

# ShowTime

*Due: Feb. 14, 2007*

## 1 Objective

The purpose of this assignment is to get you acquainted with the pinball emulator and support library with which you will be working for the rest of the course. Hopefully, it will serve to familiarize you with both the documentation and the I/O card itself (though, at this stage, you'll be working with an emulator for the most part).

## 2 Task

You are to write a program that displays the current time and date on the numeric displays of the pinball machine. It should be possible to set the time update frequency via a command line argument, and, at the very least, you should support frequencies of  $\frac{1}{10}$  and  $\frac{1}{100}$  of a second.

Keep in mind that for the digits to function properly and remain on, they must be *strobed*—that is, each must be enabled and set in series, in a continuous loop. However, the loop must be fast enough that the light from any given digit will not have started to fade by the time it receives your program's attention again. Now, it was mentioned in class that the displays do receive a “keep-alive” signal; however, this merely brings the amount of time you have for your loop up from  $2 - 3ms$  to  $18 - 20ms$ —which boils down to about  $1ms$  per digit (or pair of digits, as you'll see later). Keep this in mind when you are writing your code<sup>1</sup>.

Note that, due to the strict time constraints imposed by the hardware on the process of strobing over the displays, you'll almost certainly need to keep track of date/time timing separately from strobe timing.

You may use any language you wish so long as you get it to function with the libraries with which we provide you. We recommend C++ and Java,

---

<sup>1</sup>of course, if you strobe too quickly, the displays will misbehave in different and exciting ways as well. Be very grateful that you are working on an emulator

principally because those are the languages for which we've provided bindings. Our code is available at `/course/cs160/src/showtime` (along with a README explaining how to use it), and documentation is available on the "Information" page of the website.

Note: one problem that several people ran into with the last assignment was underestimating the cost of `printf()` and its equivalents—particularly to the terminal. When your code runs with this degree of precision, that cost starts to become important, so be careful of where you place it in your code.

### 3 Experiment

Just as with the last assignment, you are required to keep track of the  $\Delta t$  between when each update should have occurred and when it actually occurred<sup>2</sup> You are required to produce a graph of  $\Delta t$  over a one minute period for both  $\frac{1}{10}s$  and  $\frac{1}{100}s$  updates. As was the case last time, your graphs should clearly show the distribution of times for each run, and highlight outliers. This means, for instance, we'd rather not see graphs with a range from 0.1 to 0.101, or with unlabeled axes. Please also make sure to include an explanation of your experimental results.

### 4 Handing In

You should hand in:

1. The source code for your program.
2. A makefile (if you are working in c++), ant buildfile (if you are working in java), or equivalent, allowing us to quickly and easily build your code for grading.
3. Your graphs and writeup. These may be in any format, so long as they may be accessed on a department linux machine.

To hand these in, you should copy everything involved in this assignment into a single directory, and run:

---

<sup>2</sup>Remember, your program is running on an operating system with other tasks going on simultaneously, so from time to time you may notice a large outlier resulting from your process being time-sliced out. This is normal.

```
$ /course/cs160/bin/cs160handin showtime
```

Try to avoid handing in irrelevant files by cleaning up the directory before running the script. The script will yell at you if there are compiled classfiles or \*.o files sitting around.