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

Laboratory #10
Quadrature Encoder Basics

Summary

Machine control techniques often require accurate speed and/or position feedback information. In this lab, you will learn how to use a quadrature encoder to determine rotor speed and angular position. These concepts and methods can be applied to any machine equipped with a quadrature encoder.

Learning objectives

- Use a quadrature encoder to measure rotational speed and angular position.
- Use the TI C2000 enhanced Quadrature Encoder Peripheral (eQEP) to process encoder data.
- Finish measuring SCIM torque vs. speed curves.

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

- TI C2000 eQEP peripheral user guide (eQEP_User_Guide_sprufk8.pdf).
- Wind Energy notes on squirrel-cage induction machines (Chapter 2)

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

Rotor feedback, for drive control applications, is often provided through the use of an encoder. An encoder is device (optical or magnetic) which converts rotational speed and position information to an electrical signal. We will study the commonly used *quadrature encoder*. It provides a two bit digital signal, which is then processed by a MCU. The signal is produced by means of a rotating disc with lines cut about the circumference. Two optical switches turn on and off as the lines rotate past them. The optical switches are offset from each other by ¼ line. An index signal is also provided to give a definition of the home position. Figure 1 shows the relationship of the mechanical disc to electrical data outputs.

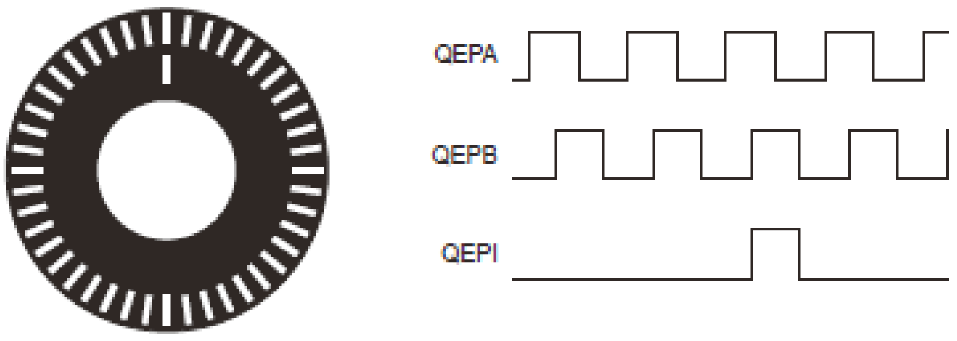

Figure 1. Encoder construction and data format (from eQEP_User_Guide_sprufk8.pdf).

Examine the *QEPA* and *QEPB* bits. Notice there are actually four discrete positions for each pulse; (*QEPA*,*QEPB*) = (0,0) (1,0) (1,1) and (0,1). This is why the encoder is called a "quadrature" encoder; there are four position events per line. For an encoder with 1000 lines per revolution, there are 4000 discrete positions, giving a resolution of 360/4000 = 0.09° per position event. Also notice that the index (*QEPI*) occurs only once per revolution.

It is possible to determine the direction of rotation (clockwise or counter-clockwise), by checking if *QEPA* is leading *QEPB*, or if *QEPB* is leading *QEPA*. The position of the rotor can be determined by integrating the number of position events as the disc rotates. The speed of the rotor, ω, can be determined either by measuring the amount of time, ΔT, between N number of positions

$$\omega = \frac{N}{\Delta T}, \tag{1}$$

or by measuring the change in position, ΔN, over a specified period of time, T,

$$\omega = \frac{\Delta N}{T}. \tag{2}$$

Study the encoder attached to your SCIM; the datasheet is provided with the Lab 10 supporting documents (*qd200_datasheet.pdf*). *Hint:* The part number on the device is 3138A001-2048-0; this is slightly different from the "ordering number" in the datasheet.
DELIVERABLE 1: How many pulses per revolution does this encoder have? What is the position resolution of this encoder?

The F28035 eQEP can be configured to implement either equation (1) or equation (2). In this lab, we will configure the eQEP to implement equation (1).

There is a limit to the speed range at which this technique can be applied. A digital counter can only count to a maximum number determined by the number of bits in its register. Depending on the frequency of the clock driving the counter, there is an upper limit to the amount of time that can occur between position events, before the counter overflows. For a 16 bit counter with a fixed clock frequency, the minimum measureable speed can be determined by

$$\Omega_{min} = \left(\frac{65535\ clock\ cycles}{\#Position\ Events} * Clock\ Period * \frac{\#Positions}{1Revolution} * \frac{1\ min}{60\ sec}\right)^{-1}, rpm. \tag{3}$$

Clock Period is derived from the MCU system clock (60 MHz for the F28035), and can be scaled by a prescale divider, allowing measurement of slower speeds. # *Position Events* can be set to a multiple of quadrature position events.

The maximum speed that can be measured, using the time between position events, occurs when there is one clock pulse between consecutive position events. The maximum measureable speed can be determined by

$$\Omega_{max} = \left(\frac{1\ clock\ cycle}{\#Position\ Events} * Clock\ Period * \frac{\#Positions}{1Revolution} * \frac{1\ min}{60\ sec}\right)^{-1}, rpm. \tag{4}$$

Example: Measuring the time between four position events, with a clock frequency of 60 MHz, and an encoder resolution of 8192 positions per revolution, yields a minimum measureable speed of

$$\Omega_{min} = \left(\frac{65535\ cycles}{4\ Positions} * \frac{1\ sec}{60*10^6\ cycles} * \frac{8192\ Positions}{1Rev} * \frac{1\ min}{60\ sec}\right)^{-1}, rpm.$$

$$\Omega_{min} = 26.8\ rpm.$$

Note: As fewer clock pulses occur between position events (higher speeds) the resolution of the speed information degrades.

DELIVERABLE 2: At what speed is the resolution reduced to 100 clock cycles per position event?

2. Quadrature Encoder – Simulink Control Model [90 minutes]

Open a new Simulink model to build a controller for using the F28035 eQEP. Add the *Target Preferences* block to configure the model for your device. Add the *F28035 eQEP* block to the model. You should also add the heartbeat function, and two PWM outputs to provide real-time analysis of system variables, as you have done in previous labs. Use *GPIO31* for heartbeat, *GPIO34* for direction, *DAC1* for angular velocity, and *DAC2* for angular position.

In the *Simulation Configuration Parameters*, go to *Code Generation/ Interface*. Set the *Target Function Library* to be *TI C28x*. This will allow for optimal code generation for some functions in the model, including multiply, divide, etc. On the *Solver* tab, check the box to *Automatically handle rate transition*.

When you are finished configuring the model, you should have something similar to that of Figure 2.

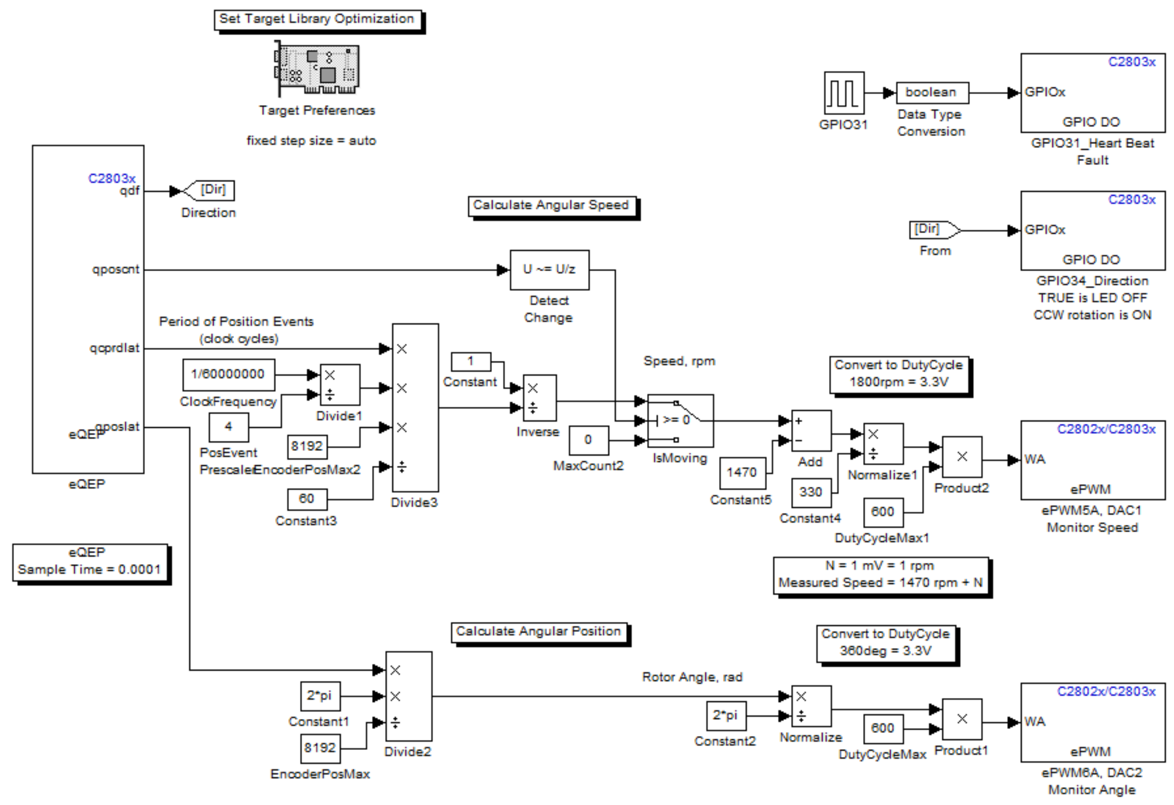

Figure 2. Simulink block diagram for reading and interpreting quadrature encoder data.

Open the eQEP block. In the *General* tab, set the *Position Counter* to operate in *Quadrature-count* mode. Define *Positive rotation* as *Clockwise*. Check the box to enable the *Quadrature direction flag output port*. Also check the box to enable *Index pulse gating on*. Set the *Sample time* to 100 µs.

In the *Position Counter* tab, set the *Maximum position counter value* to 8192, which is four times the number of lines in one revolution. Check the box to *Enable software* initialization, and set the *Initialization value* to 0. Set the *Position counter reset mode* to *Reset on an index event*; this will reset the counter to 0 each time an index pulse occurs.

In the *Speed Calculation* tab, check the box to *Enable eQEP capture*. Check the boxes for *Output capture timer period latched value* and *Output position counter latched value.*

Use equations (3) and (4) to choose an appropriate *eQEP capture timer prescaler* (this scales the system clock) and *Unit position event prescaler* (this is the number of positions to count the time between). In the dropdown menu for *Capture timer and position*, select *On unit time-out event*. Set the *Unit timer period* for a number of clock cycles in which to latch the eQEP output values; make sure this period of time equal to the *Sample time* of the eQEP block. The eQEP block is now configured.

DELIVERABLE 3: For the clock frequency, position event prescaler, and clock period of your choice, what are the maximum and minimum speeds that can be measured? If this does not seem like a reasonable range, select different values.

The rotor position, at the time of sampling the eQEP block, is obtained from the eQEP *qposlat* output. The time, *ΔT,* between *N* position events is obtained from the *qcprdlat* output, and is given in number of clock cycles.

Add additional blocks to the model to convert the eQEP outputs to actual values of angular position and rotational velocity.

Add two PWM blocks to the model to drive *DAC1* and *DAC2*. These analog signals will represent values of the speed and position calculations. Add an offset of 1470 rpm to the speed output, such that 0 V = 1470 rpm, 3.3 V = 1800 rpm; you should then see a resolution of 10 rpm/mV on the oscilloscope. Configure the angle output such that 360° = 3.3 V.

3. **Quadrature Encoder Measurements** [75 minutes]

Build, download, and run the application. Connect the SCIM to the Lab-Volt three phase power supply.

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

While monitoring the motor line current, increase the supply voltage to spin the motor. Verify that the speed and position change accordingly. Do not operate the motor beyond the rated current for an extended period of time; damage may result!

Finish constructing the torque vs. speed curves that you started in lab 9, using the speed information available from the encoder and MCU. On a single plot, plot the actual torque vs. speed curve for a frequency of 60 Hz, with a line-to-line voltage of a) 208 V b) 150 V and c) 100 V. Superimpose the theoretical torque speed curves for each case; use your MATLAB script from lab 9.

DELIVERABLE 4: If you have not already done so, submit a torque vs. speed curve that you measured for the SCIM.

If you have already completed the measurements of Lab 9, do one of the following:

1) Try to configure the MCU and eQEP to measure the speed using the number of position events, ΔN,in a specific period of time, T, as described by equation (2). The eQEP output *qposcnt* is the number of positions, ΔN, in the specified *Unit Timer Period*. You may need to change this period and the block sample time.

   See *eQEP_User_Guide_sprufk8.pdf* and *System_Control_Interrupts_sprugl8b.pdf* for additional details. These documents are available through ControlSUITE.

2) Sketch a block diagram of a motor controller implementing a speed control algorithm. Try to incorporate the position and velocity functions to your existing V/Hz Controller. You may need to modify the existing controller.

DELIVERABLE 5: Submit any plans, sketches, models, etc. for the task chosen above.

4. **Conclusion** [15 minutes]

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