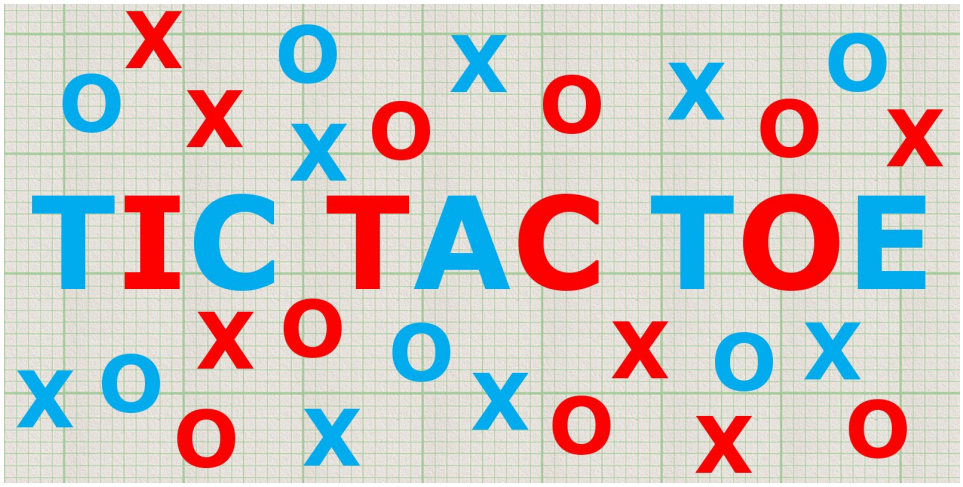
The Bit Boiz



Keaton Shelton and Johnathon Longmire

ECE 3130 Microcomputer Systems

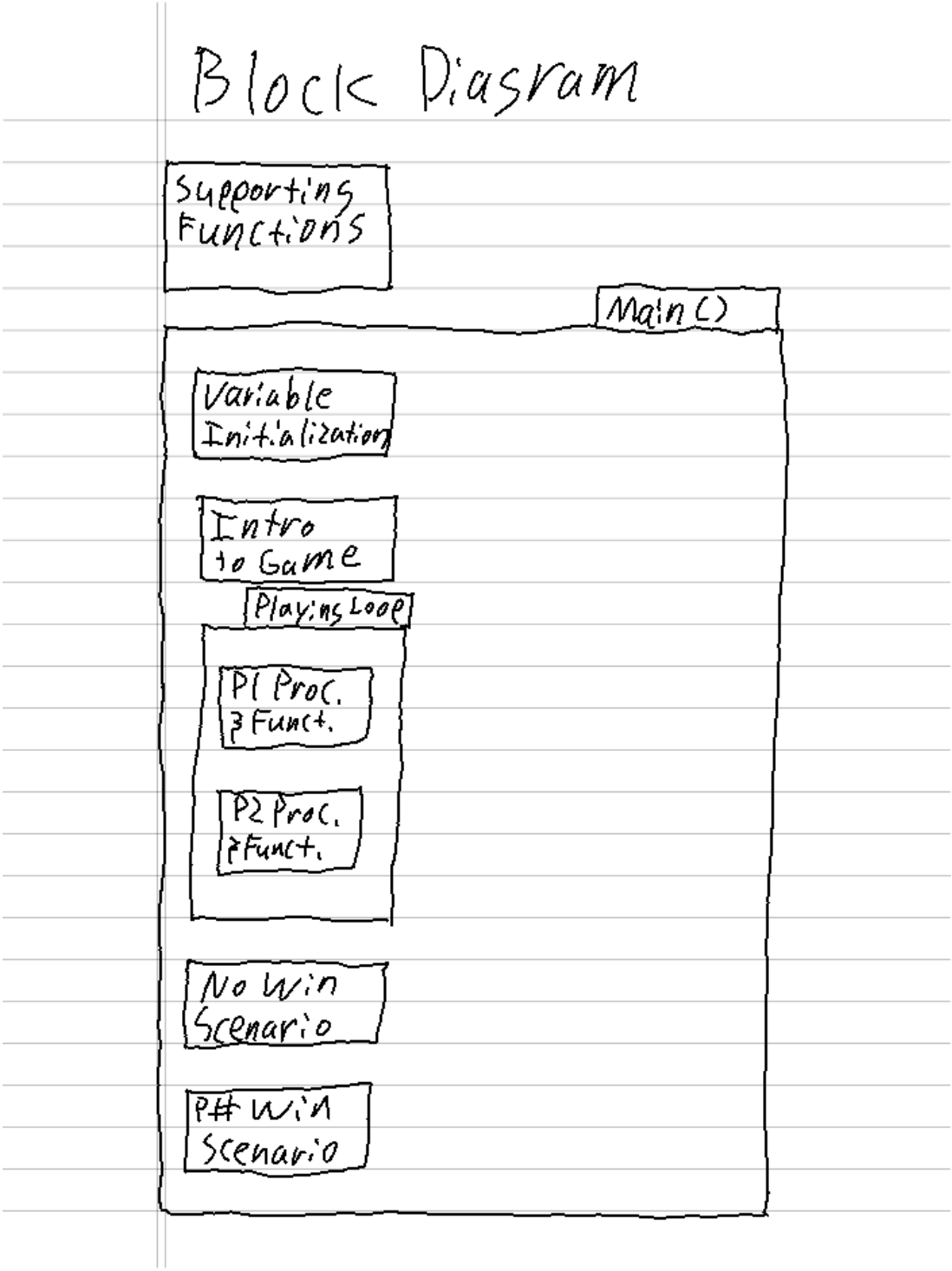
05/04/2021

Table of Contents

Introduction

The objective of our code is to create a game of Tic-Tac-Toe. To win this game, one player must have three of their characters connected either vertically, horizontally, or diagonally. This code accomplished that goal of having a winner and will also report if the game is a tie. The application for this code is to produce a game of Tic-Tac-Toe. The game itself has been around for a long time so it has shown its importance as a kids game. The code is important to that because without it, the game can't be played. The game was created using C formatting and following the rules of Tic-Tac-Toe. There are two players who try to obtain three in a row and if they achieve, they win. If not, then it is a tie. Everything involved in Tic-Tac-Toe is involved in the code. Two main problems at first in the code is displaying the board on the LCD and getting out of the playing mode to declare a winner. The first problem was resolved by fixing the syntax for importing the function. The second problem was fixed by adding an if statement to check for victory and then jumping to declare the winner. Using this software with the HCS12 allows for a complete playthrough of the Tic-Tac-Toe game and accounting for all win scenarios.

Project Specifications and Description



The software will launch and give an introduction of the game and how to input your marks onto the board. It will then declare the first player is up and show it on the seven segment displays and mark the RGB LED as white as well as making an auditory beep to declare the start of the game. The game will then run showing the layout of the board in between players on the LCD panel. The game only allows for using the 1-9 button pad and no more. If any other key is pressed the user will be presented with an error and their turn will repeat. This also occurs if the player tries to select another mark that has already been placed by the opposing player. After each turn the system will check to determine if any player has won yet. If so the player who won will be presented on the LCD panel and proceed to ask if a new game would like to be started. Also similarly if no player has won and it results in a no win scenario then the game will make fun of both players and ask if a new game would like to be started.

Detailed Algorithms

Refer to Project Specifications and Description for block diagram containing structure, and modularity of program. The block diagram goes in order from top to bottom to show the flow of the program.

Algorithms:

- player7seg(int,int*)
    - This function takes the current player number and supporting array of 7 segment codes to show the current player
- dispBoard(char[][3])
    - This function takes the contents of a 3x3 matrix and displays it on the LCD panel
- takeInput(int,char[][3])
    - This function takes the input of the current player and maps it onto the correct location in the 3x3 matrix. This function also detects for false inputs and restarts entry
- cForWin(int,char[][3])
    - This function takes the current player information along with the connected 3x3 matrix and checks for a win condition under the rules of Tic-Tac-Toe

Detail Implementation

Interface Design and Utilization:

This software takes advantage of the following HCS12 parts

- 7 Segment Display
    - Displays current player number at start of turn
- RGB LED
    - Turns white at the same time the 7 segment display is used to show the transition to a different player
- LCD Display
    - Used to display the intro, board information, input sections, and winner of the game
- Switches
    - Uses SW5 to begin the game and restart the game when prompted
- 4x4 PAD
    - Uses PAD 1-9 as input for the Tic-Tac-Toe board
- Buzzer
    - Buzzer is fired at the start of a game

Software:

```
#include <hidef.h> /* common defines and macros */
#include <mc9s12dg256.h> /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12dg256b"
#include "main_asm.h" /* interface to the assembly module */

/*
Name: player7seg(int,int*)
Purpose: This function displays the current and flips RGB LED to white
 player number on the  7 Segment Display
Inputs: int z which is the current player, int* pointer
 to array that contains the 7 Segment byte
Outputs: n/a
*/
void player7seg(int z,int *arr) {
 int x;
 int w;
 if (z == 1) {
  for (x = 0;x < 250;x++) {
   for (w = 0;w < 2;w++) {
     seg7_on(arr[w],w);
     ms_delay(5);
   }
  }
  PTP = PTP & 0b0000000;
  seg7s_off();
 }
 else {
  for (x = 0;x < 250;x++) {
   for (w = 2;w < 4;w++) {
     seg7_on(arr[w],w);
     ms_delay(5);
   }
  }
  PTP = PTP & 0b0000000;
  seg7s_off();
 }
}

/*
Name: dispBoard(char[][3])
Purpose: This function displays the
 contents of a 3x3 array on the LCD Panel
Inputs: char 3x3 matrix array containing the current board info
Outputs: n/a
```

```
*/
void dispBoard(char arr[][3]) {
  int x = 0;
  clear_lcd();
  for (x = 0;x < 3;x++) {
   data8(arr[0][x]);
   ms_delay(50);
  }
  set_lcd_addr(40);

  for (x = 0;x < 3;x++) {
   data8(arr[1][x]);
  }
  ms_delay(2000);
  clear_lcd();
  for (x = 0;x < 3;x++) {
   data8(arr[1][x]);
  }
  set_lcd_addr(40);
  for (x = 0;x < 3;x++) {
   data8(arr[2][x]);
  }
  ms_delay(2000);
  clear_lcd();
}

/*
Name: takeInput(int,char[][3])
Purpose: This function takes the input of the current player
 and maps it onto the equivalent space in the 3x3 matrix
Inputs: int pnum which is the current player number and char arr
 which is a 3x3 matrix containing the current board information
Outputs: n/a
*/
void takeInput(int pnum, char arr[][3]) {
 char *instr[] = {"Enter Key","1 to 9"};
 char *valid[] = {"Key Valid"};
 char *failed[] = {"Wrong Key Use","Keypad 1-9"};
 char letter;
 int pInput, counter, marker, x, y;
 int pArr[] = {1,2,3,4,5,6,7,8,9,0};
 clear_lcd();
 type_lcd(instr[0]);
 set_lcd_addr(40);
 type_lcd(instr[1]);
```

```
 if (pnum == 1) {
  letter = 0x58;
 }
 else {
  letter = 0x5A;
 }
 bigOOF:
 pInput = getkey();
 wait_keyup();
 clear_lcd();

 counter = 0;
 marker = pArr[0];
 while (marker != 0) {
  x = counter / 3;
  y = counter % 3;
  if ((pInput == pArr[counter]) && (arr[x][y] == 0x30)) {
   arr[x][y] = letter;
   type_lcd(valid[0]);
   ms_delay(2000);
   clear_lcd();
   break;
  }
  counter += 1;
  marker = pArr[counter];
 }
 if (marker == 0) {
  type_lcd(failed[0]);
  set_lcd_addr(40);
  type_lcd(failed[1]);
  ms_delay(2000);
  goto bigOOF;
 }
}

/*
Name: cForWin(int,char[][3])
Purpose: This function takes the current player information
 and current board information to determine if a winner has been detected
Inputs: int player containing current player information and char arr
 which is a 3x3 matrix containing the current board information
Outputs: int which returns 1 if a match has been detected
*/
int cForWin(int player, char arr[][3]) {
 int i = 0;
 int x = 0;
```

```
 if (player == 1) {
  for (i = 0;i < 3;i++) {
   if ((arr[i][0] == 0x58) && (arr[i][1] == 0x58) && (arr[i][2] == 0x58)) {
    x = 1;
    goto checked;
   }
   if ((arr[0][i] == 0x58) && (arr[1][i] == 0x58) && (arr[2][i] == 0x58)) {
    x = 1;
    goto checked;
   }
  }
  if ((arr[0][0] == 0x58) && (arr[1][1] == 0x58) && (arr[2][2] == 0x58)) {
   x = 1;
   goto checked;
  }
  if ((arr[0][2] == 0x58) && (arr[1][1] == 0x58) && (arr[2][0] == 0x58)) {
   x = 1;
   goto checked;
  }
 }
 else {
  for (i = 0;i < 3;i++) {
   if ((arr[i][0] == 0x5A) && (arr[i][1] == 0x5A) && (arr[i][2] == 0x5A)) {
    x = 1;
    goto checked;
   }
   if ((arr[0][i] == 0x5A) && (arr[1][i] == 0x5A) && (arr[2][i] == 0x5A)) {
    x = 1;
    goto checked;
   }
  }
  if ((arr[0][0] == 0x5A) && (arr[1][1] == 0x5A) && (arr[2][2] == 0x5A)) {
   x = 1;
   goto checked;
  }
  if ((arr[0][2] == 0x5A) && (arr[1][1] == 0x5A) && (arr[2][0] == 0x5A)) {
   x = 1;
   goto checked;
  }
 }
 checked:
 return x;

}

//Main Function and Variable Initialization
```

```
void main (){
    int player,vic,yaboi,x,y,z;
    char tacArr[3][3];
    //PWM_Init(120,0,0,60);
    char *intro[] = {"Welcome to","Tic-Tac-Toe","To Play Hit SW5","Use Keypad 1-9"};
    char *player1[] = {"P1 Wins!","SW5 Restart"};
    char *player2[] = {"P2 Wins!","SW5 Restart"};
    char *loss[] = {"Nobody Wins","Play Again"};
    int seg7P[] = {0x73,0x06,0x73,0x5B};
    _asm(cli);
    DDRH = DDRH | 0x00;
    PLL_init();
    lcd_init();
    led_disable();
    seg7_enable();
    keypad_enable();
    DDRT = DDRT | 0b00100000;
    hellofaYeet:


    for (x = 0;x < 3;x++) {
     for(y = 0;y < 3;y++) {
      tacArr[x][y] = 0x30;
     }
    }

    //Intro to Game
    for (x = 0;x < 3;x++){
       clear_lcd();
       type_lcd(intro[x]);
       set_lcd_addr(40);
       type_lcd(intro[x + 1]);
       ms_delay(2000);
      }
    //Sound Buzzer
    while(1) {
     if(PTH == 0b11111110) {
      for (x = 0;x < 250;x++) {
      PTT = PTT | 0x20;
      ms_delay(1);
      PTT = PTT & 0xDF;
      ms_delay(1);
      }
      for (x = 0;x< 125;x++) {
       PTT = PTT | 0x20;
       ms_delay(2);
```

```
   PTT = PTT & 0xDF;
   ms_delay(2);
  }
  break;
 }
}
//Playing Loop Begins Here
vic = 0;
yaboi = 0;
player = 1;
while(yaboi != 9){
 //P1 Process and Supporting Functions
 if (player == 1) {
  player7seg(player,seg7P);
  dispBoard(tacArr);
  takeInput(player,tacArr);
  yaboi += 1;
  vic = cForWin(player,tacArr);
  if (vic) {
   goto biggestofYeets;
  }
  player = 2;
 }
 //P2 Process and Supporting Functions
 else {
  player7seg(player,seg7P);
  dispBoard(tacArr);
  takeInput(player,tacArr);
  yaboi += 1;
  vic = cForWin(player,tacArr);
  if (vic) {
   goto biggestofYeets;
  }
  player = 1;
 }
}

//Detects For a No Win Scenario
if (yaboi == 9) {
 type_lcd(loss[0]);
 set_lcd_addr(40);
 type_lcd(loss[1]);
 ms_delay(2000);
 clear_lcd();
 type_lcd(player1[1]);
 while (1) {
```

```
      if (PTH == 0b11111110) {
       goto hellofaYeet;
      }
     }
    }

    //Displays the Winner of Game
    biggestofYeets:
    if (player == 1) {
     type_lcd(player1[0]);
     set_lcd_addr(40);
     type_lcd(player1[1]);
     while (1) {
      if (PTH == 0b11111110) {
       goto hellofaYeet;
      }
     }
    }
    else {
     type_lcd(player2[0]);
     set_lcd_addr(40);
     type_lcd(player2[1]);
     while (1) {
      if (PTH == 0b11111110) {
       goto hellofaYeet;
      }
     }
    }
}
```

Analysis

Horizontal Win Testing:

X	X	X
0	0	0
0	0	0

Results:

Detects Win for both P1 or P2.

Vertical Win Testing:

X	0	0
X	0	0
X	0	0

Results:

Detects Win for both P1 or P2.

Diagonal Win Testing:

X	0	0
0	X	0
0	0	X

Results:

Detects Win for both P1 or P2.

Conclusion and Future Work

To conclude, the program produces a game of Tic-Tac-Toe and allows two players to go against each other. The code will either declare a winner or a tie and will then be able to start a new game. Future work with the code of Tic-Tac-Toe could be to expand the board. Also, other mini games could possibly be programmed to work on the microcontroller board. Tic-Tac-Toe is widely known and this code shows it can be done on the Dragon12 microcontroller using its applications.