COE718: Embedded System Design

Final Project: Media Center

1. Objectives

The purpose of this final project is to create a media center using the MCB1700 board, uVision and all the programming concepts you have learned throughout the semester. The media center's features include a photo gallery capable of displaying various bmp files, an mp3 player that streams audio/mp3 tracks from the PC, and a game center with one or more different animated games the user can play. More details are provided below.

2. Media Center Specifications

2.1. Overall Requirements and Functionality

Upon launching the media center, a graphical interface should appear consisting of a menu of options based on the functionalities/tasks of the media center. The graphical interface should be implemented on the LCD, with its functionality including:

- A Photo Gallery
- An mp3 Player
- Game(s)
- Any additional functionality you (as the programmer and designer) wish to implement

The menu should be navigated with the MCB1700's joystick. You may also wish to add LEDs during navigation and/or execution and embed a keyboard via USB for game play.

The user should be able to select any of the menu options using the joystick. Once selected, the hardware must execute the task, being interactive with the user where necessary (i.e., user uses the joystick to browse through the photo gallery etc.) and return to the main menu once the task or user is finished with the selection.

The next subsections describes these functionalities in detail.

2.2. Photo Gallery Specifications

The photo gallery should be able to display various bitmap pictures, one at a time, on the LCD. It is your choice how to present these various pictures, i.e., through another menu of options listing the .bmp file names, a simple carousel view of the pictures which can be manoeuvred by the joystick, etc. Be creative.

It is worth noting however that a .bmp cannot be directly displayed on the LCD. Rather, the .bmp file must first be converted to a C file, which is then read in to the program as an array of unsigned chars. You may find the program GIMP useful for this type of file conversion. GIMP is capable of exporting raw image data necessary to create a C file from the original .bmp. To make the conversion, open the image with GIMP and 'export' the image as a C-source file. Enter a prefixed name (or the default one given). Ensure that you check the options "Use macros instead of struct" and "Save as RGB565(16-bit)". Press "Export". The C-file should now be created and present in the folder you selected. When opened, there will be header information in the file, describing your image. Ensure that all "guint8" variable types are changed manually

to "unsigned char" type as Keil will not be able to understand this type. This header information will also be of use when integrating the pictures into your main code. Note that you may also need to horizontally flip your desired image prior to the conversion.

To get you started, you may wish to refer to the example project provided to you in the course directory under "U:\\coe718\project\examples\LCD_Blinky". Understand what the code is trying to implement, and thereafter try executing the .axf file on the board. When integrating your C-file picture data into the main file, note that the file generated by GIMP is of type static unlike the example provided to you in LCD Blinky. Therefore, you can not refer to this file as "extern" as done in the sample code, but you must use #include "filename.c" the picture into your main code.

2.3. mp3 Player Specifications

The media center's mp3 player is responsible for streaming audio via USB from the PC to the dev board. Therefore, the USB should be able to connect to the PC when selected and disconnect when signalled to exit from the player (using the joystick etc). The potentiometer on the board should be able to adjust the volume of the on-board speaker, located underneath the LCD screen. Counter clockwise rotation of the pot it should increases the volume, whereas clockwise rotations to the pot should decrease the volume. When enabling the mp3 player, ensure that you play an mp3 or YouTube video on the PC to be streamed to the dev board to hear the audio playing on the dev board. It is also recommended to display a splash screen on the LCD when the mp3 player is selected and playing the audio. You MUST disconnect the audio player when exiting the menu option.

To get you started with playing audio from the PC to the dev board, you may wish to refer to the example provided to you in the course directory "U:\\coe718\project\examples\USBAudio". In this folder you will notice many support files used for USB implementation. Understand the generalities of each of the files and what the main file accomplishes in correlation to the support files. Also evaluate the importance of the interrupts in USB execution. The pdf file lpc17xx.keil.usb.audio provided to you in the main coe718\project directory may also be of use.

2.4 Game Selection

Implementation of the game section is solely up to the programmer. It will require the integration and interaction of the MCB1700's LCD and joystick. You may implement any game desired including Tetris, snake, pong/paddle, Pacman etc. DO NOT implement simple word games. Be creative. A one (noteworthy) game minimum is expected, with two or more games for bonus marks.

An example of the media center's menu selection screen and its options is provided in Figure 1.

Fig. 1: Sample Media Center Menu Screen

3. Hints for Getting Started

It is recommended to use the files provided to you in the project directory U:\\coe718\project\common. Copy and paste all the items in this folder into your project directory. Use the system_LPC17xx.c file provided to you in /common in your 'Startup Code' section as you have for all labs. Add the \common\inc folder to your include paths, i.e., Project >> Options for Target 'LPC1700' >> C/C++ >> Include Paths.

Now browse through the files in \common\inc and \src. Notice that the LPC17xx.h and system_LPC17xx.c files are included and re-defined in both \common and \common\inc. Once again, double check that all of these files/folders have been copied into your project folder as they contain various parts of essential code pertaining to the project.

When evaluating and executing the examples provided to you in U:\\coe718\project\examples, also be sure to make note of their "Options for Target 'LPC1700' settings, including any scatter files and flash initialization files necessary for target execution.

4. Project Deliverables

This project is due in week-12/13 at the beginning of your lab session. You are expected to deliver the following:

- You are to write a report on the media center's implementation in IEEE format. The final report should be 10-15 pages, **not** including the code and appendices. The following specifications should also be followed:
    - Avoid cutting and pasting of Figures. Only use pictures and diagrams of your own, always labelled accordingly and of a reasonable size. Remember that relevant diagrams are an effective way to describe your methodology and design.
    - Ensure that you reference as necessary using the IEEE format
    - Use a suitable font - preferably Times New Roman, size 11 or 12 points with single line spacing. Single or double column is optional.
    - The pages must be letter size, with 1.0"' top, bottom, left and right margins
    - The report must include the following sections:
        - Abstract
        - Introduction
        - Past Work/Review - *include relevant information of previous and relevant lab work on the subject you are implementing*
        - Methodology - *describe the method you used to design your project, from a top-level and general perspective*
        - Design - *give details based on your methodology for the various components, modules, functions etc you implemented in your project*
        - Experimental Results - *describe your results in words, diagrams etc as necessary*
        - Conclusion
        - References
        - Appendix - *including all your c files, any files you have adjusted given to you in any of the Keil examples etc. and anything you deem relevant to the project*

- Ensure that you include the Toronto Metropolitan (Ryerson) University title page, dated and signed, with your report attached.

- You (and your partner if applicable - Bonus project) are to present a working demo of the media center, displaying all functionalities as specified. You may also be quizzed during the demo to assess your knowledge of the topics covered and your implementation methods.

Appendices

A1. Importing Image onto the Keil MCB1700 Board

Actions for Importing Image onto the Keil MCB1700 Board

1. Download GIMP (GNU Image Manipulation Program) for your platform (either MacOS, Linux, or Windows) from: *https://www.gimp.org/.* Don't use the GIMP program installed in the lab computer as it does not have that option to export images into c array required by the board.
2. Load your image into GIMP (Preferably with white background or .jpg) using the option File > Open. Please note that any transparent pixels will be rendered as black when the image is loaded on the board. So if you have a .png image with transparent sides, it will render those sides pixels as black. Tip: If your game is on white background, make sure you have a white layer at the background, and if your game background is supposed to be black, you can leave transparent pixels as they are.
3. The board draws the image upside down. So you will need to go to Image > Transform > Flip Vertically in order to flip it beforehand so that it draws it perfectly on th board.
4. The on-board screen size is small, so you might also need to scale some of the images in order to display on the board. That can be done in Image > Scale Image. Also, keep in mind the on-board memory space and do not upload huge images to the board to render.
5. Export the image as c source file by going to File > Export As.. and choosing the format ***"C Source code (*.c)"***. Name the image with the extension ***".c"*** instead of ***".png"*** or ***".jpg"***. Click ***"Export"***. This will pop-up a dialog. Set a useful ***"Prefixed name"*** without spaces and weird characters. This is the name given to the image data array. Make sure everything is **NOT** selected except for ***"Save macros instead of struct"*** and ***"Save as RGB565 (16-bit)"*** and then click ***"Export"***.
6. Copy the .c file you just created over to the Kiel project working directory. Create a new group in your project and name it "Resources". Right click that group and select ***"Add Existing file..."*** . Select the .c file and click **"Add".** Close the dialog.
7. In your code where you want to use/draw the image, include that .c file. *#include "yourCFile.c".*
8. Use the function following to draw the image.
9. ```
   GLCD_Bitmap (unsigned int x, unsigned int y, unsigned int w, unsigned int h, unsigned char *bitmap)
   ```
10. Here x is the horizontal location of the image, y is the vertical location of the image, w is the width of the image in pixels, h is the height of the image in pixels, and bitmap is the pointer to the _DATA array in your included ".c" file.

For new version of GIMP, follow all the steps till 6 given above. Change the ".c" extension to ".h" and include *#include "yourCFile.h".* in your main file. Use the macro given by the GIMP in your code. You don't need to use "extern" in this method. In the latest version of GIMP, you will export image into struct which is casted to an array; therefore, you might not see in the same format as provided images in the project.

You can also use "https://notisrac.github.io/FileToCArray/" to convert image into c-array; however, GIMP is recommended.
   ```

A2. Memory space issues on Kiel Board when importing images

Actions for Memory space issues on Kiel Board when importing images

Many of you might have some trouble when importing substantial number of images onto the board. This might be because of duplicate memory allocation for the same image. When an image is defined in "image.c" file, it need not be defined again in "main.c" file. It should just be declared in "main.c" file.

It is important to understand the difference between ***defining*** a variable and ***declaring*** a variable:

- A variable is **defined** when the compiler allocates the storage for the variable.
- A variable is **declared** when the compiler is informed that a variable exists (and this is its type); it does not allocate the storage for the variable at that point.

You may declare a variable multiple times (though once is sufficient); you may only define it once within a given scope. Read about ***extern*** declaration of variable in detail in order to save memory in the board and do not define the image twice.

If you put a definition in a header file, you will end up with multiple definitions when multiple source files are involved. For example, suppose both **"main.c"** and **"other.c"** include **"foo.h"**. When you compile each of these files you will get **"main.o"** and **"other.o"**, both of which have a definition of ***int foo***. To do this properly, you declare your variable in the header file as ***extern int foo***. Then, in one (and only one) source file you define the variable with ***int foo***.