COE718: Media Center Final Report

Charran Thangeswaran

Department of Electrical, Computer, and Biomedical Engineering

Toronto Metropolitan University

Toronto, ON

charran.thangeswaran@torontomu.ca

***Abstract*—** This project involves the development of an interactive media center using the Keil MCB1700 Evaluation Board and the µVision software. The media center will feature a photo gallery capable of displaying bitmap pictures, an MP3 player that streams audio from the PC via USB, and a game center where the user can play at least one game; these three sections should all be located and accessed within the menu. The media center interface must be implemented on the LCD and users should be able to navigate through the system using the MCB1700's joystick. Creating this media center requires converting the bmp files into C-source arrays to display the images, implementing a USB-based MP3 player with volume control by using the board's potentiometer, and creating a game environment that uses the LCD and joystick.

I. Introduction

In this project, the media center is developed using the Keil MCB1700 Evaluation Board and Keil µVision software, while incorporating concepts gained through course material. This media center aims to provide users with a main menu that consists of a photo gallery, an MP3 player, and a game center. Each of these options must be accessible and allow the users to feel engaged and use the joystick to navigate through the media center for a seamless experience. In terms of design choices, this media center is created to not only ensure successful user interaction and experience, but also to be able to represent myself.

For the photo gallery, four bitmap files should be displayed as photos, with each photo being a hip-hop album cover. The user is able to navigate through each photo using the joystick.

Once the user selects the MP3 player from the main menu, they are taken to a screen that uses the colour scheme based on the audio streaming service, Spotify. The MP3 player streams audio via USB from the PC to the development board and users are able to adjust the volume using the board's potentiometer. The MP3 player is disconnected once the user returns to the main menu.

The game requires the integration and interaction of the MCB1700's LCD and joystick. Taking inspiration from other games, the idea for Mario Kart Maze came to mind. The objective is for Mario to drive through the maze without colliding with any blocks. If a collision occurs, then the game is over. Otherwise, the user advances to the next level, with another block being added for each level. Given that the MCB1700 has 8 LEDs, the game goes to 8 levels; an LED lights up to let the user know what level they are on.

II. Past Work/Review

This media center uses a variety of features from previous lab work.

Lab 1 introduced the Keil MCB1700 Evaluation Board and the µVision software. The concept of the LEDs flashing at a certain speed for the Blinky project was implemented for the loading screen and going through the GLCD_SPI_LPC1700.c file helped in understanding which functions would be needed to display certain text or images on the LCD. Rotating the potentiometer clockwise and counter-clockwise helped in understanding its functionality and how it could be used in the project, though to a much smaller degree than going through the USB Audio examples. Furthermore, the joystick (KBD) files attached to the appendix of Lab 1 were crucial in order for users to navigate through the media center.

While not directly used, the concepts used in Lab 3 for using threads for different scheduling methods were also used. For example, Mario Kart Maze uses the concepts of threads to update the LCD screen and render elements within the game in order for the game to function properly.

III. Methodology

As previously mentioned, this media center was implemented in a way to not only make it user-friendly, but also be a representation of myself. For each feature, including the main menu, the user can find at the bottom of the LCD the instructions to navigate through the media center.

First, the code for the main menu had to be created. Lab 1 helped in selecting which functions would be used to get the desired outputs on the LCD and determining what happens when the joystick goes a certain direction. Since blue and black are my two favourite colours, I used those for the main menu. Once I added the KBD files, I got to work on adding the options for the photo gallery, MP3 player, and game center. I also decided to make it so that whatever option the user is hovering on, that option is highlighted. By default, since the photo gallery is the top option in the media center, that option is highlighted. By moving the joystick up and down, the option the user currently hovers on is the one that is highlighted. Finally, to select an option, the user simply needs to push on the joystick.

The first feature I worked on was the photo gallery. The purpose of the photo gallery was to display bitmap pictures. However, in order to do this, the images needed to be converted into .c files. By reading through the provided project manual, I downloaded GIMP on my laptop since the lab computer is unable to export the images as C-source arrays. Once installed, I searched and loaded my four images one by one. For each image, I had to first flip it vertically since the board draws the image upside down and then I had to scale each image to make sure it fits to my liking. Once that

was completed, I exported each image as a C source code file, while making sure that everything was not selected in the pop-up dialog besides “Save macros instead of struct” and “Save as RGB565 (16-bit)”. To view each photo, it was handled in a very similar way to the main menu; the joystick allows the user to select which photo they want to see and the option the user hovers on is highlighted. Then to exit out of an image, the user moves the joystick left, making it reminiscent of a back button.

While the MP3 player is the second option of my main menu, I wanted to start working on the game, since my idea was to use bitmap images for this as well. When the user selects the game option in the main menu, they will first see an image of a Mario Kart sprite with the text “MARIO KART MAZE”. After pushing the joystick, the user will see a screen with instructions on how to play the game; the goal is to have Mario reach the other side (from the bottom to the top of the screen) and maneuver through the blocks by moving the joystick left and right. Once the user pushes the joystick again, the game will start. The user will be greeted with two initial images. One is a Mario Kart sprite in a top-down view and the other is a block. The user must use the joystick to help Mario navigate through the blocks without colliding. If the user is able to get Mario to the top of the screen, they level up. As the levels increase, the number of blocks increase which makes it harder to maneuver through them and get to the top. However if Mario collides even once, the user loses the game and returns back to the main menu. If they were to reselect the game option, they have to start back at the first level.

Finally, I started working on the MP3 player. As recommended in the project manual, I decided to look at the USB Audio example and observe how the files work and are used. I added the files and made minor changes to the code; namely, making the audio to stop streaming when the user exits the MP3 player. The audio works by reading potentiometer values and sets the speaker of the board based on those values. When the audio was played on the PC, the speaker of the board produced it for the user. Moreover, the user is able to increase or decrease the volume by adjusting the potentiometer.

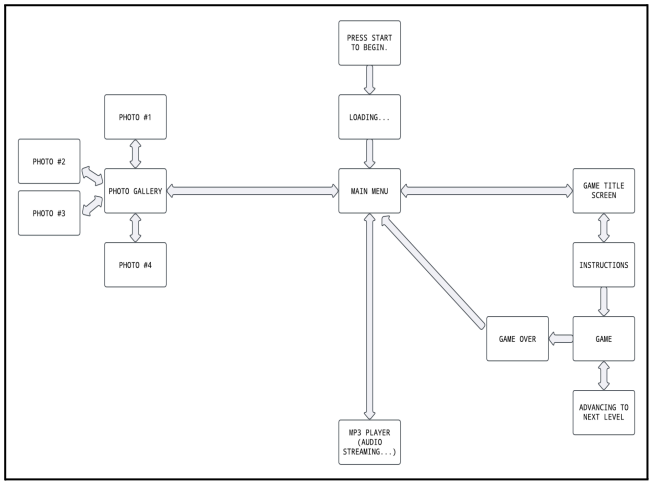

Figure 1: Flowchart of the Media Center

IV. Design

The main component of the media center is the main menu. As mentioned before, the main menu contains all the features of the media center and provides the user with an option to access each of the 3 features. Not only is the user able to navigate through these selections, but they also have the option to return back to the main menu once they are finished browsing through the selected feature. In order to approach designing the main menu, a crucial understanding of the MCB1700 LCD is needed. The first step was to print a list of features that the media center included onto the LCD. The file "GLCD.h" was included, which contains all the functions needed to use the LCD screen of the MCB1700 board. The "GLCD_DisplayString()" function from the GLCD_SPI_LPC1700.c file was used to print words onto the LCD . This function displays a string on a given line. Other functions that were used were "GLCD_Clear()", which clears the entire screen to a particular colour, "GLCD_SetBackColor()", which sets the background colour of the line, and "GLCD_SetTextColor()", which sets the text colour of the line. When the project file gets compiled and loaded onto the system, the user will have to reset, where the LCD screen gets cleared to a white screen, and then a main menu will appear, containing a list of the three features. This main menu has a colour scheme of mainly dark cyan and black, with the bottom text being white. When the user is navigating through the menu, they can easily differentiate between which option they are currently at. If the user is at the first option, the text colour will be black but highlighted as dark cyan, indicating that they are currently at that option, while the other two options will have a dark cyan text colour. This way, the user can always keep track of where they are currently at from the list of features displayed. Once the visual design for the main menu was completed, the next thing that needed to be implemented was how the user would be able to navigate and select options from the main menu; this is where the KBD files come into play (i.e. the joystick). Several different loops were implemented for navigation purposes with a counter. If the user pushes the joystick down and it was not on the third choice, then the counter will increment. If the user pushes the joystick up and it was not on the first choice, then the counter will decrement. For example, if the counter value is one, the joystick will automatically be pointing at the first option, being the photo gallery. Once the user pushes the joystick down, the counter will increment, which moves from the photo gallery to the second option, the MP3 player. If pushed down again, it will go from the second option to the third option, the game center. The opposite occurs whenever the user pushes the joystick up, where the counter will decrement, moving from the third option to the second option, and the second to the first. This is essentially how the user can go up or down when navigating through the list of features. If the user selects the photo gallery, the counter value will be 1 and it will be redirected to the Photo_Gallery() function, where another menu will be displayed. If the user selects the second option, the counter value will be 2, and it will be redirected to the MP3_Player() function to the media player. If

the user selects the third option, the counter value will be 3, and it will be redirected to the Game() function.

The photo gallery is what displays several different images onto the LCD for the user to view. Once the user selects the first option from the main menu, the Photo_Gallery() function gets called, and the user is redirected to a gallery menu. Once again, the "GLCD_Clear()" function was used to clear the entire screen to white, and then by using the "GLCD_DisplayString()", "GLCD_SetBackColor()" and "GLCD_SetTextColor()" functions, the user is presented with another list of options to choose from. This photo gallery consists of 4 bitmap images to view: the album covers for good kid, m.A.A.d city (Photo #1), 1999 (Photo #2), Madvillainy (Photo #3), and The Marshall Mathers LP (Photo #4). The gallery menu primarily uses magenta and white as the colour scheme, with options changing to blue depending on what option the user is hovering on. So if the text colour was blue, this represented the option being "highlighted", while the text colour for the other options were magenta. Similar to the main menu, the joystick concept was again used for navigation and selecting purposes. Several different loops were again implemented, and once any image from the gallery menu is selected, the LCD displays the photo. If the user wanted to return to the gallery menu, they would have to move the joystick towards the left direction. The user can also exit the gallery menu back to the main menu by moving the joystick left again. In order to load the images onto the LCD, a brief understanding of pixels were required, along with knowing the area of the LCD. The LCD is limited to displaying images that are 320x240 or smaller, anything bigger than this size will result in failure of displaying a proper image onto the LCD. Another factor that needed to be considered is the pixel on the image. The more pixels an image contains, the higher memory required to display the image. A .bmp file cannot directly be displayed onto the LCD and has to be converted into a C-source array file. As mentioned previously, the software GIMP was used to facilitate that conversion process. Once a photo was opened using this software, the image was required to be flipped vertically, otherwise, the photo would be displayed upside-down onto the LCD. After multiple attempts at scaling an appropriate size for the image, I found 180x180 pixels to be the perfect dimensions for my photo gallery. After all the requirements are fulfilled, the image can be exported as a C file. When exporting the C file, a proper name was labelled to the file and two options, "use macros instead of struct" and "save as RGB565 (16-bit)" were selected before pressing export. Once exported, I noticed after running the media center to display the images that I needed to remove the "static" in "static unsigned char name[]". Also for simplicity, I removed the “pixel_data” off of “imagename_pixel_data”. Each of the four images needed to be included as "extern unsigned char name[]", so when the "GLCD_Bitmap()" function is used, it will recognize and load the proper C file as the image. After all the images were converted into C files, and they were included into the project code file, the "GLCD_Bitmap()" function was used to display any of the photos onto the LCD screen at a given x and y position. If the user selected the first option, the gallery menu will disappear, and a photo of the first album cover will appear onto the screen. When the user moves the joystick left, it will be redirected back to the gallery menu where the user can select the next photo or push left again to return to the main screen.

When the user selects the MP3 Player option from the main menu, it clears the screen with a green background, and with a primarily black background with green text that says "AUDIO STREAMING…", which is achieved using the "GLCD_DisplayString()" function. Once the message and the photo is displayed onto the LCD, it calls the "MP3_Player()" function. This MP3 player is responsible for streaming audio from the PC to the MCB1700 development board via USB. When this option is selected, the USB should connect to the PC, and then disconnect when wanting to exit from this feature. This feature can stream any audio file, including YouTube videos, and play it on the board. The potentiometer on the board is used to adjust the volume of the built-in NXP speakers of the board. When the potentiometer is rotated counter-clockwise, the volume of the speakers should increase, whereas rotating it clockwise decreases the volume. The maximum value that the potentiometer can go is 0xFFF and the minimum value that the potentiometer can go is 0x000. The program used for the MP3 player includes an infinite while loop in its main function. The main function then repeatedly checks the USB connection, the interrupt handler and the value of the potentiometer. Once the PC is connected to the board, and the program detects some audio being played, the audio file is transferred bit by bit and played through the board’s speakers. The most challenging part of creating the MP3 player was stopping the music from playing and returning back to the main menu. In order to accomplish this task, changes had to be made in the "TIMER0_IRQHandler()" function. In the if statement, where the potentiometer value is checked every 1024th tick, another if condition was added in where the program also checks the value of the joystick. If the joystick was pushed to the left, the "NIVC_DisableIRQ()" function disables both the TIMER0_IROn and the USB_IRQn, where originally the TIMER0_IRQn was enabled using the "NVIC_EnableIRQ()" function. The "USB_Connect()" function was also set to false whereas before it was set to true. Since these functions disable or return a false value, when the main program checks for the condition of the USB, the music streaming from the board stops playing and is instead played from the PC. Once the music stops streaming onto the board, the program exits the music player feature and returns back to the main menu.

Growing up, I played a lot of old-school games, so my idea was to create a game that was retro but creative. Taking inspiration from many games I came up with the idea for Mario Kart Maze. The objective of the game was for Mario to drive his way through the block(s) which are located at different points and get to the top of the screen. The sprite chosen for this game was a top-down view of a Mario Kart sprite. This entire game was designed using Bitmap images, the GLCD functions, and by incorporating the joystick files.

Once the user selects the game option from the main menu, they get redirected to the game title screen. In the game title screen, the “Game_Check()” was called where it gives the user an introduction of what game they are about to start playing by showing another Mario Kart sprite using the "GLCD_Bitmap()" function. After pushing the joystick to proceed, the user is then directed to the instructions panel, “Instructions()”, where the user is provided brief instructions. By using the "GLCD_DisplayString()" function, two messages are displayed. The first message says, "Instructions:" with messages 2-7 displaying the actual instructions for the game. The last message says "Select=Push:" which informs the user that in order to begin, the select button should be selected. Once again, the "GLCD_Clear()" function was used to clear the entire screen to red, and then by using the "GLCD_DisplayString()", "GLCD_SetBackColor()" and "GLCD_SetTextColor()" functions. Once the user selects the joystick, the game started. Once the user selects start from the game menu, the "GLCD_Clear()" function clears the screen black and calls the "Game()" function. This initial gameplay consists of the top-down view of a Mario Kart sprite and a green block. In the setup function, the initial y-position and x-position of Mario was declared, along with the initial x- and y-position of each block. The positions of all the blocks are stored in an array of size 8. The bitmap image size of the blocks were 55x17 and the bitmap image size of Mario was 45x45. These images were loaded onto the LCD screen the same way as the photos in the gallery were loaded by converting .bmp files into C files by using the GIMP software. After initializing the positions of the block(s) and Mario, the "GLCD_Bitmap()" function was called to display all the images on the screen. In order to move Mario left or right to dodge the blocks, the user needed to use the joysticks. Nested loops were implemented in an infinite while loop to move Mario. By default, the starting x-position for Mario is 140 and the y-position is 240. A border was created to make sure Mario does not leave the game border and stays within the game frame which is also the screen frame (320x240). A few if statements were implemented to check the width of Mario and ensure that they are less than 320 on the x-position and more than 0 on the x-position, thereby successfully allowing a border to be created. Now I needed to initialize the speed for Mario to ‘drive’ with. This portion was also included in the infinite while loop. By using the "GLCD_Bitmap()" function, Mario is displayed at its initialized x- and y-positions. Afterwards, Mario’s x-speed and y-speed were initialized, being incremented by 0.1 for each direction. If the user was able to successfully maneuver through the block(s) in a level, the "GLCD_Clear" function will clear the screen to green and display a message indicating that the user has done a good job and they advance to the next level where an additional block is added at a predetermined position. At the same time, the LEDs will turn ON depending on which level the user is playing. For example, for the first level, there is one block with one LED turned on. If the user beats Level 1, they proceed to the second level with now two blocks shown on the LCD and now two LEDs are turned on. Which is why this game goes to 8 levels, given that the board has 8 LEDs. If Mario crashes into a block, the user loses the game. Two nested if statements were implemented inside a for loop which was for the collision detection between Mario and the block(s). For this, the x- and y-position of Mario and the x- and y-position of the blocks were compared. For example, if Mario's x-position and their trace was greater than or equal to the block’s x-position and Mario's x-position was less than or equal to the block’s x position and Mario's y-position was greater than or equal to the block’s y-position and Mario's y-position was less than or equal to the block’s y position, then a collision occurred. The "GLCD_Clear" function will clear the screen to black and display a “You lost” message before the user is redirected to the main menu. The progress is never saved, similar to most arcade games, so if the user wants to play the game again, they have to start at the first level.

V. Experimental Results

The results of this project are shown with the following images.

Figure 2: Start Screen

Figure 3: Main Menu

As mentioned previously, with blue and black being my two favourite colours, this was the chosen colour scheme for the start screen and main menu as shown in Figures 2 and 3

respectively. With Figure 3 specifically, the main menu contains the options for the three features: photo gallery, MP3 player, and game center. By default, the user will hover over the photo gallery option, the option is highlighted as dark cyan with the text changing to black. This applies to the other options as the user moves the joystick up and down when going through the main menu. Finally, the user is given directions near the bottom of the LCD in both Figures 2 and 3 in white to make it stand out.

Figure 4: Photo Gallery

Similar to the main menu the photo gallery contains the options for the user to open up each of the 4 photos and go through the others by operating the joystick. The option that is hovered on will simply change the text from magenta to blue without any highlighting, so that the photo can appear without any unwanted background as shown in Figures 5-8.

Figure 5: Photo #1 (good kid, m.A.A.d city)

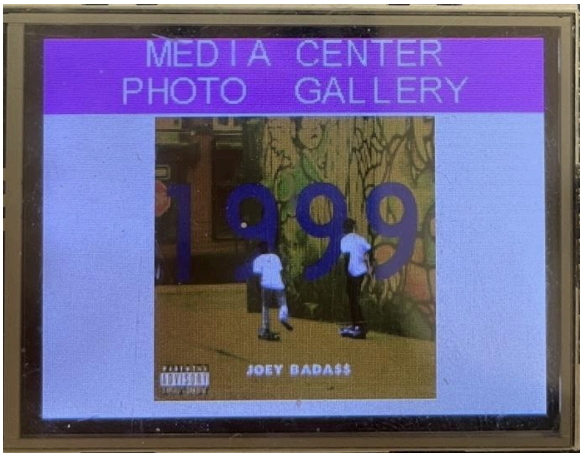

Figure 6: Photo #2 (1999)

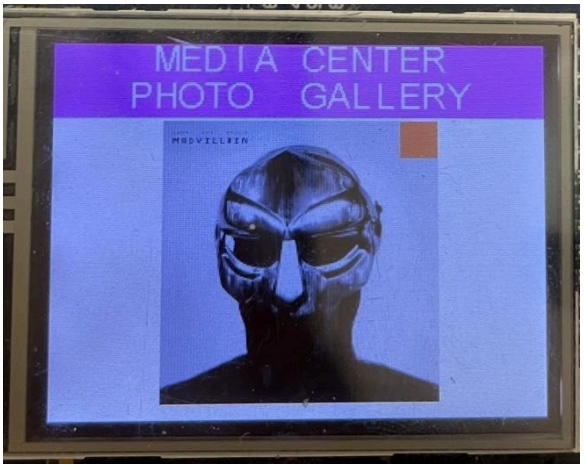

Figure 7: Photo #3 (Madvillainy)

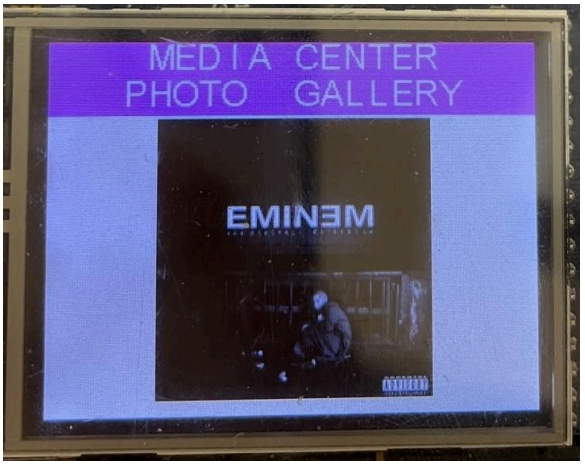

Figure 8: Photo #4 (The Marshall Mathers LP)

The best part about album covers is their square shape. But by using GIMP and the GLCD_Bitmap function, I was able to take full advantage of their shape and place the images in a perfect position.

Figure 9: MP3 Player

Figure 9 shows the display screen for when the user enters the MP3 Player, namely implementing the colour scheme used for Spotify and having the text "AUDIO STREAMING" centered and under the header.

Figure 10: Game title screen (Mario Kart Maze)

This is the start screen used for the game, Mario Kart Maze, which displays the game cover and directions for the user to decide with the joystick if they want to play the game by pushing select or exiting the game by moving the joystick to the left.

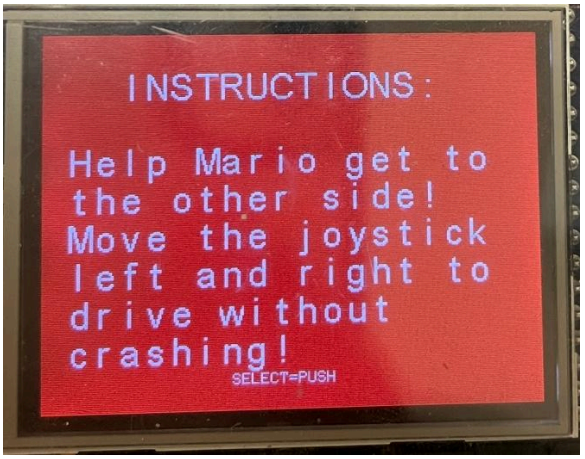

Figure 11: Instructions screen

After pushing select from the start screen, the user is then met with the instructions on the LCD that tells them how to play the game. To proceed with the game, as shown at the bottom, the user must push the joystick to confirm they understood the instructions and want to play the game. As with Figure 10, the red and white colour scheme is maintained to represent the traditional colours of Mario.

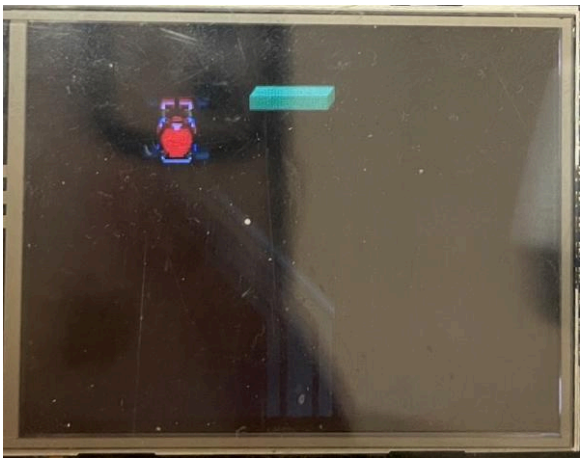

Figure 12: Gameplay screenshot

In this particular screenshot, Mario successfully moves to the left without crashing into the block in order to reach the other side, which is the objective of the game. In that particular case, the user will then be greeted with a congratulatory message as shown in Figure 13.

Figure 13: Congratulations screen

After seeing this for a couple of seconds, the user can then return to the game, only this time with an additional block being added, followed by another LED being lit up on the board.

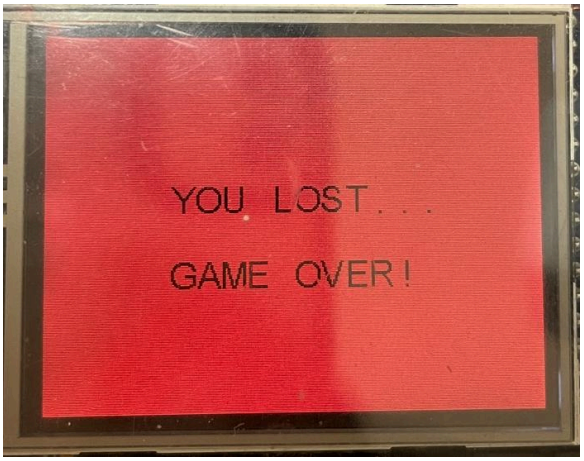

Figure 14: Game Over screen

If Mario drives into a block, the image shown in Figure 14 will be shown for a few seconds before the user is taken back to the main menu of the media center.

VI. Conclusion

Ultimately, the objectives and requirements of the media center were met, while I was still able to explore ways to make it more creative. The photo gallery was able to display my 4 favourite hip-hop albums of all time, the MP3 player was able to successfully stream audio by following the USB Audio example, and the game was able to function as intended. This project was a significant but entertaining learning experience that allowed me to understand more about embedded systems and their applications.

VII. References

Final Project: Media Center. COE718: Embedded System Design. (n.d.). https://www.ecb.torontomu.ca/~courses/coe718/labs/Media-Center.pdf

VIII. Appendix

The following appendix only consists of the primary .c files that were used/altered for the design and development of the media center.

Media_Center.c

```
#include <LPC17xx.H>
#include "string.h"
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"
#include "Media_Center.h"
#include "Photo_Gallery.h"
#include "Game.h"
#include "usbdmain.h"
#include "usb.h"
#include "usbcfg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbaudio.h"
#include "type.h"

#define __FI        1
#define DELAY_2N                    18

int sel=1;
int music=0;
int k;

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

void MP3_Player2 (void){
                    USB_Connect(TRUE);
    NVIC_EnableIRQ(TIMER0_IRQn);
    NVIC_EnableIRQ(USB_IRQn);

                    while( 1 ){
                            if (get_button() == KBD_LEFT){
                                    Media_Center();
                            }
                    }
}

int main (void) {

  LED_Init ();
  GLCD_Init();
        KBD_Init();
  GLCD_Clear(Black);

        GLCD_SetBackColor(DarkCyan);
        GLCD_SetTextColor(Black);
        GLCD_DisplayString(0, 0, __FI,(unsigned char*)"      MEDIA
CENTER   ");
        GLCD_DisplayString(1, 0, __FI,(unsigned char*)"    CHARRAN
");

        GLCD_SetBackColor(Black);
        GLCD_SetTextColor(DarkCyan);
        GLCD_DisplayString(4, 4, __FI,(unsigned char*) "PRESS
SELECT");
        GLCD_DisplayString(5, 6, __FI,(unsigned char*) "TO BEGIN.");
```

```
        GLCD_SetTextColor(White);
        GLCD_DisplayString(27, 15, 0,(unsigned char*)"
SELECT=PUSH     ");

        while (1) {
                if (get_button() == KBD_SELECT){
                        for (k=0;k<2;k++){
                                GLCD_Clear(Black);

                                GLCD_SetBackColor(Black);

                                GLCD_SetTextColor(White);

                                GLCD_DisplayString(5, 5,
__FI,(unsigned char*) "LOADING.");

                                LED_On(0); delay(5);
                                LED_Off(0);
                                LED_On(1); delay(5);
                                LED_Off(1);
                                LED_On(2); delay(5);
                                LED_Off(2);

                                LED_On(3); delay(5);
                                LED_Off(3);

                                LED_On(4); delay(5);
                                LED_Off(4);

                                LED_On(5); delay(5);
                                LED_Off(5);
                                LED_On(6); delay(5);
                                LED_Off(6);
                                LED_On(7); delay(5);
                                LED_Off(7);
                                GLCD_DisplayString(5, 5,
__FI,(unsigned char*) "LOADING..");

                                LED_On(7); delay(5);
                                LED_Off(7);
                                LED_On(6); delay(5);
                                LED_Off(6);
                                LED_On(5); delay(5);
                                LED_Off(5);

                                LED_On(4); delay(5);
                                LED_Off(4);

                                LED_On(3); delay(5);
                                LED_Off(3);

                                LED_On(2); delay(5);
                                LED_Off(2);
                                LED_On(1); delay(5);
                                LED_Off(1);
                                LED_On(0); delay(5);
                                LED_Off(0);
                                GLCD_DisplayString(5, 5,
__FI,(unsigned char*) "LOADING...");
                        }
                        Media_Center();
                }
        }
    }

    void Media_Center(){

        GLCD_Clear(Black);
        GLCD_SetBackColor(DarkCyan);
        GLCD_SetTextColor(Black);
        GLCD_DisplayString(0, 0, __FI,(unsigned char*)"      MEDIA
CENTER   ");
        GLCD_DisplayString(1, 0, __FI,(unsigned char*)"      MAIN
MENU    ");

        GLCD_SetBackColor(Black);
        GLCD_DisplayString(2, 0, __FI,(unsigned char*)"                ");
        GLCD_DisplayString(3, 0, __FI,(unsigned char*)"                ");

        GLCD_SetBackColor(Black);
        GLCD_SetTextColor(DarkCyan);
        GLCD_DisplayString(4, 0, __FI,(unsigned char*)" -> PHOTO
GALLERY");
        GLCD_DisplayString(5, 0, __FI,(unsigned char*)" -> MP3
PLAYER");
        GLCD_DisplayString(6, 0, __FI,(unsigned char*)" -> GAME");
        GLCD_SetTextColor(White);
        GLCD_DisplayString(26, 15, 0,(unsigned char*)"     TOGGLE
UP/DOWN   ");
        GLCD_DisplayString(27, 15, 0,(unsigned
char*)"SELECT=PUSH, BACK=LEFT");

        while (1) {

                if (get_button() == KBD_UP){
                        sel--;
                        if (sel == 1){

GLCD_SetTextColor(DarkCyan);

GLCD_DisplayString(5, 0, __FI,(unsigned char*)" -> MP3 PLAYER");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)" -> GAME");
                                delay(5);
                        }
                        else if (sel == 2){

GLCD_SetTextColor(DarkCyan);

GLCD_DisplayString(4, 0, __FI,(unsigned char*)" -> PHOTO GALLERY");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)" -> GAME");
                                delay(5);

                        }
                        else if (sel == 3){

GLCD_SetTextColor(DarkCyan);

GLCD_DisplayString(4, 0, __FI,(unsigned char*)" -> PHOTO GALLERY");

GLCD_DisplayString(5, 0, __FI,(unsigned char*)" -> MP3 PLAYER");
                                delay(5);
                        }

                }       else if (get_button() == KBD_DOWN){
                        sel++;
                        if (sel == 1){

GLCD_SetTextColor(DarkCyan);

GLCD_DisplayString(5, 0, __FI,(unsigned char*)" -> MP3 PLAYER");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)" -> GAME");
                                delay(5);
                        }
                        else if (sel == 2){

GLCD_SetTextColor(DarkCyan);

GLCD_DisplayString(4, 0, __FI,(unsigned char*)" -> PHOTO GALLERY");
```

```
GLCD_DisplayString(6, 0, __FI,(unsigned char*)" -> GAME");
                                                delay(5);

                                        }
                                        else if (sel == 3){

GLCD_SetTextColor(DarkCyan);

GLCD_DisplayString(4, 0, __FI,(unsigned char*)" -> PHOTO GALLERY");

GLCD_DisplayString(5, 0, __FI,(unsigned char*)" -> MP3 PLAYER");
                                                delay(5);
                                        }

                        } else if (get_button() ==  KBD_SELECT){
                                        if (sel == 1){
                                                Photo_Gallery();
                                        }
                                        else if (sel == 2){
                                                GLCD_Clear(Black);

GLCD_SetBackColor(Green);

GLCD_SetTextColor(Black);

                        GLCD_DisplayString(0, 0, __FI,(unsigned char*)"    MEDIA
CENTER    ");
                          GLCD_DisplayString(1, 0, __FI,(unsigned char*)"     MP3
PLAYER    ");


GLCD_SetBackColor(Black);

GLCD_SetTextColor(Green);

GLCD_DisplayString(4, 2, __FI,(unsigned char*) "  AUDIO");

GLCD_DisplayString(5, 4, __FI,(unsigned char*) "STREAMING...");

GLCD_DisplayString(26, 20, 0,(unsigned char*)"Go Back = L");

                                                if(music == 0){

MP3_Player ();
                                                }
                                                else if (music == 1) {

MP3_Player2 ();
                                                }

                                        }
                                        else if (sel == 3){

Game_Check();
                                        }
                        }

                        switch(sel) {
                                case 1:

GLCD_SetBackColor(DarkCyan);
                GLCD_SetTextColor(Black);
                  GLCD_DisplayString(0, 0, __FI,(unsigned char*)"    MEDIA
CENTER    ");
                    GLCD_DisplayString(1, 0, __FI,(unsigned char*)"     MAIN
MENU    ");

                GLCD_SetBackColor(Black);
                        GLCD_DisplayString(2,   0,   __FI,(unsigned   char*)"
");
        GLCD_DisplayString(3, 0, __FI,(unsigned char*)"                    ");

GLCD_SetBackColor(DarkCyan);
                                GLCD_SetTextColor(Black);
                                        GLCD_DisplayString(4,           0,
__FI,(unsigned char*)" -> PHOTO GALLERY          ");

                                        GLCD_SetBackColor(Black);
                                GLCD_SetTextColor(DarkCyan);
                                        GLCD_DisplayString(5,           0,
__FI,(unsigned char*)" -> MP3 PLAYER             ");
                                        GLCD_DisplayString(6,           0,
__FI,(unsigned char*)" -> GAME                   ");

          break;
        case 2:
          GLCD_SetBackColor(DarkCyan);
                GLCD_SetTextColor(Black);
                  GLCD_DisplayString(0, 0, __FI,(unsigned char*)"    MEDIA
CENTER    ");
                    GLCD_DisplayString(1, 0, __FI,(unsigned char*)"     MAIN
MENU    ");

                GLCD_SetBackColor(Black);
                        GLCD_DisplayString(2,   0,   __FI,(unsigned   char*)"
");
        GLCD_DisplayString(3, 0, __FI,(unsigned char*)"                    ");

                                GLCD_SetBackColor(DarkCyan);
                                GLCD_SetTextColor(Black);
                                        GLCD_DisplayString(5,           0,
__FI,(unsigned char*)" -> MP3 PLAYER             ");

                                        GLCD_SetBackColor(Black);
                                GLCD_SetTextColor(DarkCyan);
                                        GLCD_DisplayString(4,           0,
__FI,(unsigned char*)" -> PHOTO GALLERY          ");
                                        GLCD_DisplayString(6,           0,
__FI,(unsigned char*)" -> GAME                   ");

          break;
        case 3:
          GLCD_SetBackColor(DarkCyan);
                GLCD_SetTextColor(Black);
                  GLCD_DisplayString(0, 0, __FI,(unsigned char*)"    MEDIA
CENTER    ");
                    GLCD_DisplayString(1, 0, __FI,(unsigned char*)"     MAIN
MENU    ");

                GLCD_SetBackColor(Black);
                        GLCD_DisplayString(2,   0,   __FI,(unsigned   char*)"
");
        GLCD_DisplayString(3, 0, __FI,(unsigned char*)"                    ");

                                GLCD_SetBackColor(DarkCyan);
                                GLCD_SetTextColor(Black);
                                        GLCD_DisplayString(6,           0,
__FI,(unsigned char*)" -> GAME                   ");

                                        GLCD_SetBackColor(Black);
                                GLCD_SetTextColor(DarkCyan);
                                        GLCD_DisplayString(4,           0,
__FI,(unsigned char*)" -> PHOTO GALLERY          ");
                                        GLCD_DisplayString(5,           0,
__FI,(unsigned char*)" -> MP3 PLAYER             ");

          break;
                        }
    }
  }
```

Photo_Gallery.c

```
#include <LPC17xx.H>
#include "string.h"
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"
#include "Media_Center.h"
#include "Photo_Gallery.h"
#include "Photo_Viewer.h"

#define __FI       1
#define DELAY_2N                    18
int sel2=1;

void delay2 (int cnt) {
  cnt <<= DELAY_2N;
  while (cnt--);
}

void Photo_Gallery(){
        GLCD_Clear(White);

        GLCD_SetBackColor(Magenta);
        GLCD_SetTextColor(White);
        GLCD_DisplayString(0, 0, __FI,(unsigned char*)"      MEDIA
CENTER   ");
        GLCD_DisplayString(1, 0, __FI,(unsigned char*)"      PHOTO
GALLERY   ");

        GLCD_SetBackColor(White);
        GLCD_SetTextColor(Magenta);
        GLCD_DisplayString(3, 0, __FI,(unsigned char*)"      PHOTO
#1");
        GLCD_DisplayString(4, 0, __FI,(unsigned char*)"      PHOTO
#2");
        GLCD_DisplayString(5, 0, __FI,(unsigned char*)"      PHOTO
#3");
        GLCD_DisplayString(6, 0, __FI,(unsigned char*)"      PHOTO
#4");


        GLCD_SetTextColor(Black);
        GLCD_DisplayString(26, 15, 0,(unsigned char*)"      TOGGLE
UP/DOWN   ");
        GLCD_DisplayString(27,              15,              0,(unsigned
char*)"SELECT=PUSH, BACK=LEFT");

        while (1) {

                if (get_button() == KBD_UP){
                        sel2--;
                        if (sel2 == 1){

GLCD_SetTextColor(Magenta);

GLCD_DisplayString(4, 0, __FI,(unsigned char*)"     PHOTO #2");

GLCD_DisplayString(5, 0, __FI,(unsigned char*)"     PHOTO #3");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)"      PHOTO #4");

                                delay2(5);
                        }
                        else if (sel2 == 2){

GLCD_SetTextColor(Magenta);

GLCD_DisplayString(3, 0, __FI,(unsigned char*)"     PHOTO #1");

GLCD_DisplayString(5, 0, __FI,(unsigned char*)"     PHOTO #3");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)"      PHOTO #4");

                                delay2(5);

                        }
                        else if (sel2 == 3){


GLCD_SetTextColor(Magenta);

GLCD_DisplayString(3, 0, __FI,(unsigned char*)"     PHOTO #1");

GLCD_DisplayString(4, 0, __FI,(unsigned char*)"     PHOTO #2");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)"      PHOTO #4");

                                delay2(5);
                        }
                        else if (sel2 == 4){

GLCD_SetTextColor(Magenta);

GLCD_DisplayString(3, 0, __FI,(unsigned char*)"     PHOTO #1");

GLCD_DisplayString(4, 0, __FI,(unsigned char*)"     PHOTO #2");

GLCD_DisplayString(5, 0, __FI,(unsigned char*)"      PHOTO #3");

                                delay2(5);
                        }


                }       else if (get_button() == KBD_DOWN){
                        sel2++;
                        if (sel2 == 1){

GLCD_SetTextColor(Magenta);

GLCD_DisplayString(4, 0, __FI,(unsigned char*)"     PHOTO #2");

GLCD_DisplayString(5, 0, __FI,(unsigned char*)"     PHOTO #3");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)"      PHOTO #4");

                                delay2(5);
                        }
                        else if (sel2 == 2){

GLCD_SetTextColor(Magenta);

GLCD_DisplayString(3, 0, __FI,(unsigned char*)"     PHOTO #1");

GLCD_DisplayString(5, 0, __FI,(unsigned char*)"     PHOTO #3");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)"      PHOTO #4");

                                delay2(5);

                        }
                        else if (sel2 == 3){

GLCD_SetTextColor(Magenta);

GLCD_DisplayString(3, 0, __FI,(unsigned char*)"     PHOTO #1");

GLCD_DisplayString(4, 0, __FI,(unsigned char*)"     PHOTO #2");

GLCD_DisplayString(6, 0, __FI,(unsigned char*)"      PHOTO #4");

                                delay2(5);
                        }
```

```
                                        else if (sel2 == 4){
GLCD_SetTextColor(Magenta);
GLCD_DisplayString(3, 0, __FI,(unsigned char*)"     PHOTO #1");
GLCD_DisplayString(4, 0, __FI,(unsigned char*)"     PHOTO #2");
GLCD_DisplayString(5,  0,  __FI,(unsigned  char*)"          PHOTO  #3");
                                                delay2(5);
                                        }

                        } else if (get_button() ==  KBD_SELECT){
                                        if (sel2 == 1){
                                                Photo_Viewer(1);
                                        }
                                        else if (sel2 == 2){
                                                Photo_Viewer(2);

                                        }
                                        else if (sel2 == 3){

                                                Photo_Viewer(3);

                                        }
                                        else if (sel2 == 4){
                                                Photo_Viewer(4);
                                        }

                        } else if (get_button() ==  KBD_LEFT){

                                        Media_Center();
                                }

                        switch(sel2) { //Highlighting options on the LCD when
hovering
                                case 1:
                                        GLCD_SetTextColor(Blue);
                                        GLCD_DisplayString(3,        0,
__FI,(unsigned char*)"     PHOTO #1");
                                break;
                                case 2:
                                        GLCD_SetTextColor(Blue);
                                        GLCD_DisplayString(4,        0,
__FI,(unsigned char*)"     PHOTO #2");
                                break;
                                case 3:
                                        GLCD_SetTextColor(Blue);
                                        GLCD_DisplayString(5,        0,
__FI,(unsigned char*)"     PHOTO #3");
                                break;
                                case 4:
                                        GLCD_SetTextColor(Blue);
                                        GLCD_DisplayString(6,        0,
__FI,(unsigned char*)"     PHOTO #4");
                                break;
                        }
                }

        }
```

Photo_Viewer.c

```
#include <LPC17xx.H>
#include "string.h"
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"
#include "Media_Center.h"
#include "Photo_Gallery.h"
#include "Photo_Viewer.h"

#define __FI       1

extern unsigned char GKMC[];
extern unsigned char JOEY1999[];
extern unsigned char MADVILLAINY[];
extern unsigned char MMLP[];

int Photo_Viewer(int j){

        GLCD_ClearLn(8,1);
        GLCD_ClearLn(9,1);

        while (1) {

                        if (j == 1){
                                        GLCD_Bitmap    (70,
50, 180, 180, GKMC);
                        }
                        else if (j == 2){

                                        GLCD_Bitmap    (70,
50, 180, 180, JOEY1999);
                        }
                        else if (j == 3){
                                        GLCD_Bitmap    (70,
50, 180, 180, MADVILLAINY);
                        }
                        else if (j == 4){
                                        GLCD_Bitmap    (70,
50, 180, 180, MMLP);
                        }

                        if (get_button() ==  KBD_LEFT){
                                if (j == 1){
                                        Photo_Gallery();
                                }
                                else if (j == 2){

                                        Photo_Gallery();

                                }
                                else if (j == 3){

                                        Photo_Gallery();

                                }
                                else if (j == 4){
                                        Photo_Gallery();

                                }

                        }
                }
        }
```

Game.c

```
#include <LPC17xx.H>
#include "string.h"
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"
#include "Media_Center.h"
#include "Photo_Gallery.h"
#include "Game.h"

#define __FI       1
#define DELAY_2N                18

int mario_x = 180;
int mario_y = 240;
```

```
    int bounce=0;
    int game_over=0, win = 0, blocknum = 1, once = 0;
    int paddle, i;
    int x1[] = {130, 70, 110, 150, 190, 230, 240, 60};
                                    //array for the x-postion of the block
    int x2[] = {169, 109, 149, 189, 229, 269, 279, 99};
                                    //array for the 2nd x-postion of the block
    int y1[] = {30, 90, 60, 110, 50, 150, 100 ,160};
                                    //array for the y-postion of the
block
    int y2[] = {69, 129, 99, 149, 89, 189, 139, 199};
                                    //array for the 2nd y-postion of
the block

    extern unsigned char MARIO[];
    extern unsigned char BLOCKS[];
    extern unsigned char MARIOINTRO[];

    //This function creates a delay
    void delay3 (int cnt) {
      cnt <<= DELAY_2N;
      while (cnt--);
    }

    //Introduction Menu Screen
    void Game_Check(void){
            GLCD_Clear(White);

            GLCD_SetBackColor(Red);
            GLCD_SetTextColor(White);
            GLCD_DisplayString(0, 0, __FI,(unsigned char*)"      MEDIA
CENTER   ");
            GLCD_DisplayString(1, 0, __FI,(unsigned char*)"      GAME
");
            GLCD_Bitmap (110, 60, 100, 101, MARIOINTRO);

            GLCD_SetBackColor(White);
            GLCD_SetTextColor(Red);
            GLCD_DisplayString(7, 2, __FI,(unsigned char*)"MARIO KART
MAZE");

            GLCD_SetTextColor(Black);
            GLCD_DisplayString(27,          15,          0,(unsigned
char*)"SELECT=PUSH, BACK=LEFT");
            while(1){
                    if (get_button() == KBD_SELECT){
                            Instructions();
                    }
                    else if(get_button() == KBD_LEFT){

                                    Media_Center();
                            }
            }
    }

    //Instructions Menu Screen
    void Instructions(void){
            GLCD_Clear(Red);

            GLCD_SetBackColor(Red);
            GLCD_SetTextColor(White);
            GLCD_DisplayString(1,          3,          __FI,(unsigned
char*)"INSTRUCTIONS:");
            GLCD_DisplayString(3, 1, __FI,(unsigned char*)"Help Mario get
to");
            GLCD_DisplayString(4, 1, __FI,(unsigned char*)"the other
side!");
            GLCD_DisplayString(5, 1, __FI,(unsigned char*)"Move the
joystick");
            GLCD_DisplayString(6, 1, __FI,(unsigned char*)"left and right
to");
            GLCD_DisplayString(7, 1, __FI,(unsigned char*)"drive without");
            GLCD_DisplayString(8, 1, __FI,(unsigned char*)"crashing!");
            GLCD_DisplayString(27,          15,          0,(unsigned          char*)"
SELECT=PUSH      ");
            while(1){
                    if (get_button() == KBD_SELECT){
                                    Game();
                    }
            }
    }

    //---------- Game ----------
    void Game (void) {
            if(once == 0){
                    LED_Init ();
                    KBD_Init();
                    once++;
            }
            game_over = 0;
            win = 0;
            mario_x = 140;
            mario_y = 240;

            GLCD_Clear(Black);
            GLCD_Bitmap (mario_x, mario_y, 45, 45, MARIO);

    while(1){
            for(i=0;i<blocknum;i++){
                            GLCD_Bitmap          (x1[i],y1[i],55,17,
BLOCKS);
                            LED_On(i);
            }

            while (1) {
                    if (get_button() == KBD_RIGHT){
                            if (mario_x+40 < 320){
                                    mario_x++;
                            }
                    }
                    else if (get_button() == KBD_LEFT){
                            if (mario_x-1 > 0){
                                    mario_x--;
                            }
                    }

                    if (bounce ==0){
                            mario_y=mario_y-0.1;
                    }

                            GLCD_Bitmap (mario_x, mario_y, 45, 45,
MARIO);


            //Collision Detection
            for(i=0;i<blocknum;i++){
                    if(mario_x-10 < x2[i] && x1[i] < mario_x+39){
                            if(mario_y+27 < y2[i] && y1[i] <
mario_y+66){
                                    game_over = 1;
                            }
                    }
            }

            if(mario_y <= 0)
                    win = 1;

            if(game_over == 1 || win == 1)
                    break;
            }

    //If Mario reaches top of the screen
            if (win == 1){
                    GLCD_Clear(Green);
```

```
                        GLCD_SetBackColor(Green);
                        GLCD_SetTextColor(Black);
                        GLCD_DisplayString(4,   5,   __FI,(unsigned
char*)"GOOD JOB!");
                        GLCD_DisplayString(6,   4,   __FI,(unsigned
char*)"LEVEL UP..");
                        delay3(250);
                        blocknum++;
                        Game();
                }

                //If Mario collides with block
                else if (game_over == 1){
                        GLCD_Clear(Red);
                        GLCD_SetBackColor(Red);
                        GLCD_SetTextColor(Black);
                        GLCD_DisplayString(4,   5,   __FI,(unsigned
char*)"YOU LOST...");
                        GLCD_DisplayString(6,   5,   __FI,(unsigned
char*)"GAME OVER!");
                        blocknum=1;

LED_Off(0);LED_Off(1);LED_Off(2);LED_Off(3);LED_Off(4);LED_Off(5)
;LED_Off(6);     LED_Off(7);
                        delay3(250);
                        Media_Center();
                }
        }
        }
```

usbdmain.c

```
#include "LPC17xx.h"
#include "type.h"
#include "usb.h"
#include "usbcfg.h"
#include "usbhw.h"
#include "usbcore.h"
#include "usbaudio.h"
#include "usbdmain.h"
#include "KBD.h"
#include "Media_Center.h"

extern  void SystemClockUpdate(void);
extern uint32_t SystemFrequency;
uint8_t  Mute;                          /* Mute State */
uint32_t Volume;                        /* Volume Level */

#if USB_DMA
uint32_t *InfoBuf = (uint32_t *)(DMA_BUF_ADR);
short *DataBuf = (short *)(DMA_BUF_ADR + 4*P_C);
#else
uint32_t InfoBuf[P_C];
short DataBuf[B_S];                     /* Data Buffer */
#endif

uint16_t  DataOut;                      /* Data Out Index */
uint16_t  DataIn;                       /* Data In Index */

uint8_t   DataRun;                      /* Data Stream Run State */
uint16_t  PotVal;                       /* Potenciometer Value */
uint32_t  VUM;                          /* VU Meter */
uint32_t  Tick;                         /* Time Tick */


/*
 * Get Potenciometer Value
 */

void get_potval (void) {
  uint32_t val;

  LPC_ADC->CR |= 0x01000000;            /* Start A/D Conversion */
  do {
    val = LPC_ADC->GDR;                 /* Read A/D Data Register */
    } while ((val & 0x80000000) == 0);  /* Wait for end of A/D
Conversion */
  LPC_ADC->CR &= ~0x01000000;           /* Stop A/D Conversion */
  PotVal = ((val >> 8) & 0xF8) +        /* Extract Potenciometer Value */
           ((val >> 7) & 0x08);
}


/*
 * Timer Counter 0 Interrupt Service Routine
 *   executed each 31.25us (32kHz frequency)
 */

void TIMER0_IRQHandler(void)
{
  long  val;
  uint32_t cnt;

  if (DataRun) {                        /* Data Stream is running */
    val = DataBuf[DataOut];             /* Get Audio Sample */
    cnt = (DataIn - DataOut) & (B_S - 1);   /* Buffer Data Count */
    if (cnt == (B_S - P_C*P_S)) {       /* Too much Data in Buffer */
      DataOut++;                        /* Skip one Sample */
    }
    if (cnt > (P_C*P_S)) {              /* Still enough Data in Buffer */
      DataOut++;                        /* Update Data Out Index */
    }
    DataOut &= B_S - 1;                 /* Adjust Buffer Out Index */
    if (val < 0) VUM -= val;            /* Accumulate Neg Value */
    else         VUM += val;            /* Accumulate Pos Value */
    val  *= Volume;                     /* Apply Volume Level */
    val >>= 16;                         /* Adjust Value */
    val  += 0x8000;                     /* Add Bias */
    val  &= 0xFFFF;                     /* Mask Value */
  } else {
    val = 0x8000;                       /* DAC Middle Point */
  }

  if (Mute) {
    val = 0x8000;                       /* DAC Middle Point */
  }

  LPC_DAC->CR = val & 0xFFC0;           /* Set Speaker Output */

  if ((Tick++ & 0x03FF) == 0) {         /* On every 1024th Tick */
    get_potval();                       /* Get Potenciometer Value */
    if (VolCur == 0x8000) {             /* Check for Minimum Level */
      Volume = 0;                       /* No Sound */
    } else {
      Volume = VolCur * PotVal;         /* Chained Volume Level */
    }
    val = VUM >> 20;                    /* Scale Accumulated Value */
    VUM = 0;                            /* Clear VUM */
    if (val > 7) val = 7;               /* Limit Value */
  }

  LPC_TIM0->IR = 1;                     /* Clear Interrupt Flag */

//----------------- Stopping the streaming -----------------
        if (get_button()==KBD_LEFT){
                NVIC_DisableIRQ(TIMER0_IRQn);
                NVIC_DisableIRQ(USB_IRQn);
                USB_Connect(FALSE);
        }
//----------------- Stopping the streaming -----------------

}
```

```
/*****************************************************************
*****************
**   Main Function  main()
*****************************************************************
*****************/
void MP3_Player (void)
{
 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 */
 NVIC_EnableIRQ(TIMER0_IRQn);

 USB_Init();                                  /* USB Initialization
*/
 USB_Connect(TRUE);                   /* USB Connect */

 /********* The main Function is an endless loop **********/
 music = 1;
        while( 1 ){
                if (get_button() == KBD_LEFT){
                 Media_Center();
                }
        }
}
/*****************************************************************
*****************
**                   End Of File
*****************************************************************
*****************/
```