A) Requirements document

1. Overview

1.1. Objectives: Why are we doing this project? What is the purpose?

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.

1.2. Process: How will the project be developed?

The project will be developed using the TM4C123 board. There will be switches or a keypad. The system will be built on a solderless breadboard and run on the usual USB power. The system may use the on board switches and/or the on board LEDs. Alternatively, the system may include external switches. The speaker will be external. There will be at least four hardware/software modules: switch/keypad input, time management, LCD graphics, and 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.

1.3. Roles and Responsibilities: Who will do what? Who are the clients?

EE445L students are the engineers and the TA is the client. Students are expected to modify this document to clarify exactly what they plan to build. Students are allowed to divide responsibilities of the project however they wish, but, at the time of demonstration, both students are expected to understand all aspects of the design.

1.4. Interactions with Existing Systems: How will it fit in?

The system will use the TM4C123 board, a ST7735 color LCD, a solderless breadboard, and be powered using the USB cable.

1.5. Terminology: Define terms used in the document.

Power budget, device driver, critical section, latency, time jitter, and modular programming. See textbook for definitions.

1.6. Security: How will intellectual property be managed?

The system may include software from Tivaware 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.

2. Function Description

2.1. Functionality: What will the system do precisely?

The clock must be able to perform five functions. 1) It will display hours and minutes in both graphical and numeric forms on the LCD. The graphical output will include the 12 numbers around a circle, the hour hand, and the minute hand. The numerical output will be easy to read. 2) It will allow the operator to set the current time using switches or a keypad. 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. 5) It will allow the operator to stop the sound. An LED heartbeat will show when the system is running.

2.2. Scope: List the phases and what will be delivered in each phase.

Phase 1 is the preparation; phase 2 is the demonstration; and phase 3 is the lab report. Details can be found in the lab manual.

2.3. Prototypes: How will intermediate progress be demonstrated?

A prototype system running on the TM4C123 board, ST7735 color LCD, and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration and lab report.

2.4. Performance: Define the measures and describe how they will be determined.

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 LCD I/O occurs in the main program. The average current on the +5V power will be measured with and without the alarm sounding.

2.5. Usability: Describe the interfaces. Be quantitative if possible.

There will be two to 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; and 4) display mode. The user should be able to set the time (hours, minutes) and be able to set the alarm (hour, minute). Exactly how the user interface works is up to you. After some amount of inactivity the system reverts to the main menu. The user should be about to control some aspects of the display configuring the look and feel of the device. The switches MUST be debounced, so only one action occurs when the operator touches a switch once.

The LCD 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 can be a simple square wave. The sound amplitude will be just loud enough for the TA to hear when within 3 feet.

2.6. Safety: Explain any safety requirements and how they will be measured.

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.

3. Deliverables

3.1. Reports: How will the system be described?

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

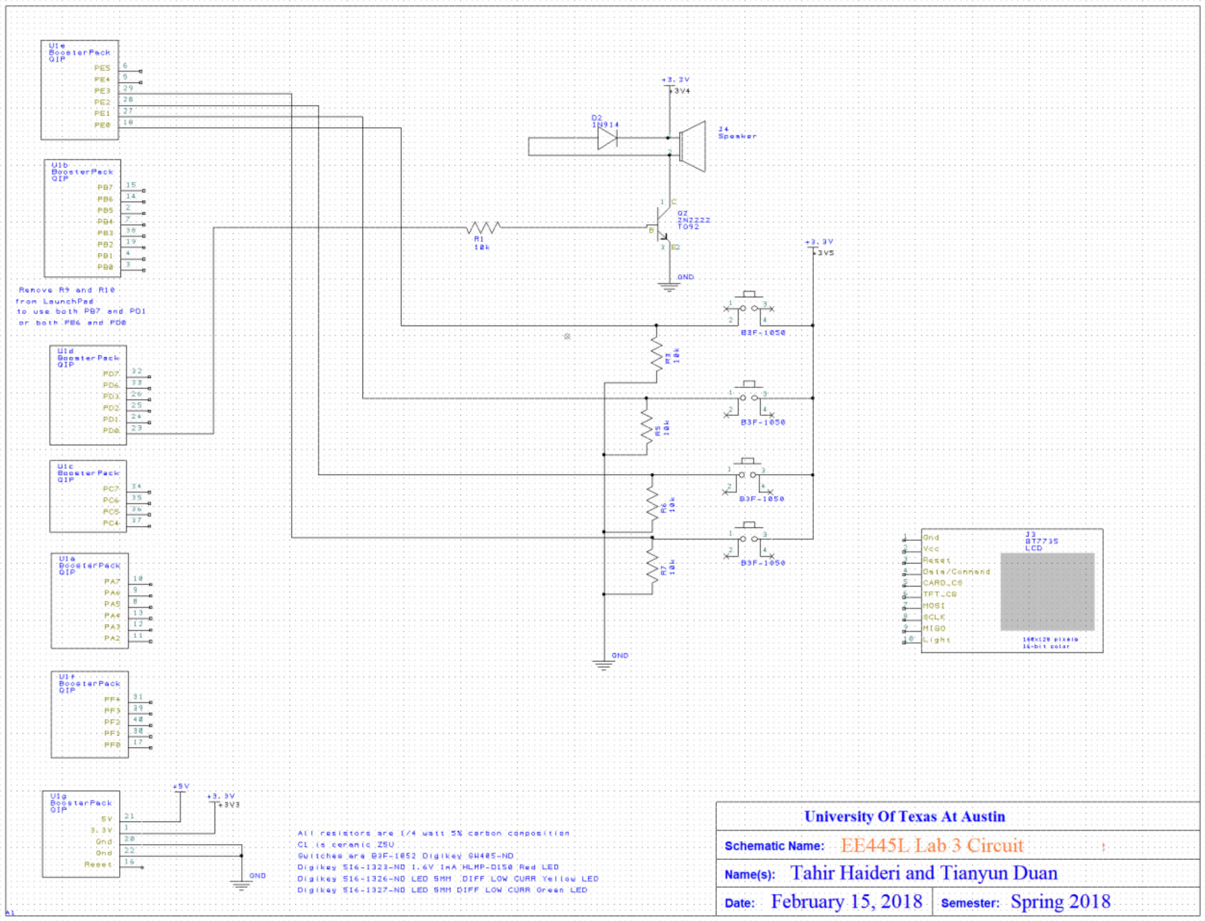
3.2. Audits: How will the clients evaluate progress?

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

3.3. Outcomes: What are the deliverables? How do we know when it is done?

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

B) Hardware Design



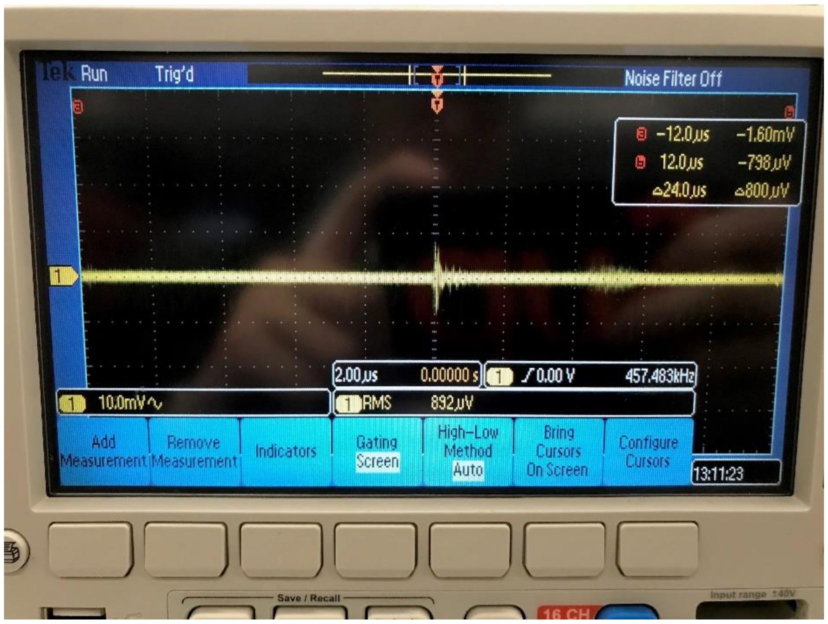
C) Software Design

Our design is the same as the one shown in Figure 3.2. We made our own .c and .h files for a higher level of abstraction, but the call graph is still the same.

D) Measurement Data

Plot the +5 and +3.3 supply voltages versus time and record the rms magnitudes

+3.3 V:



Tek Run Trig'd Noise Filter Off

a −12.0µs −1.60mV
b 12.0µs −798µV
Δ24.0µs Δ800µV

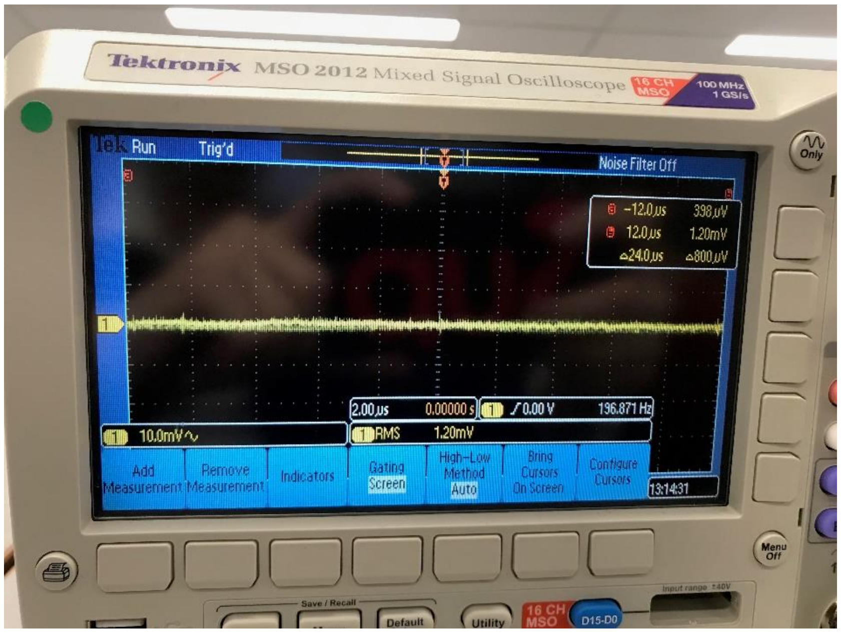
2.00µs 0.00000 s 1 ∫ 0.00 V 457.483kHz

1 10.0mV

1 RMS 892µV

13:11:23

+5 V:



Tek Run Trig'd Noise Filter Off

a −12.0µs 398µV
b 12.0µs 1.20mV
Δ24.0µs Δ800µV

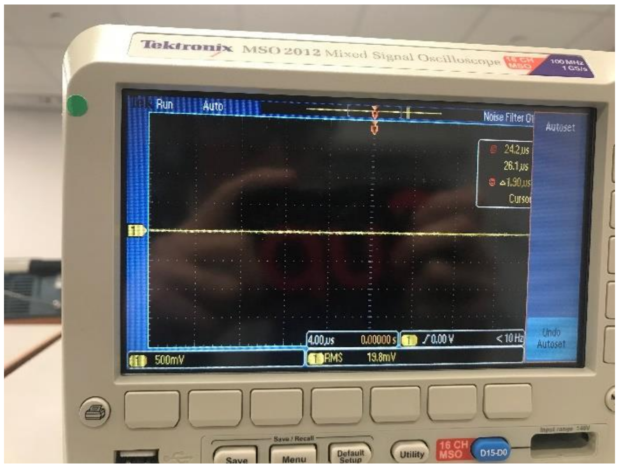
2.00µs 0.00000 s 1 ∫ 0.00 V 196.871 Hz

1 10.0mV

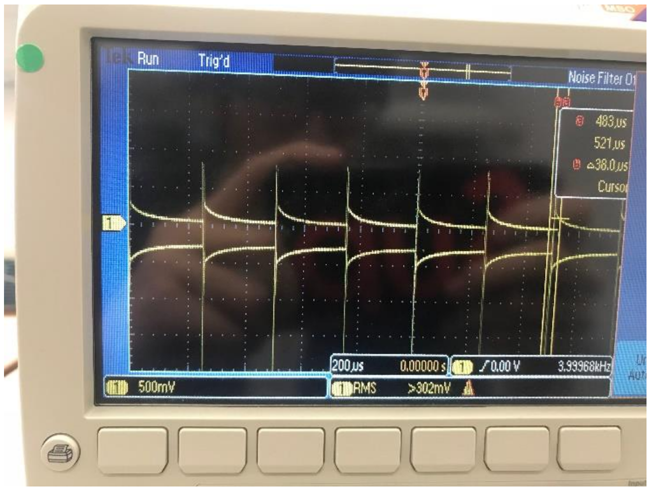
1 RMS 1.20mV

13:14:31

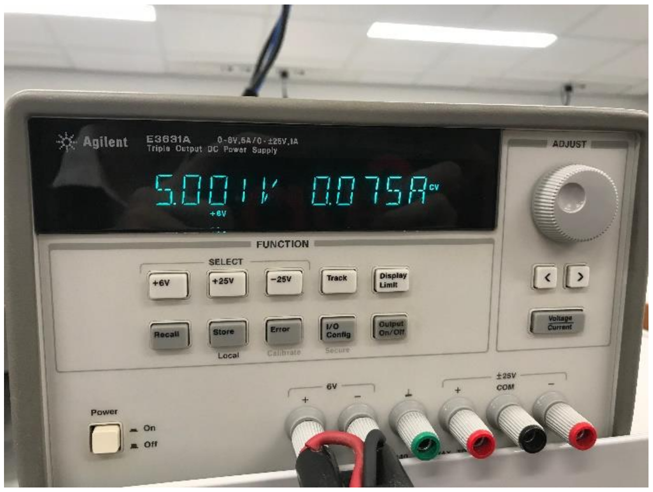
Plot the speaker voltage (or output voltage) versus time during an alarm sound
Alarm off:



Alarm on:



Measurements of current required to run the alarm clock, with and without the alarm
Alarm off:



Alarm on:

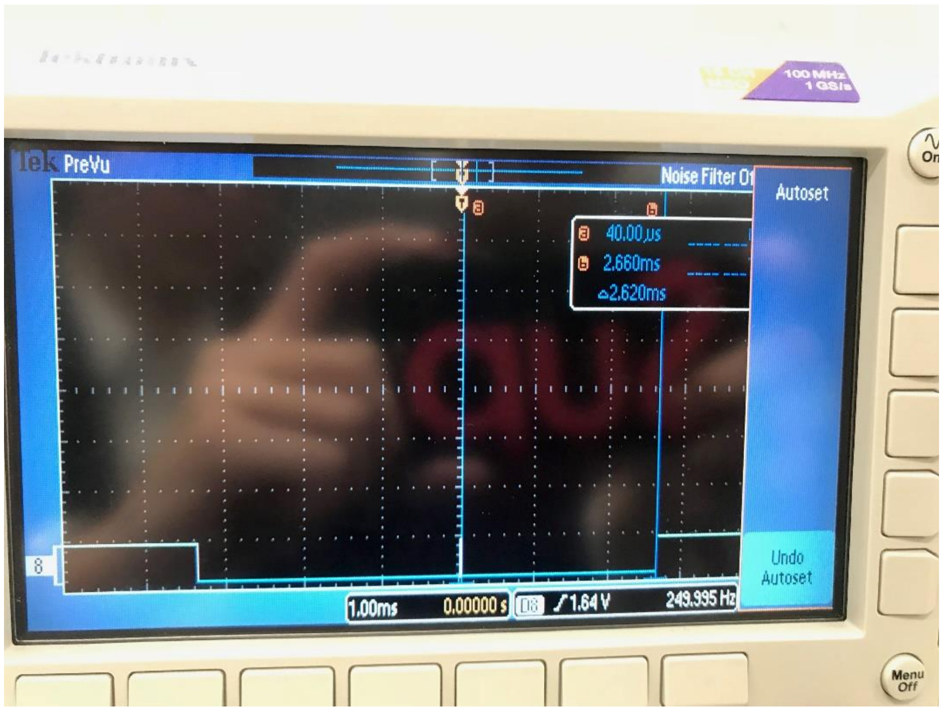
E) Analysis and Discussion

1) Give two ways to remove a critical section.

- Disable interrupt before you change a variable shared by 2 threads
- Use a mutex

2) How long does it take to update the LCD with a new time?

2.62ms, as shown below (I toggle PF2 twice before drawing, and toggle it once more after it's drawn)



3) What would be the disadvantage of updating the LCD in the background ISR?

Takes too long, which shouldn't happen in a ISR.

4) Did you redraw the entire clock for each output? If so, how could you have redesigned the LCD update to run much faster, and create a lot less flicker?

No. We only erase the previous hands and draw the new hands.

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

1. Have a sleep mode where the LCD sleeps after a period of time with no user interaction.
2. Use a smaller speaker that requires less power
3. Use a slower clock (we could use the 16MHz mode instead of the 80MHz mode)

Procedure 1):

2.62 ms is the time it takes to update the graphical time on the LCD. PE0-PE3 are GPIO ports used for the 4 switches. The switches interface with the program via edge-triggered interrupts. However, during set time and set alarm mode, interrupts on the switches are disabled and inputs are read via busy wait. The speaker is connected to PD0. All read and write commands to the speaker are included in the speaker module. The global variables are as follows:

```
uint32_t currentMode = 0;
uint32_t prevMode = 0;
uint32_t setTimeFlag = 0;
uint32_t currentHour = 0;
uint32_t currentMinute = 0;
uint32_t currentSeconds = 0;
uint32_t prevHour = 0;
uint32_t prevMinute = 0;
uint32_t prevSeconds = 0;
uint32_t setAlarmFlag = 0;
uint32_t alarmHour = 0;
uint32_t alarmMinute = 0;
uint32_t alarmActive = 0;
uint32_t soundFlag = 0;
uint32_t changeModeFlag = 0;
```

```
uint32_t updateTimeFlag = 1;
uint32_t changeThemeFlag = 0;
uint16_t THEME_COLOR = 0x0000;
uint16_t theme_index = 0;
```

ISRs don't have write access to global variables that are changed by the main.c to avoid critical sections. ISRs can set the associated flags which are acknowledged by the main and the relevant variable is changed. One problem that we were having initially was due to ISR having write access to the currentMode counter. Therefore, at times this would lead to multiple mode screens being displayed simultaneously. Therefore, a layer of abstraction was added where ISRs don't have write access to critical variables and instead set the relevant flags.