Tarang Khandpur - tk8435
Karime Saad - ks38728
September 20, 2017

EE445L Lab 3 Preparation

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.

6) The clock will have the ability to switch to a different American Timezone. (Eastern, Central, Mountain, Pacific) by using a switch.

7) The clock will also have a mode to switch to a Binary Clock

8) It will also allow for different themes (different bitmaps)

9) It will have a stopwatch

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 **eight** 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.

5) change timezone

6) change clock theme

7) switch to binary clock

8) switch to stopwatch

The user should be able to set the time (hours, minutes) and be able to set the alarm (hour, minute). 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 user will be able to access a stopwatch interface where they can start the timer, and press a button to lap the current time and use another button to reset the stopwatch.**

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. **It will also allow the user to change the display to a binary clock. The clock will show the user's selected timezone, and theme.**

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.

LCD Module

LCD.c

```
#include <stdint.h>
#include "ADCSWTrigger.h"
#include "../inc/tm4c123gh6pm.h"
#include "PLL.h"
#include "Timer1.h"
#include "Timer2.h"
#include "ST7735.h"
#include "stdio.h"
#include "Title.h"
#include "math.h"

void DrawSlantedLine(uint32_t numberOfLines);
void ST7735_Line(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2,uint16_t
color);
void DrawClockFace(void);


/******Global Variables******/


/**************Name: ResetScreenBlack***************
 Author: Karime Saad, Tarang Khandpur
 Description: Clears the screen to all Black.
 Inputs: none
 Outputs: none
*/
void ResetScreenBlack(void){
      ST7735_InitR(INITR_REDTAB);
      ST7735_FillScreen(ST7735_BLACK);
  ST7735_SetCursor(0,0);
}
/**************Name: ResetScreenWhite***************
 Author: Karime Saad, Tarang Khandpur
 Description: Clears the screen to all White.
 Inputs: none
```

```
 Outputs: none
*/
void ResetScreenWhite(void){
      ST7735_InitR(INITR_REDTAB);
      ST7735_FillScreen(ST7735_WHITE);
  ST7735_SetCursor(0,0);
}

/*******Name: DelayWait10ms*********
// Description: Subroutine to wait 10 msec
// Inputs: None
// Outputs: None
// Notes: This function was provided to us
***********************************/
void DelayWait10ms(uint32_t n){

      uint32_t volatile time;
  while(n){
    time = 727240*2/91;  // 10msec
    while(time){

            time--;
    }
    n--;
  }
}

/*************Name: DrawSlantedLine**************
 Author: Karime Saad, Tarang Khandpur
 Description: Draws lines at a 45 degree angle from
                                          the top left corner towards the
bottom
                                          right corner of the LCD.
 Inputs: number of Lines you want to be drawn
 Outputs: none
*/
void DrawSlantedLine(uint32_t numberOfLines){
      ST7735_FillScreen(ST7735_WHITE);
      for (uint32_t i = 1; i < numberOfLines; i++){
            ST7735_Line (0,i*10,i*10,0, ST7735_BLUE);
      }
      DelayWait10ms(1000);
}
/********* Name: DrawClockFace **********
 Author: Karime Saad, Tarang Khandpur
 Description: Draws a second hand clock to the

 Inputs: number of Lines you want to be drawn
 Outputs: none
*/
void DrawClockFace(void){
            ST7735_FillScreen(ST7735_WHITE);
            int32_t radius = 50;
            uint16_t xVal;
            uint16_t yVal;
            for (int i = 0; i < 60; i++){
                  ST7735_FillScreen(ST7735_WHITE);
```

```
                  float_t angle = (90 - (6 * i)) * 22/7 * 1.0/180;
                  float temp = (radius * 1000 * cos(angle) /1000);
                  xVal = 64 + temp;
                  temp = -1 * (radius * 1000 * sin(angle) /1000);
                  yVal = 80 + temp;
                  ST7735_Line (64,80,xVal,yVal, ST7735_BLUE);
            };
            ST7735_FillScreen(ST7735_WHITE);
}
```

LCD.h

```
#include <stdint.h>

/**************Name: ResetScreenBlack***************
 Author: Karime Saad, Tarang Khandpur
 Description: Clears the screen to all Black.
 Inputs: none
 Outputs: none
*/
void ResetScreenBlack(void);

/**************Name: ResetScreenWhite***************
 Author: Karime Saad, Tarang Khandpur
 Description: Clears the screen to all White.
 Inputs: none
 Outputs: none
*/
void ResetScreenWhite(void);

/*******Name: DelayWait10ms**********
// Description: Subroutine to wait 10 msec
// Inputs: None
// Outputs: None
// Notes: This function was provided to us
*************************************/
void DelayWait10ms(uint32_t n);

/**************Name: DrawSlantedLine**************
 Author: Karime Saad, Tarang Khandpur
 Description: Draws lines at a 45 degree angle from
                                          the top left corner towards the
bottom
                                          right corner of the LCD.
 Inputs: number of Lines you want to be drawn
 Outputs: none
*/
void DrawSlantedLine(uint32_t numberOfLines);

/********** Name: DrawClockFace ***********
 Author: Karime Saad, Tarang Khandpur
 Description: Draws a second hand clock to the

 Inputs: number of Lines you want to be drawn
 Outputs: none
*/
void DrawClockFace(void);
```

Speaker Module

Speaker.c

```
#include <stdint.h>
#include "../inc/tm4c123gh6pm.h"

/*********PortD_Init***********
 Initializes Speaker on Port D*/

void PortD_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x08;         // 1) activate port D
  while((SYSCTL_PRGPIO_R&0x08)==0){};    // allow time for clock to stabilize
                                     // 2) no need to unlock PD3-0
  GPIO_PORTD_AMSEL_R &= ~0x0F;       // 3) disable analog functionality on
PD3-0
  GPIO_PORTD_PCTL_R &= ~0x0000FFFF; // 4) GPIO
  GPIO_PORTD_DIR_R |= 0x0F;          // 5) make PD3-0 out
  GPIO_PORTD_AFSEL_R &= ~0x0F;       // 6) regular port function
  GPIO_PORTD_DEN_R |= 0x0F;          // 7) enable digital I/O on PD3-0
}
```

Speaker.h

```
void PortD_Init(void);
```

Switches Module

Switches.c

```
#include <stdint.h>
#include "../inc/tm4c123gh6pm.h"
#include "ST7735.h"
#include "stdio.h"

#define PF2   (*((volatile uint32_t *)0x40025010))

void PortF_Init(void){
      GPIO_PORTF_DIR_R |= 0x06;             // make PF2, PF1 out (built-in
LED)
  GPIO_PORTF_AFSEL_R &= ~0x06;           // disable alt funct on PF2, PF1
  GPIO_PORTF_DEN_R |= 0x06;              // enable digital I/O on PF2, PF1
                                         // configure PF2 as GPIO
  GPIO_PORTF_PCTL_R = (GPIO_PORTF_PCTL_R&0xFFFFF00F)+0x00000000;
  GPIO_PORTF_AMSEL_R = 0;                // disable analog functionality on PF
```

```
  PF2 = 0;                                             // turn off LED
}
```

Switches.h

```
void PortF_Init(void);
```

Timer Module

Timer.c

```
#include <stdint.h>
#include "../inc/tm4c123gh6pm.h"
#include "PLL.h"
#include "Timer1.h"
#include "Timer2.h"
#include "ST7735.h"
#include "stdio.h"

void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
long StartCritical (void);    // previous I bit, disable interrupts
void EndCritical(long sr);    // restore I bit to previous value
void WaitForInterrupt(void);  // low power mode

void Timer3A_Init10KHzInt(void);
void SysTick_Init(void);       // initialize SysTick timer
void SysTick_Wait(uint32_t delay);
void SysTick_Disable(void);
void Timer3A_Disable(void);
void Timer0A_Init100HzInt(void);
void Timer0A_Handler(void);


#define PF2   (*((volatile uint32_t *)0x40025010))


void Timer3A_Init10KHzInt(void){
  volatile uint32_t delay;
  DisableInterrupts();
  // ** general initialization **
  SYSCTL_RCGCTIMER_R |= 0x08;      // activate timer3
  delay = SYSCTL_RCGCTIMER_R;      // allow time to finish activating
  TIMER3_CTL_R &= ~TIMER_CTL_TAEN; // disable timer3A during setup
  TIMER3_CFG_R = 0;                // configure for 32-bit timer mode
  // ** timer3A initialization **
  TIMER3_TAMR_R = TIMER_TAMR_TAMR_PERIOD; // configure for periodic mode,
down count

  TIMER3_TAILR_R = 8000 - 3;         // start value for 10,000 Hz interrupts;
80mhz / 10khz = 8000
```

```
  TIMER3_TAPR_R = 0;         // bus clock resolution
  TIMER3_IMR_R |= TIMER_IMR_TATOIM;// enable timeout (rollover) interrupt
  TIMER3_ICR_R = TIMER_ICR_TATOCINT;// clear timer3A timeout flag
  TIMER3_CTL_R |= TIMER_CTL_TAEN;  // enable timer3A 32-b, periodic

  // ** interrupt initialization **
// see table 9.1 of old book, 5.1 of new book
  NVIC_PRI8_R = (NVIC_PRI8_R&0x00FFFFFF)|0x20000000; // Timer3A=priority 1
top 3 bits
  NVIC_EN1_R = 1<<(35-32);  // enable interrupt 35 in NVIC.
      //NVIC_EN0_R bit 31-0 control IRQ 31-0; NVIC_EN1_R bit 15-0 control ira
47-32
      // timer3a_handler has irq 35 and use 31-29 bits in NVIC_PRI8_R to
control the priority
}
void Timer3A_Handler(void){
  TIMER3_ICR_R = TIMER_ICR_TATOCINT;    // acknowledge timer3A timeout
}

void Timer3A_Disable(void){
      TIMER3_CTL_R &= ~TIMER_CTL_TAEN;
}



// Initialize SysTick with busy wait running at bus clock.
void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;                   // disable SysTick during setup
//  NVIC_ST_RELOAD_R = NVIC_ST_RELOAD_M;  // maximum reload value
  NVIC_ST_RELOAD_R = 7920;  // set period for 99 micro sec so 99 *10^-6 /
12.5 * 10^-9 =  reload value 7920
  NVIC_ST_CURRENT_R = 0;                // any write to current clears it
      NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R & 0x00FFFFFF) | 0x20000000;
//priority 1 bits 31-29 , 0x4 will priority 2
  NVIC_ST_CTRL_R =
NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC+NVIC_ST_CTRL_INTEN; // enable
SysTick with core clock
}

void SysTick_Disable(void){
      NVIC_ST_CTRL_R = 0;                   // disable SysTick during setup
}

void SysTick_Handler(void){
      int i = 0;
      i++;
}
// Time delay using busy wait.
// The delay parameter is in units of the core clock. (units of 20 nsec for
50 MHz clock)
void SysTick_Wait(uint32_t delay){
  volatile uint32_t elapsedTime;
  uint32_t startTime = NVIC_ST_CURRENT_R;
  do{
    elapsedTime = (startTime-NVIC_ST_CURRENT_R)&0x00FFFFFF;
  }
  while(elapsedTime <= delay);
```

```
}
// Time delay using busy wait.
// This assumes 50 MHz system clock.
void SysTick_Wait10ms(uint32_t delay){
  uint32_t i;
  for(i=0; i<delay; i++){
    SysTick_Wait(500000);  // wait 10ms (assumes 50 MHz clock)
  }
}

// This debug function initializes Timer0A to request interrupts
// at a 100 Hz frequency.  It is similar to FreqMeasure.c.
void Timer0A_Init100HzInt(void){
  volatile uint32_t delay;
  DisableInterrupts();
  // **** general initialization ****
  SYSCTL_RCGCTIMER_R |= 0x01;      // activate timer0
  delay = SYSCTL_RCGCTIMER_R;      // allow time to finish activating
  TIMER0_CTL_R &= ~TIMER_CTL_TAEN; // disable timer0A during setup
  TIMER0_CFG_R = 0;                // configure for 32-bit timer mode
  // **** timer0A initialization ****
                                   // configure for periodic mode
  TIMER0_TAMR_R = TIMER_TAMR_TAMR_PERIOD;
  TIMER0_TAILR_R = 799999;         // start value for 100 Hz interrupts
//      TIMER0_TAILR_R = 799999/10;         // start value for 1000 Hz
interrupts
  TIMER0_IMR_R |= TIMER_IMR_TATOIM;// enable timeout (rollover) interrupt
  TIMER0_ICR_R = TIMER_ICR_TATOCINT;// clear timer0A timeout flag
  TIMER0_CTL_R |= TIMER_CTL_TAEN;  // enable timer0A 32-b, periodic,
interrupts
  // **** interrupt initialization ****
                                   // Timer0A=priority 2
      NVIC_PRI4_R = (NVIC_PRI4_R&0x00FFFFFF)|0x40000000; // top 3 bits
  NVIC_EN0_R = 1<<19;              // enable interrupt 19 in NVIC
}

void Timer0A_Handler(void){

  TIMER0_ICR_R = TIMER_ICR_TATOCINT;    // acknowledge timer0A timeout
  PF2 ^= 0x04;                    // profile
  PF2 ^= 0x04;                    // profile

      PF2 ^= 0x04;                    // profile

}



void (*PeriodicTask)(void);   // user function

// ***************** TIMER1_Init ****************
// Activate TIMER1 interrupts to run user task periodically
// Inputs:  task is a pointer to a user function
//          period in units (1/clockfreq)
// Outputs: none
void Timer1_Init(void){
  SYSCTL_RCGCTIMER_R |= 0x02;   // 0) activate TIMER1
```

```
  TIMER1_CTL_R = 0x00000000;    // 1) disable TIMER1A during setup
  TIMER1_CFG_R = 0x00000000;    // 2) configure for 32-bit mode
  TIMER1_TAMR_R = 0x00000002;   // 3) configure for periodic mode, default
down-count settings
  TIMER1_TAILR_R = 0xFFFFFFFF-1;    // 4) reload value
  TIMER1_TAPR_R = 0;            // 5) bus clock resolution
  TIMER1_ICR_R = 0x00000001;    // 6) clear TIMER1A timeout flag
//  TIMER1_IMR_R = 0x00000001;    // 7) arm timeout interrupt
//  NVIC_PRI5_R = (NVIC_PRI5_R&0xFFFF00FF)|0x00008000; // 8) priority 4
// interrupts enabled in the main program after all devices initialized
// vector number 37, interrupt number 21
//  NVIC_EN0_R = 1<<21;           // 9) enable IRQ 21 in NVIC
  TIMER1_CTL_R = 0x00000001;    // 10) enable TIMER1A
}

void Timer1A_Handler(void){
  TIMER1_ICR_R = TIMER_ICR_TATOCINT;// acknowledge TIMER1A timeout
}
```

Timer.h

```
void Timer3A_Init10KHzInt(void);

// Initialize SysTick with busy wait running at bus clock.
void SysTick_Init(void);

// This debug function initializes Timer0A to request interrupts
// at a 100 Hz frequency.  It is similar to FreqMeasure.c.
void Timer0A_Init100HzInt(void);

void Timer0A_Handler(void);


void (*PeriodicTask)(void);   // user function

// ***************** TIMER1_Init ****************
// Activate TIMER1 interrupts to run user task periodically
// Inputs:  task is a pointer to a user function
//          period in units (1/clockfreq)
// Outputs: none
void Timer1_Init(void);

void Timer1A_Handler(void);
```

Main Program

main.c

```
/********* main.c **************
 Author: Tarang Khandpur, Karime Saad
 Description: Main program to test Lab 3 Fall 2017
 Date: September 19, 2017
```

```
 Runs on TM4C123
 Uses ST7735.c LCD.

*******************************/

/* This example accompanies the book
   "Embedded Systems: Real Time Interfacing to Arm Cortex M
Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2015

 Copyright 2015 by Jonathan W. Valvano, valvano@mail.utexas.edu
    You may use, edit, run or distribute this file
    as long as the above copyright notice remains
 THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
 VALVANO SHALL NOT, IN ANY sCIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL,
 OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/
 */

// center of X-ohm potentiometer connected to PE3/AIN0
// bottom of X-ohm potentiometer connected to ground
// top of X-ohm potentiometer connected to +3.3V
#include <stdint.h>
#include "ADCSWTrigger.h"
#include "../inc/tm4c123gh6pm.h"
#include "PLL.h"
#include "LCD.h"
#include "Switches.h"
#include "Speaker.h"
#include "Timer.h"
#include "ST7735.h"
#include "stdio.h"
#include "Title.h"
#include "math.h"

#define SCREEN_WIDTH 128

/**** Function Declaration ****/

void DisableInterrupts(void); // Disable interrupts
void EnableInterrupts(void);  // Enable interrupts
long StartCritical (void);    // previous I bit, disable interrupts
void EndCritical(long sr);    // restore I bit to previous value
void WaitForInterrupt(void);  // low power mode

void ST7735_OutNum(char *ptr);



/****** Global Variables *******/

int main(void){
```

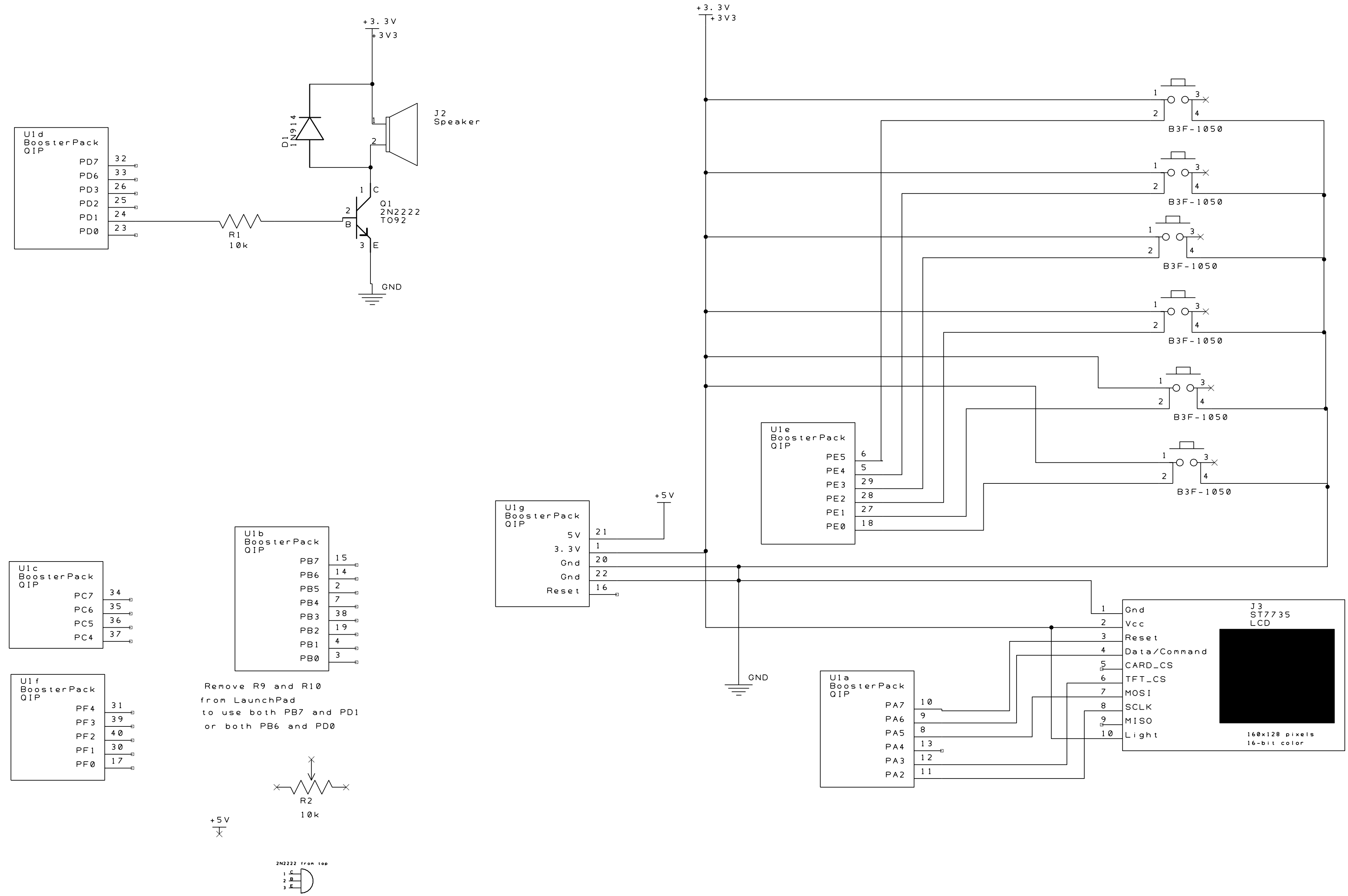
```
  PLL_Init(Bus80MHz);                   // 80 MHz
  SYSCTL_RCGCGPIO_R |= 0x20;            // activate port F
      Timer0A_Init100HzInt();               // set up Timer0A for 100 Hz
interrupts
      ResetScreenBlack();

      Timer1_Init();
            // System Clock timer
      PortD_Init(); //Initialize Speaker
      PortF_Init(); //Initialize Switches

      DrawClockFace();

}
```

Circuit Diagram - below (next page)



All resistors are 1/4 watt 5% carbon composition
C1 is ceramic Z5U
Switches are B3F-1052 Digikey SW405-ND
Digikey 516-1323-ND 1.6V 1mA HLMP-D150 Red LED
Digikey 516-1326-ND LED 5MM DIFF LOW CURR Yellow LED
Digikey 516-1327-ND LED 5MM DIFF LOW CURR Green LED

University Of Texas At Austin

Schematic Name: EE445L Lab 3 Circuit Template

Name(s): Karime Saad, Tarang Khandpur

Date: September 20, 2017 **Semester:** Fall 2017

A1