EE445L - Lab03 Report

Kevin Gilbert
Gilberto Rodriguez

Professor Bard
Lab: Monday/Wednesday 5-6:15

February 14, 2014

Objectives and Requirements Document

- Develop a graphics driver for the OLED that can plot lines and circles
- Design a hardware/software interf ace for a keyboard or individual switches
- Design a hardware/software driver for generating a simple tone on a speaker
- Measure supply current necessary to run the embedded system
- Implement a digital alarm clock using periodic interrupts

0.1 Requirements Document

0.1.1 Objectives

The objectives of this project are to design, build and test an alarm clock. Educationally, students are learning how to design and test modular software and how to perform switch/keypad input in the background.

0.1.2 Process

The project will be developed using the LM3S1968 board. The system will be built on a solderless breadboard and run on the usual USB power. The system will use external switches, one on board switch, and the on board speaker. There will be four hardware/software modules:

1. switch/keypad input
2. time management
3. OLED graphics
4. sound output

The process will be to design and test each module independently from the other modules. After each module is tested, the system will be built and tested.

0.1.3 Roles and Responsibilities

0.1.4 Interactions with Existing Systems

The system will use the LM3S1968 board, a solderless breadboard, and be powered using the USB cable.

0.1.5 Terminology

1. Power budget - estimate of the operation time of a battery-powered embedded system by dividing the enery storage by the average current requried to run the system

2. device driver - a collection of software routines that perform I/O functions

3. critical section - locations within a sofware module, which if an interrput were to occur at one of these locations, then an error could occur (e.g., data lost, corrupted data, program crash, etc)

4. latency - response time of the computer to external events

5. time jitter - undesired deviation from true periodicity of an assumed periodic signal in electronics and telecommunications, often in relation to a reference clock source

6. modular programming - a style of software development that divides the software problem into distinct and independent modules

0.1.6 Security

The system may include software from StellarisWare and from the book. No software written for this project may be transmitted, viewed, or communicated with any other EE445L student past, present, or future (other than the lab partner of course). It is the responsibility of the team to keep its EE445L lab solutions secure.

0.2 Function Description

0.2.1 Functionality

The clock must be able to perform five functions.

1. It will display hours, minutes, and seconds in both graphical and numeric forms on the OLED. The graphical output will include the 12 numbers around a circle, the hour hand, the minute hand, and the second hand. The numerical output will be easy to read.

2. It will allow the operator to set the current time using switches.

3. It will allow the operator to set the alarm time including enabling/disabling alarms. 4) It will make a sound at the alarm time.

4. It will allow the operator to stop the sound.

An LED heartbeat will show when the system is running.

0.2.2 Prototypes

A prototype system running on the LM3S1968 board and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration, and lab report.

0.2.3 Performance

The system will be judged by three qualitative measures. First, the software modules must be easy to understand and well-organized. Second, the clock display should be beautiful and effective in telling time. Third, the operation of setting the time and alarm should be simple and intuitive. The system should not have critical sections. All shared global variables must be identified with documentation that a critical section does not exist. Backward jumps in the ISR should be avoided if possible. The interrupt service routine used to maintain time must complete in as short a time as possible. This means all OLED I/O occurs in the main program. The average current on the +5V power will be measured with and without the alarm sounding.

0.2.4 Usability

There will be four switch inputs. In the main menu, the switches can be used to activate

1. set time
2. set alarm
3. turn on/off alarm
4. display mode.

In set time and alarm modes, two switches add and subtract hours and the other two add and subtract minutes. After 10 seconds of inactivity the system reverts to the main menu. This functionality can be enabled by uncommenting a *#define* in SysTickInts.c in which we disabled this feature to allow debugging. The display mode switch toggles between graphical and numeric displays. The switches will be debounced, so only one action occurs when the operator touches a switch once. The OLED display shows the time using graphical display typical of a standard on the wall clock. The 12 numbers, the minute hand, and the hour hand are large and easy to see. The clock can also display the time in numeric mode using numbers. The alarm sound is a simple square wave. The sound amplitude will be just loud enough for the TA to hear when within 3 feet.

0.2.5 Safety

The alarm sound will be VERY quiet in order to respect other people in the room during testing. Connecting or disconnecting wires on the protoboard while power is applied may damage the board.

0.3 Deliverables

0.3.1 Reports

A lab report described below is due by the due date listed in the syllabus. This report includes the final requirements document.

0.3.2 Audits

The preparation is due at the beginning of the lab period on the date listed in the syllabus.

0.3.3 Outcomes

There are three deliverables: preparation, demonstration, and report.

Problem 1. *You can either call the start and end critical sections functions from Startup.s to disable interrupts, or remove global dependencies.*

Problem 2. How long does it take to update the OLED with a new time?
The ISR takes a few clock cycles to update the time variables. Drawing to the OLED takes considerably more time, and is not located in the ISR. The buffer is roughly 6200 bytes in size, and must be transfered to the OLED.

Problem 3. What would be the disadvantage of updating the OLED in the background ISR?
Updating the OLED in the background ISR would increase the time in the ISR and miss other interrputs that would fire such as a timer interrput or button interrupt.

Problem 4. Did you use a OLED clear function? If so, how could you have redesigned the OLED update to run much faster?
We did use an OLED clear function. There was a lot of space on the borders that remained the same. Selectively clearing portions would have been more efficient.

Problem 5. Assuming the system were battery powered, list three ways you could have saved power.

We could have slowed down the PLL to conserve power, as well as utilize hibernate mode between interrupts and operations. We could also use lower power external speakers as the internal ones seem to use a lot of power.

Measurements

-> RMS voltage 3.31, 4.94. -> Current 56mA without speaker, 0.58mA-270mA

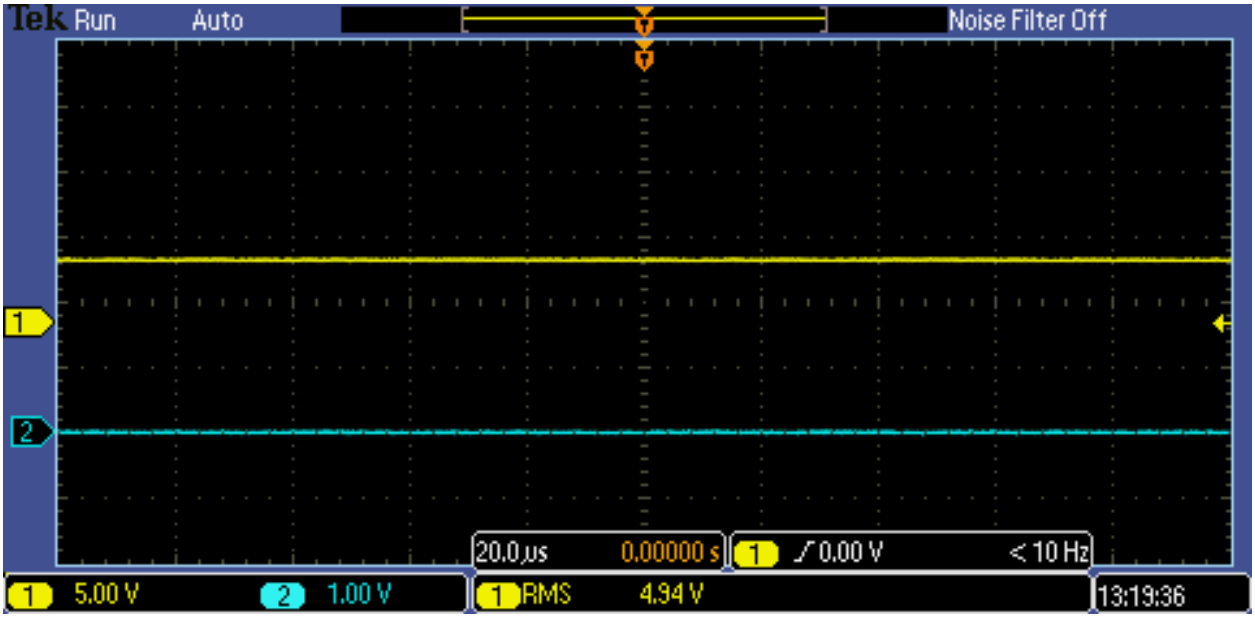

Figure A: 5V vs time

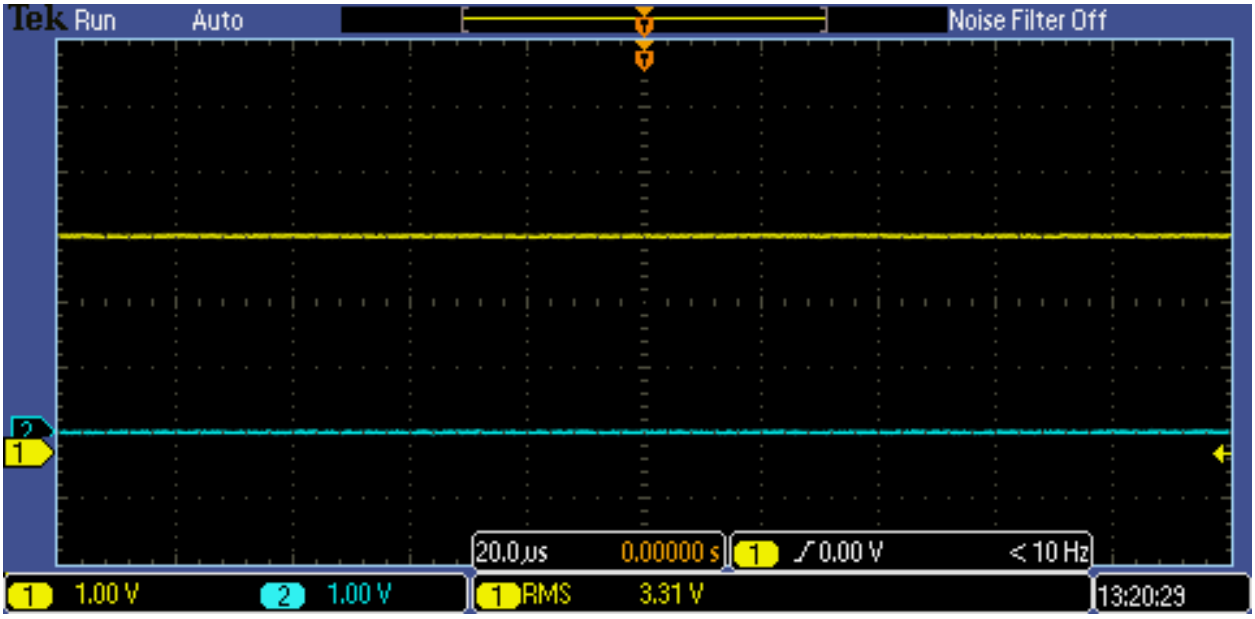

Figure B: 3.3V vs time

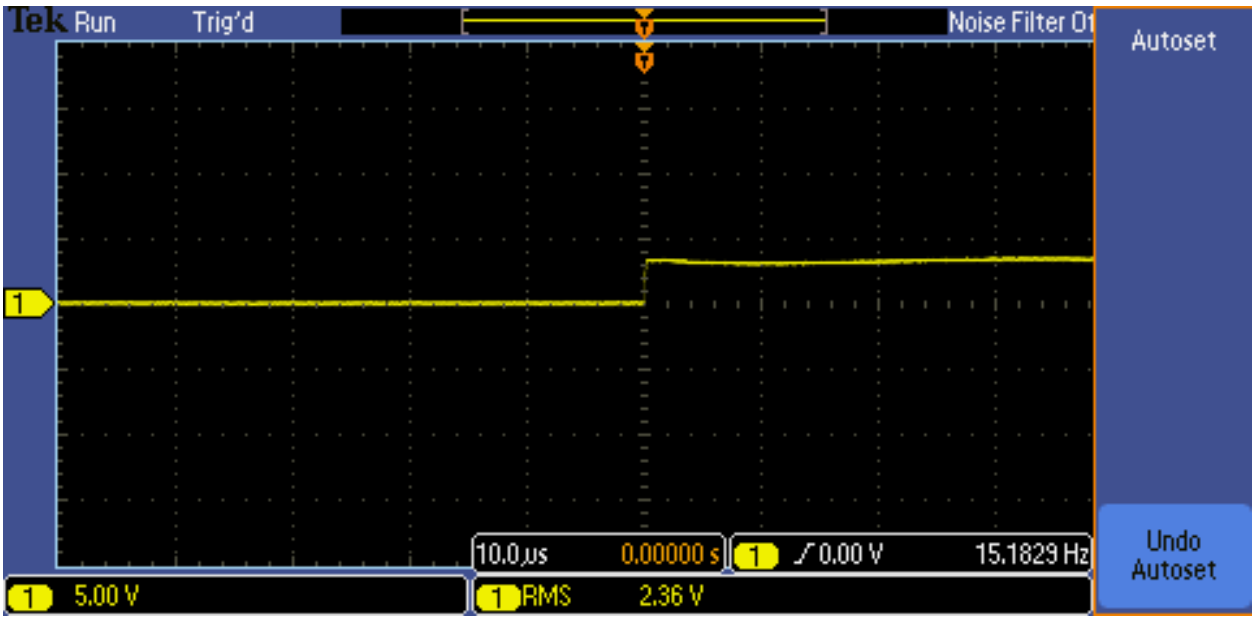

Figure C: Voltage when alarm is on