EC331

Project Report

To:	Dr. Song
From:	Colin Hill and Peter Haugen
Date:	6/7/2004
Project:	Pic based Tic-Tac-Toe System

Introduction:

For our EC331 project we successfully designed and implemented a PIC based Tic-Tac-Toe game using the PIC16874.

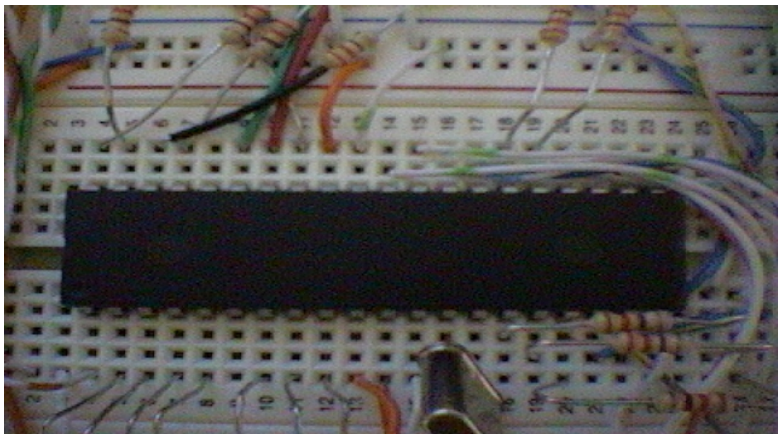

Figure 1: PIC16f784

Specifications:

The design needed to utilize the PIC16F874 due to the large number of I/O lines needed. The main display will consists of 9 Dual Color LEDs that will show game board status. There is an extra Dual Color LED to indicate whose turn it is. Game input is received from a membrane keypad.

Sub-System:	Number of I/O pins needed:
Main LED display	20
Keypad	8

Total:	28 I/O channels

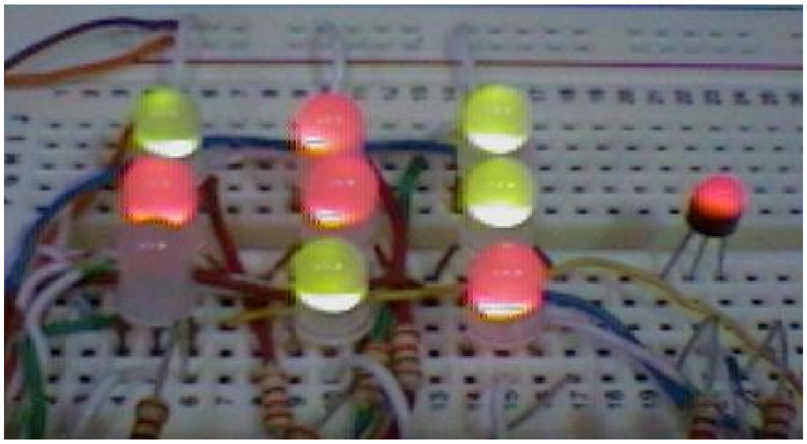

Figure 2: Main Game Display

Tasks and priorities completed:

To insure at a minimum partial success of our game, our project was completed in the following order.

1. Built main 9 LED game display and got it and the keypad working so two humans could play the game against each other.

2. Added the functionality of a simple computer player so a single user could play against the computer.

3. Optimized the code and computer player for speed and intelligence.

4. Added Computer vs. Computer mode.

5. Worked out any remaining bugs and simplified user interface.

Task completed:	**Completion date:**
Basic 9 LED display and keypad interface	End of Lab time on November 1st
Basic 2 player code written and debugged	End of Lab time on November 1st
Added computer player	November 3rd
Added computer vs. computer mode	November 3rd

Strategy:

The Dual Color LED display is connected to the PIC and the LEDs change color (Red/Green) depending on which player is occupying that space on the board. If a player presses a key that corresponds to a space on the board that is already occupied the LED in that position will turn Orange. The LEDs will extinguish if no player is occupying that space. User input is received via the following keys on the membrane keypad connected to the PIC.

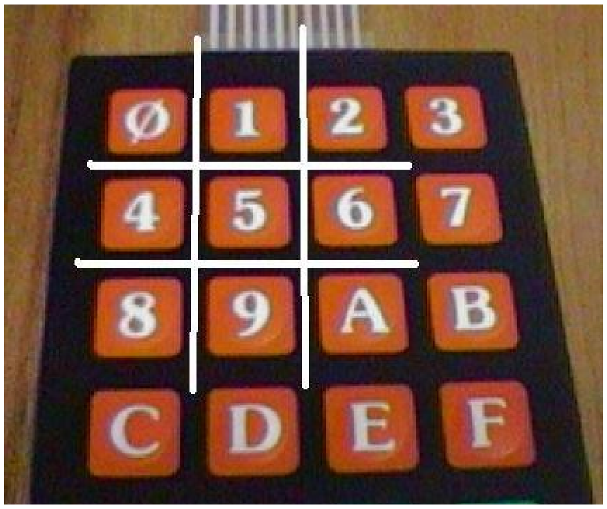

Figure 3: Position Keys

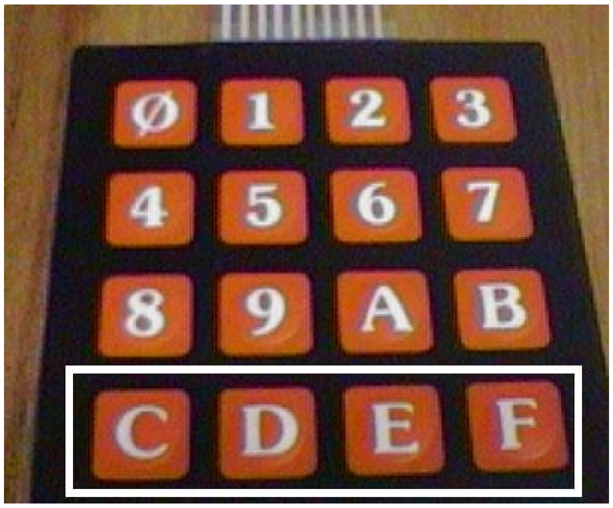

Figure 4: Game Selection Keys

These keys (0-A) are used because they form a square box on the keypad similar to the appearance of the Tic-Tac-Toe board. The only other keys available during game play are “C, D, E, F”. Depressing the “F” key will start a new game in the two-player mode in which one human player (Red) will play against another human player (Green). The turn indicator on the main display will show which player’s turn it is. Depressing the “E” key will start a new game in the one player mode in which one human player (Red) will play against the “computer” (Green) with the human (Red) getting the first move. Depressing the “D” key will start a new game in the one player mode in which one human player (Red) will play against the “computer” (Green) with the computer (Green) getting the first move. Finally, depressing the “C” key will start a new game in the Computer vs. Computer or demo mode in which the Pic will use it’s code to try and beat itself during game play. During any of the games if a player has connected three locations causing a win the game will flash the LEDs in the winning row, column, or diagonal to show that the player has won the game. Once the game has been won the only option is to press one of the new game election keys (C-F) and start a new game.

<u>Software Design:</u>

The software was written using a demo version of the High-Tech C compiler. This particular compiler was chosen because it was the only one with a demo/free version that worked for the PIC16F874.

The main program simply initializes variables and port directions before going into the main game loop. Its flow can be seen in Figure 1. The game loop consists of checking the game type and the turn indicator to determine whether the computer or human makes the next move. After a move, the board is checked for a win or a scratch game. If there is a win, the first detected winning combination is flashed on the board. If there is a scratch game, the entire board is flashed.

There are four main game modes. The first is computer versus computer. In this case, the main program loop will have the computer move for both red and green. The second is computer versus player. In this mode, computer will play green and the human will play red. The computer will move first if green moves first (This is selected with “D” on the keypad).

Checking for wins and scratches

In the program, there are three 9-bit variables that hold the state of the board. One of them stores where red has marks. Another stores where green has marks. The third stores whether or not a square has any mark in it.

To check for a win, the software uses an array of the eight winning bit-masks. It logical ANDs these masks with the board state of the last player to move. If all three bits are set, this is a win.

After checking for a win, the third bit-field that stores the state of the board regardless of color is checked to see if all squares are full. If they are, then the game is a scratch because a win would have been detected prior to this check.

How the computer moves

The computer player uses a common mini-max algorithm to generate its moves. This entails scoring the board for each possible move, and then scoring it again for each possible response move. For each move, the minimum response score is picked. It is then added to the score for the move alone. After all possible scores have been tallied, the maximum scoring move is made by the computer. The one exception is that the computer, if moving first, will randomize its move to make the game more interesting. This is because the algorithm thinks the middle square is always the best move.

The board is scored by examining the number of marks for each side in each winning row, column, and diagonal. If the current player has marks in a winning combination, and the other player does not, then 10^(number of current marks) is added to the score. If the other player has marks in a winning combination, and the other player does not, then 10^(number of other marks) is subtracted from the score. Note: this causes rows, columns, and diagonals with both color marks in them to not be scored, since they cannot count towards a win.

With this minimization of response and maximization of score algorithm, the computer will make semi-intelligent moves, like trying to win, and blocking the opposing player.

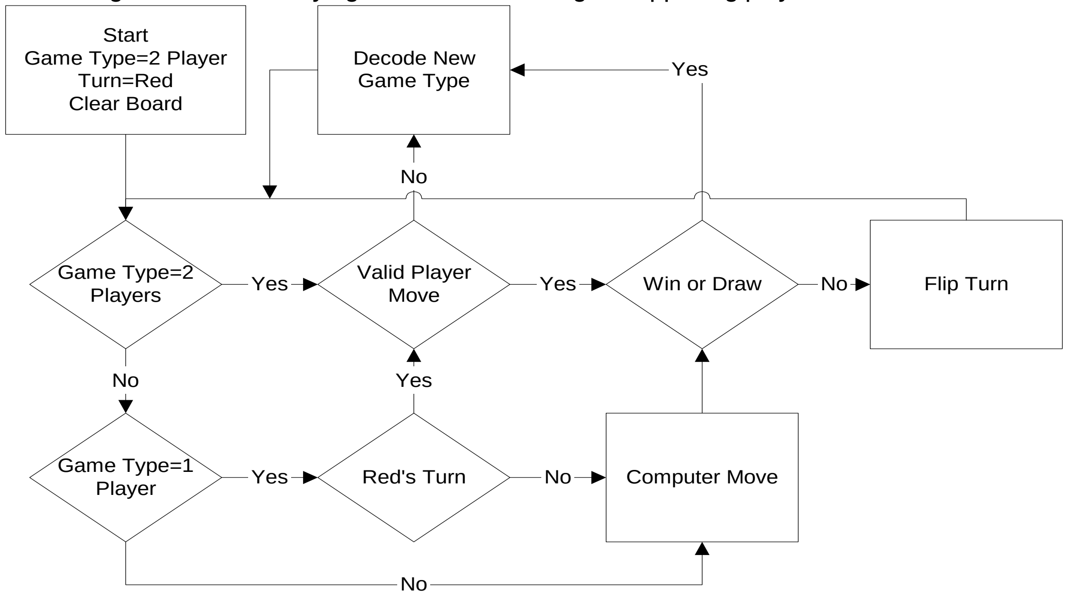

Figure 5: Main Program Flow

Budget and Final Design:

Item:	**Cost:**
PIC 16F874	Checked out from Dr. Song
Keypad	Checked out form Instrument Room
10 LED's	~$20
Large Breadboard	Already Have ~$24
Total new cost to us:	~$20

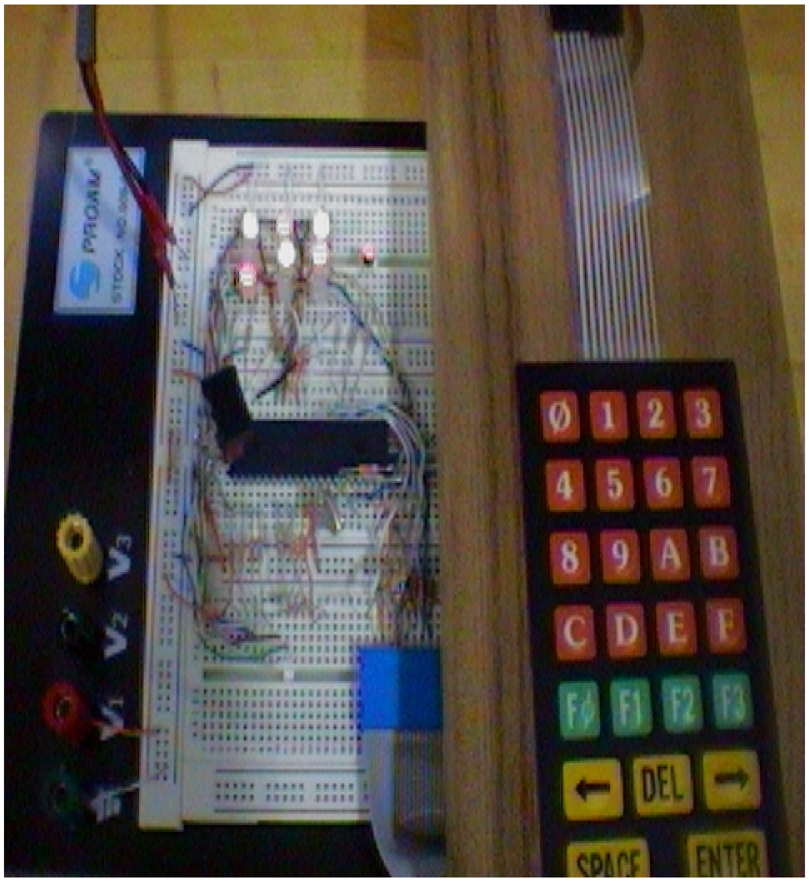

Figure 6: Final Design

Attachments: A) Schematic, B) Program Code