# Software Design Document for Final Project: Frogger Game

University at Buffalo
EE379 Spring 2019

The information enclosed in this document and the source code generated for this final project assignment were generated by the named students on this cover page.

| | | | | |
|---|---|---|---|---|
| **Author** | Serena LaFave | **Revision History** | | |
| **Engineer** | Serena LaFave | **Rev.** | **Date** | **Session** |
| **Engineer** | Yanting Li | Orig | 5/16/19 | Thurs. 10AM |

# Table of Contents

# List of Figures

# 1. Introduction

## 1.1 Overview

This system uses the LandTiger LPC1768 development board to implement an embedded system that runs the videogame “Frogger”. The game’s main requirements and description are outlined below:

- At the bottom of the screen, there is a row that is a starting point for the frog.
- Five rows of cars and trucks appear on a road that occupies the bottom part of the display. There are two rows of truck and three rows of cars. The rows move in alternating directions.
- In a similar fashion to the cars and trucks, there are five rows of turtles and logs in a river occupying the upper part of the display.
- Between the road and the river, there is a safe zone.
- If the frog is hit by a vehicle or falls off a river object, the frog dies and is respawned at the starting point.
- The frog is controllable with the board’s joystick.
- Five houses occupy the top of the screen. Once a frog lands a house, a new frog appears at the bottom of the screen.
- The user wins the game by placing a frog into each house. The user loses by dying four times.

## 1.2 Document Scope

This document discusses the hardware platform, inputs and outputs, design of the system, API documentation of the software, and testing on the embedded system development board.

## 1.3 Intended Audience

The audience for this document is the EE379 TA for the Thursday 10 AM lab session.

## 1.4 Revision History

| **Date** | **Revision** | **Modified Sections** | **Description of Changes** |
|---|---|---|---|
| 5/16/19 | Original | All | Original |

## 2. Software Design: Embedded System Overview

Figure 1 shows the schematic of the LandTiger LPC1768 development board's digital joystick, which is used in this project to move the frog. JOY SEL – JOY UP are GPIO Port 1 pins P1.25 – P1.29. They are connected to the board's CPU.

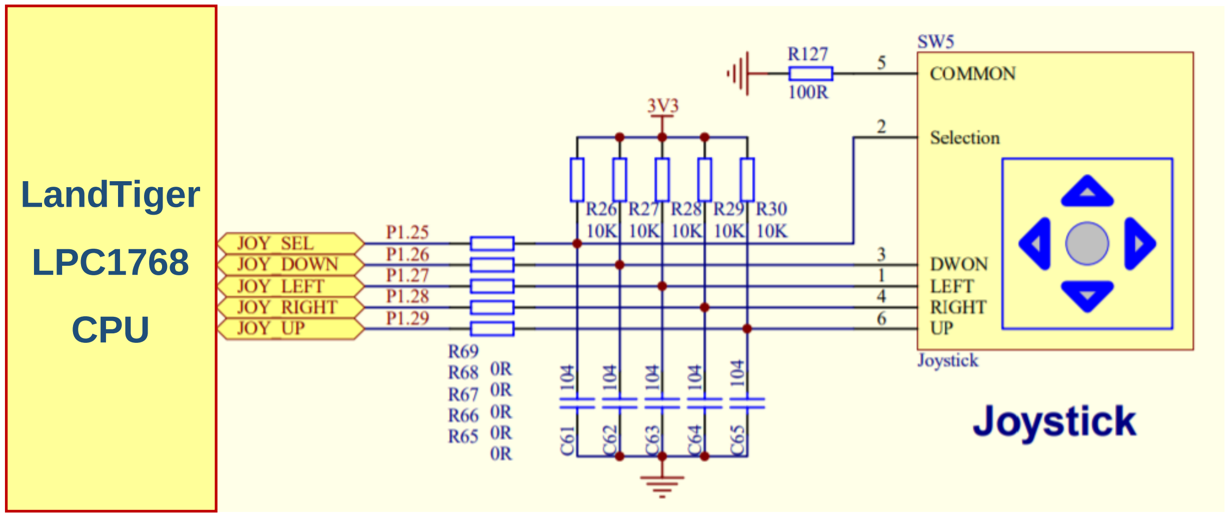

Figure 1: Joystick Schematic

# 3. Software Design: Application Programming Interface

## 3.1 int dispArr[13][13]

To simplify layout control of the game, the display is programmed as a 13x13 array of single digit integers ranging from 0 to 8. Each integer corresponds to a game object, such as a log or a piece of road. The following functions were programmed to utilize the array.

## 3.2 drawObject (examples: drawTurtle, drawLog)

**Return type:** void

**Parameters:** int row, column

**Description:**

Nine functions with the above properties are used to draw the following objects of the display: Water, turtle, log, house, safe, road, truck, car, and row. Each function uses the current row and column number of the display array to draw the object: First, a text color is set using GLCD_SetTextColor(). Next, a for loop multiplies a counter with 20 (the number of pixels constituting the height of one array element) with the current column number as the starting point. The counter is incremented until the starting point has increased by another 20 pixels. A nested for loop does the same process, utilizing the current row number to draw 18 pixels across. During each increment, a pixel is drawn using GLCD_PutPixel().

Sometimes, multiple nested for loop sets are used to change the apparent dimensions of the game object. For example, drawTurtle uses two pixels of blue, 14 pixels of green, and another two pixels of blue from top to bottom. This helps separate the turtle from the rows of logs above and below it.

## 3.3 drawBackground

**Return type:** void

**Parameters:** none

**Description:**

The entire code of drawBackground is held within two nested for loops that increment the current row and column of the display array from 0 to 12. For every element in the array, a series of if statements calls the specific drawObject function that corresponds with the number within it.

For game objects that require an action, further instructions are written within the specific if statement. For death objects, including car, truck, and water, there is another if statement after drawObject is called. This statement determines if the center coordinates of the frog are within the pixel boundaries of the current element. If it is, it sets the frog's coordinates back to the starting point and subtracts one from lives, the variable that keeps track of the user's remaining lives.

For objects log and turtle, the frog needs to stay on the object while it moves across the screen. If the frog's center coordinates are within a log element, the value of variable logs is changed from 0 to 1. The same method changes the value of the variable turtle for when the frog lands on a turtle. These variables will be utilized in function main, described later, to keep the frog on the object.

For the houses at the top of the display, an if statement again checks the frog's coordinates. If they are within its boundaries, the function "houses", which will be described later, is called.\

## 3.4 drawFrog

**Return type:** void

**Parameters:** int cxbody, cybody, wbody, unsigned short color

**Description:**

This function drew the frog. cxbody and cybody marked the center coordinates of the frog. wbody determined the size of the frog. First, the text color was set using GLCD_SetTextColor(color). Then, nested for loops incremented the current pixel from edge to edge of a wbody x wbody square. However, using an if statement, GLCD_PutPixel() was only called when the current pixel was within the boundaries of a circle with diameter wbody. The boundaries were determined using the distance formula. The parameters of drawFrog were set to draw the frog as a small pink circle ten pixels in diameter.

## 3.5 moveFrog

**Return type:** void

**Parameters:** none

**Description:**

This function allowed communication between the program and the joystick on the LandTiger board. Four if statements used mask values corresponding to the joystick's four directions to check if they were activated by the user. The embedded system overview of this communication is described in Section 2. Before moving the frog, nested if statements made sure the frog would not move if it was too close to a boundary of the game board. Then, if the up or down directions were activated, the center coordinates of the frog were incremented by 18 pixels in the appropriate direction. For left and right movements, the same method was used to move the frog by 20 pixels. These pixel values allowed the frog to move from the center of one display array element to the center of a neighboring one.

## 3.6 moveOddRows and moveEvenRows

**Return type:** void

**Parameters:** none

**Description:**

These two functions controlled the movement of the car, truck, log, and frog objects. Though the road and water appeared to be an unmoving background, these functions moved those objects along with the others. moveOddRows used nested for loops to increment row and column values. The row values started from row one and incremented by two until row 11. These rows contained the log and truck objects which were to move across the display from left to right. Within the for loop set, the current column was incremented by one while the current row remained the same. This shifted each element of the rows to the right, causing the game objects to move in that direction. An if statement reset the current column position back to zero once it reached the end of the row.

moveEvenRows used the same method, but started with row two and shifted the columns to the left.

## 3.7 houses

**Return type:** void

**Parameters:** none

**Description:**

This function allowed a frog to "stay" in a house once it was able to reach it. Five if statements corresponding the five houses checked if the center coordinates of the frog were within the pixel boundaries of each house. If the coordinates of the frog were within house 1, for example, a variable h1 would be changed from 0 to 1. Then, the coordinates of the frog would be reset to the starting position. Variables h1-h5 were utilized in main(), which is described later, to continually draw a frog in the house once the user had reached it.

## 3.8 main

**Return type:** int

**Parameters:** none

**Description:**

At the start of the main function, the joystick keys are set as outputs using LPC_GPIO -> FIODIR. The GLCD is setup initiated and cleared using GLCD_Init() and GLCD_Clear(Black). After this is the infinite while loop. drawBackground, moveFrog, moveOddRows, and moveEvenRows are called. drawFrog is also called, with initial coordinates set as the starting point (130, 230).

GLCD_DisplayChar() is used, displaying the value of variable lives at character coordinates (1,18). An if statement follows, containing instructions for when the value of lives reaches zero: GLCD_DisplayString()

is used to display the string “You Lose!”, starting at character coordinates (6,0). The game is then stopped using return(0).

Next, a series of if statements utilize the h1-h5 variables described in the function houses. If h1 equals 1, i.e. a frog was entered into house 1, drawFrog will be called at pixel coordinates (30, 12). Thus, it will be drawn continuously after a frog has entered the house once. The rest of the if statements follow the same method, calling drawFrog to the appropriate coordinates for each house.

An if statement contains instructions for when h1-h5 are all equal to one, meaning the game has been completed. GLCD_DisplayString() is used to display the string “You Win!” at character coordinates (6,0). The game is then stopped using return(0).

Two if statements utilize variables turtle and logs, described in function drawBackground. If the variable is equal to 1, meaning the frog is on the moving game object, the frog’s center x-coordinate is incremented by 20 pixels in the appropriate direction. This allows the frog to ride along with log or turtle until it reaches a game board boundary, at which point it is pushed into the water.

Lastly, outside of the if statements, variables turtle and logs are both reset to 0 so that the frog will not move on its own when it is on a non-relevant game object.

# 4. Testing and Verification

## 4.1 Objective Verification

The functionality of the Frogger game was verified by deployment on the LandTiger LPC1768 development board. After loading the program onto the board using the Keil uLink device, the game appeared on the GLCD and started. All game requirements were demonstrated by playing the game, including running into cars or trucks, falling into water, riding a log or turtle to the game board boundary, running out of lives, and entering all five houses.

# Glossary

## List of Abbreviations

- GLCD: Graphics Liquid Crystal Display
- GPIO: General Purpose Input-Output
- API: Application Programming Interface

## Hardware References

LPC1769/68/67/66/65/64/63 Datasheet:
https://www.nxp.com/docs/en/data-sheet/LPC1769_68_67_66_65_64_63.pdf

LPC1768 LandTiger Development Board Schematics:
https://os.mbed.com/media/uploads/wim/landtiger_v2.0_schematic.pdf