Frogger Final Project

University at Buffalo, The State University at New York

EE 379 Spring 2019

			Revision History			
Written By	Justin Bender					
Engineer	Ridwan Sadiq		**Rev**	**Date**	**Session**	**Approved/Grade**
Engineer	Justin Bender		Orig	05/14/19	Thurs. 10 AM	
Teaching Assistant	Avinash Kumar					
University at Buffalo The State University of New York						**EE 379 S19 LAB 4**

Table of Contents

List of Figures

1. Introduction

1.1 Overview

The final project of this course entailed programming the classic arcade game, Frogger, using the C programming language. This game was developed using the Keil μVision 5 Integrated Development Environment and the output was to be implemented onto our Cortex M3 LandTiger board. It was to be displayed on the board's GLCD screen and controlled using the joystick in order to move the frog. The general objective is to move the frog from the safe zone on one end of the screen, to the lily pad "homes" on the opposite end. Between the safe zone and homes, the frog must cross a road of moving trucks and cars, as well as water which can be navigated through by jumping on moving logs and turtles. The frog starts with four lives and each time the frog is hit by a truck or car, or falls in the water, a life is lost. Each time a home is reached, a new frog is spawned at the starting safe zone. To win, a player must successfully reach all five homes with different frogs.

1.2 Document Scope

The scope of this document is to provide an accurate description of how the game Frogger is programmed in C and how this code is implemented onto our LandTiger board. This design document discusses the program developed to run the game, the hardware involved in testing the system, the API documentation of the software, and the output of the program tested on the embedded system development board.

1.3 Intended Audience

The intended audience for this document is the EE 379 session TAs and the professor, as well as the lab partners on team 7 in the Thursday 10 AM session.

1.4 Revision History

Date	Revision	Modified Sections	Description of Changes
05/14/19	Original	All	Original

2. Software Design

2.1 Embedded System Overview

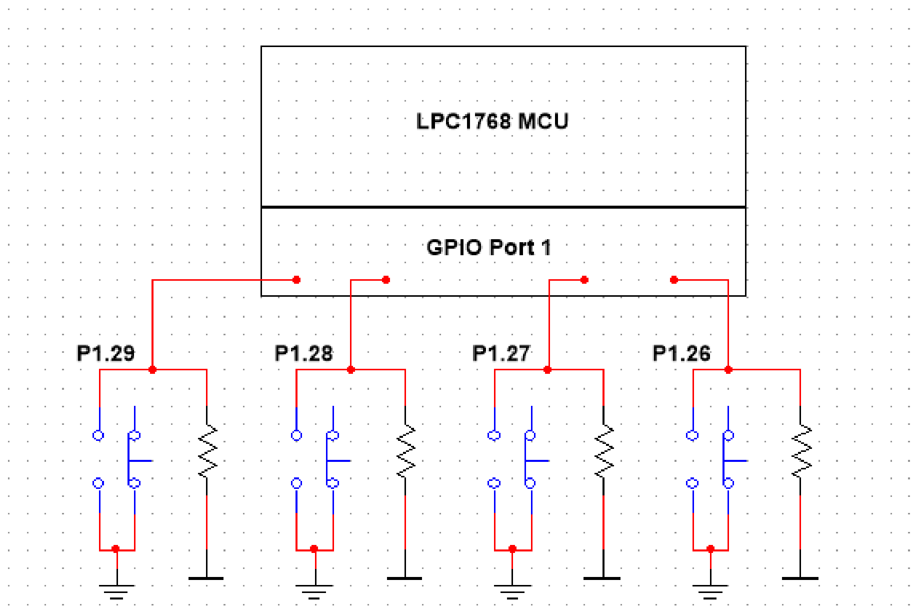

Figure 1: Embedded System Overview

2.2 Application Programming Interface

The software application used to write the code used within this project was Keil µVision 5. In this code, my partner and I had written 6 functions in total. The return type, parameters, and description of each of these functions follows in the documentation below.

2.2.1 Function main

Return type	int
Parameters	None
Description	This function is the main execution of our program. Other functions shown below are called in this for their different purposes described. We start by first initializing the LandTiger board screen using GLCD_init() and then clear this screen using GLCD_Clear(Black). After this, we then declare all variables used and set their values if needed. This includes the mask values for the joystick pins, starting number of lives, starting score, and number of homes. We also initialize the UART using UARTInit() from the UART.h header file so we are able to control our frog using a PC keyboard. We next

	create a 13x13 character array called "level" to display just about every part of our game. This is further described in Section 2.2.5 in the function "initialize". We then set the frogs starting and current locations to the desired location within our array. Calling functions drawSquare, dispLives, and dispScore we create the right side of the screen where the lives left and current score are displayed. We then create an infinite while loop to run the game. Within this while loop we start by setting up the UART then create if statements to control the frog using either the WASD keyboard keys or the LandTiger board joystick. The joystick is set up using mask values to read the pin on the board and move it to the specific direction in the array if pressed. In order for us to next set up collision with objects in the game, we used if statements to check if a frog was in the specific location of that object within the array. If it is, we decrease the frog's live by 1 and set the frog back to its base location. If the frog has no lives left, "GAME OVER" is displayed on a black screen. When the frog reaches a home, the home is deactivated and the frog is sent back to its base location using the same collision detection techniques. When there are no homes left to for the frog to go to, "CONGRATS, YOU WON" is displayed on the screen. To move all of the necessary objects on the screen such as logs, turtles, cars, and trucks, we used a for loop nested with if statements and a second for loop within them. This allowed us to shift the objects to the left or right in the array, as well as vary their speed if shifting by 2 places. Lastly, we call the function "initialize" to draw the objects to the specific array indexes, and call the function "drawFrog" to draw our frog at a specified array location.

2.2.2 Function drawSquare

Return type	void
Parameters	int cx, int cy, int width, int length, unsigned short color
Description	This function serves many different purposes within our program. In brief summary, this function creates a rectangle of pixels on the GLCD screen defined by the parameters entered. These parameters include integers cx and cy which specify the center location of the rectangle, integers width and length to specify the boundaries to which these pixels will be drawn, and unsigned short color to specify the color of the pixels which will draw this rectangle. Using GLCD_PutPixel within nested for loops in this function, we are able to place pixels in every position which they loop over, defined by the parameters. This function is used to create much of our game, including the safe zones, water, road, cars, trucks, turtles, logs, and homes.

2.2.3 Function dispLives

Return Type	void
Parameters	char x
Description	This is used to display the amount of lives the user has left. By creating an unsigned character array we are able to display a string, in this case, “Lives”. The headers GLCD.h and GLCD_UTILS.h allow us access to functions used within this such as GLCD_SetTextColor to set the color of our string of text, GLCD_DisplayString to display our string on the right side of the screen, and GLCD_DisplayChar to display our input parameter for the number of lives the user has left or starting with.

2.2.4 Function dispScore

Return Type	void
Parameters	char x
Description	Very similar to our previously described function, dispLives, this one does the same except rather than displaying lives, it displays the score of the user. Each time the user moves the frog up by one, the score increases by a point. Once the user runs out of lives, the score is reset. This is done by again using a character array in order to create the string “Score”. This is displayed to the right side of the screen along with the lives and the integer “score” in our main function is incremented by 1 each time the user moves the frog forward one place.

2.2.5 Function initialize

Return Type	void
Parameters	char level[13][13], frogLoc[]
Description	This function is very important to our program as it draws almost everything displayed on the GLCD screen. Within our main function we created a 13x13 array called “level” used as one of the input parameters. This function uses nested for loops to loop through each position in the array, and a multitude of if statements within these loops to check if a position in that array is equal to a certain integer. Each time the integer specific to that if statement is found, a drawSquare function is used to create whichever piece of the game needs to be put in that position. These game pieces include cars, trucks, logs, turtles, etc.

2.2.6 Function drawFrog

Return Type	void
Parameters	int x, int y, int width, int length, unsigned short color
Description	Similar to our function drawSquare, this function is used to draw the frog sprite used within our game. This function includes parameters to set the central position, width, length, and color of our frog. Using multiple drawSquare functions within this, we can make rectangles of different lengths, widths, positions and colors in order to create our frog sprite.

3. Testing and Verification

This section will display the result of the program we had written to create the game Frogger. All code was written in Keil µVision 5 IDE and tested on the LandTiger development board.

3.1 Final Project Testing

The functionality of this project was verified in deployment on the LandTiger LPC1768 development board. After programming the board using the Keil ulink device, the screen displayed the Frogger game we had programmed with the objects moving as specified. We also were able to successfully use the boards joystick as well as WASD keys to move our frog within the game. Through further testing the game, we were able to see the increase in score upon moving, decrease in lives upon dying, and displayed messages upon both winning and losing the game.

List of Abbreviations

- IDE: Integrated Development Environment
- GLCD: Graphical Liquid Crystal Display
- UART: Universal Asynchronous Receiver and Transmitter