

Additional information on Lab #5:

The configuration data is located in

C:\NBurn\MOD5234\include\ETPU\functions_API\eptu_sm.h

The source code is in

C:\NBurn\MOD5234\system\ETPU\functions_API\etpu_sm.c

Pin Sequences:

The TPU and code for this processor allows us to specify one pin sequence for each motor that we configure. In an ideal world, we would specify a pin sequence for each channel that we use to run the motors. For us, that would be four channels: TPU12, TPU13, TPU14, TPU15. The enables and the drives cannot be expressed as one pin sequence that is time-shifted for the each channel, so we need to isolate the channels that are time-shifted versions of each other.

In the full step version that is provided to you, the ENABLE channels are always “on”, therefore we don’t include them as a part of our motor. The drive channels are variable and, more importantly, time shifted versions of each other. The ENABLE channels are set high by using the two $J2[\text{channel} + 7] = 1$ and $J2[\text{channel} + 8] = 1$ calls in stepper.cpp.

In the half step version that you are required to implement, you must determine which of the channels are time-shifted versions of each and group them together into a “virtual” motor. Given the new configurations above that should be fairly straightforward. The pin sequences are included in the new configurations. Convince yourself that these pin sequences are correct.

Code changes:

I have removed the back-door global variables and inserted two new configurations.

I have added two new configurations to those available for passing to fs_etpu_sm_init

CMPE401_ETPU_SM_2PHASE_HALF_STEP_ENA
CMPE401_ETPU_SM_2PHASE_HALF_STEP_DRIVE

C:\NBurn\MOD5234\include\ETPU\functions_API\eptu_sm.h

C:\NBurn\MOD5234\system\ETPU\functions_API\etpu_sm.c

I have uploaded the student.cpp/.h and index.htm files to CVS to make sure that everyone has the same starting code. Make sure to save your existing code before

checking out new files from CVS. Using the Import... menu option to get a specific file is OK too.

Expectations:

Ex 1:

Once you download the new code from CVS, inserting data into the text fields will succeed. The Steps text box will move the motor by the indicated number of steps. Inserting 100 into the text box will move the motor one rotation clockwise if your hardware is correct and the stepper motor cable is plugged in with the correct polarity. Inserting -100 into the text box will move the motor one rotation counter-clockwise. This assumes that your hardware is hooked up properly.

The other text boxes will succeed from a form point of view but will do nothing because that is the topic of exercise #2.

Ex2:

Students will have range checking in place and the speed profile can be changed for full step mode. Junk in the Start period and Slew period fields will be rejected. If the Start period and Slew period fields are rejected, two possible results are acceptable. The motor is stepped the number of steps requested with a known good speed profile, or the motor is not stepped at all. Your choice. Bizarre motor behaviour is not acceptable. I will not be testing the Step range checking because this is not part of your coding exercise. I do not expect you to correct provided code unless explicitly told to.

Ex3:

Half step mode works correctly and speed profile changes for both half-step and full-step mode work correctly.

Test Suite:

I will be posting my test suite to the web ASAP so that you can be aware of what will be tested and what will be the expected result.

Nancy
November 24th, 2008