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 TAis 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)**