mbed Lab 5: Excuse me, but haven't we been here before?

CS 220

Dec. 4

Introduction

For this lab, you'll use a button and the KL25Z's touch slider and tri-color LED (the "haven't we been here before" part of the lab title) along with some of the mbed system's interrupt facilities (the "excuse me" part of the lab title) in order to "simultaneously" control three tasks on the KL25Z. This will shed a bit of light on how an operating system manages its trick of running hundreds of processes/threads "simultaneously" on a computer that has only a handful of CPU cores. Each of the three tasks controls one of the tri-color LED's individual LEDs. One of the tasks simulates a CPU intensive process, consuming five seconds of CPU time before relinquishing the CPU to the other tasks. This task controls the red LED. The remaining two tasks are user-oriented and would be considered to be I/O intensive tasks by an operating system. One of these tasks uses the button to control the blue LED. The other task uses the KL25Z's touch slider (the gray-ish rectangle at the end of the board — slide a finger along it) to control the brightness of the green LED.

Lab Objectives

1. Use the `InterruptIn` interface to get the KL25Z to immediately respond to external events.
2. Use the `Ticker` interface to sample an external device's state at fixed time intervals and take an action.
3. Observe how pulse width modulation can be used to modulate the LED's insane brightness.

Lab Procedure

Read each of the following steps **completely** before acting on them.

1. Insert a button into your breadboard and wire one side of the button to one of the KL25Z's ground pins and the other side of the button to pin `PTA1`. If you don't remember how the button should be oriented on the breadboard, ask me or refer back to the first mbed lab.
2. Go to the course web site on phoenix, scroll down to the info for this lab and download the `Interrupts` project. You're going to import this project into your mbed developer account.
3. Log in to your mbed developer account and go to the `Compiler` page. Click the `Import` button (near the top, on the left), select the `Upload` tab (near the middle of the page), and then click the `Browse...` button (bottom of the page.) Navigate to the `Interrupts` project that you downloaded in the previous step, select it, and then click the `Import!` button (off to the right of the `Upload` tab).

4. Open `main.cpp` and study it. Pay particular attention to the `while` loop in `main()`:

```
while (1) {
   cpuIntensive();

   // Use the touch sensor to scale the green LED's brightness
   // to be between LED_ON and LED_OFF.
   greenLed = LED_ON + (LED_OFF - LED_ON) * (1.0 - tsi.readPercentage());

   if (button)
      blueLed = LED_OFF;
   else
      blueLed = LED_ON;
}
```

   This loop simulates three tasks meant to execute "simultaneously" — a CPU intensive task that runs in five second CPU bursts and two I/O intensive tasks, one associated with the KL25Z's touch slider and the other associated with the button.

5. What do you think will happen if you interact with the button or the touch slider while the CPU intensive task is running?

   Try it out — connect your KL25Z to the host system, compile and download the `Interrupts` program, and run the program.

6. Notice how the CPU intensive task "hogs" the CPU. Typically, an operating system gives such tasks a lower priority than I/O intensive tasks. You'll now use some interrupt techniques to give the I/O intensive tasks priority over the CPU intensive task.

7. Go the the mbed site's Handbook page, scroll down to the `InterruptIn` interface, and take a look at the documentation.

   In your program, change `button`'s type to `InterruptIn` and comment-out the code acting on `button` in the `while` loop.

   Looking at the `InterruptIn` interface documentation, you're going to use the `rise()` and `fall()` functions to attach functions that are run when the button is pressed (resulting in a fall event) and when the button is released (resulting in a rise event). These two functions, which you'll need to write, should control the blue LED in such a way that it lights when the button is pushed and is dark otherwise.

8. Now, take a look at the documentation for the `Ticker` interface.

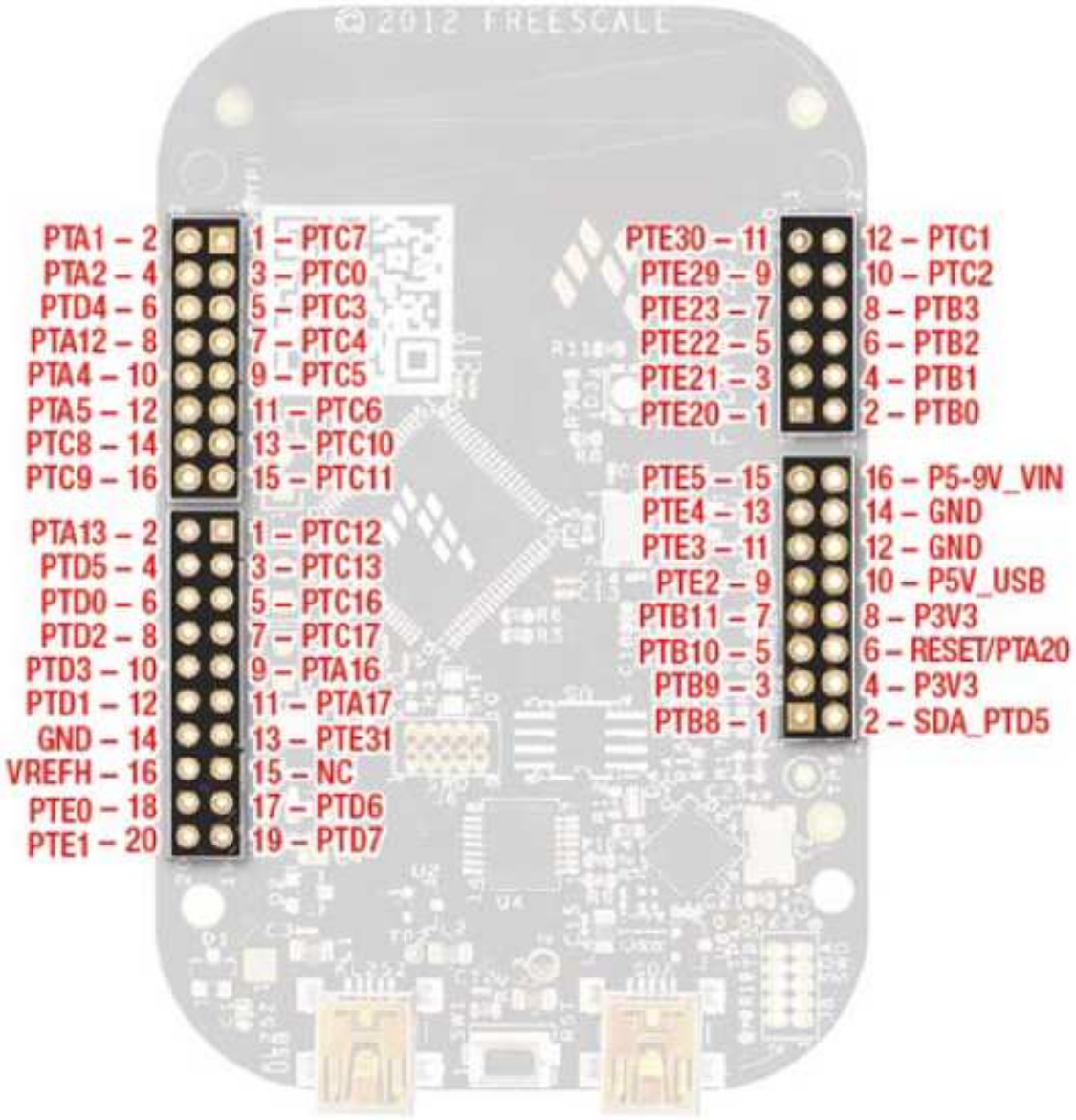
   In your program, comment-out the code acting on the touch slider in the `while()` loop.

   Create a variable of type `Ticker` and attach a function (you'll need to write this function, too) that samples the touch slider and updates the green LED's value. Choose a sampling interval that is short enough to be completely interactive (in human terms), but not so short that the sampling function ends up hogging the CPU.

9. At this point, the only statement in your `while()` loop should be the call to `cpuIntensive()`.

   Compile, download, and run your program. How is its responsiveness to the button and touch slider now?

KL25Z Pin Diagram



USB connector-end of the KL25Z