bus ground. Damage to the oscilloscope will result. Use isolated differential probes for the duration of this project.

CAUTION!
Verify your hardware configuration with your group members and your T.A.

3. **Software Setup** [15 minutes]

Open CCS and import the example project as you did in Lab 13. If the project already exists in the workspace, right click on the project and remove it from the workspace. Import the original file from

C:\TI\controlSUITE\development_kits\HVMotorCtrl+PfcKit_v1.6\HVACI_Sensored.

Right-click on the project and set it as the active project, it should turn bold.

Go to *Project Properties*, and verify the C2000 compiler is set to *v5.2.3* or higher.

Refer to your progress of Lab 13. Set the build level appropriately. To set the *Build Level*, open the file *HVACI_Sensored-Settings.h*, and scroll to line 27, where you can then set the build level.

Right-click on the project and set the *Build Configuration* as *F2803x_RAM*. Build the project and launch a debug session. There should be no errors during the build.

Click *Enable silicon real-time* mode and *Enable polite real-time mode* as described in *HVMotorCtrl+PFCKit_HowToRunGuide.pdf.*

Caution! Disable these two real-time modes before performing a *Reset* to avoid connectivity problems.

To watch the value of system variables, go to View/Expressions and add the variables you wish to monitor. Enter the variable names as described in Table 1 on Page 19. You can add variables as needed.

Add two graphs as described on Page 19, and click to *Continuously Refresh*. Make sure the *Start Address A* and *Start Address B* have the following entries:

DLOG_4CH_buff1
DLOG_4CH_buff2
DLOG_4CH_buff3
DLOG_4CH_buff4.

Data Log and *DAC* signals can be identified in *HVACI_Sensored.c* in the respective build level section, between lines 506 and 1038.

Continue following the instructions of *Sensored FOC of ACI.pdf* to complete all five build levels.

4. **Continue with the Incremental Build Process** [140 minutes]

Follow the instructions of *Sensored FOC of ACI.pdf* to complete the incremental build process, and verify the contents of the field oriented control application.

Note: When you move to a new build level, you may have to adjust the gain values that you tuned in the previous build levels; each build level has default values.

Attention: Do not get ahead of yourself. There are many steps in this procedure, and it is critical that you pay attention to details. If something does not seem to be working properly, go back and make sure that what you expect to happen is what is truly meant to be happening for the build level you are on. This example project is designed to work with the kit and motor you are using. Understand the block diagram associated with each build level.

DELIVERABLE 2: In build level 3, what are the *Proportional, P,* and *Integral, I,* gain values you chose to control the current?

DELIVERABLE 3: In build level 3, what is the slip, in pu? Reduce the value of *IdRef* to 0.01pu. What is the new slip, and why did it change the way it did? What happens if you set *IdRef* to 0, and why?

CAUTION!
When you come to build level 5, do not follow the example project instructions regarding the startup procedure, as described on page 33. Instead, do the following:

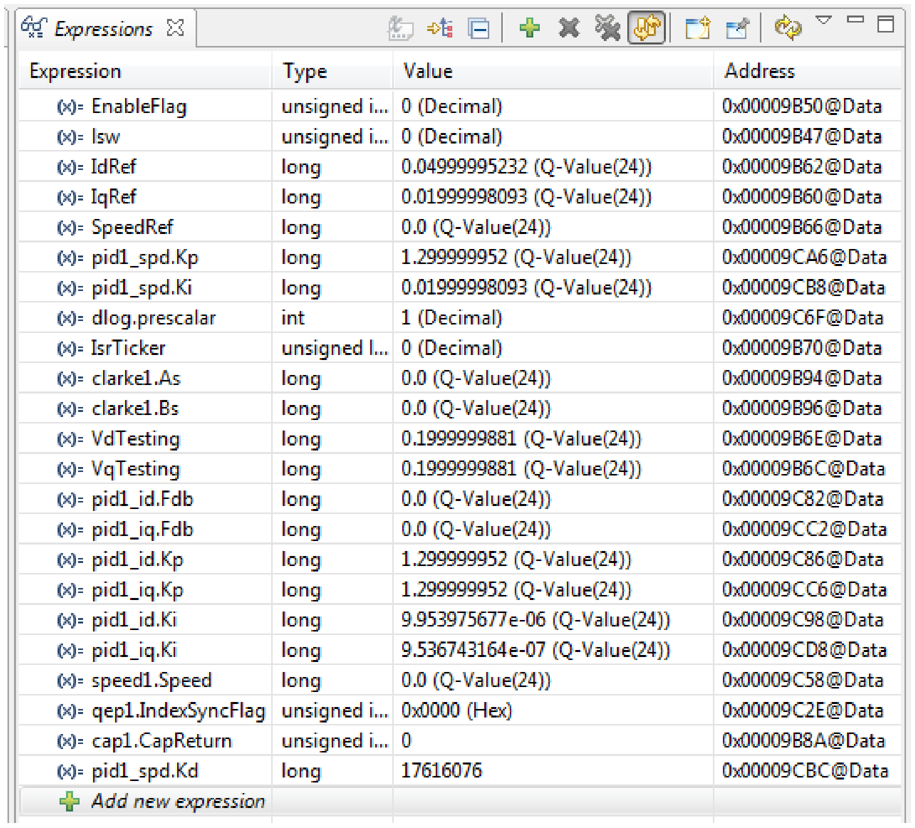
1) Build, download, enable real-time, and run the application.
2) Ensure *Enable Flag* is 0, and set *SpeedRef* to -0.3 or another desired value.
   Set speed is negative to make the sensed direction match the commanded one.
3) Turn up the DC-bus to the proper level (294 V for 208 V line-to-line using SVM).
4) Set *Enable Flag* to 1. The motor should ramp up to the set speed.
5) Verify that *speed1.Speed* follows *SpeedRef*.

Also, notice that the qd current PID gain values are set to a default value for each build level. While running the motor with the default values, you will probably observe that the qd currents of *DLOG_4CH_buff1* and *DLOG_4CH_buff2* are not sinusoidal, and there is excessive vibration. If so, the integral gains of the qd current PID regulator are probably too high.

Set *pid1_id.Ki* and *pid1_iq.Ki* to zero; the currents should now be sinusoidal, but the feedback values *pid1_id.Fdb* and *pid1_iq.Fdb* do not follow the reference values *IdRef* and *IqRef*. In build level 3, you found a set of *proportional* and *integral* gains such that the actual qd currents followed the reference qd currents. Set the current regulator gains to the values you found in Build Level 3. Do not change the value of *IqRef*; this is actually the output of the speed PID regulator, as shown in the diagram on page 35. If the feedback *speed1.Speed* does not follow the reference *SpeedRef*, adjust the gains for the speed PID regulator *pid1.spd.Kp* and *pid1.spd.Ki*. Although a more rigorous approach is usually necessary in obtaining desired control gains, precise machine control is not the objective of this lab. You can simply use trial and error to settle on a set of gains.

DELIVERABLE 4: In build level 5, run the motor connected to the dynamometer with a reference speed of 0.3 pu. Run the dynamometer in either *Pos. CT Brake* or *Neg. CT Brake* mode, depending on your direction of rotation. Observe stator current *Isq* via *DAC2*. Step change the load, either increasing or decreasing, and observe the response of *Isq*. Qualitatively, how does *Isq* respond to the change? How does the frequency change? How does the amplitude change? Does the rotor speed change? Why does the machine respond the way it does? Change the reference speed. Does the machine track the speed command?

For your reference, the variables you may be interested in are shown here. Additionally, you can explore the project code for other variables of interest.



Expression	Type	Value	Address
(x)= EnableFlag	unsigned i...	0 (Decimal)	0x00009B50@Data
(x)= Isw	unsigned i...	0 (Decimal)	0x00009B47@Data
(x)= IdRef	long	0.04999995232 (Q-Value(24))	0x00009B62@Data
(x)= IqRef	long	0.01999998093 (Q-Value(24))	0x00009B60@Data
(x)= SpeedRef	long	0.0 (Q-Value(24))	0x00009B66@Data
(x)= pid1_spd.Kp	long	1.299999952 (Q-Value(24))	0x00009CA6@Data
(x)= pid1_spd.Ki	long	0.01999998093 (Q-Value(24))	0x00009CB8@Data
(x)= dlog.prescalar	int	1 (Decimal)	0x00009C6F@Data
(x)= IsrTicker	unsigned l...	0 (Decimal)	0x00009B70@Data
(x)= clarke1.As	long	0.0 (Q-Value(24))	0x00009B94@Data
(x)= clarke1.Bs	long	0.0 (Q-Value(24))	0x00009B96@Data
(x)= VdTesting	long	0.1999999881 (Q-Value(24))	0x00009B6E@Data
(x)= VqTesting	long	0.1999999881 (Q-Value(24))	0x00009B6C@Data
(x)= pid1_id.Fdb	long	0.0 (Q-Value(24))	0x00009C82@Data
(x)= pid1_iq.Fdb	long	0.0 (Q-Value(24))	0x00009CC2@Data
(x)= pid1_id.Kp	long	1.299999952 (Q-Value(24))	0x00009C86@Data
(x)= pid1_iq.Kp	long	1.299999952 (Q-Value(24))	0x00009CC6@Data
(x)= pid1_id.Ki	long	9.953975677e-06 (Q-Value(24))	0x00009C98@Data
(x)= pid1_iq.Ki	long	9.536743164e-07 (Q-Value(24))	0x00009CD8@Data
(x)= speed1.Speed	long	0.0 (Q-Value(24))	0x00009C58@Data
(x)= qep1.IndexSyncFlag	unsigned i...	0x0000 (Hex)	0x00009C2E@Data
(x)= cap1.CapReturn	unsigned i...	0	0x00009B8A@Data
(x)= pid1_spd.Kd	long	17616076	0x00009CBC@Data
Add new expression			

5. **Conclusion** [10 minutes]

Write about one or two things you learned in this lab that you think are important or interesting, and why.