LIGHT EM UP: A CAUSE & EFFECT

GAME BY CIRCUIT CITEE

Authors:

Jaclyn Nord
Lauren Malhe
Irere Romeo Kwihangana

Table of Contents

1 Introduction

The Problem:

The problem our group decided to work on was presented by ADEC, which is the Association for the Disabled of Elkhart County. This organization is a group that is "working in partnership with all to help people with physical and developmental disabilities, live fuller, richer, more meaningful lives". They have several campuses and group residences throughout Elkhart and St. Joseph counties. More information on this group can be found at www.adecinc.com.

ADEC works with individuals with disabilities in a variety of ways, including conceptual understanding. They were interested in a device that would help in this area, particularly, a device that would help demonstrate the cause and effect relationship of a physical action. How to use the device would need to be easily understood by the user, and also demonstrate the relationship between a cause and effect in an obvious way. This device would be best in a game form which could be understood by all age groups. The game would need to be usable on a flat surface, or upright. It needs to be easy to use for someone with no engineering background, and durable in order to survive long term use.

The Solution:

Our team believes that this problem can be solved with a game similar to a Whack-a-Mole: a game aimed at demonstrating the cause and effect relationship. The game would come in the form of a matrix with large, easy to push buttons, which are capable of lighting up. The game itself could then be used in a variety of ways, with the priority being the Whack-a-Mole type

game. This game would work by having 9 large buttons in a 3x3 matrix that can light up individually or simultaneously. A button will be lit, and the user must press the lit button to continue with the game. When a user presses the lit button, the light inside the button turns off and another button lights up. The user will continue to play until they do not press the correct (lit) button in the allocated time. The user will be able to see how the cause (pushing a lit button), leads to the effect (one button turning off and another turning on). This game will be accessible for different individuals by offering differing levels of challenge for the game.

2 Detailed System Requirements

After consulting with ADEC, Circuit CitEE has determined a set of requirements for the game. They are broken down by stakeholders requirements and system requirements. The stakeholders are the staff members at ADEC and the players. They will be the ones setting up the system for use. The players are the people who ADEC serve. The requirements are as follows:

- The player shall be able to push a lit button
    - Requirements:
        - Light within button
        - Ability to turn light off in button
        - Physical button to be pressed
- The player or staff member shall be able to select a mode of difficulty (easy, medium, hard, progressive) before starting to play
    - Requirements:
        - method to select mode of difficulty
            - ability to scroll through menu and select difficulty

        - modes of difficulty to scroll through
            - code to differentiate between levels
            - code that will allow the user to run different levels of difficulty
                - difference in time allowed for user to press button
- The player shall be able to see their score on an LED screen
    - Requirements:
        - physical LCD screen
        - ability to track player's score
        - ability to send recorded score to LCD
- The staff member shall be able to turn on the system
    - Requirements:
        - ON/OFF switch that controls the flow of power to the game system
- The staff member shall be able to see the status of the system (on or off)
    - Requirement:
        - LED on power switch to indicate status of system
            - lit LED if ON
            - unlit LED if OFF
- The system shall be able to be placed on a flat surface
    - Requirements:
        - system that can function in a flat position
        - physical box that can be positioned on a flat surface
        - interface that is conducive to being played on a flat surface
- The system's buttons shall have different colors

    - Requirements:
        - LEDs capable of producing varying colors
            - RGB LEDs
- The system shall recognize a button press from the player
    - Requirements:
        - physical button to be pressed
        - mechanism to detect button press
        - connection to circuit board to register push
        - software code to recognize change in button state
- The system shall be able to turn on an LED when button is pressed
    - Requirements:
        - physical button to be pressed
        - mechanism to detect button press
        - connection to circuit board to register push
        - software code to recognize change in button state
        - software code dependent upon change in button state that creates command to initialize new LED
        - hardware (NMOSfet) that will allow the necessary current to reach the chosen new LED
- The system shall be able to run on a modified ATX power supply
- The cost of design and production shall not exceed $500

3 Detailed project description

3.1 System theory of operation

As a general overview, the system theory is as follows:

The software for the gaming system is written in MPLABX, and contains the information necessary to run the system. The software is broken down into several parts, starting with the initialization of the pic32 microcontroller. It contains the code necessary to run the simple whack-a-dome game, which is broken down into a series of commands that will light LEDs based on certain situational criteria.

The computer containing the MPLABX code is connected to the designed circuit board through a Pickit3. The Pickit3 takes the code from the MPLABX and converts it to information that the PIC32 microcontroller can understand. Once the code has been loaded to the PIC32, the Pickit3 can be removed from the circuit board, and the rest of the system can be run through the circuit board.

The PIC32 microcontroller is the source of control for the rest of the game system. For the simple whack-a-dome game, the system is run as follows:

When the whack-a-dome game is initialized, the PIC32 microcontroller generates a random number between 0 and 8, and another random number between 0 and 2 that controls the LED color choice. The random number between 0 and 8 determines which button on the 3x3 board will be lit first. The number is sent from the PIC32 microcontroller to the PCA9635 LED driver. The driver then uses that information to initialize the appropriate LED for the corresponding button. The PCA9635 LED driver turns on the LEDs based on the randomly

generated color choice from the PIC32 microcontroller. These RGB LEDs are located inside each of the large dome buttons, causing the buttons to light up according to the randomly generated button choice.

The LEDs stay lit in the chosen button until a) the allocated amount of time for the game has passed, or b) the user presses the button in the allocated amount of time.

If the first option, a), occurs, and the user does not press the lit button in the allocated amount of time, the end game mode is triggered, and the game ends. If the second option, b), occurs, the game continues.

The game determines if the button has been pressed in the allocated amount of time through the PIC32 microcontroller. The big dome button has a built in push button that was used to determine if the button was being pressed. The common pin on the sensor is connected to voltage (3.3 V). The normally open pin on the sensor is connected through a resistor to an input pin from PORT B on the PIC32 microcontroller. When the button is pressed, a connection is made between the voltage from the common node and the normally open node. This connection sends the PIC32 microcontroller input pin high, which tells the microcontroller that the button has been pressed.

If the input pin for the selected button on the microcontroller is not set high within the allocated time, option a) occurs, and the end game mode is triggered. If the end game mode is triggered, the final score is displayed upon the LCD screen, and the game resets itself.

If the input pin for the selected button on the microcontroller is set high within the allocated time, then option b) occurs, and the game continues. The PIC32 registers the selected button as being pressed, and the game sequence can continue. The PIC32 microcontroller sends a command to the PCA9635 to turn off the currently lit LEDs within the button. At the same time,

another set of random numbers is generated to select a new button and a new LED output color. This sequence will continue until the user fails to set the normally open input pin high by pressing the button in the allocated amount of time.

Concurrent with the game being played, the software is keeping track of the number of times the user correctly hits the selected button within the allocated amount of time. The number can then be passed from the PIC32 microcontroller to the LCD screen to allow the user to track progress.

Power is supplied through a standard wall outlet to the board through a voltage regulator. Power supply can also be controlled through a power switch located on the product to turn in on or off.

The physical appearance of the product is a rectangular plastic HDPE encasing the 3x3 matrix of buttons, the LCD display, and the menu buttons.

3.2 System Block diagram

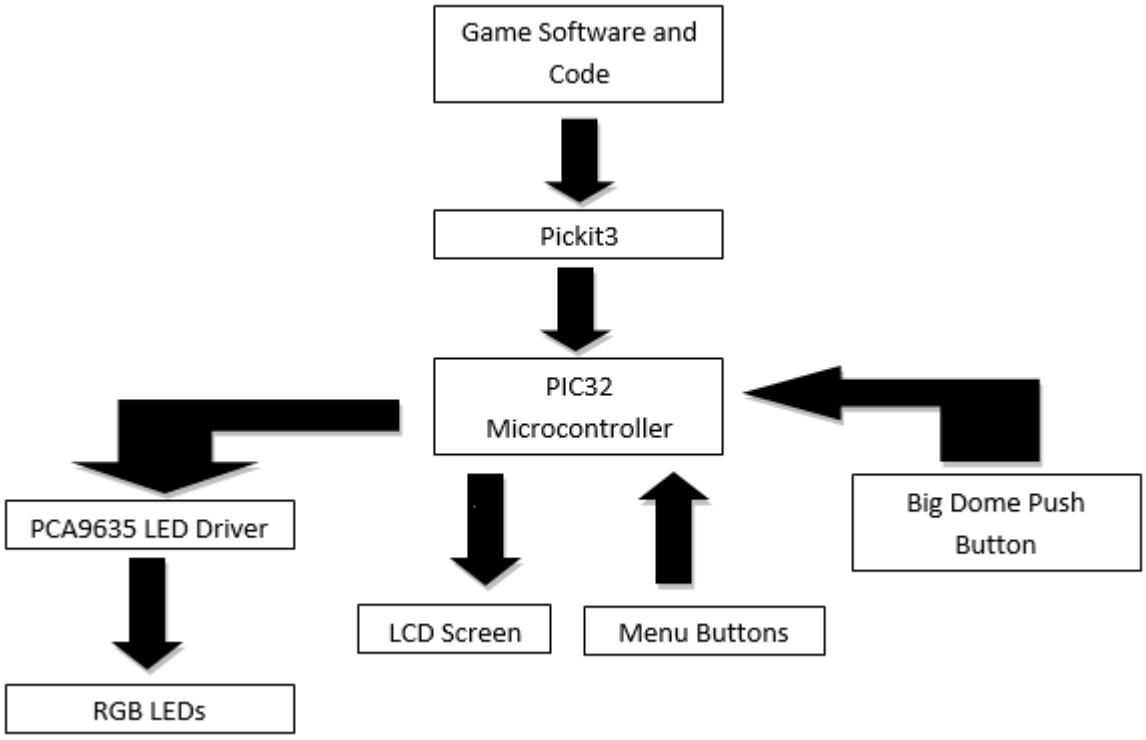


Figure 1

3.3 Detailed operation user interface subsystem

Subsystem 1 is the user interface, and everything that is included in the functionality of it. The user interface base is made of an HDPE plastic that has been drilled through to create spaces for the rest of the necessary components. The first of these components is the large dome push button.

The large dome push button was ordered from SparkFun, an online electronics part website. The button can be seen below. Figure 2 shows the topside view of the large dome push button. The clear plastic top allows light to pass through, causing the button to appear illuminated. The clear plastic was frosted in order to increase the viewing angle of the LED and reduce the brightness to increase the user's comfort with the product. This is the part of the large dome push button that is visible on the user interface. The big dome push button has a large

surface area, 100mm diameter, which makes it easy for

Figure 2

the user to push down.

The button interfaces to the game itself through a modified setup for the big dome push button. The big dome push button is delivered with an insert as shown in Figure 3. As can be seen in the figure below, the big dome push button is equipped with a built in LED that corresponds to the color of the dome button - in this case, white. Circuit CitEE decided the single LED provided did not offer enough variation in color options, and made alterations to the switch in Figure 3. The

LED is easily removed from the switch, and so was removed. This made room for a new LED to be placed within the big dome push button. The LED

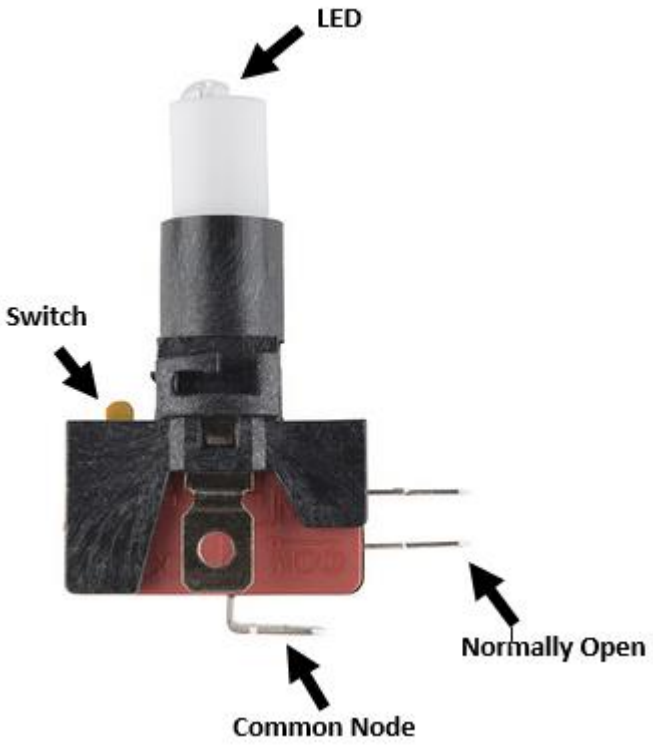


Figure 3

replacement choice is discussed below.

The internal parts of the big dome push button are necessary for the interfacing between the button press itself and the recognition of the button press by the game system through the PIC32. Labelled on Figure 3 are the common node and the normally open node. These two nodes are used to determine if a button press has occurred. The common node is connected to a voltage source of 3.3V. The normally open pin is not normally connected to the common node, so in a waiting state the normally open pin has a voltage of 0V. The connection between the common

node and the normally open node is shown below. The connection between the common node and the normally open node is an open wait connection.

Figure 4

The arrow in Figure 5 indicates a switch press. When the user presses down on the big dome push button, the switch press is moved in a downward motion. One side of the switch press pushes against the switch that is indicated in Figure 3. When this occurs, the connection between the common node and the normally open node is closed, allowing the voltage from the common node to pass to the normally open node. This high voltage is passed to the game system, telling it that the button has been pressed.

Figure 5

Circuit CitEE decided to use 9 of the big dome push buttons in the final user interface. This allowed the game board to be set up as a 3x3 matrix. Using a 3x3 matrix solved several design problems for the game system. Because of the physical limitations of many of the clients, the user interface needs to be small enough that every aspect of it is within an easy reach distance of the user.

The 3x3 matrix of the big dome push buttons also allows for future expansion in terms of game usage for the system. A 3x3 matrix allows the board to be used, for example, as a tic tac toe board.

Next to the big dome push button interface is the menu interface, which consists of two parts: the LCD screen, and the menu controller. The LCD screen chosen was the NHD-0420D3Z-FL-GBW, which is the same LCD screen used in the senior design kits. This LCD screen was chosen because it provided the necessary number of lines, a desirable color scheme and backlight, and could be powered with the voltage available for the game system. Circuit CitEE also had experience with this LCD screen, making it an attractive choice for further use. The LCD screen chosen can be seen below in Figure 6.

Figure 6

The LCD screen interfaces with the rest of the game through the PIC32 microcontroller, which is covered in the subsystem 3 overview. The LCD screen allows the user to move through a menu that he/she can see through the use of the menu keys. The LCD screen is a visual representation of what the user is choosing but it does not actually input any information to the software or hardware subsystems. The actual flow of information can be seen below in Figure 7.

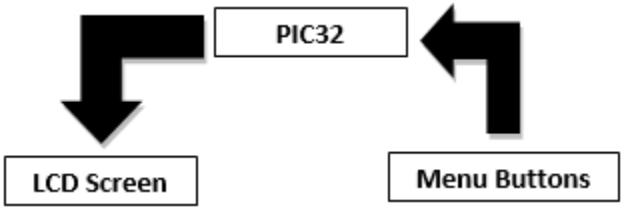


Figure 7

As the flow chart above shows, the LCD screen is just a visual representation being sent from the PIC32. The actual input information itself is coming from the menu buttons.

The menu buttons are a group of tactile buttons ordered from Sparkfun. These buttons are arranged to allow the user an UP, DOWN, LEFT, RIGHT, and ENTER option. An example of the buttons used is shown below in Figure 8.

Figure 8

A tactile button works by only allowing the circuit to close when the button has been pressed. Two pins from opposite sides are connected to the circuit in question, and when the button is pressed down, the two pins are connected to each other. By connecting the two pins, a high voltage can be sent from the menu buttons to the PIC32 microcontroller, which will take that information and relay it to the LCD screen. Information on how this occurs can be found in the section for subsystems 2 and 3.

3.4 Detailed operation software subsystem

Subsystem 2 is the software aspect of the game system. Game code is programmed using C in MPLABX. MPLABX is the program that our project used to communicate with the PIC32 microcontroller via a PICKIT3 programming device.

The outline of the game code included:

Random number generation

Within the game code, random numbers were generated in order to select which button would be lit and again generated to select which color LED would be lit: red, blue, or green. Figure 9a demonstrates the code used to randomize which button was selected to be lit. Figure 9b demonstrates the code used to randomize which color LED (red, green, or blue) was chosen to be lit. The colors of the LEDs are referred to as red_Gamex, green_Gamex, and blue_Gamex respective to the LEDs which were lit.

The random number between 0 and 8 was used to determine if the correct button was being pressed. This was determined through an input pin connected to each big dome press button (mechanism for this described in section 5.5 on subsystem hardware). This number is also

sent to the PCA9635 LED driver along with the random number generated to determine which LED is lit. This tells the PCA9635 LED driver which LED to light.

```
//Choose button to light
void choose_button(void){
    char button;
    button = rand()%8; //generates a random integer between 0 and 8
    if (button==0){
       // PORTBbits.RB0;
        Game0();
    }
    else if (button==1){
       // PORTBbits.RB1;
        Game1();
    }
    else if (button==2){
       // PORTBbits.RB2;
        Game2();
    }
    else if (button==3){
       // PORTBbits.RB3;
        Game3();
    }
```

Figure 9a

```
void Game0(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue0_Game();}
        else if (select==1){
                green0_Game();}
        else
            {red0_Game();}
            }
    else{
          Write(DEVADD, OUT0_REG, led0_on);
          Write(DEVADD, OUT0_REG, led_off);
    }
}
```

Figure 9b

The code is constantly checking the state of the PORT B input pin of the button that is being lit to see if it is high or low. If the input is high, this signals the code that the user has

pressed the button, and the next step of the game can be run. However, if the input remains low past the allotted amount of time, the code will step into the end game sequence.

Debouncing code: keeping an integer count

In order to escape the hardware issue of buttons bouncing, debouncing code needed to be included, so that a score could be incremented each time a button was successfully pressed in the allotted amount of time. This was done by creating an integer and tracking the value of the button's input pin (each button was connected to its own PORT B pin that was set to be an input for the PIC32 microcontroller). Each time the input pin went high after being low, the integer increases by 1. This integer can then be outputted to the LCD screen.

Internal clock: for time delays

The PIC32 microcontroller had a built in internal oscillator that provided an internal clock. This clock was used to generate time delays within the code in order to track the allotted time the user is given to press the button.

Initialize

The main purpose of the Initialize function was to initialize everything needed to start using I2C in order to communicate with the PCA9635 LED driver. The baud rate had to be set to run at 100 kHz. The slew rate had to be disabled, and I2C had to be turned on.

Wait

The Wait function was used to wait until the bus was idle, check for when the master interrupt flag was set, and then clear the master interrupt flag, so that information could be transferred.

Start

The Start function was used to send the “start” bit to the PCA9635 LED driver.

Stop

The Stop function was used to send the “stop” bit to the PCA9635 LED driver.

Send_byte

The send_byte function was used to send data to the I2CxTRN register and then check for an acknowledge or not acknowledge bit.

Write

The Write function provided the data that needed to be sent to I2CxTRN register of the PCA9635 register in the appropriate order. First the slave address (the address of the PCA9635 LED driver) was sent, then the register address (exs: Mode 1 register, Mode 2 register, LED registers), and finally the value to set the register to.

Display score

The display_score function initializes the LCD display, then initializes the serial port, then sets the output device to the LCD display, turns the LCD display on, clears the LED display, sets the position to write to on the LCD display, then displays the user’s score at that position.

3.5 Detailed operation of hardware subsystem

Subsystem 3 is the hardware aspect of the game system. The hardware subsystem includes all hardware aspects that are not part of the user interface hardware. In general, this hardware includes the circuit board that is used to run the game system. The circuit board for the game system includes the PIC32 microcontroller, the necessary components to power the RGB LEDs, the connections for the buttons, the connection for the LED, the voltage regulator necessary for the board, the connection for the power switch, and the LED drivers necessary to

control the RGB LEDs. The schematic shown below in Figure 10 shows the components and their connections for the circuit board.

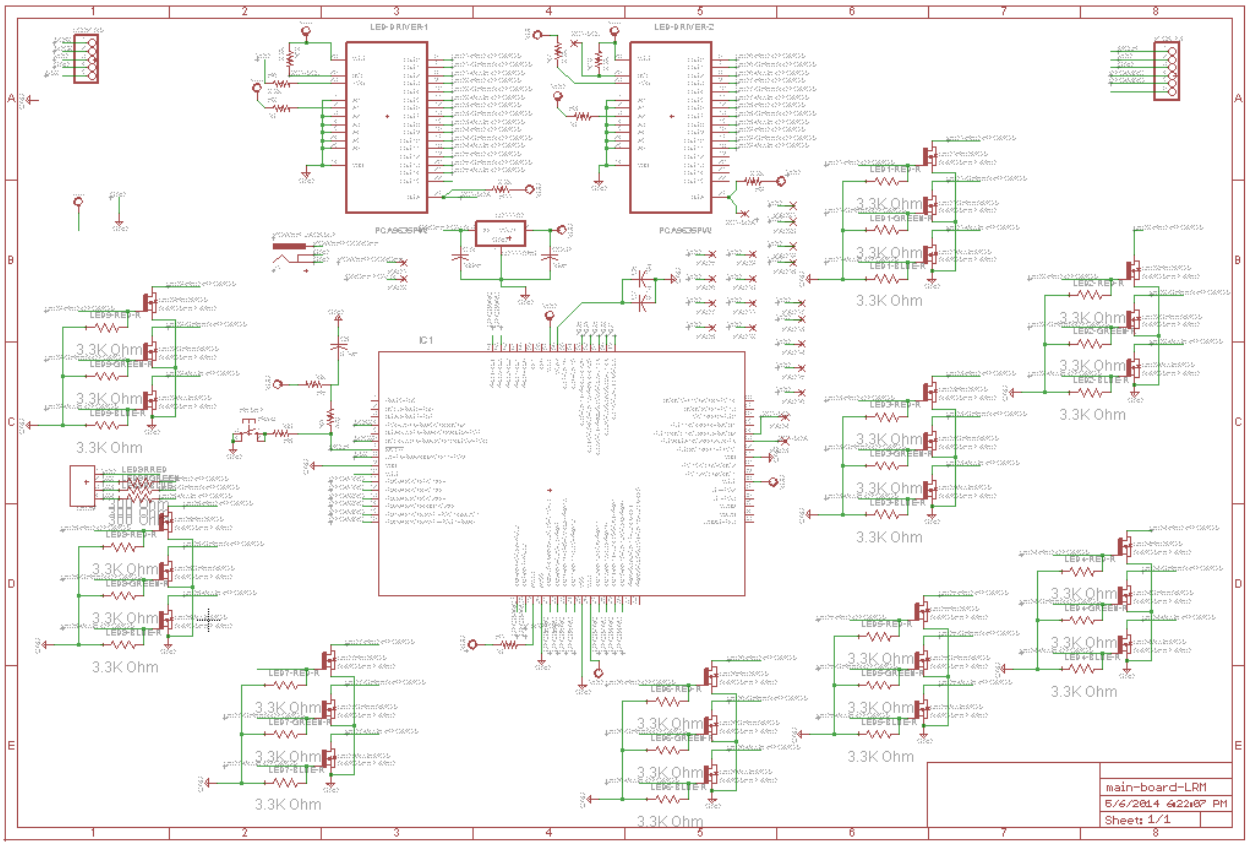


Figure 10

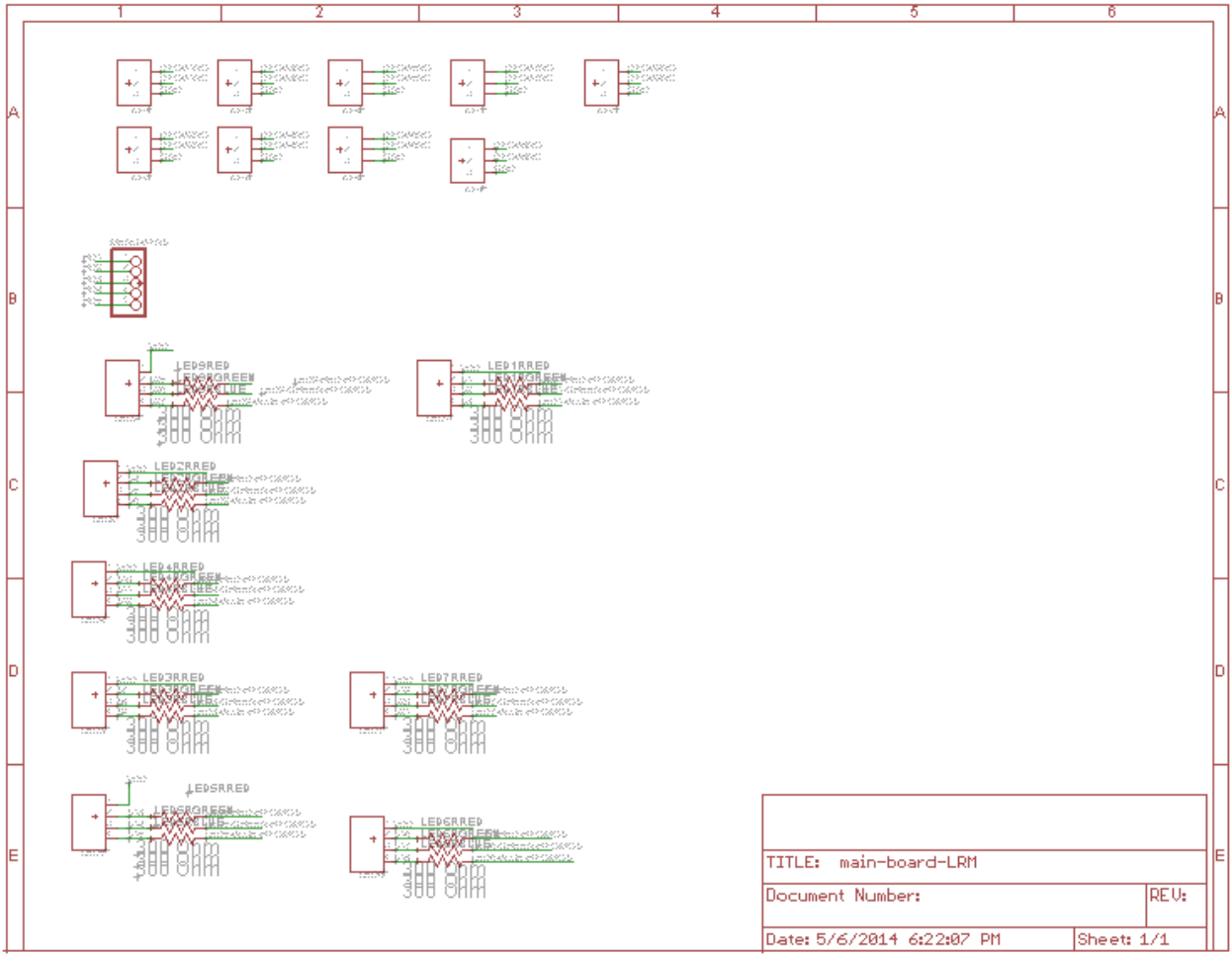


Figure 11

A PIC32 microcontroller was used as the central source of control for the game system. It received its code through a Pickit3 connected to MPLABX. Code stored in MPLABX is transferred through here and stored in the PIC32 microcontroller, which then uses the code to control the game system. The PIC32 microcontroller is shown in Figure 12, below.

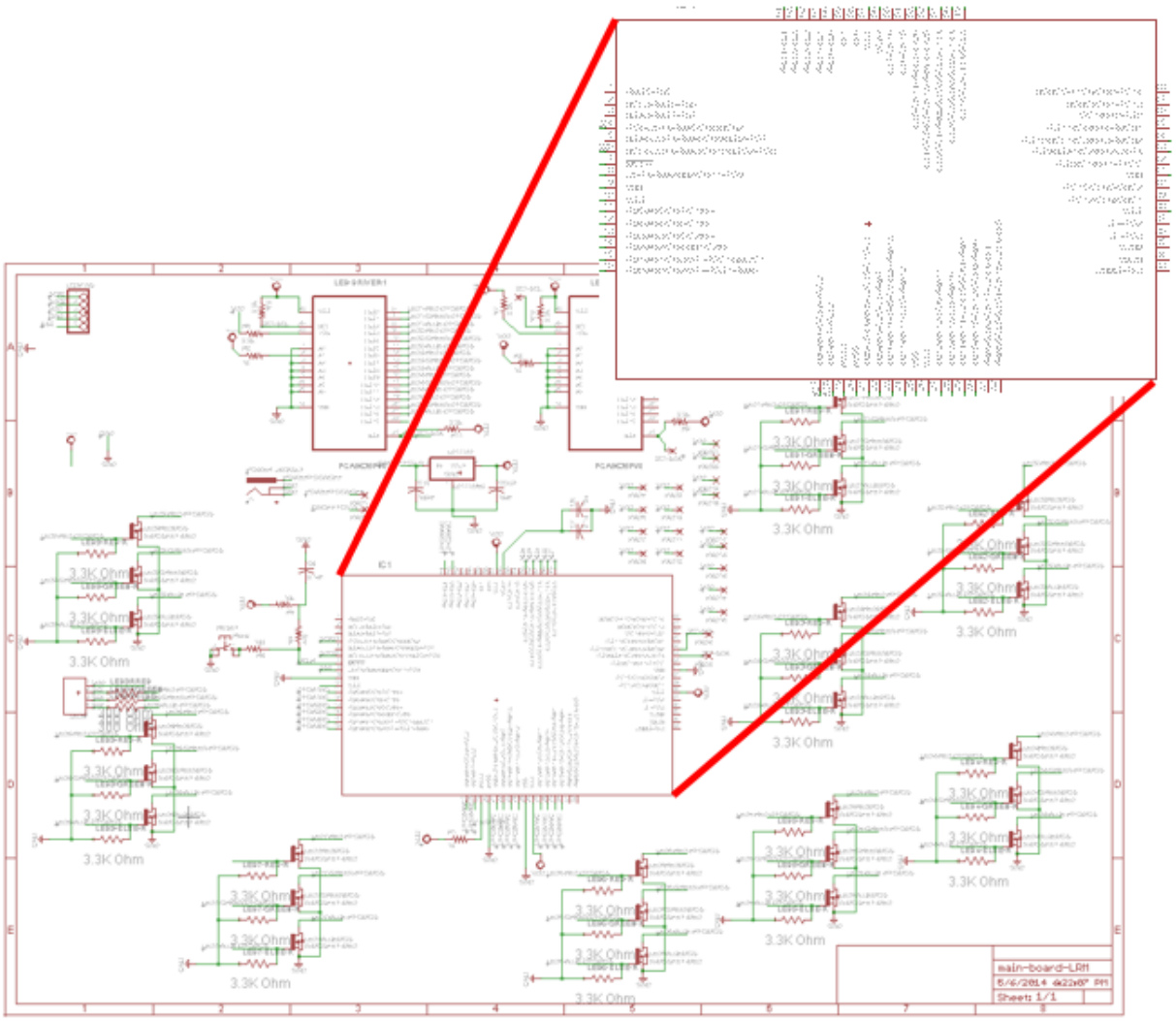


Figure 12

The PIC32 microcontroller is important in several vital functions, and can be considered the brain of the game system. The PIC32 microcontroller is used to determine if a button has been pressed, to give the LED drivers the command to run, to output to the LCD screen, and to run the logic needed for the games.

The PIC32 microcontroller can be used to determine if a button has been pressed. The physical circuit for the button press is discussed in detail in the user interface system, but will be briefly covered here. The big dome push button has a normally open node attached to it, and this node is attached to the microcontroller through an input pin in PORT B. When the big dome

push button is pressed down, the normally open node is given a high voltage, which is sent to the input pin on the microcontroller. The software needed to run the game in the microcontroller takes this information and uses it to determine the next step. More information on how this works can be found in the software subsection, 5.4.

The PIC32 microcontroller can also be used to give the command to drive the LEDs. The PIC32 microcontroller is connected to the PCA9635 LED driver through two pins. The PCA9635 LED driver will be discussed later in this section. The two pins are used to output SDA and SCL information. These connections to the PCA9635 LED driver allow the PIC32 microcontroller to control which RGB LEDs are turned on and off. The decision making process for this can be found in section 5.4, software subsystem.

The PIC32 microcontroller is also responsible for controlling the LCD screen. The PIC32 microcontroller decides what information needs to be sent (see software subsection 5.4), and then outputs this information through output pins to the LCD screen. The logic for the game itself is also run inside the PIC32 microcontroller. See section 5.4 for more details on this logic.

Finally, the PIC32 microcontroller is used to register the input from the menu tactile buttons, and relay this information to the LCD screen. The input from the menu tactile buttons is similar to the setup for the big dome press buttons. When the tactile button is pressed, the pin that is connected to an input pin on the PIC32 microcontroller is triggered high, which tells the logic in the software that the button has been pressed. Each tactile button on the menu setup is connected to a different input pin, allowing the PIC32 microcontroller to track button presses.

As mentioned above, the PIC32 microcontroller is attached to the PCA9635 LED driver that controls which RGB LEDs turn on. The PCA9635 LED driver is a 16 pin LED driver, which means that 5 RGB LEDs can be controlled by the LED driver.Two LED drivers were used for

this game system. Each PCA9635 is given a specific address. This address is set by a combination of pins being set low and high. The first device address used was C0h and the second was C2h. The PIC32 microcontroller is then able to send information to the correct PCA9635 LED driver by sending the address information first, guaranteeing the correct driver is used. Once the correct LED driver is chosen, the correct LED will be lit. This happens in the following way:

The RGB LED requires a 350mA source to drive each LED, but when the LED actually receives the current needs to be determined by the LED driver. This is accomplished by controlling the LED through an NMOS transistor. The NMOS transistor is connected as shown in Figure 13, below. The gate of the transistor is connected to the PCA9635 LED driver as well as to ground through a resistor, and it is through this gate that the LED is controlled. The source of the NMOS is ground, and the sink is the LED connected to V_{DD}. When the PCA9635 LED driver is commanded to turn on a specific LED, a high output signal is sent from the corresponding output pin. This high output allows the power source to pass through the transistor to the LED, lighting it. As long as this high signal is maintained, the LED will remain lit.

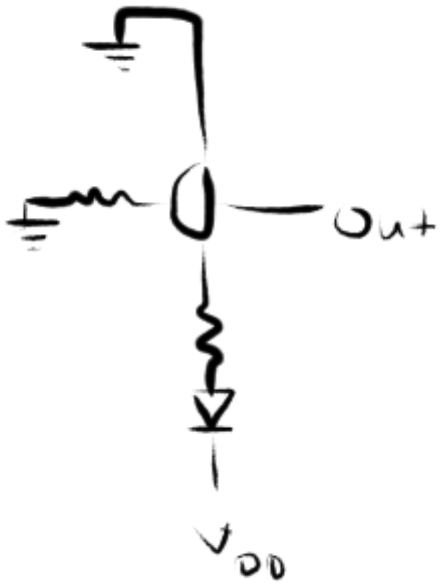


Figure 13

The voltage regulator chosen for this game system was the LM1117. This particular voltage regulator converted the power source to 3.3VDC, which was required for our board to function. The power to the voltage regulator was interrupted by a toggle switch placed between them. When the switch is set to OFF, the wall source power will not reach the voltage regulator, and therefore the board. When the switch is set to the ON position, the voltage regulator receives power, and converts it for board use.

4 System Integration Testing

4.1 How the set of subsystems were tested

In order to test the integrated subsystem, I2C communication, a USBee logic analyzer was used. The USBee allowed us to see what information was actually being transmitted from the master, the PIC32 microcontroller, to the slave, the PCA9635 LED driver. The USBee showed the SCL, SDA, and ground signals. By tracking these three signals, we were able to see start, stop, acknowledge, not acknowledge, device address, write, read, and data bits being transmitted. This was helpful for knowing if our software and hardware were functioning properly and was also crucial to troubleshooting. An example of these events on the USBee logic analyzer can be seen in Figure 14.

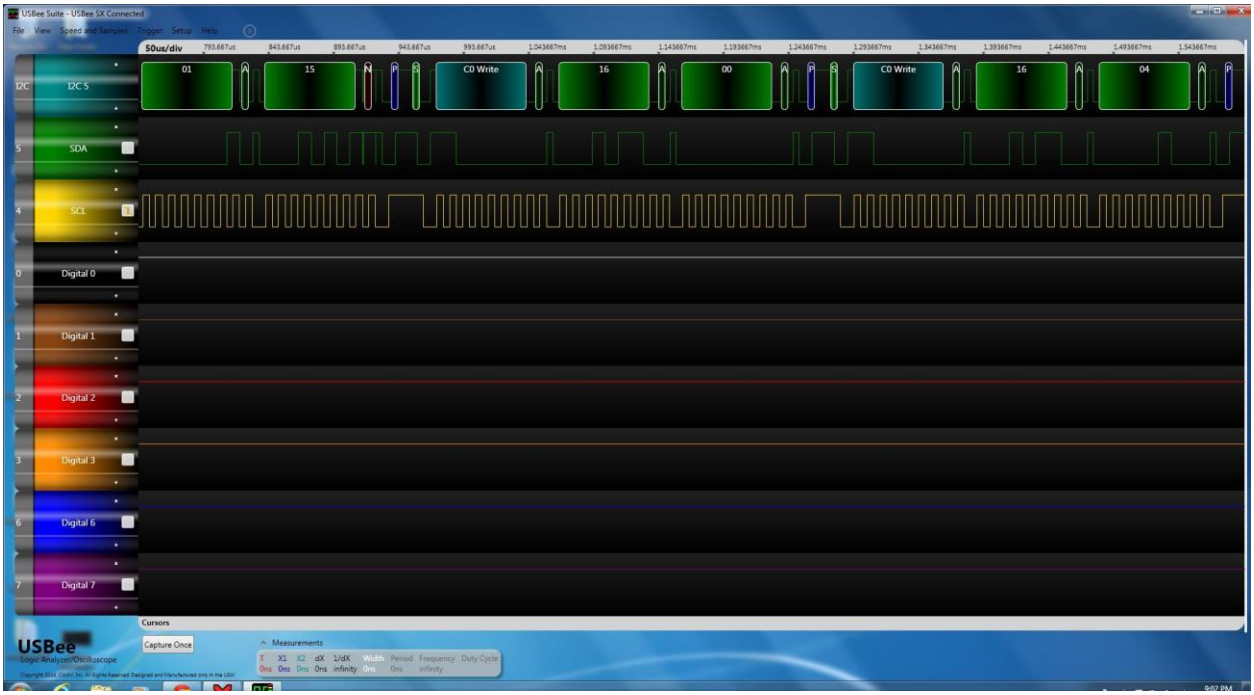


Figure 14

The actual game was tested by downloading the software to the microcontroller and simply playing it to see if the code was enacted in the expected manner.

4.2 Testing demonstrates that overall design meets requirements

By using the USBee logic analyzer to show that the necessary events were taking place, this proved that the system was meeting its requirement to effectively communicate through I2C.

By playing the game to see if the code performed in the expected manner, we were able to demonstrate that the overall system met the requirement to show that when a button that is lit is pressed within an allotted amount of time, its light will be turned off and a new button will light up. Or if the user does not press the button in the allotted amount of time the game would then end.

5 Users Manual/Installation manual

5.1 How to install your product

Assuming game code has been downloaded to the PIC32 microcontroller, installation is simple. The user must plug the device into a power outlet, and turn the switch on. The blue LED lighting on the power switch will indicate that the device is now on. The power switch can be seen in Figure 15.

Figure 15

5.2 How to setup your product

In order to set up the product, the user must first ensure the product is turned on (see section 5.1). Once the product has been turned on, the user should use the menu buttons, located above the LCD display, to select and begin a game.

A reset button is also located on the product, so that the user can select a different game at any point during the process.

5.3 How the user can tell if the product is working

The user can tell if the product is working by tracking their score on the LCD display. If the score is incrementing in accordance with them successfully pressing a lit button in the allotted amount of time, then the game is working.

If buttons are not lighting once a game has been started, then the product is not working.

If buttons are turning off before the user is pressing a button or before the allotted amount of time has passed, then the product is not working.

5.4 How the user can troubleshoot the product

The user may troubleshoot the product by pressing the reset button on the product. If that fails to correct the problem, the user may then turn the power switch to the off position then back to the on position.

If neither of these solutions corrects the problems, then the user should bring the device to an Electrical Engineer, so that they may check that the wiring is all connected and then redownload the software for the product. Further troubleshooting would include checking that data pins are still functioning and have not been blown.

6 To-Market Design Changes

Before bringing Light Em Up to market, there are a couple changes that we suggest be made. Although we agree with the simple aesthetics of the current box shape, we would further research alternative forms for the enclosure. If the current form factor is kept, the edges of the box should be rounded to avoid any injuries that could be caused by hitting the edges. If the current form factor is to be revisited, further research into materials and form factors should be done to best meet the needs of our users.

Another design change we would incorporate is the user interface and enabling easier data entry and acquisition for the supervisor of the game. To make it easier for the ADEC supervisors to enter user data and retrieve user data, the ability to use a keyboard to enter user

data and metrics would be beneficial. One such solution would be to equip the game with a small webserver that would enable tracking the players of the game from a browser, whether it be from your PC or mobile phone.

Although the use of a 4 line LCD would enable display of the score, the small screen is not as exciting as a big screen. To make the scoring of the game much more exciting, we would enable the use of a game scoring board that uses big 7 segment LED segments to display the scores in similar fashion to the the games in Chuckee Cheese or Dave and Buster.

7 Conclusions

Light Em Up has the ability to demonstrate cause and effect to persons with mental handicaps by showing that when one lit button is pressed, the light will turn off, and a new button will light up. By keeping track of the score, we have created the opportunity for users to track their progress overtime. This meets the requirements set forth by ADEC in a fun and visually stimulating manner. Light Em Up has a lot of potential to be used in a variety of settings. While that potential was not fully tapped by our group, future senior design groups may be able to pick up where we left off and fine tune the concept.

8 Appendices

cases.c file at the end of listings

Relevant parts or component data sheets (do NOT include the data sheets for the microcontroller or other huge files but give good links to where they may be found.)

Data Sheets

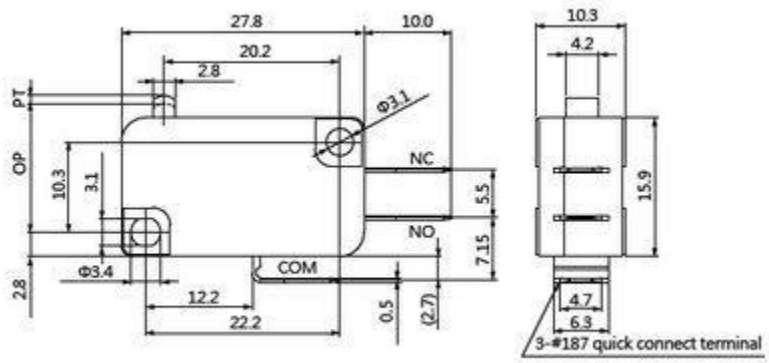
PIC32MX795F512H - http://ww1.microchip.com/downloads/en/DeviceDoc/61156H.pdf

PCA9635 - http://www.nxp.com/documents/data_sheet/PCA9635.pdf

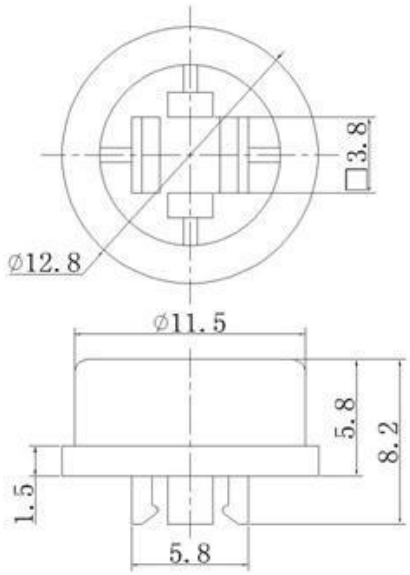
3W RGB LED - http://www.dx.com/p/3w-led-emitter-on-star-multicolored-rgb-4530#.U2kJevldUjk

New Haven LCD Display - http://www.newhavendisplay.com/specs/NHD-0420D3Z-FL-GBW.pdf

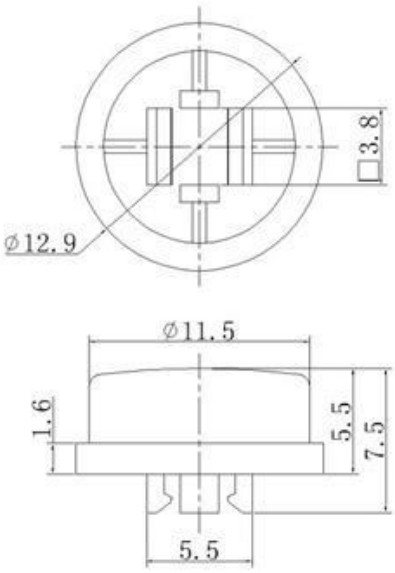
3 terminal micro-switch - https://www.sparkfun.com/products/9506



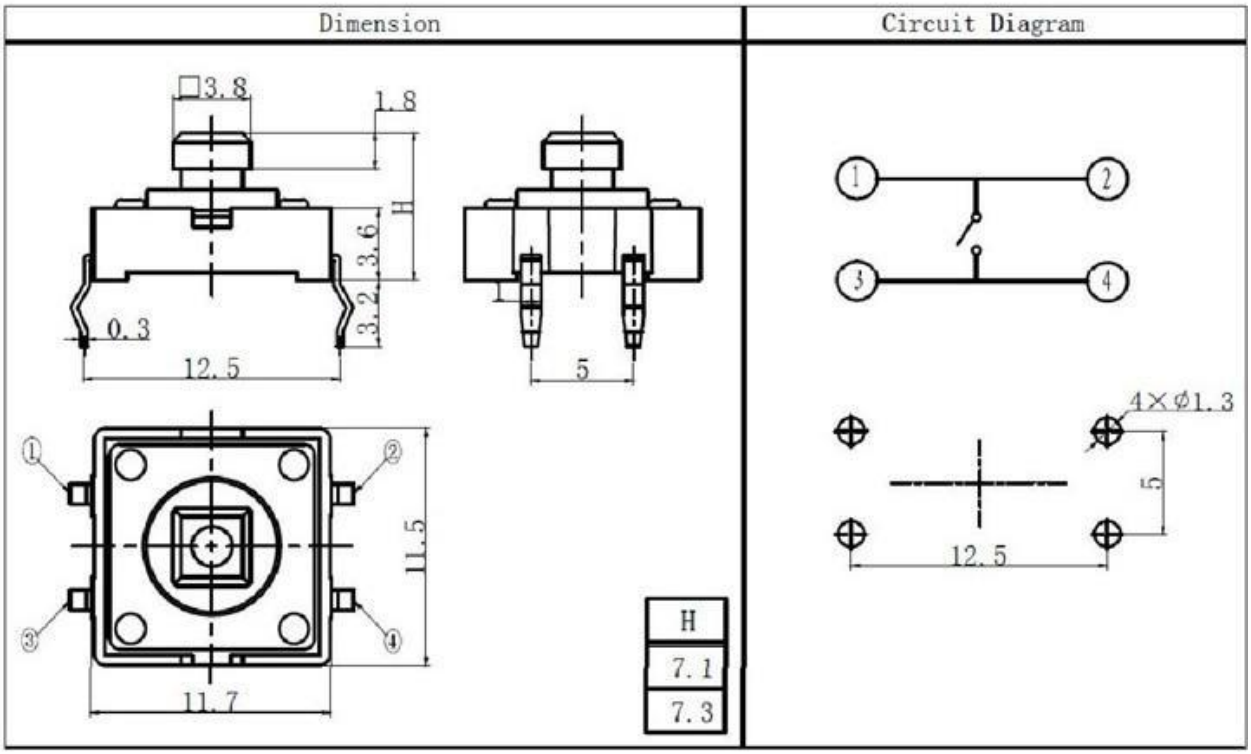
Tactile buttons - https://www.sparkfun.com/products/10302



SC203 CAP



SC215 CAP



```
/*
 * File:   cases.c
 * Author: jnord
 *
 * Created on April 30, 2014, 5:07 PM
 */


#define i2c_mif IFS0bits.I2C1MIF
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#include <plib.h>
#include "configbitsrev8.h"
#include "kit32r7lib.h"
#include "pca9635.h"
#define DEVADD 0xC0
#define DEVADD2 0xC2
#define not_idle (I2C1CON & 0x1F) //lower 5 bits of con reg

//i2c functions
void Initialize(void);
void Wait(void);
void Start(void);
void Stop(void);
char send_byte(int data);
void Restart(void);
void Write(char slave_address, char reg_address, char value);
char Read(void);

void Game0(void);
void blue0_Game(void);
void green0_Game(void);
void red0_Game(void);

void Game1(void);
void blue1_Game(void);
void green1_Game(void);
void red1_Game(void);

void Game2(void);
void blue2_Game(void);
void green2_Game(void);
void red2_Game(void);

void Game3(void);
void blue3_Game(void);
```

```
void green3_Game(void);
void red3_Game(void);

void Game4(void);
void blue4_Game(void);
void green4_Game(void);
void red4_Game(void);

void Game5(void);
void blue5_Game(void);
void green5_Game(void);
void red5_Game(void);

void Game6(void);
void blue6_Game(void);
void green6_Game(void);
void red6_Game(void);

void Game7(void);
void blue7_Game(void);
void green7_Game(void);
void red7_Game(void);

void Game8(void);
void blue8_Game(void);
void green8_Game(void);
void red8_Game(void);

void disp_Score(int score);
void choose_button(void);
//define parameters

char mode1 = MODE_REG1;
char mode1_val = 00;
char mode2 = MODE_REG2;
char mode2_val = 0x15;

//OUT0_REG
char led0_on = 0x01;
char led1_on =0x04 ;
char led2_on = 0x10;
char led3_on = 0x40;

//OUT1_REG
char led4_on = 0x01;
char led5_on =0x04 ;
char led6_on = 0x10;
```

```
char led7_on = 0x40;

//OUT2_REG
char led8_on = 0x01;
char led9_on =0x04 ;
char led10_on = 0x10;
char led11_on = 0x40;

//OUT3_REG
char led12_on = 0x01;
char led13_on =0x04 ;
char led14_on = 0x10;
char led15_on = 0x40;
char all_led_on = 0x55;


//Common led off
char led_off =0x00;

   int x, y, z;
   int end_game;
   int score = 0;
 unsigned char ack;
int main(int argc, char** argv) {
 //in order to put in an inifinite loop need to figure out what
conditions need to be reinitialized
//      TRISE =0;
//            LATE = 0xFF00;
//            delay_ms(1000);
//            LATE = 0x00FF;
//            delay_ms(1000);
//            LATE = 0xFF00;
    AD1PCFG=0xFFFF;
    TRISB= 0xFFFF;
Initialize();
//Mode 1 Register: Turn internal oscillator on
Write(DEVADD, mode1, mode1_val);

//Mode 2 : Invert bits to use mosFET
Write(DEVADD, mode2, mode2_val);


//turn all leds on
Write(DEVADD, OUT0_REG, all_led_on);
delay_ms(250);
Write(DEVADD, OUT1_REG, all_led_on);
delay_ms(250);
```

```
Write(DEVADD, OUT2_REG, all_led_on);
delay_ms(250);
Write(DEVADD, OUT3_REG, all_led_on);
delay_ms(250);
Write(DEVADD2, OUT0_REG, all_led_on);
delay_ms(250);
Write(DEVADD2, OUT1_REG, all_led_on);
delay_ms(250);
Write(DEVADD2, OUT2_REG, all_led_on);
delay_ms(250);

        //if button is not pressed turn the blue led on

        end_game=0;
        while(end_game==0){
            x=1;
            y=1;
            z=1;
            choose_button();
        }



return(EXIT_SUCCESS);
}
//Master interrupt flag


//Initialize Function
void Initialize(void){
     I2C1BRG = 0x2f; // 100khz with 10Mhz pb clock
 I2C1CONbits.DISSLW = 1; // disable sler
 I2C1CONbits.ON = 1;
 TRISDbits.TRISD5 = 0;
  TRISDbits.TRISD6 = 0;
   TRISDbits.TRISD7 = 0;
   TRISE = 0xFF00;
   LATE = 0;

}

//Wait Function
void Wait(void){
//wait for idle
     while(not_idle);
     //wait for interrupt flag
while(!i2c_mif);
```

```
     //clear interrupt flag
          i2c_mif = 0;
                return;
}

//Start Function
void Start(void){
     i2c_mif = 0; // SSP int flag
  while(not_idle);
 I2C1CONbits.SEN = 1; // send "start bit"
 Wait(); // wait till done

}

//Stop Fucntion
void Stop(void){
    /* do a stop */

   i2c_mif = 0; // master int flag
   while(not_idle);

 I2C1CONbits.PEN = 1;

 Wait();
 //delay_ms(250);

}

//Restart Function
void Restart(void){
     i2c_mif = 0; // SSP int flag
  while(not_idle);
 I2C1CONbits.RSEN = 1; // send "start bit"
 Wait(); // wait till done

}

//Send Function
char send_byte(int data){
i2c_mif = 0; // SSP int flag
   while(not_idle);
  I2C1TRN = data;  // send to device address

Wait();

 if (I2C1STATbits.ACKSTAT)
   ack = 0; // no ack
```

```
 else
   ack = 1; // ack
}

//Write Function
void Write(char slave_address, char reg_address, char value){
Start();
send_byte(slave_address);
send_byte(reg_address);
send_byte(value);
Stop();

}

//Choose button to light
void choose_button(void){
    char button;
    button = rand()%8; //generates a random integer between 0
and 8
    if (button==0){
       // PORTBbits.RB0;
        Game0();
    }
    else if (button==1){
       // PORTBbits.RB1;
        Game1();
    }
    else if (button==2){
       // PORTBbits.RB2;
        Game2();
    }
    else if (button==3){
       // PORTBbits.RB3;
        Game3();
    }
    else if (button==4){
       // PORTBbits.RB4;
        Game4();
    }
    else if (button==5){
       // PORTBbits.RB5;
        Game5();
    }
    else if (button==6){
       // PORTBbits.RB6;
        Game6();
    }
```

```
    else if (button==7){
       // PORTBbits.RB7;
        Game7();
    }
    else if (button==8){
       // PORTBbits.RB8;
        Game8();
    }
    else
        {
        Write(DEVADD, OUT2_REG, all_led_on);
        }

}
//Display score to LCD display
void disp_Score(int score){
           LCD_init(100000UL);
           serial_init(57600UL);
           set_output_device(2);
           LCD_display_on();
           LCD_clear();
           LCD_setpos(0,0);
//           printf(score);
           LCD_setpos(1,0);


           set_output_device(1);
       //    printf(score);
}
//Game Function
void blue0_Game(void){
//if(PORTBbits.RB0==0){
//Write(DEVADD, OUT0_REG, led0_on); //blue light on
while(x==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB0==0){

          Write(DEVADD, OUT0_REG, led0_on); //blue light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }
```

```
//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT0_REG, led0_on); //green
light on
                    Write(DEVADD, OUT0_REG, led_off);
                    delay_ms(1000);
                    x=0;
                    count=201;

                }
            }

         if(lose==1){
//                       Write(DEVADD, OUT0_REG, all_led_on);
//                       delay_ms(500);
                      Write(DEVADD, OUT0_REG, led0_on);
                      delay_ms(250);
                      Write(DEVADD, OUT0_REG, led1_on);
                      delay_ms(250);
                      Write(DEVADD, OUT0_REG, led2_on);
                      delay_ms(250);
                      x=0;
                      end_game=1;
                }
    }
            }

//Green Game Function
void green0_Game(void){
    //if(PORTBbits.RB0==0){
        while(y==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB0==0){

          Write(DEVADD, OUT0_REG, led1_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
```

```
                    Write(DEVADD, OUT0_REG, led1_on); //green
light on
                    Write(DEVADD, OUT0_REG, led_off);
                    delay_ms(1000);
                    y=0;
                    count=201;

                }
            }

         if(lose==1){
//                       Write(DEVADD, OUT0_REG, all_led_on);
//                       delay_ms(500);
                      Write(DEVADD, OUT0_REG, led0_on);
                      delay_ms(250);
                      Write(DEVADD, OUT0_REG, led1_on);
                      delay_ms(250);
                      Write(DEVADD, OUT0_REG, led2_on);
                      delay_ms(250);
                      y=0;
                      end_game=1;
                }
    }
            }
//Red Game Function
void red0_Game(void){
       while(z==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB0==0){

          Write(DEVADD, OUT0_REG, led2_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT0_REG, led2_on); //green
light on
                    Write(DEVADD, OUT0_REG, led_off);
                    delay_ms(1000);
                    z=0;
```

```
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT0_REG, all_led_on);
//                      delay_ms(500);
                      Write(DEVADD, OUT0_REG, led0_on);
                      delay_ms(250);
                      Write(DEVADD, OUT0_REG, led1_on);
                      delay_ms(250);
                      Write(DEVADD, OUT0_REG, led2_on);
                      delay_ms(250);
                      z=0;
                      end_game=1;
                }
    }
            }
//end add
//Game function

void Game0(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue0_Game();}
        else if (select==1){
                green0_Game();}
        else
            {red0_Game();}
            }
    else{
          Write(DEVADD, OUT0_REG, led0_on);
          Write(DEVADD, OUT0_REG, led_off);
    }
}

//Game Function
void blue1_Game(void){

//    while(x==1){
//if(PORTBbits.RB1==0){
//        //Write(DEVADD, OUT3_REG, led_off);
//Write(DEVADD, OUT0_REG, led3_on); //blue light on
       while(x==1){
```

```
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB1==0){

          Write(DEVADD, OUT0_REG, led3_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT0_REG, led3_on); //green
light on
                    Write(DEVADD, OUT0_REG, led_off);
                    delay_ms(1000);
                    x=0;
                    count=201;

                }
            }

         if(lose==1){
//                       Write(DEVADD, OUT0_REG, all_led_on);
//                       delay_ms(500);
                     Write(DEVADD, OUT0_REG, led3_on);
                     delay_ms(250);
                     Write(DEVADD, OUT1_REG, led4_on);
                     delay_ms(250);
                     Write(DEVADD, OUT1_REG, led5_on);
                     delay_ms(250);
                     x=0;
                     end_game=1;
                }
    }
            }
//end add
//Green Game Function
void green1_Game(void){
//        while(y==1){
//if(PORTBbits.RB1==0){
//        //Write(DEVADD, OUT3_REG, led_off);
//Write(DEVADD, OUT1_REG, led4_on); //green light on
       while(y==1){
```

```
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB1==0){

          Write(DEVADD, OUT1_REG, led4_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT1_REG, led4_on); //green
light on
                    Write(DEVADD, OUT1_REG, led_off);
                    delay_ms(1000);
                    y=0;
                    count=201;

                }
            }

         if(lose==1){
//                       Write(DEVADD, OUT0_REG, all_led_on);
//                       delay_ms(500);
                     Write(DEVADD, OUT0_REG, led3_on);
                     delay_ms(250);
                     Write(DEVADD, OUT1_REG, led4_on);
                     delay_ms(250);
                     Write(DEVADD, OUT1_REG, led5_on);
                     delay_ms(250);
                     y=0;
                     end_game=1;
                }
    }
            }
//end add
//Red Game Function
void red1_Game(void){
//        while(z==1){
//if(PORTBbits.RB1==0){
//        //Write(DEVADD, OUT3_REG, led_off);
//Write(DEVADD, OUT1_REG, led5_on); //red light on
       while(z==1){
```

```
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB1==0){

          Write(DEVADD, OUT1_REG, led5_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT1_REG, led5_on); //green
light on
                    Write(DEVADD, OUT1_REG, led_off);
                    delay_ms(1000);
                    z=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT0_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD, OUT0_REG, led3_on);
                    delay_ms(250);
                    Write(DEVADD, OUT1_REG, led4_on);
                    delay_ms(250);
                    Write(DEVADD, OUT1_REG, led5_on);
                    delay_ms(250);
                    z=0;
                    end_game=1;
                }
    }
            }
//end add
//Game function

void Game1(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
```

```
            blue1_Game();}
        else if (select==1){
                green1_Game();}
        else
            {red1_Game();}
            }
    else{
          Write(DEVADD, OUT0_REG, led3_on);
          Write(DEVADD, OUT0_REG, led_off);
          Write(DEVADD, OUT1_REG, led4_on);
          Write(DEVADD, OUT1_REG, led_off);
    }
}
void blue2_Game(void){

//    while(x==1){
//if(PORTBbits.RB2==0){
//Write(DEVADD, OUT1_REG, led6_on); //blue light on
       while(x==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB2==0){

          Write(DEVADD, OUT1_REG, led6_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//               if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT1_REG, led6_on); //green
light on
                    Write(DEVADD, OUT1_REG, led_off);
                    delay_ms(1000);
                    x=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT1_REG, all_led_on);
//                      delay_ms(500);
```

```
                    Write(DEVADD, OUT1_REG, led6_on);
                    delay_ms(250);
                    Write(DEVADD, OUT1_REG, led7_on);
                    delay_ms(250);
                    Write(DEVADD, OUT2_REG, led8_on);
                    delay_ms(250);
                    x=0;
                    end_game=1;
                }
    }
            }
//end add
//Green Game Function
void green2_Game(void){
//        while(y==1){
//if(PORTBbits.RB2==0){
//Write(DEVADD, OUT1_REG, led7_on); //green light on
       while(y==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB2==0){

          Write(DEVADD, OUT1_REG, led7_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT1_REG, led7_on); //green
light on
                    Write(DEVADD, OUT1_REG, led_off);
                    delay_ms(1000);
                    y=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT1_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD, OUT1_REG, led6_on);
```

```
                    delay_ms(250);
                    Write(DEVADD, OUT1_REG, led7_on);
                    delay_ms(250);
                    Write(DEVADD, OUT2_REG, led8_on);
                    delay_ms(250);
                    y=0;
                    end_game=1;
                }
    }
            }
//end add
//Red Game Function
void red2_Game(void){
//        while(z==1){
//if(PORTBbits.RB2==0){
//Write(DEVADD, OUT2_REG, led8_on); //red light on
       while(z==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB2==0){

          Write(DEVADD, OUT2_REG, led8_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT2_REG, led8_on); //green
light on
                    Write(DEVADD, OUT2_REG, led_off);
                    delay_ms(1000);
                    z=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT1_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD, OUT1_REG, led6_on);
                    delay_ms(250);
```

```
                    Write(DEVADD, OUT1_REG, led7_on);
                    delay_ms(250);
                    Write(DEVADD, OUT2_REG, led8_on);
                    delay_ms(250);
                    z=0;
                    end_game=1;
                }
    }
            }
//end add
//Game function

void Game2(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue2_Game();}
        else if (select==1){
                green2_Game();}
        else
            {red2_Game();}
            }
    else{
          Write(DEVADD, OUT1_REG, led6_on);
          Write(DEVADD, OUT1_REG, led_off);
          Write(DEVADD, OUT2_REG, led8_on);
          Write(DEVADD, OUT2_REG, led_off);
    }
}//Game Function
void blue3_Game(void){

//    while(x==1){
//if(PORTBbits.RB3==0){
//Write(DEVADD, OUT2_REG, led9_on); //blue light on
       while(x==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB3==0){

          Write(DEVADD, OUT2_REG, led9_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
```

```
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT2_REG, led9_on); //green
light on
                    Write(DEVADD, OUT2_REG, led_off);
                    delay_ms(1000);
                    x=0;
                    count=201;

                }
            }

         if(lose==1){
//                       Write(DEVADD, OUT2_REG, all_led_on);
//                       delay_ms(500);
                      Write(DEVADD, OUT2_REG, led9_on);
                      delay_ms(250);
                      Write(DEVADD, OUT2_REG, led10_on);
                      delay_ms(250);
                      Write(DEVADD, OUT2_REG, led11_on);
                      delay_ms(250);
                      x=0;
                      end_game=1;
                }
    }
            }
//end add

//Green Game Function
void green3_Game(void){
//        while(y==1){
//if(PORTBbits.RB3==0){
//Write(DEVADD, OUT2_REG, led10_on); //green light on
       while(y==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB3==0){

          Write(DEVADD, OUT2_REG, led10_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
```

```
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT2_REG, led10_on); //green
light on
                    Write(DEVADD, OUT2_REG, led_off);
                    delay_ms(1000);
                    y=0;
                    count=201;

                }
            }

         if(lose==1){
//                       Write(DEVADD, OUT2_REG, all_led_on);
//                       delay_ms(500);
                      Write(DEVADD, OUT2_REG, led9_on);
                      delay_ms(250);
                      Write(DEVADD, OUT2_REG, led10_on);
                      delay_ms(250);
                      Write(DEVADD, OUT2_REG, led11_on);
                      delay_ms(250);
                      y=0;
                      end_game=1;
                }
    }
            }
//end add
//Red Game Function
void red3_Game(void){
//        while(z==1){
//if(PORTBbits.RB3==0){
//Write(DEVADD, OUT2_REG, led11_on); //red light on
       while(z==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB3==0){

          Write(DEVADD, OUT2_REG, led11_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }
```

```
//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT2_REG, led11_on); //green
light on
                    Write(DEVADD, OUT2_REG, led_off);
                    delay_ms(1000);
                    z=0;
                    count=201;

                }
            }

         if(lose==1){
//                       Write(DEVADD, OUT2_REG, all_led_on);
//                       delay_ms(500);
                      Write(DEVADD, OUT2_REG, led9_on);
                      delay_ms(250);
                      Write(DEVADD, OUT2_REG, led10_on);
                      delay_ms(250);
                      Write(DEVADD, OUT2_REG, led11_on);
                      delay_ms(250);
                      z=0;
                      end_game=1;
                }
    }
            }
//end add
//Game function

void Game3(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue3_Game();}
        else if (select==1){
                green3_Game();}
        else
            {red3_Game();}
            }
    else{
          Write(DEVADD, OUT2_REG, led9_on);
          Write(DEVADD, OUT2_REG, led_off);
    }
}//Game Function
void blue4_Game(void){
```

```
//    while(x==1){
//if(PORTBbits.RB4==0){
//Write(DEVADD, OUT3_REG, led12_on); //blue light on
       while(x==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB4==0){

          Write(DEVADD, OUT3_REG, led12_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT3_REG, led12_on); //green
light on
                    Write(DEVADD, OUT3_REG, led_off);
                    delay_ms(1000);
                    x=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT3_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD, OUT3_REG, led12_on);
                    delay_ms(250);
                    Write(DEVADD, OUT3_REG, led13_on);
                    delay_ms(250);
                    Write(DEVADD, OUT3_REG, led14_on);
                    delay_ms(250);
                    x=0;
                    end_game=1;
                }
    }
            }
//end add
//Green Game Function
void green4_Game(void){
```

```
//        while(y==1){
//if(PORTBbits.RB4==0){
//Write(DEVADD, OUT3_REG, led13_on); //green light on
       while(y==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB4==0){

          Write(DEVADD, OUT3_REG, led13_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT3_REG, led13_on); //green
light on
                    Write(DEVADD, OUT3_REG, led_off);
                    delay_ms(1000);
                    y=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT3_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD, OUT3_REG, led12_on);
                    delay_ms(250);
                    Write(DEVADD, OUT3_REG, led13_on);
                    delay_ms(250);
                    Write(DEVADD, OUT3_REG, led14_on);
                    delay_ms(250);
                    y=0;
                    end_game=1;
                }
    }
            }
//end add
//Red Game Function
void red4_Game(void){
//        while(z==1){
```

```
//if(PORTBbits.RB4==0){
//Write(DEVADD, OUT3_REG, led14_on); //red light on
       while(z==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB4==0){

          Write(DEVADD, OUT3_REG, led14_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT3_REG, led14_on); //green
light on
                    Write(DEVADD, OUT3_REG, led_off);
                    delay_ms(1000);
                    z=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT3_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD, OUT3_REG, led12_on);
                    delay_ms(250);
                    Write(DEVADD, OUT3_REG, led13_on);
                    delay_ms(250);
                    Write(DEVADD, OUT3_REG, led14_on);
                    delay_ms(250);
                    z=0;
                    end_game=1;
                }
    }
            }
//end add
//Game function

void Game4(void){
    if (end_game==0){
```

```
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue4_Game();}
        else if (select==1){
                green4_Game();}
        else
            {red4_Game();}
            }
    else{
          Write(DEVADD, OUT3_REG, led12_on);
          Write(DEVADD, OUT3_REG, led_off);
    }
}//Game Function
void blue5_Game(void){

//    while(x==1){
//if(PORTBbits.RB5==0){
//Write(DEVADD, OUT3_REG, led15_on); //blue light on
       while(x==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB5==0){

          Write(DEVADD, OUT3_REG, led15_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD, OUT3_REG, led15_on); //green
light on
                    Write(DEVADD, OUT3_REG, led_off);
                    delay_ms(1000);
                    x=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD, OUT3_REG, all_led_on);
```

```
//                      delay_ms(500);
                      Write(DEVADD, OUT3_REG, led15_on);
                      delay_ms(250);
                      Write(DEVADD2, OUT0_REG, led0_on);
                      delay_ms(250);
                      Write(DEVADD2, OUT0_REG, led1_on);
                      delay_ms(250);
                      x=0;
                      end_game=1;
                  }
    }
             }
//end add

//Green Game Function
void green5_Game(void){
//        while(y==1){
//if(PORTBbits.RB5==0){
//Write(DEVADD2, OUT0_REG, led0_on); //green light on
       while(y==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB5==0){

          Write(DEVADD2, OUT0_REG, led0_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT0_REG, led0_on); //green
light on
                    Write(DEVADD2, OUT0_REG, led_off);
                    delay_ms(1000);
                    y=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD2, OUT0_REG, all_led_on);
```

```
//                      delay_ms(500);
                      Write(DEVADD, OUT3_REG, led15_on);
                      delay_ms(250);
                      Write(DEVADD2, OUT0_REG, led0_on);
                      delay_ms(250);
                      Write(DEVADD2, OUT0_REG, led1_on);
                      delay_ms(250);
                      y=0;
                      end_game=1;
                  }
    }
             }
//end add
//Red Game Function
void red5_Game(void){
//        while(z==1){
//if(PORTBbits.RB5==0){
//Write(DEVADD2, OUT0_REG, led1_on); //red light on
       while(z==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB5==0){

          Write(DEVADD2, OUT0_REG, led1_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT0_REG, led1_on); //green
light on
                    Write(DEVADD2, OUT0_REG, led_off);
                    delay_ms(1000);
                    z=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD2, OUT0_REG, all_led_on);
//                      delay_ms(500);
```

```
                        Write(DEVADD, OUT3_REG, led15_on);
                        delay_ms(250);
                        Write(DEVADD2, OUT0_REG, led0_on);
                        delay_ms(250);
                        Write(DEVADD2, OUT0_REG, led1_on);
                        delay_ms(250);
                        z=0;
                        end_game=1;
                    }
    }
             }
//end add
//Game function

void Game5(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue5_Game();}
        else if (select==1){
                green5_Game();}
        else
            {red5_Game();}
            }
    else{
          Write(DEVADD, OUT3_REG, led15_on);
          Write(DEVADD, OUT3_REG, led_off);

          Write(DEVADD2, OUT0_REG, led0_on);
          Write(DEVADD2, OUT0_REG, led_off);
          Write(DEVADD2, OUT0_REG, led1_on);
          Write(DEVADD2, OUT0_REG, led_off);
    }
}//Game Function
void blue6_Game(void){

//    while(x==1){
//if(PORTBbits.RB6==0){
//Write(DEVADD2, OUT0_REG, led2_on); //blue light on
       while(x==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB6==0){

          Write(DEVADD2, OUT0_REG, led2_on); // light on
```

```
            delay_ms(10);
             count++;
             if (count==200){
                 lose=1;
             }
             }

//                 if(PORTBbits.RBx==1) button is pressed
             else{
                     Write(DEVADD2, OUT0_REG, led2_on); //green
light on
                     Write(DEVADD2, OUT0_REG, led_off);
                     delay_ms(1000);
                     x=0;
                     count=201;

                 }
             }

          if(lose==1){
//                       Write(DEVADD2, OUT0_REG, all_led_on);
//                       delay_ms(500);
                       Write(DEVADD2, OUT0_REG, led2_on);
                       delay_ms(250);
                       Write(DEVADD2, OUT0_REG, led3_on);
                       delay_ms(250);
                       Write(DEVADD2, OUT1_REG, led4_on);
                       delay_ms(250);
                       x=0;
                       end_game=1;
                 }
    }
             }
//end add

//Green Game Function
void green6_Game(void){
//         while(y==1){
//if(PORTBbits.RB6==0){
//Write(DEVADD2, OUT0_REG, led3_on); //green light on
        while(y==1){
         int count=0;
         int lose=0;
         while(count<200){
            if(PORTBbits.RB6==0){

            Write(DEVADD2, OUT0_REG, led3_on);// light on
```

```
            delay_ms(10);
             count++;
             if (count==200){
                 lose=1;
             }
             }

//                 if(PORTBbits.RBx==1) button is pressed
             else{
                     Write(DEVADD2, OUT0_REG, led3_on); //green
light on
                     Write(DEVADD2, OUT0_REG, led_off);
                     delay_ms(1000);
                     y=0;
                     count=201;

                 }
             }

          if(lose==1){
//                       Write(DEVADD2, OUT0_REG, all_led_on);
//                       delay_ms(500);
                       Write(DEVADD2, OUT0_REG, led2_on);
                       delay_ms(250);
                       Write(DEVADD2, OUT0_REG, led3_on);
                       delay_ms(250);
                       Write(DEVADD2, OUT1_REG, led4_on);
                       delay_ms(250);
                       y=0;
                       end_game=1;
                 }
    }
             }
//end add
//Red Game Function
void red6_Game(void){
//        while(z==1){
//if(PORTBbits.RB6==0){
//Write(DEVADD2, OUT1_REG, led4_on); //red light on
       while(z==1){
         int count=0;
         int lose=0;
         while(count<200){
            if(PORTBbits.RB6==0){

            Write(DEVADD2, OUT1_REG, led4_on); // light on
            delay_ms(10);
```

```
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT1_REG, led4_on);  //green
light on
                    Write(DEVADD2, OUT1_REG, led_off);
                    delay_ms(1000);
                    z=0;
                    count=201;

                }
            }

         if(lose==1){
//                        Write(DEVADD2, OUT0_REG, all_led_on);
//                        delay_ms(500);
                      Write(DEVADD2, OUT0_REG, led2_on);
                      delay_ms(250);
                      Write(DEVADD2, OUT0_REG, led3_on);
                      delay_ms(250);
                      Write(DEVADD2, OUT1_REG, led4_on);
                      delay_ms(250);
                      z=0;
                      end_game=1;
                }
    }
            }
//end add
//Game function

void Game6(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue6_Game();}
        else if (select==1){
                green6_Game();}
        else
            {red6_Game();}
            }
    else{
```

```
          Write(DEVADD2, OUT0_REG, led2_on);
          Write(DEVADD2, OUT0_REG, led_off);
          Write(DEVADD2, OUT1_REG, led4_on);
          Write(DEVADD2, OUT1_REG, led_off);
    }
}//Game Function
void blue7_Game(void){

//    while(x==1){
//if(PORTBbits.RB7==0){
//Write(DEVADD2, OUT1_REG, led5_on); //blue light on
       while(x==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB7==0){

           Write(DEVADD2, OUT1_REG, led5_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT1_REG, led5_on);  //green
light on
                    Write(DEVADD2, OUT1_REG, led_off);
                    delay_ms(1000);
                    x=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD2, OUT1_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD2, OUT1_REG, led5_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT1_REG, led6_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT1_REG, led7_on);
                    delay_ms(250);
                    x=0;
```

```
                    end_game=1;
                }
    }
            }
//end add

//Green Game Function
void green7_Game(void){
//        while(y==1){
//if(PORTBbits.RB7==0){
//Write(DEVADD2, OUT1_REG, led6_on); //green light on
       while(y==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB7==0){

           Write(DEVADD2, OUT1_REG, led6_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT1_REG, led6_on);  //green
light on
                    Write(DEVADD2, OUT1_REG, led_off);
                    delay_ms(1000);
                    y=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD2, OUT1_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD2, OUT1_REG, led5_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT1_REG, led6_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT1_REG, led7_on);
                    delay_ms(250);
                    y=0;
```

```
                    end_game=1;
                }
    }
            }
//end add

//Red Game Function
void red7_Game(void){
//        while(z==1){
//if(PORTBbits.RB7==0){
//Write(DEVADD2, OUT1_REG, led7_on); //red light on
       while(z==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB7==0){

           Write(DEVADD2, OUT1_REG, led7_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//               if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT1_REG, led7_on);  //green
light on
                    Write(DEVADD2, OUT1_REG, led_off);
                    delay_ms(1000);
                    z=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD2, OUT1_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD2, OUT1_REG, led5_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT1_REG, led6_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT1_REG, led7_on);
                    delay_ms(250);
                    z=0;
```

```
                    end_game=1;
                }
    }
            }
//end add
//Game function

void Game7(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue7_Game();}
        else if (select==1){
                green7_Game();}
        else
            {red7_Game();}
            }
    else{
          Write(DEVADD2, OUT1_REG, led5_on);
          Write(DEVADD2, OUT1_REG, led_off);
    }
}//Game Function
void blue8_Game(void){

//    while(x==1){
//if(PORTBbits.RB8==0){
//Write(DEVADD2, OUT2_REG, led8_on); //blue light on
       while(x==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB8==0){

           Write(DEVADD2, OUT2_REG, led8_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT2_REG, led8_on);  //green
light on
                    Write(DEVADD2, OUT2_REG, led_off);
```

```
                    delay_ms(1000);
                    x=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD2, OUT2_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD2, OUT2_REG, led8_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT2_REG, led9_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT2_REG, led10_on);
                    delay_ms(250);
                    x=0;
                    end_game=1;
                }
    }
            }
//end add
//Green Game Function
void green8_Game(void){
//        while(y==1){
//if(PORTBbits.RB8==0){
//Write(DEVADD2, OUT2_REG, led9_on); //green light on
       while(y==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB8==0){

           Write(DEVADD2, OUT2_REG, led9_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT2_REG, led9_on);  //green
light on
                    Write(DEVADD2, OUT2_REG, led_off);
                    delay_ms(1000);
```

```
                    y=0;
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD2, OUT2_REG, all_led_on);
//                      delay_ms(500);
                    Write(DEVADD2, OUT2_REG, led8_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT2_REG, led9_on);
                    delay_ms(250);
                    Write(DEVADD2, OUT2_REG, led10_on);
                    delay_ms(250);
                    y=0;
                    end_game=1;
                }
    }
            }
//end add
//Red Game Function
void red8_Game(void){
//        while(z==1){
//if(PORTBbits.RB8==0){
//Write(DEVADD2, OUT2_REG, led10_on); //red light on
       while(z==1){
        int count=0;
        int lose=0;
        while(count<200){
           if(PORTBbits.RB8==0){

           Write(DEVADD2, OUT2_REG, led10_on); // light on
           delay_ms(10);
            count++;
            if (count==200){
                lose=1;
            }
            }

//                if(PORTBbits.RBx==1) button is pressed
            else{
                    Write(DEVADD2, OUT2_REG, led10_on);  //green
light on
                    Write(DEVADD2, OUT2_REG, led_off);
                    delay_ms(1000);
                    z=0;
```

```
                    count=201;

                }
            }

         if(lose==1){
//                      Write(DEVADD2, OUT2_REG, all_led_on);
//                      delay_ms(500);
                      Write(DEVADD2, OUT2_REG, led8_on);
                      delay_ms(250);
                      Write(DEVADD2, OUT2_REG, led9_on);
                      delay_ms(250);
                      Write(DEVADD2, OUT2_REG, led10_on);
                      delay_ms(250);
                      z=0;
                      end_game=1;
                }
    }
            }
//end add
//Game function

void Game8(void){
    if (end_game==0){
    int select;
    select= rand()%2; //generate an random int between 0 and 2
        if (select==0){
            blue8_Game();}
        else if (select==1){
                green8_Game();}
        else
            {red8_Game();}
            }
    else{
          Write(DEVADD2, OUT2_REG, led8_on);
          Write(DEVADD2, OUT2_REG, led_off);
    }
}
//Read Function
//Ack asks for next byte; Nack ends request for bytes
//ACKDT bit
char Read(void){
     char data = 0;
     //clear I2C1 master interrupt flag
i2c_mif = 0;
     //wait for idle:
while(not_idle)
```

```
      I2C1CONbits.RCEN = 1; //set module to receive
//wait
while(I2C1STATbits.RBF)
data = I2C1RCV; //transfers contents of the register to data
return data;

}
```