

91.450, Robotics I
Fall 2002
Prof. Yanco

Lab 5: Shaft Encoders and Gears

Out: Thursday, 3 October 2002

Due: Thursday, 10 October 2002

Shaft encoders can be used to count the number of times a wheel spins, or in general the number of digital pulses seen by an input. For this lab, we will use optical encoders that use optical switches whose beam is periodically broken by a slotted wheel (we will use a Lego pulley wheel with holes as our slotted wheel).

Also in this lab, you will experiment with gears and gear trains.

Part I: Shaft Encoders

Read the following information from the IC documentation, as well as the attached information on encoders.

The following information is from the IC documentation:

Shaft encoders are implemented using the input timer capture feature on the 6811. Therefore processing time is only used when a pulse is actually being recorded, and even very fast pulses can be counted. Digital ports 0 and 1 are two input capture channels which are available for use on the Handy Board, so two channels of shaft encoding are supported.

The encoding software keeps a running count of the number of pulses each enabled encoder has seen. The number of counts is set to 0 when a channel is first enabled and when a user resets that channel. Because the counters are only 16-bits wide, they will overflow and the value will appear negative after 32,767 counts have been accumulated without a reset.

As shaft encoders are an optional feature, the library routines which read them are not loaded on start up. In order to load the following routines for use in your programs, load the file `encoders.lis`. This file is in the standard IC library directory.

The actions of the shaft encoders are commanded and the results are read using the following routines. The argument `encoder` to each of the routines specifies which shaft encoder the function should affect. This value should be 0 for digital port 0 or one for digital port 1. Arguments out of the range 0 to 1 have no useful effect.

```
void enable_encoder(int encoder)
```

Enables the given encoder to start counting pulses and resets its counter to zero. By default encoders start in the disabled state and must be enabled before they start counting.

```
void disable_encoder(int encoder)
```

Disables the given encoder and prevents it from counting. Each shaft encoder uses processing time every time it receives a pulse while enabled, so they should be disabled when you no longer need the encoder's data.

```
void reset_encoder(int encoder)
```

Resets the counter of the given encoder to zero. For an enabled encoder, it is more efficient to reset its value than to use `enable_encoder()` to clear it.

```
int read_encoder(int encoder)
```

Returns the number of pulses counted by the given encoder since it was enabled or since the last reset, whichever was more recent.

Install encoders on your robot's wheels. See the attached slides for encoder attachment advice. Enable the encoders and play with reading and resetting them as line commands.

Now write a program that attempts to maintain constant velocity on the drive wheel by varying the power level delivery to the motor (Martin p. 137, Exercise 4). Experiment with the system by holding the robot in the air and applying pressure to the drive wheel. Is the system able to maintain the velocity? Why happens if you suddenly remove the pressure?

Part II: Gears and Gear Trains

Read Martin, Sections 4.2 and 4.5. Do Exercise 4.2.2 on p. 146.

Do Exercise 3 in Section 4.5.4 on p. 173 of Martin. Show me your two designs. (You should have enough Lego to build two different chasses.)

I'd also recommend that you try Exercise 2 on p. 173, but there's nothing to turn in for this.