**Acknowledgement**

We would like to thank Professor Dhananjay V. Gadre for his guidance and support throughout this project. We would also like to thank the CEDT lab for providing us with facilities to fabricate our board and its members who provided us with valuable help throughout the fabrication process.

**Code Description**:

To implement tic-tac-toe, we have maintained a 3x3 board in RAM {BOARD} which provides various functions with the current state of the game. The game proceeds by alternatively calling functions corresponding to each player or a player and the AI depending on the mode selected. This process goes on until a terminal state is achieved, a terminal state being either victory or getting tied. Various menus have been constructed by using simple keypad input which uses polling to get user input. The LCD has been used to display various strings stored in the ROM depending on the situation. Two more sub-boards exist in RAM {TBOARD,BMAP} which store a transformed version of the main board state. These sub-boards have been used to update the LED matrix with the current state of the game.

Finally, the AI component of our project has been implemented using the Minimax algorithm. The AI functions determine the most optimal move under the given situation recursively and update the state of the internal board. AIMOVE is the overall function which mimics a MAX function to determine the move to play and actually modifies the board. The MAX function iterates over empty positions and calls the MIN function for each position and similarly the MIN function iterates over the empty positions and calls the MAX function for each position. Internally, the AI is always represented as ‘01H’ while the player is represented as ‘0FEH’. The head of both MAX and MIN functions contain a terminal state checker and information is moved between these functions using the accumulator. If a terminal state hasn’t been achieved MAX iterates over empty spots placing ‘01H’ and calls MIN to evaluate the result. MAX picks the place which gives the best result for ‘01H’. MIN works in a similar manner but looks for the case which gives the best result for ‘0FEH’. As soon as an optimal position is detected by either MIN or MAX it immediately exits returning the value.

Code Layout:

The code can broadly be divided into the following sections:

1. Decision making functions for the AI.

2. I/O Functions to take input from the keypad as well as display information on the LCD panel and LED matrix.

3. Game functions which have been used to construct Tic-Tac-Toe

4. Miscellaneous functions which include various delays as well as debugging functions

1) AI Internal Functions:

AIMOVE: Main AI function which makes the move for the AI on the board. Iterates over empty positions and makes calls to MIN to determine which move to make.

MIN: Function which mimics an optimally playing player by iterating over empty positions and making calls to MAX to determine the optimal position for the player. However, does not make changes to the board while doing so. Returns result in accumulator.

MAX: Function which determines the best move for the AI to make given that the calling MIN function made a move prior to it being called. It also iterates over the empty positions to determine the move.

NOTE: Both MIN and MAX have a state checker at the start to determine if the move played by the calling function causes a terminal state. In such a case no iteration is performed inside MIN/MAX and the terminal state result is returned by MIN/MAX.

2)I/O Functions:

**INPUT FUNCTIONS:**

KEYIN: Function which determines the input from the keypad connected to port C of the PPI. This function pulses each of the output lines (Active low logic pulse) one at a time and checks the input lines for a response. The input nibble is compared to various combinations representing a keypress. Depending on which pulse and the input nibble combination the row and column is found and thus the exact key is uniquely identified.

**OUTPUT FUNCTIONS:**

Liquid Crystal Display

LCDINIT: Function which initialises and configures the LCD into the 2-line mode.

LCDSTRINGDISP: Function which takes the memory location of a string and keeps printing out the string till a null character (00H) has been found. String address must be loaded in the HL register before calling this function.

LCDL2: Function which brings the LCD cursor to the next line and prints a string in the similar manner as LCDSTRINGDISP.

CMD: Function to send a command byte to the LCD.

DATA: Function to send a data byte to the LCD.

LED Matrix

UPDATE2: Function which makes a call to TXBOARD to transform the internal board and then uses ‘TBOARD’ to create another sub-board ‘BMAP’ whose values are taken to display the board state on the LED matrix which is connected to the PPI.

TXBOARD: Function which transforms internal game board ‘BOARD’ into ‘TBOARD’. ‘TBOARD’ contains either an inverted version or the same version as the ‘BOARD’ depending on whether the COLOURFLAG has been set or not.

SETWPOS: Loads three RAM locations beginning at label ‘WPOS’ with the ‘WINLIST’ row of the winning combination. This is used as a reference to blink the winning combination at the end.

BLINKRSD: Function which blinks the winning combination stored at ‘WPOS’ once and it is utilised in the functions {DUMMYPLAYERWIN, DUMMYPLAYER1WIN, DUMMYPLAYER2WIN, DUMMYAIWIN} to blink repeatedly.

3) Game functions:

CWIN: Function which evaluates the 3x3 board ‘BOARD’ and determines if someone has won. It returns ‘01H’ if ‘01H’ wins, returns ‘0FEH’ if ‘0FEH’ wins and returns ‘00H’ if no one wins. The values are returned through the accumulator. It uses the ‘WINLIST’ to jump to all possible winning combinations.

CFULL: Function which determines if the 3x3 board ‘BOARD’ is full. It returns ‘00H’ if the board is full and ‘01H’ otherwise through the accumulator.

PLAYERMOVE: Function used in AI vs Player games which takes input from the player about which position to play at using the keypad. The numerical portion of the keypad has a direct 1:1 mapping with the board positions.

AIMOVE: As described above this function plays a move according to the minimax algorithm.

4) Miscellaneous functions:

INIT\_RAM: Function which initialises various RAM locations at the start of each game.

DELAY 10/30/100MS: Functions to implement various delays in milliseconds.

SETSOD: Function which sets the SOD led.

RESETSOD: Function which resets the SOD led.

BLINKSOD: Function which blinks the SOD led.

;-------------------------------------------------------------------------------------------------------------------------;  
MEMORY MAP:

8200-8208H: ‘BOARD’

8100-8108H: ‘BMAP’

8400-8408H: ‘TBOARD’

8300H: ‘COLOURFLAG’

LOOKUP TABLE USED:

‘WINLIST’ : Contains offsets to various winning combinations which have been stored in a manner that adding successive offsets gives the actual offset.