Lab #2: A Task

In this exercise, you will write a new task class. The exercise will take one week; you will work in teams as usual. Skills to practice in this lab include the following:

- Creating a task class with the ME 405 task class structure
- Instantiating multiple tasks for multiple motors
- Inter-task communication in a thread-safe manner
- Documenting code with Doxygen
- Testing your code thoroughly to ensure that it is reliable

1 The Project

Write a task class which controls a motor using the motor driver you wrote in the previous lab. You should be able to create multiple motor drivers and motor tasks as follows, with as many parameters as needed:

```
// Create the motor driver objects, one for each motor
motor_drv* p_motor_1 = new motor_drv (parm1A, parm2A, parm3A, ...);
motor_drv* p_motor_2 = new motor_drv (parm1B, parm2B, parm3B, ...);

// Create motor driver tasks, one for each motor
new my_motor_task ("Motor1", ..., p_motor_1);
new my_motor_task ("Motor2", ..., p_motor_2);
```

Your motor task class should receive commands to set its power level with a thread-safe shared variable of type `TaskShare<int16_t>`. There should be another shared variable which controls the mode (braking, power, etc.) You can use a `bool`, a `uint8_t`, or an enumeration as you prefer. For examples of how task shared variables work, look at the code in `task_comm` under your `lab` directory, which can also be downloaded by getting a `zip` file from
`http://wind.calpoly.edu/hg/examples`

It is recommended to take an existing task class such as `task_brightness`, renaming and modifying its files – but make sure you don't leave any features of the old task when making your new task files, including comments and `#define`'s. Everything must be updated.

Next, modify `task_user` so that you can use the user interface to control the mode and power level of each motor independently. This means that the user interface task is writing the shared variables which are read by the motor control tasks. Note that if you would like to use one or two potentiometers with the A/D converter to control the motor speeds, that's fine; if you'd prefer to use clever keyboard commands, that's fine too. The keyboard or potentiometer must be read within the user interface task.

When this task has been completed, your motor driver tasks should be *very* modular: you can copy the motor driver and motor task files into any old AVR program and use them with little if any modification, as long as there are brushed DC motors connected to VNH3SP30 or compatible driver chips. The only difference between different motor driver tasks will be the parameters given to the task constructor as each task object is created. If a *third* motor driver using a different OCR register were added to the system, your task should be able to control that by changing only constructor parameters.

Test your motor drivers thoroughly. *Try* to cause problems so that they can be fixed or, if you can't fix them before the due date, at least reported. Known bugs are better than hidden ones. Your testing should include running the motors at different speeds, braking one while powering the other, and so on in many combinations.

2 Deliverables

On the due date, you'll demonstrate your program in lab and turn in the following:

- A one page memo discussing what your program does, how it does it, how well it works (discuss test results), and the location of the Mercurial repository in which your files are stored.
- A printout of the Doxygen documentation for your new task class. This should not include the title page, file documentation, nor documentation for other classes. It should be printed double sided. *Have you looked over the printout before handing it in?*
- Printouts of the source files you have written or substantially modified, also double sided.

3 References

ME405 library – see classes `TaskBase` and `TaskShare` in particular:
`http://wind.calpoly.edu/ME405/doc/`

FreeRTOS:
`http://www.freertos.org/a00106.html`

avr-libc:
`http://www.nongnu.org/avr-libc/user-manual/`

An excellent study break:
`http://xkcd.com`