Toronto Metropolitan University

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

Course Title:	Embedded Systems Design
Course Number:	COE718
Semester/Year (e.g.F2016)	F2022

Instructor:	Gul Khan

Assignment/Lab Number:	Final Report
Assignment/Lab Title:	

Submission Date:	
Due Date:	December 2, 2022

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Valenti	Anthony	500948260	2	AV

*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

Abstract

This report shows the steps behind developing a media center system to be deployed on the MCB1700 board. The media center software is all designed using C and implements three functionalities, a photo gallery, a media player and a connect four game. A user interacts with the media center by using the joystick on the MCB1700 board.

Introduction

The goal of this project is to develop a media center using the MCB1700 board and Keil uVision to develop all of the associated software written in C. The media center implements three main functionalities, a photo gallery, an mp3 audio streamer and a playable game of connect four.

Upon launching the media center the user is able to interact with the menu on the LCD screen using the board's joystick. The joystick is moved in the up/down directions to cycle through the three main menu options while clicking the joystick selects the option that is currently highlighted on the screen. The board's LED lights are used to indicate the current status of the media center, LED 0 represents the menu state while LED's 1 through 3 represent the photo gallery, music player and the connect four game respectively.

The user can exit back to the main menu by clicking the joystick in, except when playing the connect four game in which the user can instead move the joystick to the left to return back to the main menu.

Past Work / Review

The most relevant past work pertaining to this final project was conducted in labs 1 and 2. These labs established the fundamentals of working with the LCD screen, LED lights and interfacing with the board's joystick.

These functionalities can be accomplished in a variety of ways, however in this instance of the project they were implemented using the given .C and .h files from labs 1 and 2, specifically the GLCD, LED and KBD files. Including the GLCD file in the projects directory and initializing the screen via the CLGD_Init() method, allows full control of the LCD screen.

The main way of interacting with the LCD is writing strings to the screen to be displayed. This is accomplished with the GLCD_DisplayString() function. This function has three parameters that can be used to customize the output which are the row, column and string. The row and column are assigned by passing in an integer number that represents which row and column the text will be rendered in. The higher the row number the lower the string will be displayed on the LCD, and the higher the column number the string will be moved to the right. Other GLCD functions are used for stylistic purposes such as the GLCD_Clear(), GLCD_SetBackColor() and GLCD_SetTextColor() functions. In these functions a color is passed through as a parameter and the GLCD will clear the screen with that color, set

the background color of text and set text color respectively. Combining these discussed functions of the GLCD files allows all the necessary manipulation of the LCD screen required to complete the project.

Next is the LED file which is used to interface with the board's LED lights. After importing the LED files into the correct directory the LED_Init() method can be called to initialize the interfacing of the lights. From here LED lights 0-7 can be turned on and off using the provided methods LED_On() and LED_Off(). Each of these functions takes a single integer as a parameter which corresponds to a LED light on the board. Refer to **Figure 1** as a reference for the LED lights, the light at the top is associated with index 0 while the number at the bottom would be 7.

Finally, the last provided file from previous work used in collaboration with the media center project is the KBD file. These files allow access to the board's joystick after successfully importing and calling the KBD_Init() method. The joystick has five main directions each with an associated value up,down,left,right and inward or a clicking motion. The joystick motions are processed via the get_button() method within the KBD file which gets the associated value of the current position of the joystick. The current position of the joystick can then be compared to its known values defined as KBD_SELECT, KBD_UP, KBD_DOWN, KBD_LEFT, KBD_RIGHT. Based on this comparison the joystick can be used to execute code conditionally based on user input from the joystick.

In conclusion, previous work from labs 1 and 2 was mainly used to accomplish the results obtained in the media center. The specific work from these labs used in this project is interfacing with the board's LCD screen, LED lights and joystick. Each of these accessories are controlled with the provided source code files and their respective methods provided through the GLCD.C , LED.C , and KBD.C files.

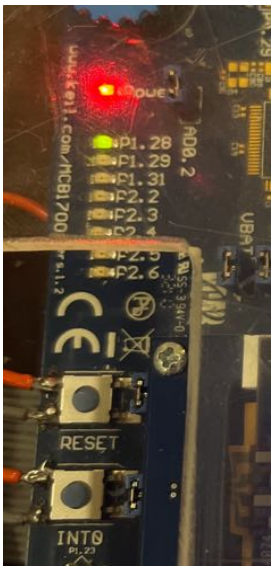

Figure 1: The MCB1700 board LED's

Methodology

The overall methodology for approaching the project is pretty simple and straightforward. The end goal is clearly outlined in the project report guidelines and can be used as a reference for the project's functional requirements. Functional requirements for the project are considered mandatory and do not have room for modifications. In the instance of this

project there are four functional requirements, implement an interactable menu, photo gallery, media streamer and game. The non-functional requirements are those that are not integral to the operation of the project and thus have been left up to the decision of the developer (myself in this instance). Some non-functional requirements for this project are that the media center should be organized, visually appealing, responsive, and easy to use.

When analyzing the functional requirements to be implemented the project guideline was referenced to achieve the results for the photogallery and audio streamer. The project guideline set forth the methodology in which displaying images on the LCD screen can be achieved. The first step is to vertically flip and scale the chosen image in order to be displayed correctly on the LCD. Next, the image file must be converted to a .c file using the GIMP program. After importing the resulting .c file into the correct project directories, the image can be rendered using the GLCD_BitMap() function. This function can be called while passing the parameters of x,y coordinates to be rendered at, length/width of the image and finally the pixel data array for the image.

The audio streaming functionality is achieved completely through the provided sample code and no code modifications/additions are required for any functional requirements.

The most complex methodology went into the game feature to be implemented as this was completely open-ended in its requirements. Originally I had planned to implement a snake game but the results yielded too many errors and it lacked originality amongst the other projects being conducted. This led to the implementation of connect four, a simple but fun game. The logic behind this game is quite simple, two players defined as ‘X’ and ‘O’ alternate taking turns until a tie or winner is reached. The game can be broken down into four states: start, take a turn, check, and finish. Within the start state the board must be rendered as well as all the relevant variables are initialized. Next is the taking a turn state, this state must keep track of which player is taking a turn as well as allow the user to select where they would like to place their piece. After the turn is completed a series of checks are completed to see if an individual player has accomplished four consecutive pieces in the horizontal, vertical, or diagonal directions. If any of these checks return true then the game finally transitions to the finish state in which the winner is declared and the option to exit or play-again is provided.

Selecting the proper run-time environment components also contributes to the process of creating this project. Certain components within Keil can be selected to provide board support APIs such as LED or GPIO pin support. However since my

implementation for these features was accomplished using the previously described LED.c and KBD.c files these components were not selected. Ultimately the project environment matches the example usb audio project as it contains all the much more complex settings required to enable streaming audio over USB between the MCB1700 board and the computer. In previous projects the heap size would sometimes need to be increased in order to handle the memory requirements of the program, however the default heap size in the pre-defined project settings was large enough to support the successful execution of the project.

The non-functional requirements made for this project were achieved in a variety of ways. The first being using the highlighting effect on the menu screen to show users which option they are currently selecting. This effect not only looks visually appealing but it is also very user friendly as it instantly shows the currently selected options in an identifiable way. The code responsible for the highlighting effect is also very simple, easy to understand, and reusable. Whenever the user moves the joystick a counter is incremented/decremented and this counter conditionally executes a block of code that changes the background and text color to give the highlighted appearance.

Furthermore, each program in the media center is displayed on the LCD screen with a title banner on the top for organizational reasons as well as just filling up some white space which does not look visually appealing.

Along with these non-functional requirements made for the final product, I also wanted my written code to be organized and easy to understand/read. This is considered good coding practice as organized code is easier to debug and find mistakes, it also makes it much easier to implement any modifications. Keeping the code organized and readable is achieved in two different ways, one being utilizing functions and the other being separating code into different files. In the C programming language a function is a block of code that is only executed when called to and it is very useful for executing code that runs multiple times. Functions also improve readability by separating chunks of code into functions with appropriate titles, for example the code that contributes to displaying information on the LCD screen can be contained in a function called “display”.

The second factor that improves code readability is separating code into separate files. This is by-far the most common practise to improve code readability, it allows all the code pertaining to a specific function be contained in its own file. So in the instance of the project, code is separated into files such as menu.c, photogallery.c , game.c , etc. Each title is a descriptor of what the code within accomplishes. To implement all these different files together header files must be used, which allow the functions

within these files to be called upon from different files. For this project the menu.c file acts as the “main” file calling the functions from the other components to run them.

In conclusion, the methodology behind this project can be analyzed through its functional and nonfunctional requirements. The functional requirements were predominately outlined within the project guidelines, with the exception of the playable game. The non functional requirements pertain mostly to stylistic and organizational decisions. Ultimately, both functional and nonfunctional requirements play an integral part when considering the methodology behind the media center.

Design

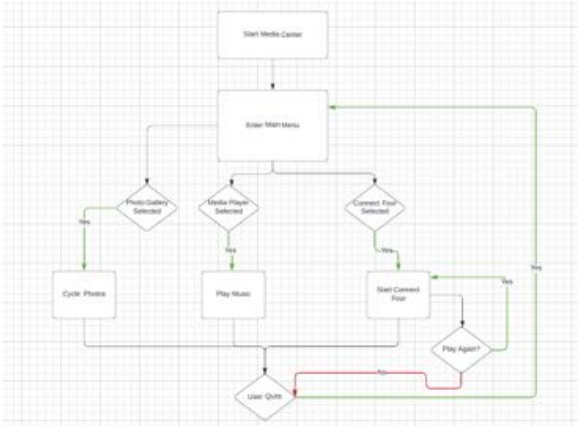

Figure 2: Flowchart diagram to represent desired program flow.

The design process is started by designing a flowchart as seen in ***figure 2*** *to plan out how the media center will flow. Upon starting the media center from a flash reload the user will be brought to the main menu. Within this main menu the user has three decisions or options, to start the photo gallery, the media player or to play connect four. Upon selecting one of these options the media center will launch the selected functionality and the user will be displayed the corresponding information. From this state, the user will be allowed to “quit” in which they will be brought back to the main menu. This flowchart serves as the template when writing the code, it is clear from this diagram that at minimum a menu file along with a photo gallery, media player and “connect four” game files will be used to design the system.*

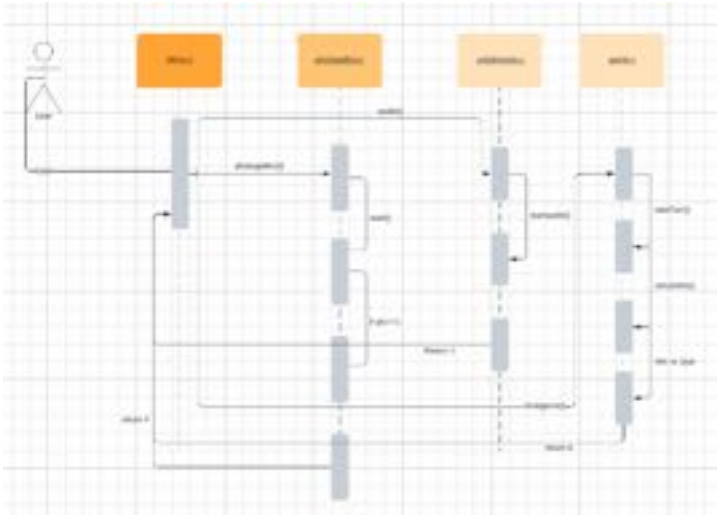

Figure 3: Sequence diagram of media center

To obtain a closer look into the design of each functionality we can look at the sequence diagram which shows the interactions of specific functions within files. The sequence diagram

provides a more in depth look into the flow of the program, specifically which functions we will use to follow the flow outlined by the flowchart. From the sequence diagram above in **figure 3** we can see that the user interacts with the menu.c file which contains the program's main method. From here depending on the user selection, the specific functions will be called to execute the other functionalities. For example, if the user selects the game function, the startgame() method will be called from the game.c file. From here the takeTurn() and checkWin() functions will be used to carry out the logic of connect four. Finally, upon completion or user request to quit, the game.c file will return 0 signifying the end of that execution and returning the user to the main menu. It is important to note that this sequence diagram was used in the early stages of development and therefore does not reflect all of the functions and function names used in actual implementation. Instead the sequence diagram provides an outline of which functions that can be used to accomplish the desired program flow. The sequence diagram also shows how the different files will interact with each other during program execution.

Results

Combining all the work done in the design stage of the project yields the final results. Depicted in **Figure 4** above is the final implementation of the menu. The functional requirements are all clearly met, the user can scroll through the three options presented using the joystick, as well as LED 0 is lit up

Figure 4: Interactive menu screen

depicting the “menu” state as described previously. The non-functional requirements are also clearly met, as previously discussed the menu screen is simple to understand, and visually appealing by reducing the amount of on screen white space. The ability to scroll through menu selections is accomplished by a series of if statements that increments/decrements a counter variable depending on the input from the joystick. This counter variable is used to conditionally render which option should be highlighted blue, and it also is used to determine which option the user selects. For example if the user selects the photo gallery option, the counter variable will equal to 1. This variable is than passed to the switch game function in the form of “switchgame(1)” , which then calls the “photogallery()” function that deals with

the operation of the photo gallery. Before rendering the associated elements of the photo gallery functionality on the LCD screen, the screen must be cleared using the GLCD_Clear(White) command which acts as a sort of eraser to ensure new text is not rendered on top of old text. The menu functionality exists within an infinite while loop, meaning that to exit back to the menu from other functions, a value of 0 is returned and the program resumes its execution within the menu loop.

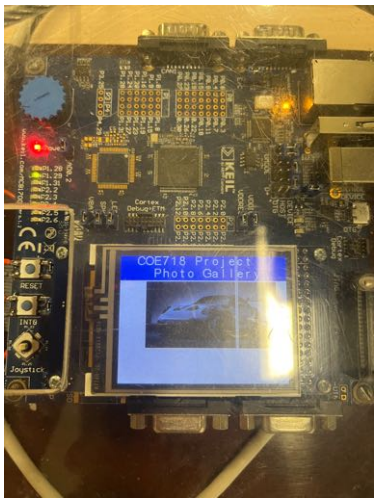

Figure 5: Photogallery picture 1

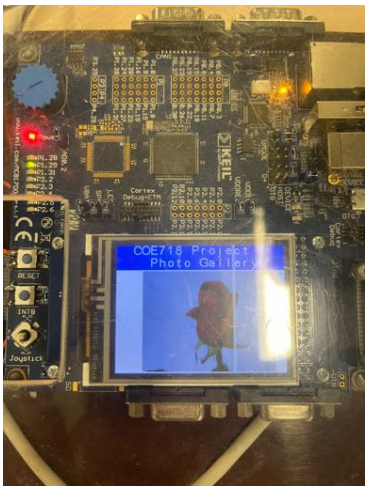

Figure 6: Photogallery picture 2

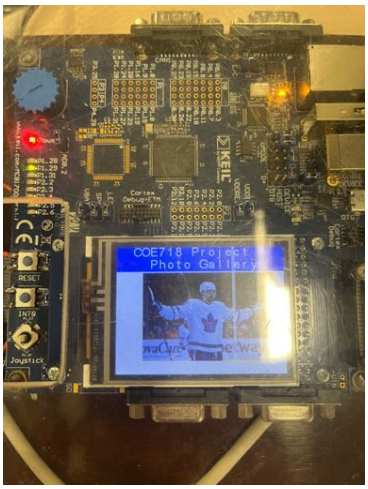

Figure 7: Photogallery picture 2

Once a user selects the photo gallery option they will enter the photo gallery state. The first image displayed is seen in **Figure 5,** the user can then move the joystick left or right to cycle through the other images seen in **Figure 6 and 7.** LED 1 is also turned on to indicate that the program is now in the photo gallery state, while LED 0 is turned off because we have exited the menu state for now. Each image is rendered using the GLCD_Bitmap() function as previously described. The ability to cycle through images is again accomplished using a counter variable. Once the user moves the joystick left or right the counter will be incremented/decremented. The value of the counter variable is used to conditionally render the image, for example the default value for the counter is 1, so the image of the porsche is rendered. Once the user moves the joystick to the right, the counter is incremented and the image of the rose will now be depicted. Some of

the complications implementing this function come from the fact that the images take up alot of board memory, so the heap size was increased to support this. Along with this complication, the images must be correctly sized to fit onto the small LCD screen as well as vertically flipped before importing. All the functional requirements for this feature previously outlined were achieved along with the non-functional requirements. The cycling between images is responsive and easy to use, and the title above the images makes it clear to the user they are in the photo gallery. At any point within the photo gallery functionality the user can return to the main menu by pressing in the joystick, the photogallery() function will return 0 and code execution will resume within the menu loop. Furthermore, LED 1 will be turned off and LED 0 will be turned on as the program is now back in the menu state.

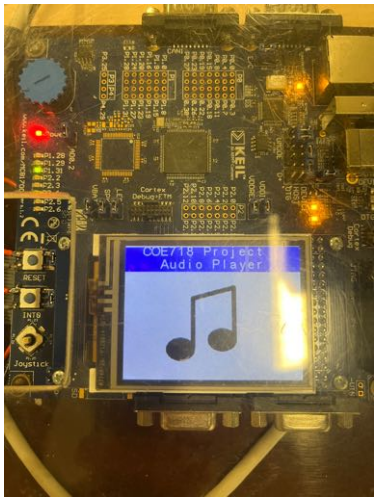

Figure 8: Music Player screen

Upon selecting the music player option from the menu the user will be brought to the screen shown above in **Figure 8.** This is the music player functionality of the media center which allows the user to play any song on the computer which will be streamed to the speakers on the MCB1700 board. The user is also able to use the potentiometer on the board to adjust the volume. Once entering this part of the media center the computer will be presented with two prompts the user must accept, and also ensure that the volume output on the computer settings is at a reasonable level. The functionality for this program was accomplished using the provided usb audio code from the sample project. Notice that the LED 2 is lit up on the board to indicate that the program is now in the audio player state. The only unique code contributed to the proper use of this feature was added to allow a user to exit the audio player functionality and return to the menu. This was accomplished by using some of the included functions to first disable the usb connection and then re-establish it once the user re-enters the audio player functionality. The non-functional requirements for simplicity and design were accomplished in this implementation by displaying a music note in the center of the screen and also including the title banner to indicate the audio player state. Again like in the photo gallery functionality, if a user pressed down the joystick, the usbaudio() function will return 0 and the

program will return to the main while loop in the menu file. From here the user is able to select another functionality of the media center, the last to be explored is the connect four game.

The final feature implementation to discuss is the playable connect four game. This was the most time consuming implementation for the entire project as it was completely open ended and there was no template to work from. Upon selecting connect four from the menu the user will be brought to the screen shown in **Figure 9** as well as LED 3 will be turned on to indicate the program is now in the game state. The connect four game is designed for two players, one player denoted by piece 'X' and the other piece 'O' each held in a pieces array. The players alternate turns placing their corresponding pieces into columns 1-7 and the first to achieve four

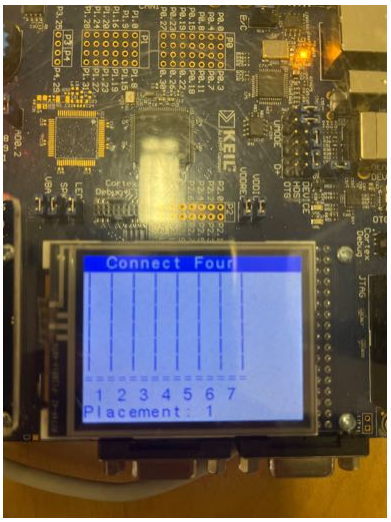

Figure 9: Connect four game screen

pieces in row either horizontally, vertically or diagonally will win. The players move the joystick up/down to increment/decrement the "placement" variable on screen. This placement variable is used to depict which column the player would like to place their piece. After reaching the desired number the player will press down on the joystick to confirm/select their placement, their respective piece will be displayed on the updated game board and then the other player may take their turn.

Figure 10: Connect four win screen

The graphics for the game are all contained within the display() method, which is called every time the game board must be updated, so on game load/replay or after a player completes their turn. Player moves are contained within an array, and this array is checked against a series of conditions to confirm if a player has achieved four in a row vertically, horizontally or diagonally. If any of these checks return

1, then the player who just completed their turn has won. After victory the LCD screen will look like **Figure 10.** The title will show which player won, either ‘X’ or ‘O’, or if a tie has occurred the screen will read a tie message instead. The user can select ‘yes’ to play again in which the board will be cleared/reset and the user will be brought back to the game screen shown in **Figure 9.** If the user selects to not play again then the startgame() function will return 0, returning the user to the menu screen. Unlike the other functionalities, clicking on the joystick is not used to exit the game, instead moving the joystick left when in the game screen will exit the connect four game and return to the menu.

Future Work

In the future, some improvements that could be made for the media center are enabling input via a USB keyboard. This would allow for easier interaction from users as the on-board joystick is very small and uncomfortable to interact with. Implementing USB keyboard functionality would also allow two-player games such as Pong to be developed where Player 1 uses the joystick while Player 2 uses the keyboard.

Further improvements such as adding more dynamic animations to the connect four game could also be implemented to provide the user with a more enjoyable experience. Such animations could include adding a “dropping” effect to the pieces once added by a player, or even a confetti animation in the background of the winner screen.

Conclusion

In conclusion, a photo gallery, media player and a playable connect four game were implemented in C and deployed on the MCB1700 board to build a media center embedded system. Users control the system through the on board joystick and can control volume from the potentiometer. An interactive menu is used to navigate between the three functionalities available, and LED lights on the board turn on to indicate the current state of the media center.

Some of the functions used to accomplish the media center design are provided through given template files such as GLCD.c, KBD.c and usbaudio.c files. Some of the specific functions used from these templates are the GLCD_DisplayString(), GLCD_Clear(), get_value(), USB_Connect() and USB_Reset(). The joystick plays an integral role in the operation of the media center as it is used for user input and also is used to conditionally render different graphics for the LCD screen.

Looking forward, some improvements to the project that can be made are adding USB keyboard support and further developing LCD animations/graphics. Implementation of a USB keyboard offers more possibilities for multiplayer games. While further developing animations improves the user experience.

Appendix

menu.c

```
#include "LPC17xx.h"                    /* LPC17xx definitions */
#include "type.h"
#include <stdio.h>
#include "GLCD.h"
#include "LED.h"
#include "KBD.h"
#include "game.h"
#include "mediaplayer.h"
#include "photogallery.h"
#define __FI        1

#define __USE_LCD   1

void switchGame(int);



int main (void) {
                int current = 1;
  LED_Init();
  KBD_Init();
  GLCD_Init();
                #ifdef __USE_LCD
                        GLCD_Init();                            /* Initialize graphical LCD (if enabled */

                        GLCD_Clear(White);                      /* Clear graphical LCD display   */
                        GLCD_SetBackColor(Blue);
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(0, 0, __FI, "  COE718 Project       ");
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(1, 0, __FI, "       Menu            ");
                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);
                #endif
  while (1) {
                        LED_On(0);
                        GLCD_SetBackColor(Blue);
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(0, 0, __FI, "  COE718 Project       ");
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(1, 0, __FI, "       Menu            ");
                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);
                        if(get_button()==KBD_DOWN){
                                GLCD_SetBackColor(White);
                                GLCD_SetTextColor(White);
                                GLCD_DisplayString(3,0,__FI,"                       ");
```

```
		GLCD_DisplayString(4,0,__FI,"                    ");
		GLCD_DisplayString(5,0,__FI,"                    ");
		if(current<3){
			current++;
		}
		if(current>3){
			current=1;
		}

	}
	if(get_button()==KBD_UP){
		GLCD_SetBackColor(White);
		GLCD_SetTextColor(White);
		GLCD_DisplayString(3,0,__FI,"                    ");
		GLCD_DisplayString(4,0,__FI,"                    ");
		GLCD_DisplayString(5,0,__FI,"                    ");
		if(current>1){
			current--;
		}
		if(current<1){
			current=3;
		}

	}
	if(get_button()==KBD_SELECT){
		switchGame(current);
	}



	if(current==1){
		GLCD_SetBackColor(Blue);
		GLCD_SetTextColor(White);
		GLCD_DisplayString(3,0,__FI," Photo Gallery          ");

		GLCD_SetBackColor(White);
		GLCD_SetTextColor(Blue);
		GLCD_DisplayString(4,0,__FI," Music Player           ");
		GLCD_DisplayString(5,0,__FI," Connect Four             ");


	}
	if(current==2){

		GLCD_SetBackColor(White);
		GLCD_SetTextColor(Blue);
		GLCD_DisplayString(3,0,__FI," Photo Gallery          ");

		GLCD_SetBackColor(Blue);
		GLCD_SetTextColor(White);
		GLCD_DisplayString(4,0,__FI," Music Player           ");
```

```
                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);
                        GLCD_DisplayString(5,0,__FI," Connect Four          ");


                }
                if(current==3){

                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);
                        GLCD_DisplayString(3,0,__FI," Photo Gallery         ");
                        GLCD_DisplayString(4,0,__FI," Music Player          ");
                        GLCD_SetBackColor(Blue);
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(5,0,__FI," Connect Four          ");


                }


        }

}



void switchGame(int option){
        LED_Off(0);
        GLCD_Clear(White);
        if(option==1){
                LED_On(1);
                photogallery();
                LED_Off(1);
        }
        if(option==2){
                LED_On(2);
                audio();
                GLCD_Clear(White);
                LED_Off(2);
        }
        if(option==3){
                LED_On(3);
                startgame();
                LED_Off(3);
        }


}
```

Photogallery.c

```
#include "LPC17xx.h"                    /* LPC17xx definitions */
#include "type.h"
#include "string.h"
#include "GLCD.h"
#include "LED.h"
#include "porsche.c"
#include "matthews.c"
#include "rose.c"
#include "KBD.h"



#define __FI        1


/*----------------------------------------------------------------------------
  Main Program
 *----------------------------------------------------------------------------*/
int photogallery () {
        /* Main Program                    */
        int pic=1;
        while(1){

                if(pic==1){
                        GLCD_SetBackColor(Blue);
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(0, 0, __FI, "  COE718 Project       ");
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(1, 0, __FI, "    Photo Gallery          ");
                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);

                        GLCD_Bitmap(50,50,250,125,PORSCHE_PIXEL_DATA);
                }
                if(pic==2){
                        GLCD_SetBackColor(Blue);
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(0, 0, __FI, "  COE718 Project       ");
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(1, 0, __FI, "    Photo Gallery          ");
                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);
                        GLCD_Bitmap(50,50,250,166,MATTHEWS_PIXEL_DATA);
```

```
		}
		if(pic==3){
			GLCD_SetBackColor(Blue);
			GLCD_SetTextColor(White);
			GLCD_DisplayString(0, 0, __FI, "  COE718 Project       ");
			GLCD_SetTextColor(White);
			GLCD_DisplayString(1, 0, __FI, "    Photo Gallery          ");
			GLCD_SetBackColor(White);
			GLCD_SetTextColor(Blue);
			GLCD_Bitmap(50,50,250,188,ROSE_PIXEL_DATA);
		}

		if(get_button()==KBD_RIGHT){
			if(pic<=3){
				GLCD_Clear(White);
				pic++;
			}
			if(pic>3){
				GLCD_Clear(White);
				pic=1;
			}
		}
		if(get_button()==KBD_LEFT){
			if(pic>0){
				GLCD_Clear(White);
				pic--;
			}
			if(pic<=0){
				GLCD_Clear(White);
				pic=3;
			}
		}

		if(get_button()==KBD_SELECT){
			GLCD_Clear(White);
			return 0;
		}

	}

	return 0;


}
```

Game.c

```
#include <stdio.h>
#include "LPC17xx.H"
#include <string.h>
#include <stdlib.h>
#include "KBD.h"
#include "GLCD.h"

#define ROWS                                    6
#define COLS                                    7
#define __FI       1


int checkHorizontal(char *board);
int checkFour(char *board, int, int, int, int);
int checkVertical(char *board);
int checkDiag(char *board);
int takeTurn(char *board, int player, const char*);
int checkWin(char *board);
void display(char *board);
void delay();


void delay(int count){
          count<<=20;
          while(count--);

}


int startgame(){
   const char *PIECES = "XO";
   char board[ROWS * COLS];
            int turn, done = 0;
   memset(board, ' ', ROWS * COLS);
   GLCD_Clear(White);
   for(turn = 0; turn < ROWS * COLS && !done; turn++){
      display(board);
      while(!takeTurn(board, turn % 2, PIECES)){
         display(board);
      }
      done = checkWin(board);
   }
   display(board);
```

```
if(turn == ROWS * COLS && !done){
                        int select=1;
                        GLCD_Clear(White);                      /* Clear graphical LCD display   */
                        GLCD_SetBackColor(Blue);
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(0, 0, __FI, "  COE718 Project       ");
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(1, 0, __FI, "  Connect Four        ");
                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);
                        GLCD_DisplayString(2, 0, __FI, "       Tie Game!           ");
                        GLCD_DisplayString(4, 0, __FI, "  Play Again?         ");
                        GLCD_SetBackColor(Blue);
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(5, 0, __FI, "  Yes                 ");
                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);
                        GLCD_DisplayString(5, 0, __FI, "  No                  ");
                        while(1){
                                if(get_button()==KBD_DOWN){
                                        GLCD_SetBackColor(White);
                                        GLCD_DisplayString(5, 0, __FI, "                    ");
                                        GLCD_DisplayString(6, 0, __FI, "                    ");
                                        select=2;
                                }
                                if(get_button()==KBD_UP){
                                        GLCD_SetBackColor(White);
                                        GLCD_DisplayString(5, 0, __FI, "                    ");
                                        GLCD_DisplayString(6, 0, __FI, "                    ");
                                        select=1;
                                }
                                if(get_button()==KBD_SELECT){
                                        if(select==1){
                                                startgame();
                                        }
                                        if(select==2){
                                                GLCD_Clear(White);
                                                return 0;
                                        }

                        }
                                if(select==1){
                                        GLCD_SetBackColor(Blue);
                                        GLCD_SetTextColor(White);
                                        GLCD_DisplayString(5, 0, __FI, "  Yes                 ");
                                        GLCD_SetBackColor(White);
                                        GLCD_SetTextColor(Blue);
                                        GLCD_DisplayString(5, 0, __FI, "  No                  ");
                                }
                                if(select==2){
                                        GLCD_SetBackColor(White);
```

```
                                        GLCD_SetTextColor(Blue);
                                        GLCD_DisplayString(5, 0, __FI, "  Yes                ");
                                        GLCD_SetBackColor(Blue);
                                        GLCD_SetTextColor(White);
                                        GLCD_DisplayString(5, 0, __FI, "  No                 ");
                                }


}
        }else {
                        int select=1;
                        char winner [20];
                        GLCD_Clear(White);
                        turn--;
                        sprintf(winner,"   Player %c wins!        ",PIECES[turn%2]);
                        GLCD_DisplayString(2, 0, __FI, (unsigned char*)winner);
                        GLCD_DisplayString(4, 0, __FI, "  Play Again?          ");
                        GLCD_SetBackColor(Blue);
                        GLCD_SetTextColor(White);
                        GLCD_DisplayString(5, 0, __FI, "  Yes                ");
                        GLCD_SetBackColor(White);
                        GLCD_SetTextColor(Blue);
                        GLCD_DisplayString(6, 0, __FI, "  No                 ");
                        while(1){
                                if(get_button()==KBD_DOWN){
                                        GLCD_SetBackColor(White);
                                        GLCD_DisplayString(5, 0, __FI, "                     ");
                                        GLCD_DisplayString(6, 0, __FI, "                     ");
                                        select=2;
                                }
                                if(get_button()==KBD_UP){
                                        GLCD_SetBackColor(White);
                                        GLCD_DisplayString(5, 0, __FI, "                     ");
                                        GLCD_DisplayString(6, 0, __FI, "                     ");
                                        select=1;
                                }
                                if(get_button()==KBD_SELECT){
                                        if(select==1){
                                                startgame();
                                        }
                                        if(select==2){
                                                GLCD_Clear(White);
                                                return 0;
                                }

                        }
                                if(select==1){
                                        GLCD_SetBackColor(Blue);
                                        GLCD_SetTextColor(White);
                                        GLCD_DisplayString(5, 0, __FI, "  Yes                ");
                                        GLCD_SetBackColor(White);
```

```
                                        GLCD_SetTextColor(Blue);
                                        GLCD_DisplayString(6, 0, __FI, "  No                ");
                                }
                                if(select==2){
                                        GLCD_SetBackColor(White);
                                        GLCD_SetTextColor(Blue);
                                        GLCD_DisplayString(5, 0, __FI, "  Yes               ");
                                        GLCD_SetBackColor(Blue);
                                        GLCD_SetTextColor(White);
                                        GLCD_DisplayString(6, 0, __FI, "  No                ");
                                }


    }
    }

    return 0;

}
void display(char *board){
    int row, col;
                GLCD_Clear(White);                      /* Clear graphical LCD display   */
                GLCD_SetBackColor(Blue);
                GLCD_SetTextColor(White);
                GLCD_DisplayString(0, 0, __FI, "  COE718 Project       ");
                GLCD_SetTextColor(White);
                GLCD_DisplayString(1, 0, __FI, "  Connect Four        ");
                GLCD_SetBackColor(White);
                GLCD_SetTextColor(Blue);
                GLCD_DisplayString(8,0,__FI," 1 2 3 4 5 6 7           ");
                for(row = 0; row < ROWS; row++){
                        int count=1;
                        for(col = 0; col < COLS; col++){
                                GLCD_DisplayChar(row+2,col+count,__FI,board[COLS *
row+col]);
                                count++;
                        }
    }

}
int takeTurn(char *board, int player, const char *PIECES){
    int row, col = 1;
                char place[20];
                sprintf(place,"Placement: %d",col);
                GLCD_DisplayString(9, 0, __FI, (unsigned char*)place);
    while(1){
                        delay(5);
                        if(get_button()==KBD_DOWN){
                                if(col>=1){
                                        col--;
                                }
```

```
                        if(col<1){
                                col=7;
                        }
                        GLCD_DisplayString(9, 0, __FI,"                    ");

                        sprintf(place,"Placement: %d",col);
                        GLCD_DisplayString(9, 0, __FI, (unsigned char*)place);
                }
                if(get_button()==KBD_UP){
                        if(col<=7){
                                col++;
                        }
                        if(col>7){
                                col=1;
                        }
                        GLCD_DisplayString(9, 0, __FI,"                    ");

                        sprintf(place,"Placement: %d",col);
                        GLCD_DisplayString(9, 0, __FI, (unsigned char*)place);
                }
                if(get_button()==KBD_SELECT){
                        break;
                }
    }
    col--;
    for(row = ROWS - 1; row >= 0; row--){
        if(board[COLS * row + col] == ' '){
            board[COLS * row + col] = PIECES[player];
            return 1;
        }
    }
    return 0;

}
int checkWin(char *board){
    return (checkHorizontal(board) || checkVertical(board) || checkDiag(board));

}
int checkFour(char *board, int a, int b, int c, int d){
    return (board[a] == board[b] && board[b] == board[c] && board[c] == board[d] && board[a] != '
');

}
int checkHorizontal(char *board){
    int row, col, idx;
    const int WIDTH = 1;

    for(row = 0; row < ROWS; row++){
        for(col = 0; col < COLS - 3; col++){
            idx = COLS * row + col;
            if(checkFour(board, idx, idx + WIDTH, idx + WIDTH * 2, idx + WIDTH * 3)){
```

```
            return 1;
        }
    }
  }
  return 0;

}
int checkVertical(char *board){
  int row, col, idx;
  const int HEIGHT = 7;

  for(row = 0; row < ROWS - 3; row++){
    for(col = 0; col < COLS; col++){
      idx = COLS * row + col;
      if(checkFour(board, idx, idx + HEIGHT, idx + HEIGHT * 2, idx + HEIGHT * 3)){
        return 1;
      }
    }
  }
  return 0;

}
int checkDiag(char *board){
  int row, col, idx, count = 0;
  const int DIAG_RGT = 6, DIAG_LFT = 8;

  for(row = 0; row < ROWS - 3; row++){
    for(col = 0; col < COLS; col++){
      idx = COLS * row + col;
      if(count <= 3 && checkFour(board, idx, idx + DIAG_LFT, idx + DIAG_LFT * 2, idx +
DIAG_LFT * 3) || count >= 3 && checkFour(board, idx, idx + DIAG_RGT, idx + DIAG_RGT * 2,
idx + DIAG_RGT * 3)){
        return 1;
      }
      count++;
    }
    count = 0;
  }
  return 0;

}
```

Usbdmain.c

```
**ONLY SHOWING THE CODE I CONTRIBUTED TO THE GIVEN FILE**
        if(get_button()==KBD_SELECT){
                NVIC_DisableIRQ(TIMER0_IRQn);
                NVIC_DisableIRQ(USB_IRQn);
                USB_Connect(FALSE);
                USB_Reset();
                USB_Connect(TRUE);

        }



int audio (void)
{

  volatile uint32_t pclkdiv, pclk;
        GLCD_SetBackColor(Blue);
        GLCD_SetTextColor(White);
        GLCD_DisplayString(0, 0, __FI, "  COE718 Project       ");
        GLCD_SetTextColor(White);
        GLCD_DisplayString(1, 0, __FI, "    Audio Player          ");
        GLCD_SetBackColor(White);
        GLCD_SetTextColor(Blue);
        GLCD_Bitmap(50,50,MUSIC_WIDTH,MUSIC_HEIGHT,MUSIC_PIXEL_DATA);
}
```