Toronto Metropolitan University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Embedded Systems Design
Course Number:	COE 718
Semester/Year (e.g.F2016)	F2023

Instructor:	Dr. Gul Khan

Assignment/Lab Number:	
Assignment/Lab Title:	Project: Multimedia Center

Submission Date:	
Due Date:	

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Patel	Apurva	500876938	04	A.P

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: https://www.torontomu.ca/content/dam/senate/policies/pol60.pdf

Final Project – Multi-Media Centre

COE718 – Embedded Systems Design

Apurva Patel

Section 14

500876938

F2023

I. Abstract

Create a media center on the MCB1700 board with a menu displayed on the LCD, navigated using the joystick. Features include a photo gallery, mp3 player streaming from PC, and games. Users can select options with the joystick, execute tasks interactively, and return to the main menu when done. The photo gallery displays BMP pictures converted to C files. The mp3 player streams audio from the PC via USB, with volume control using a potentiometer. Game selection and implementation are open-ended, with 2 games that were implemented.

II. Introduction

The project involves creating a media center on the MCB1700 board with a user-friendly graphical interface displayed on the LCD. Navigation through the main menu of the media center is facilitated using the MCB1700 board joystick. The menu encompasses a photo gallery, a mp3 player capable of streaming audio from a connected PC, and a game center with the flexibility for creative game implementations, such as Flappy Birds and Snake. Additionally, the project encourages creativity by enabling the programmer to implement additional features.

The photo gallery functionality involves displaying bmp pictures of certain dimensions that the LCD/Board can support, converted to C files using GIMP for integration into the program. The mp3 player connects to the PC via USB, allowing users to adjust volume using a potentiometer. A splash screen on the LCD provides visual feedback when the mp3 player is active. The games developed for this project are Snake [simple] and Flappy Birds [graphical], both of which use the board's joystick [user input] for implementation.

III. Methodology

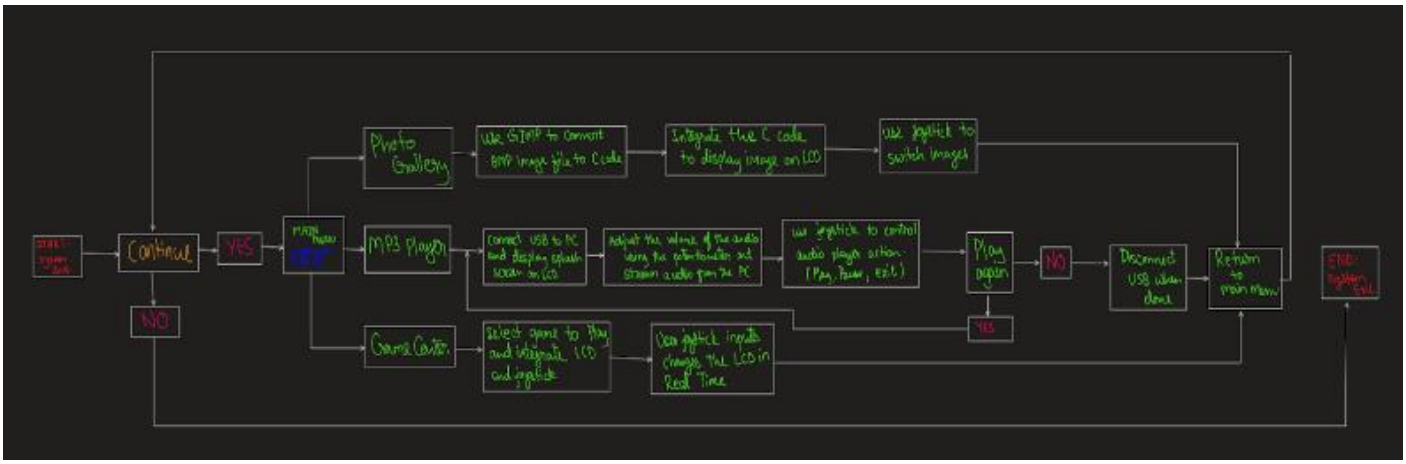
The methodology used for the media project will be discussed here. I used functions to execute all the necessary parts of the program. Firstly, once the program is uploaded to the flash memory on the board, it will display the menu on the LCD panel. Users are able to choose between the 3 options (Photo Gallery, MP3 Player, Game) and select any option and the program will execute that command. At any time, users may press the LEFT on the joystick to return to the menu to choose another option. If the Photo Gallery is selected, users are prompted to press RIGHT on the joystick to display the images one after the other. Once the user is finished with the photos of the Photo Gallery, they may press the Left joystick to return back to the main menu.

Another option a user may select is the MP3 Player. Once the MP3 player is selected, users may play an outsourced audio file (from the PC, youtube.com) that will be played off the MCB1700 board. The potentiometer on the MCB1700 board can be used to control the volume of the song played. Again, users may press the Left joystick to return back to the main menu.

The last two options that users may select are the Games: Flappy Bird and Snake. When either of the Game options are selected using the Push Select Button on the joystick, they will be prompted with a screen on the LCD panel that describes how to play the game Flappy Bird or Snake. If the User selects Flappy Birds, they are asked to press Select to start the game. Once the Select button has been pressed, users will see a Flappy Bird on the left-hand side of the LCD panel. Using the Select button, the user can let the Bird flap into the sky. If nothing is pressed, the Flappy Bird will slowly fall. The goal of the game is to keep the Flappy Bird hovering in the sky without it touching the top or bottom of the LCD panel. In addition, there are Tubes that appear from the right side of the screen and move towards the Flappy Bird on the left. Another challenge appears as the user must avoid these tubes by either letting the bird rise or fall, dodging the tubes approaching it. There is a score counter that keeps track of the distance that the Flappy Bird has traveled. If the Flappy Bird touches the bottom/top of the screen or touches the tubes, the game is over, and a game-ending screen will appear on the LCD panel. This screen will indicate the score that the user has achieved. At any time during the game or after the game, the user may press the Left joystick to return to the main menu and choose another option. However, if the User chooses to play the Snake game, they will be prompted with instructions and once the User presses select, the game starts. The basic gameplay involves the snake approaching

its target/food and eating it in order to increase the length of the snake. The User can navigate the snake using the joystick and if the User approaches the wall or sides of the LCD display, the game ends. Once the game ends, the User is given the option to play again or exit the game and move to the main menu.

IV. Flowchart



V. Design

The comprehensive implementation of the media center is the focal point of discussion here. The majority of the program is structured around the utilization of subroutines to execute specific functions, simplifying the overall implementation. Upon uploading the program to the flash memory of the MCB1700 board, the main function initiates the menu function. The menu function commences by utilizing GLCD commands to clear the LCD panel, followed by the display of menu options. The selection variable undergoes incrementation or decrementation based on the joystick's position. Specific GLCD code corresponding to the selected menu option highlights the chosen one. Regardless of the joystick's selection, the menu option triggers the execution of the corresponding function. This approach not only streamlines the program but also enhances its modularity, allowing for efficient navigation through various media center functionalities. The use of functions facilitates a clear and organized structure, contributing to the overall ease of implementation in the MCB1700 board's flash memory.

As an illustration, when the user is opting for the Photo Gallery, the program seamlessly transitions into the execution of the gallery() function, directing the flow to this specific subroutine. Within the confines of the gallery() function, the LCD panel undergoes a refreshing process to present an array of user options. Enclosed within a while loop, the entire function post-LCD update allows users to effortlessly revert to the main menu by pressing the Left joystick. Within this loop, the program dynamically responds to joystick inputs, determining the corresponding action to display the relevant image on the LCD panel. On the activation of the right direction on the joystick, a mechanism is set in motion to transition between images. This entails an initial clearing of the LCD panel, followed by the rendering of an LCD Bitmap occupying a space of 200 x 180 on the display. The decision to compress the images to this size stems from memory constraints on the board, particularly due to the substantial space consumption by the gaming section. This meticulous approach ensures an optimized and efficient utilization of resources, overcoming challenges associated with memory limitations on the MCB1700 board.

Upon selecting the MP3 player option, the program seamlessly initiates the execution of the audio_main() function. This function is responsible for the playback of user-selected audio files from the PC through the integrated speaker on the MCB1700 board. The audio output is transmitted through the board's speaker, providing a seamless audio experience. Notably, the MCB1700 board's potentiometer is used to control the volume of the MP3 player, ensuring user-friendly and customizable audio settings. This integration adds a layer of convenience and user control to the media center's audio functionalities.

Upon execution of the game() [Flappy Bird] or snakegame() [Snake] function, the program seamlessly transitions to the respective function calls within the main() function. This pivotal section of the code orchestrates the execution of the game() and snakegame() functions based on the user's selection, offering a dynamic and interactive gaming experience. The Flappy Bird game function is structured using a procedural programming paradigm. At the program's outset, essential variables are initialized, encompassing the x and y coordinates of the Small Tube, Medium Tube, and Large Tube. This extends to the Tubes appearing upside down, denoted as BottomSmallTubeX/Y, BottomMedTubeX/Y, and

BottomLargeTubeX/Y. The game initiation hinges on detecting a press of the Select joystick, triggering the commencement of the Flappy Bird game. The positioning of the Flappy Bird Bitmap at coordinates (5,120) denotes its placement at x = 5 and y = 120 on the LCD panel, with a fixed x coordinate. As the game unfolds, the Flappy Bird's y coordinate undergoes a decrement of 1 pixel per clock cycle, simulating a gradual descent. Pressing the Select joystick prompts a 2-pixel increment in the Flappy Bird's y coordinate, signifying ascent. This fundamental mechanism underpins the Flappy Bird's movement dynamics in the program. Simultaneously, tubes emerge from the right side of the screen with fixed y coordinates. Each clock cycle witnesses a decrement of 1 pixel in the x values of the Tubes, propelling them leftward. The comprehensive game() function is encapsulated within an iterative while loop, ensuring continuous and responsive gameplay. The game's fluidity and responsiveness are maintained by the ongoing evaluation of joystick inputs and consequent adjustments in the game state. This meticulous approach guarantees the synchronization of the Flappy Bird's movement and the Tube dynamics, delivering an engaging and immersive gaming experience. The while loop's iteration facilitates real-time updates and responses to user inputs, creating a seamless and enjoyable interaction within the Flappy Bird game.

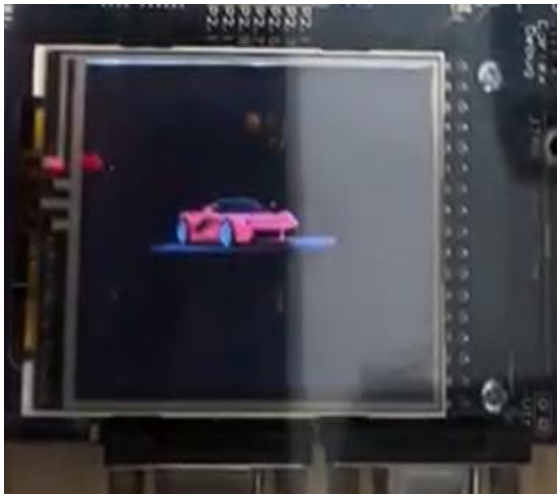
Within this iterative loop, an assessment is conducted to discern whether the Left joystick is engaged. If indeed triggered, the program gracefully exits the loop, seamlessly returning to the main menu. Alongside the pivotal game() function, a strategically integrated reset() function emerges, orchestrating the recalibration of all x and y coordinates for both tubes and the Flappy Bird itself. Complementing this, a gameover() function takes center stage whenever a scenario leading to game termination transpires, triggering a screen refresh on the LCD. Following the meticulous implementation of the Flappy Bird and Tubes' coordinated movement, the code rigorously examines collision factors. Manifesting as a series of if-else statements, the program scrutinizes the x coordinates of the Tubes, ensuring they fall within the range of 30 to 35. Simultaneously, the y position of the Flappy Bird is scrutinized in tandem with the various tube sizes, generating a robust collision detection mechanism. This intricate process is iterated six times, each corresponding to a distinct scenario and tube size. Upon the validation of these conditions, the gameover() and reset() functions are invoked, signaling that Flappy Bird failed to fulfill its fundamental objective. Integrating a scoring system, the program diligently tallies points whenever the Flappy Bird navigates unscathed through the labyrinth of tubes. This score counter incrementation dynamically captures the user's success in avoiding collisions, enriching the gaming experience with an element of achievement. The entire game unfolds in a meticulously structured manner, adhering to a top-to-bottom execution paradigm. Joystick interactions serve as triggers, prompting responsive behaviors within the game. This structural approach not only enhances code readability but also streamlines the execution flow, ensuring a coherent and efficient gaming experience. In contrast, the Snake game exhibits a more straightforward design, devoid of the complexities associated with coordinating multiple images. Commencing with the initialization of crucial variables encompassing the snake's x and y coordinates on the LCD display, score, snake speed, and length, the game function sets the stage for an immersive gaming encounter. Upon the user's initiation of the snake game, a choice between two modes is presented: Border mode and Borderless mode. In Border mode, the game culminates when the snake collides with the display's sides, and the score is promptly displayed. In stark contrast, the Borderless mode perpetuates the game even if the snake encroaches upon the LCD display's edges. The game function dynamically adapts based on the user's mode selection, tailoring the gaming experience to their preferences. Subsequent to a collision leading to game termination in Border mode, the score is prominently showcased, affording the user the choice to either continue or revert to the main menu. Intriguingly, if the user opts for the borderless mode, the game persists until the reset button on the board is activated, prompting a comprehensive reset and a return to the main menu. This thoughtful inclusion ensures a seamless transition between game sessions and reinforces the user's control over the gaming environment.

VI. Observations and Results:

- Main Menu:
  In the figure below, you can see the Main Menu on the LCD panel. You can see the background is Black and the text at the top indicates the student’s name and course code in Red text. The menu options are in Red as well. Depending on which menu is selected, that corresponding menu option is highlighted with a blue color. This is done so it is easy for the user to know which menu option is selected.

- Gallery:

  In the figure below, the Gallery is selected. The LCD updates to this when the Gallery Photo option is selected in the main menu. Here, once the user selects the Gallery option, the images are displayed with the dimensions: 200x180. The size is small because the higher the dimensions more the resources will be using up the memory and might cause the other functionalities to not work. When the Right joystick is pressed, car1's image will be displayed on the LCD panel, and if the Right joystick is pressed 2 or 3 times, the LCD will broadcast car2's or car3's image respectively as seen in the images below:



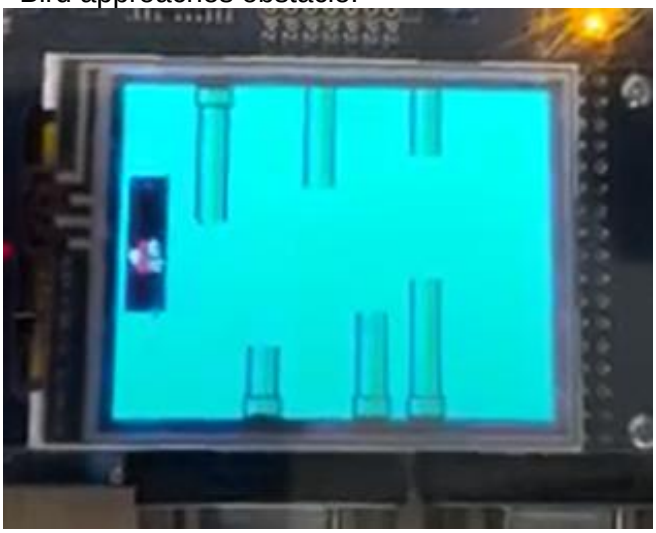
- Audio:

  In the figure below, the LCD panel displays the Main Menu while the MP3 Player is selected.

When the audio player is selected, the audio_main() function call is triggered and that refreshes the LCD panel to indicate that the MP3 Player has been selected.

When the PC plays an MP3 file, the music will play off the MCB1700 board and the potentiometer will control the volume of the board.

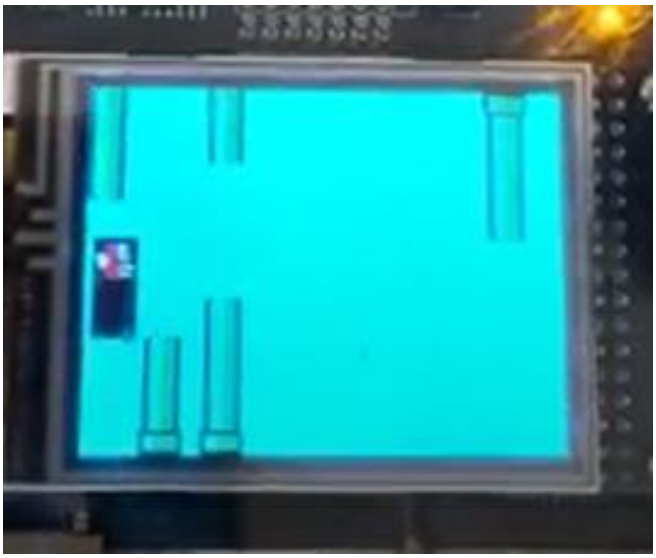
Bird approaches obstacle:



- Game Center:

Flappy Birds: In the figure below, the game controls and options are loaded before the game starts, when the user presses the select button. This LCD screen is shown when the user presses the Flappy Bird option in the Main Menu. This screen displays how the game is to be played and the goal of the game. This game is simple to play (it was difficult to implement due to the graphical work), all the user needs to do is keep the bird in the air whilst avoiding obstacles coming towards it.

Bird passing through the obstacle:



Bird crashing into an obstacle and the game is over, return to main menu:

Snake: Once the user returns to the main menu and selects the Snake game to implement as shown in the image below, the snakegame() subroutine is called.

Snake moves through the area and towards the target:

Snake approaches and hits the target, every time the target is hit by the snake it appends to the snake's body:

Snake crashes and game ends:

VII. Conclusion

The project was developed, tested, and implemented smoothly, however, there were a few difficulties faced while developing the Flappy Bird game as it was a complex mix of several images and real-time user input. The board did not support multiple images to load and display due to less and insufficient memory. However, this issue was debugged and solved by reusing several resources for the Flappy Bird game and this in turn solved the display issue for the photo gallery. Furthermore, a few bugs were identified while demoing to the TA but overall, this project helped me understand the concept of real-time programming and was successfully implemented.

VIII. References

1) Khan, Dr. Gul. (n.d.). Final Project: Media Center. COE718: Embedded System Design. https://www.ecb.torontomu.ca/%7Ecourses/coe718/labs/Media-Center.pdf

Appendix:

Ignore the box, used for IEEE formatting purposes.

Blinky.c:

```
//-------------------------------------------------------------------------------
 * Name:    Blinky.c
 * Purpose: LED Flasher and Graphic Demo
 * Note(s):
 *-------------------------------------------------------------------------------
 * This file is part of the uVision/ARM development tools.
 * This software may only be used under the terms of a valid, current,
 * end user licence from KEIL for a compatible version of KEIL software
 * development tools. Nothing else gives you the right to use this software.
 *
 * This software is supplied "AS IS" without warranties of any kind.
 *
 * Copyright (c) 2008-2011 Keil - An ARM Company. All rights reserved.
 *-------------------------------------------------------------------------------
 //const unsigned char image2[] =  Use this for the image .c files
#include <LPC17xx.H>                    // NXP LPC17xx definitions
#include "LPC17xx.h"
#include "string.h"
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"
#include "usbdmain.h"
#include "snake.h"
#include <stdio.h>
#include "usb.h"
#include "usbcfg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbaudio.h"
#include "usbdmain.h"
#include "type.h"
#include "Gallery.h"
#include "stdlib.h"
extern int audio_main (void);

//////////////////////////////////////////////////Flappy
Bird////////////////////////////////////////////////////////////////////////////
#define __FI        1                       // Font index 16x24
#define __USE_LCD   0                                   // Uncomment to use
the LCD

// Define picture variables
extern unsigned char FlappyBird[];
extern unsigned char TubesLarge[];
extern unsigned char upTubesLarge[];

// Initialize x and y position value of pictures
int bird_Xaxis = 5;
int bird_Yaxis = 120;
int TopLargeTubeX = 320;
int TopLargeTubeY = 0;
int TopMedTubeX = 400;
int TopMedTubeY = 0;
int TopSmallTubeX = 480;
int TopSmallTubeY = 0;
```

```
int BottomLargeTubeX = 480;
int BottomLargeTubeY = 140;
int BottomLMedTubeX = 440;
int BottomLMedTubeY = 165;
int BottomLSmallTubeX = 360;
int BottomLSmallTubeY = 190;
int flappyDelay = 0;
char flappyBirdScore[10];
int scoreNum = 0;

struct __FILE { int handle;  };
FILE __stdout;
FILE __stdin;

void flaapyBirdGameOver(){
    int gameExitButton;
            #ifdef __USE_LCD
                GLCD_Clear(Black);                          // Clear graphical
LCD display
                GLCD_SetBackColor(Black);
                GLCD_SetTextColor(Blue);
                GLCD_SetTextColor(Red);
                GLCD_DisplayString(2, 0, __FI, "  Score:  ");
                sprintf(flappyBirdScore, "%d", scoreNum);
                GLCD_DisplayString(2,12, __FI,  (unsigned
char*)flappyBirdScore);
                GLCD_DisplayString(3, 0, __FI, "  Game Over          ");
                GLCD_DisplayString(4, 0, __FI, "  Press LEFT         ");
                GLCD_DisplayString(5, 0, __FI, "  To Return to Menu  ");
            #endif
    while(1){
        gameExitButton = get_button();
        if (gameExitButton == KBD_LEFT){
            return;
            }
        }
}


// Reset function to reset all variables to initial values
void valuesInit(){
    scoreNum = 0;
    bird_Xaxis = 5;
    bird_Yaxis = 120;
    TopLargeTubeX = 320;
    TopLargeTubeY = 0;
    TopMedTubeX = 400;
    TopMedTubeY = 0;
    TopSmallTubeX = 480;
    TopSmallTubeY = 0;
    BottomLargeTubeX = 480;
    BottomLargeTubeY = 140;
    BottomLMedTubeX = 440;
    BottomLMedTubeY = 165;
    BottomLSmallTubeX = 360;
    BottomLSmallTubeY = 190;

}
```

```
// Game function to initialize and play Flappy Bird game
void game(){
    int gamebutton;
    #ifdef __USE_LCD
            GLCD_Clear(Cyan);                           // Clear graphical LCD
display
            GLCD_SetBackColor(Cyan);
            GLCD_SetTextColor(Red);
            GLCD_DisplayString(0, 0, __FI, "Flappy Birds    ");
            GLCD_DisplayString(1, 0, __FI, "Goal: Stay Alive       ");
            GLCD_SetTextColor(Blue);
            GLCD_DisplayString(2, 0, __FI, "Select - Play Game     ");
            GLCD_DisplayString(3, 0, __FI, "Controls:  ");
            GLCD_DisplayString(4, 0, __FI, "Select - Bird Flaps  ");
            GLCD_DisplayString(5, 0, __FI, "Left - Quit to Main Menu  ");
    #endif


    while(1){ //game inside while loop

    gamebutton = get_button();
    if (gamebutton == KBD_SELECT){ //check if Select option picked
    for (flappyDelay = 0; flappyDelay < 1; flappyDelay++){ //for loop to clear
screen once
            GLCD_Clear(Cyan);
    }
            gamebutton = get_button();


            while(gamebutton != KBD_LEFT){ //entire game held inside while loop
waiting for left to be pressed to exit game

                    // Reset x and y values of tubes once reached end of screen
                    if (TopLargeTubeX < -30){
                            TopLargeTubeX = 320;
                    }

                    if (TopMedTubeX < -30){
                            TopMedTubeX = 400;
                    }

                    if (TopSmallTubeX < -30){
                            TopSmallTubeX = 480;
                    }

                    if (BottomLargeTubeX < -30){
                            BottomLargeTubeX = 480;
                    }

                    if (BottomLMedTubeX < -30){
                            BottomLMedTubeX = 440;
                    }

                    if (BottomLSmallTubeX < -30){
                            BottomLSmallTubeX = 360;
                    }
```

```
                    // Placing pictures on LCD panel
                    GLCD_Bitmap (bird_Xaxis, bird_Yaxis, 30, 30, FlappyBird);
                    // Tubes touching the ceiling
                    GLCD_Bitmap (TopLargeTubeX, TopLargeTubeY, 30, 100,
TubesLarge);
                    GLCD_Bitmap (TopMedTubeX, TopMedTubeY, 30, 75, TubesLarge);
                    GLCD_Bitmap (TopSmallTubeX, TopSmallTubeY, 30, 50,
TubesLarge);

                    //Tubes touching the floor
                    GLCD_Bitmap (BottomLargeTubeX, BottomLargeTubeY, 30, 100,
upTubesLarge);
                    GLCD_Bitmap (BottomLMedTubeX, BottomLMedTubeY, 30, 75,
upTubesLarge);
                    GLCD_Bitmap (BottomLSmallTubeX, BottomLSmallTubeY, 30, 50,
upTubesLarge);



                    // Incrementing/Decrementing values of tubes and bird
                    gamebutton = get_button();
                    bird_Yaxis = bird_Yaxis + 1;

                    TopLargeTubeX = TopLargeTubeX - 1;
                    TopMedTubeX = TopMedTubeX - 1;
                    TopSmallTubeX = TopSmallTubeX - 1;

                    BottomLargeTubeX = BottomLargeTubeX - 1;
                    BottomLMedTubeX = BottomLMedTubeX - 1;
                    BottomLSmallTubeX = BottomLSmallTubeX - 1;

                    if (gamebutton == KBD_SELECT){ //select moves bird up 2
pixels
                            bird_Yaxis = bird_Yaxis - 2;
                    }

                    // Tube and Bird collision detection
                    if (TopLargeTubeX <= 35 && TopLargeTubeX >= 30 && bird_Yaxis
>= 0 && bird_Yaxis <= 70){
                            flaapyBirdGameOver(); //flaapyBirdGameOver function
thrown
                            valuesInit();    //reset function thrown
                            return;
                    }

                    else if (TopMedTubeX <= 35 && TopMedTubeX >= 30 &&
bird_Yaxis <= 45){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    else if (TopSmallTubeX <= 35 && TopSmallTubeX >= 30 &&
bird_Yaxis <= 20){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }
```

```
                    else if (BottomLargeTubeX <= 35 && BottomLargeTubeX >= 30 &&
bird_Yaxis >= 140){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    else if (BottomLMedTubeX <= 35 && BottomLMedTubeX >= 30 &&
bird_Yaxis >= 165){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    else if (BottomLSmallTubeX <= 35 && BottomLSmallTubeX >= 30
&& bird_Yaxis >= 190){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    // Tube and Bird collision for top and bottom of bird
                    else if (TopLargeTubeX < 35 && TopLargeTubeX > 5 &&
bird_Yaxis <= 70){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    else if (TopMedTubeX < 35 && TopMedTubeX > 5 && bird_Yaxis
<= 45){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    else if (TopSmallTubeX < 35 && TopSmallTubeX > 5 &&
bird_Yaxis <= 20){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    else if (BottomLargeTubeX < 35 && BottomLargeTubeX > 5 &&
bird_Yaxis >= 140){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    else if (BottomLMedTubeX < 35 && BottomLMedTubeX > 5 &&
bird_Yaxis >= 165){
                            flaapyBirdGameOver();
                            valuesInit();
                            return;
                    }

                    else if (BottomLSmallTubeX < 35 && BottomLSmallTubeX > 5 &&
bird_Yaxis >= 190){
```

```
                        flaapyBirdGameOver();
                        valuesInit();
                        return;
                }

                else{ //incrementing score
                        scoreNum++;
                }

                // Bird collison detection on top and bottom of screen
                if (bird_Yaxis > 210 || bird_Yaxis < 0){
                        flaapyBirdGameOver();
                        valuesInit();
                        return;
                }

            }
            GLCD_Clear(Yellow);
            return;
        }


        if (gamebutton == KBD_LEFT){ //able to exit game at any time
        GLCD_Clear(Yellow);
                return;
        }
    }
}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////
////////////////////////////////////////////Snake////////////////////////////
////////////////////////////////////////////////////////
#define DELAY_2N 20

int snakePos_X; //horizontal position
int snakePos_Y; //vertical position
int snakeSize; //snakeSize of the body
int currentDirection = 0; //current direction
int previousDirection = 0; //previous direction
int currentJoystkVal = 0; //current joystick val
int previousJoystkVal = KBD_RIGHT; //previous joystick val
char str[20],str1[20],str2[20],str3[20];
int snakeCoordinates[100][2]; //snake coordinates.
int delx, dely; //used to figure out where to turn
int speed; //how fast the snake will move
int xtarget, ytarget; //target coordinates
int collision = 0;
int displayBorder = 0;
int snakescore = 0;

void target(){
    int i;
    xtarget = rand()%9;
    ytarget = rand()%20;
    for(i=0;i<snakeSize;i++){
        if(xtarget == snakeCoordinates[i][0])
            if(ytarget == snakeCoordinates[i][1])
                target();
    }
```

```
    GLCD_DisplayChar(xtarget,ytarget,1,0x81);
}

void delay (int count){
    count <<= DELAY_2N;
    while(count--);
}

void setbody(){
    int i; //counting

    for(i=0;i<snakeSize;i++){
        switch(currentDirection){
            case 0://right
                snakeCoordinates[i][0] = snakePos_X;
                snakeCoordinates[i][1] = snakePos_Y-i;
                break;
            case 1://left
                snakeCoordinates[i][0] = snakePos_X;
                snakeCoordinates[i][1] = snakePos_Y+i;
                break;
            case 2://down
                snakeCoordinates[i][0] = snakePos_X+i;
                snakeCoordinates[i][1] = snakePos_Y;
                break;
            case 3://up
                snakeCoordinates[i][0] = snakePos_X-1;
                snakeCoordinates[i][1] = snakePos_Y;
                break;
        }
    }
}

void addbody(){
    int n=1;
    snakeSize++;
    snakescore = snakescore + 2*n;
    if(speed != 0)
        speed--;
    n++;
}

void check(){
    int i;

    //target check
    if(xtarget == snakeCoordinates[0][0])
        if(ytarget == snakeCoordinates[0][1]){
            addbody();
            target();
        }

    //tail collision check
    for(i=1;i<snakeSize;i++){
        if(snakeCoordinates[0][0] == snakeCoordinates[i][0])
            if(snakeCoordinates[0][1] == snakeCoordinates[i][1])
                collision = 1;
    }
```

```
    //collision to wall
    if(displayBorder == 1){
        //check right wall
        if(snakeCoordinates[0][1] == 19 && snakeCoordinates[1][1] == 18)
            collision = 1;
        //check left wall
        if(snakeCoordinates[0][1] == 0 && snakeCoordinates[1][1] == 1)
            collision = 1;
        //check bottom wall
        if(snakeCoordinates[0][0] == 9 && snakeCoordinates[1][0] == 8)
            collision = 1;
        //check top wall
        if(snakeCoordinates[0][0] == 0 && snakeCoordinates[1][0] == 1)
            collision = 1;
    }
}

void updatebody(){
    int i;
    if(currentDirection == 0){//move right
        for(i=snakeSize;i>0;i--){
            if(i -1 == 0){
                snakeCoordinates[0][1] = snakePos_Y;
                snakeCoordinates[0][0] = snakePos_X;
            }else{
                GLCD_DisplayChar(snakeCoordinates[i-1][0],snakeCoordinates[i-
1][1],1,' ');
                snakeCoordinates[i-1][1] = snakeCoordinates[i-2][1];
                snakeCoordinates[i-1][0] = snakeCoordinates[i-2][0];
            }
        }
        for(i=1;i<snakeSize;i++){
            GLCD_DisplayChar(snakeCoordinates[0][0],snakeCoordinates[0][1],1,0x8
B);
            GLCD_DisplayChar(snakeCoordinates[i][0],snakeCoordinates[i][1],1,0x8
2);
        }
        delay(speed);
    }else if(currentDirection == 1){//move left
        for(i=snakeSize;i>0;i--){
            if(i -1 == 0){
                snakeCoordinates[0][1] = snakePos_Y;
                snakeCoordinates[0][0] = snakePos_X;
            }else{
                GLCD_DisplayChar(snakeCoordinates[i-1][0],snakeCoordinates[i-
1][1],1,' ');
                snakeCoordinates[i-1][1] = snakeCoordinates[i-2][1];
                snakeCoordinates[i-1][0] = snakeCoordinates[i-2][0];
            }
        }
        for(i=1;i<snakeSize;i++){
            GLCD_DisplayChar(snakeCoordinates[0][0],snakeCoordinates[0][1],1,0x8
9);
            GLCD_DisplayChar(snakeCoordinates[i][0],snakeCoordinates[i][1],1,0x8
2);
        }
        delay(speed);
    }else if(currentDirection == 2){//move down
        for(i=snakeSize;i>0;i--){
            if(i -1 == 0){
```

```
                snakeCoordinates[0][1] = snakePos_Y;
                snakeCoordinates[0][0] = snakePos_X;
            }else{
                GLCD_DisplayChar(snakeCoordinates[i-1][0],snakeCoordinates[i-
1][1],1,' ');
                snakeCoordinates[i-1][1] = snakeCoordinates[i-2][1];
                snakeCoordinates[i-1][0] = snakeCoordinates[i-2][0];
            }
        }
        for(i=1;i<snakeSize;i++){
            GLCD_DisplayChar(snakeCoordinates[0][0],snakeCoordinates[0][1],1,0x8
7);
            GLCD_DisplayChar(snakeCoordinates[i][0],snakeCoordinates[i][1],1,0x8
2);
        }
        delay(speed);
    }else if(currentDirection == 3){//move up
        for(i=snakeSize;i>0;i--){
            if(i-1 == 0){
                snakeCoordinates[0][1] = snakePos_Y;
                snakeCoordinates[0][0] = snakePos_X;
            }else{
                GLCD_DisplayChar(snakeCoordinates[i-1][0],snakeCoordinates[i-
1][1],1,' ');
                snakeCoordinates[i-1][1] = snakeCoordinates[i-2][1];
                snakeCoordinates[i-1][0] = snakeCoordinates[i-2][0];
            }
        }
        for(i=1;i<snakeSize;i++){
            GLCD_DisplayChar(snakeCoordinates[0][0],snakeCoordinates[0][1],1,0x8
5);
            GLCD_DisplayChar(snakeCoordinates[i][0],snakeCoordinates[i][1],1,0x8
2);
        }
        delay(speed);
    }
    check();
}

void direction(int joyval){

    switch(joyval){
        case KBD_UP:
            if (previousJoystkVal == KBD_LEFT || previousJoystkVal ==
KBD_RIGHT){
                snakePos_X--;
                if (snakePos_X < 0){
                    snakePos_X = 9;
                }
                currentDirection = 3;
                previousDirection = currentDirection;
                previousJoystkVal = currentJoystkVal;
                updatebody();
            }
            break;
        case KBD_DOWN:
            if (previousJoystkVal == KBD_LEFT || previousJoystkVal ==
KBD_RIGHT){
                snakePos_X++;
```

```
            if (snakePos_X > 9){
                snakePos_X = 0;
            }
            currentDirection = 2;
            previousDirection = currentDirection;
            previousJoystkVal = currentJoystkVal;
            updatebody();
        }
        break;
    case KBD_LEFT:
        if (previousJoystkVal == KBD_UP || previousJoystkVal == KBD_DOWN){
            snakePos_Y--;
            if (snakePos_Y < 0){
                snakePos_Y = 20;
            }
            currentDirection = 1;
            previousDirection = currentDirection;
            previousJoystkVal = currentJoystkVal;
            updatebody();
        }
        break;
    case KBD_RIGHT:
        if (previousJoystkVal == KBD_UP || previousJoystkVal == KBD_DOWN){
            snakePos_Y++;
            if (snakePos_Y > 20 ){
                snakePos_Y = 0;
            }
            currentDirection = 0;
            previousDirection = currentDirection;
            previousJoystkVal = currentJoystkVal;
            updatebody();
        }
        break;
    default:
        switch(currentDirection){
            case 0://right
                snakePos_Y++;
                if (snakePos_Y > 20){
                    snakePos_Y = 0;
                }
                updatebody();
                check();
                break;
            case 1://left
                snakePos_Y--;
                if (snakePos_Y < 0){
                    snakePos_Y = 20;
                }
                updatebody();
                check();
                break;
            case 2://down
                snakePos_X++;
                if (snakePos_X > 9){
                    snakePos_X = 0;
                }
                updatebody();
                check();
                break;
            case 3://up
```

```
                    snakePos_X--;
                    if (snakePos_X < 0){
                        snakePos_X = 9;
                    }
                    updatebody();
                    check();
                    break;
            }
            break;
    }
}

void clearsnake(){
    int i;
    for(i=0;i<snakeSize;i++){
        snakeCoordinates[i][0]=1;
        snakeCoordinates[i][1]=1;
    }
}

int snakegame(){
    int joystkDifficulty, joystkOpt;
    int gameMode, modesel;
    int highscore=0;
    int flaapyBirdGameOver, tryagain;
    char scores[60];
    int done=0;

    GLCD_Init();
    KBD_Init();
    LED_Init();

    while(!done){
        modesel = 1;
        gameMode = 1;
        tryagain = 1;
        currentDirection = 0;
        previousDirection = 0;
        currentJoystkVal = 0;
        previousJoystkVal = KBD_RIGHT;
        speed = 15;
        snakeSize = 2;
        snakePos_X = 5;
        snakePos_Y = 10;
        GLCD_Clear(Black);
        GLCD_SetBackColor(Black);
        GLCD_SetTextColor(Green);
        GLCD_DisplayString(2,0,1,"Select Border gameMode: ");
        GLCD_DisplayString(4,0,1,"-->  ON             ");
        GLCD_DisplayString(5,0,1,"     OFF            ");
        GLCD_DisplayString(29,0,0,"          press joystick to select the game
mode            ");
        while(modesel == 1){
            joystkDifficulty = get_button();
            switch(joystkDifficulty){
                case KBD_DOWN:
                    GLCD_DisplayString(4,0,1,"     ON             ");
                    GLCD_DisplayString(5,0,1,"-->  OFF            ");
                    gameMode = 2;
```

```
                    break;
                case KBD_UP:
                    GLCD_DisplayString(4,0,1,"-->  ON             ");
                    GLCD_DisplayString(5,0,1,"     OFF            ");
                    gameMode = 1;
                    break;
                case KBD_SELECT:
                    if(gameMode == 1)
                        displayBorder = 1;
                    if(gameMode == 2)
                        displayBorder = 0;
                    modesel = 0;
                    GLCD_Clear(Black);
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Red);
                    break;
            }
        }
        setbody();
        target();
        while(collision == 0){
            currentJoystkVal = get_button();
            direction(currentJoystkVal);
            sprintf(str,"     snakescore:[%d]",snakescore);
            GLCD_DisplayString(0,0,0,(unsigned char *)str);
        }
        if(collision == 1){
            GLCD_Clear(Red);
            GLCD_SetBackColor(Black);
            GLCD_SetTextColor(Red);
            if(snakescore>=highscore)
                    highscore = snakescore;
            sprintf(scores,"     [snakescore: %d]           [HIGH snakescore:
%d]",snakescore,highscore);
            GLCD_DisplayString(0,0,0,(unsigned char *)scores);
            GLCD_DisplayString(2,0,1,"      GAME OVER     ");
            GLCD_DisplayString(7,0,1,"Wanna try again? ");
            GLCD_DisplayString(8,0,1," --> YES");
            GLCD_DisplayString(9,0,1,"     NO ");
            flaapyBirdGameOver = 0;
            while(flaapyBirdGameOver == 0){
                joystkOpt = get_button();
                switch(joystkOpt){
                    case KBD_DOWN:
                        GLCD_DisplayString(8,0,1,"     YES");
                        GLCD_DisplayString(9,0,1," --> NO ");
                        tryagain = 0;
                        break;
                    case KBD_UP:
                        GLCD_DisplayString(8,0,1," --> YES");
                        GLCD_DisplayString(9,0,1,"     NO ");
                        tryagain = 1;
                        break;
                    case KBD_SELECT:
                        if(tryagain == 0){
                            GLCD_Clear(Black);
                            done =1;
                            return 0;
                        }
                        if(tryagain == 1){
```

```
                            flaapyBirdGameOver = 1;
                            collision = 0;
                            displayBorder = 0;
                            clearsnake();
                        }
                        break;
                }
            }
        }
        delay(5);
    }
    return 0;
}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////



//-------------------------------------------------------------------------
  Main Program
 *-------------------------------------------------------------------------
int main (void)
{   // Main Program
    int inputSelected = 0; //inputSelected to see which program is user the
choosing
                                        //'1' for Gallery, '2' for audio file,
'3' for game
    int joystkVal = 0;   //track the current joystick value
    int joystkPrev = 0;  //track the previous value for the joystick

    KBD_Init();

    LED_Init ();
    GLCD_Init();

    GLCD_Clear  (Black);
    SysTick_Config(SystemCoreClock/100);
    GLCD_SetBackColor(Black);
    GLCD_SetTextColor(Red);
    GLCD_DisplayString (0, 2, 1, "COE 718 Project");
    GLCD_DisplayString (1, 5, 1, "MAIN MENU");
    GLCD_DisplayString (2, 4, 1, "Apurva Patel");

    for (;;)        //loop forever
    {
        joystkVal = get_button();   //read the joystick

        if (joystkVal !=joystkPrev) //this means that the user used the joystick
            {
            if (joystkVal == KBD_DOWN)
                {
                    inputSelected +=1;  //we are have only 3 modes so only 3
values of inputSelected
                    inputSelected = inputSelected %5;       //are accepted,
overflow is mapped back to beginning
                }
            else if (joystkVal == KBD_UP)
                {
                    inputSelected -=1;
```

```
                    if (inputSelected <=0)
                            inputSelected = 1;
                }
            else if(joystkVal == KBD_RIGHT)
                {
                    if (inputSelected == 1)
                    {
                        gallery(1);    //start the Gallery function
                        inputSelected = 0;
                    }
                    else if (inputSelected == 2)
                    {
                        audio_main();
                        delay(10);
                        GLCD_Clear(Black);
                        GLCD_SetBackColor(Black);
                        GLCD_SetTextColor(Red);
                        GLCD_DisplayString (0, 2, 1, "COE 718 Project");
                        GLCD_DisplayString (1, 5, 1, "MAIN MENU");
                        GLCD_DisplayString (2, 4, 1, "Apurva
Patel");
                        inputSelected =0;
                    }
                    else if (inputSelected == 3)
                    {
                        GLCD_Clear(Black);
                        game();
                        GLCD_Clear(Black);
                        GLCD_SetBackColor(Black);
                        GLCD_SetTextColor(Red);
                        GLCD_DisplayString (0, 2, 1, "COE 718 Project");
                        GLCD_DisplayString (1, 5, 1, "MAIN MENU");
                        GLCD_DisplayString (2, 4, 1, "Apurva Patel");
                        inputSelected = 0;
                    }
                    else if(inputSelected ==4)
                    {
                        GLCD_Clear(Black);
                        snakegame();
                        GLCD_Clear(Black);
                        GLCD_SetBackColor(Black);
                        GLCD_SetTextColor(Red);
                        GLCD_DisplayString (0, 2, 1, "COE 718 Project");
                        GLCD_DisplayString (1, 5, 1, "MAIN MENU");
                        GLCD_DisplayString (2, 4, 1, "Apurva Patel");
                        inputSelected = 0;
                    }
                }
            joystkPrev = joystkVal;
        }

        //this is to update the LCD each time the user interfaces with the
joystick
        if (inputSelected == 1) //for displaying now
            {
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Blue);
                    GLCD_DisplayString(4,0,1, "Gallery");
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Red);
```

```
                    GLCD_DisplayString(5,0,1, "Audio");
                    GLCD_DisplayString(6,0,1, "Flappy Bird");
                    GLCD_DisplayString(7,0,1, "SNAKE");

            }
        else if(inputSelected == 2)
            {
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Red);
                    GLCD_DisplayString(4,0,1, "Gallery");
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Blue);
                    GLCD_DisplayString(5,0,1, "Audio");
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Red);
                    GLCD_DisplayString(6,0,1, "Flappy Bird");
                    GLCD_DisplayString(7,0,1, "SNAKE");
            }
        else if(inputSelected == 3)
            {
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Red);
                    GLCD_DisplayString(4,0,1, "Gallery");
                    GLCD_DisplayString(5,0,1, "Audio");
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Blue);
                    GLCD_DisplayString(6,0,1, "Flappy Bird");
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Red);
                    GLCD_DisplayString(7,0,1, "SNAKE");
            }
        else if (inputSelected == 4)
            {
                        GLCD_SetBackColor(Black);
                        GLCD_SetTextColor(Red);
                        GLCD_DisplayString(4,0,1, "Gallery");
                        GLCD_DisplayString(5,0,1, "Audio");
                        GLCD_DisplayString(6,0,1, "Flappy Bird");
                        GLCD_SetBackColor(Black);
                        GLCD_SetTextColor(Blue);
                        GLCD_DisplayString(7,0,1, "SNAKE");
                        GLCD_SetBackColor(Black);
                        GLCD_SetTextColor(Red);
            }
        else
            {
                    GLCD_SetBackColor(Black);
                    GLCD_SetTextColor(Red);
                    GLCD_DisplayString(4,0,1, "Gallery");
                    GLCD_DisplayString(5,0,1, "Audio");
                    GLCD_DisplayString(6,0,1, "Flappy Bird");
                    GLCD_DisplayString(7,0,1, "SNAKE");
            }
    }
}
```

Photo Gallery:

```
/*-------------------------------------------------------------------------
```

```
 * Name:    Blinky.c
 * Purpose: LED Flasher and Graphic Demo
 * Note(s):
 *----------------------------------------------------------------------------
 * This file is part of the uVision/ARM development tools.
 * This software may only be used under the terms of a valid, current,
 * end user licence from KEIL for a compatible version of KEIL software
 * development tools. Nothing else gives you the right to use this software.
 *
 * This software is supplied "AS IS" without warranties of any kind.
 *
 * Copyright (c) 2008-2011 Keil - An ARM Company. All rights reserved.
 *----------------------------------------------------------------------------*/

#include <LPC17xx.H> /* LPC17xx definitions */
#include "Gallery.h"
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"


extern unsigned char ClockLEDOn;
extern unsigned char ClockLEDOff;
extern unsigned char ClockANI;
extern unsigned int counter;

 extern unsigned char image1[];
 extern unsigned char image2[];
 extern unsigned char image3[];




void imageToLCD(int which)      //function for displaying image
{
    int delay = 0;
    if (which==0)
    {
        GLCD_Clear(Black);
        GLCD_Bitmap(45,20,200,180,image1);
    }
    else if (which==1)
    {
        GLCD_Clear(Black);
        GLCD_Bitmap(45,20,200,180,image2);
    }
    else if (which==2)
    {
        GLCD_Clear(Black);
        GLCD_Bitmap(45,20,200,180,image3);
    }
}

void gallery (int mode)
{
    int imageZoom=0;
    int image     =  0, DELAY = 0;      //variable that saves which picture to
display
    int clockTimeOut   =  0;
```

```
    unsigned char *picture_ptr =0;
    int prevImage = get_button();
    int joystickInput = get_button();
    counter =0;
    imageToLCD(image);
    while (clockTimeOut <1)     //if the joystick has not pressed twice, we stay
on photo viewer
    {
        joystickInput = get_button();       //read the joystick
        if (joystickInput != prevImage) //if sth change, then know what change
it was
            {
                if (joystickInput == KBD_RIGHT)
                    {
                            image = image+1; //increment image
                            image = image%3;    //we are only displaying 3
pictures
                          imageToLCD(image);    //display whatever image
                            imageZoom = 0;
                    }
                else if (joystickInput ==KBD_LEFT)
                    {
                        image = image-1; //decrement image
                        if (image < 0)
                                image = 2;
                        imageToLCD(image);  //display whatever image
                        imageZoom = 0;
                    }
                else if (joystickInput ==KBD_SELECT)
                {
                    clockTimeOut ++;
                }
              prevImage = joystickInput;
                counter =0;
        }
    }
    clockTimeOut = 0;   //before going out, set the clockTimeOut back to zero
    GLCD_Clear(Black);
    GLCD_DisplayString (0, 2, 1, "COE 718 Project");
    GLCD_DisplayString (1, 5, 1, "MAIN MENU");
    GLCD_DisplayString (2, 4, 1, "Apurva Patel");
}
```

Audio:

> Split Screen for MP3 Player:

```
int audio_main (void)
{
    int i,j;
    volatile uint32_t pclkdiv, pclk;
  /* SystemClockUpdate() updates the SystemFrequency variable */
  SystemClockUpdate();

  LPC_PINCON->PINSEL1 &=~((0x03<<18)|(0x03<<20));
  /* P0.25, A0.0, function 01, P0.26 AOUT, function 10 */
  LPC_PINCON->PINSEL1 |= ((0x01<<18)|(0x02<<20));
```

```
  /* Enable CLOCK into ADC controller */
  LPC_SC->PCONP |= (1 << 12);

  LPC_ADC->CR = 0x00200E04;      /* ADC: 10-bit AIN2 @ 4MHz */
  LPC_DAC->CR = 0x00008000;      /* DAC Output set to Middle Point */

  /* By default, the PCLKSELx value is zero, thus, the PCLK for
  all the peripherals is 1/4 of the SystemFrequency. */
  /* Bit 2~3 is for TIMER0 */
  pclkdiv = (LPC_SC->PCLKSEL0 >> 2) & 0x03;
  switch ( pclkdiv )
  {
    case 0x00:
    default:
      pclk = SystemFrequency/4;
    break;
    case 0x01:
      pclk = SystemFrequency;
    break;
    case 0x02:
      pclk = SystemFrequency/2;
    break;
    case 0x03:
      pclk = SystemFrequency/8;
    break;

    }

  LPC_TIM0->MR0 = pclk/DATA_FREQ - 1;    /* TC0 Match Value 0 */
  LPC_TIM0->MCR = 3;                     /* TCO Interrupt and Reset on MR0 */
  LPC_TIM0->TCR = 1;                     /* TC0 Enable */

    GLCD_Init();
    GLCD_Clear(Blue);
    GLCD_SetTextColor(Black);
    GLCD_SetBackColor(Blue);
    GLCD_DisplayString(0,0,1,"____________________");
    GLCD_DisplayString(1,0,1,"*** AUDIO MODE!! ***");
    GLCD_DisplayString(2,0,1,"____________________");
    GLCD_DisplayString(10,0,0,"   Select Joystick to return to main menu    ");
    for(i=0;i<120;i++){
        GLCD_SetTextColor(Yellow);
        GLCD_PutPixel(0+i,150);
        GLCD_PutPixel(0+i,151);
        GLCD_PutPixel(0+i,149);
    }
    for(i=0;i<120;i++){
        GLCD_PutPixel(200+i,150);
        GLCD_PutPixel(200+i,151);
        GLCD_PutPixel(200+i,149);
    }
    for(i=0;i<15;i++){
        GLCD_PutPixel(120+i,150+2*i);
        GLCD_PutPixel(120+i,151+2*i);
        GLCD_PutPixel(120+i,149+2*i);
        GLCD_PutPixel(120+i,148+2*i);
    }
    for(i=0;i<35;i++){
        GLCD_PutPixel(135+i,180-2*i);
```

```
        GLCD_PutPixel(135+i,181-2*i);
        GLCD_PutPixel(135+i,179-2*i);
        GLCD_PutPixel(135+i,178-2*i);
    }
    for(i=0;i<20;i++){
        GLCD_PutPixel(170+i,110+3*i);
        GLCD_PutPixel(170+i,111+3*i);
        GLCD_PutPixel(170+i,109+3*i);
        GLCD_PutPixel(170+i,108+3*i);
    }
    for(i=0;i<10;i++){
        GLCD_PutPixel(190+i,170-2*i);
        GLCD_PutPixel(190+i,171-2*i);
        GLCD_PutPixel(190+i,169-2*i);
        GLCD_PutPixel(190+i,168-2*i);
    }

  NVIC_EnableIRQ(TIMER0_IRQn);

  USB_Init();                /* USB Initialization */
    NVIC_EnableIRQ(USB_IRQn);               /* enable USB interrupt */
    USB_Reset();
    USB_SetAddress(0);
    USB_Connect(TRUE);       /* USB Connect */
  /********* The main Function is an endless loop **********/
  //while( 1 );
    return 0;
}
```