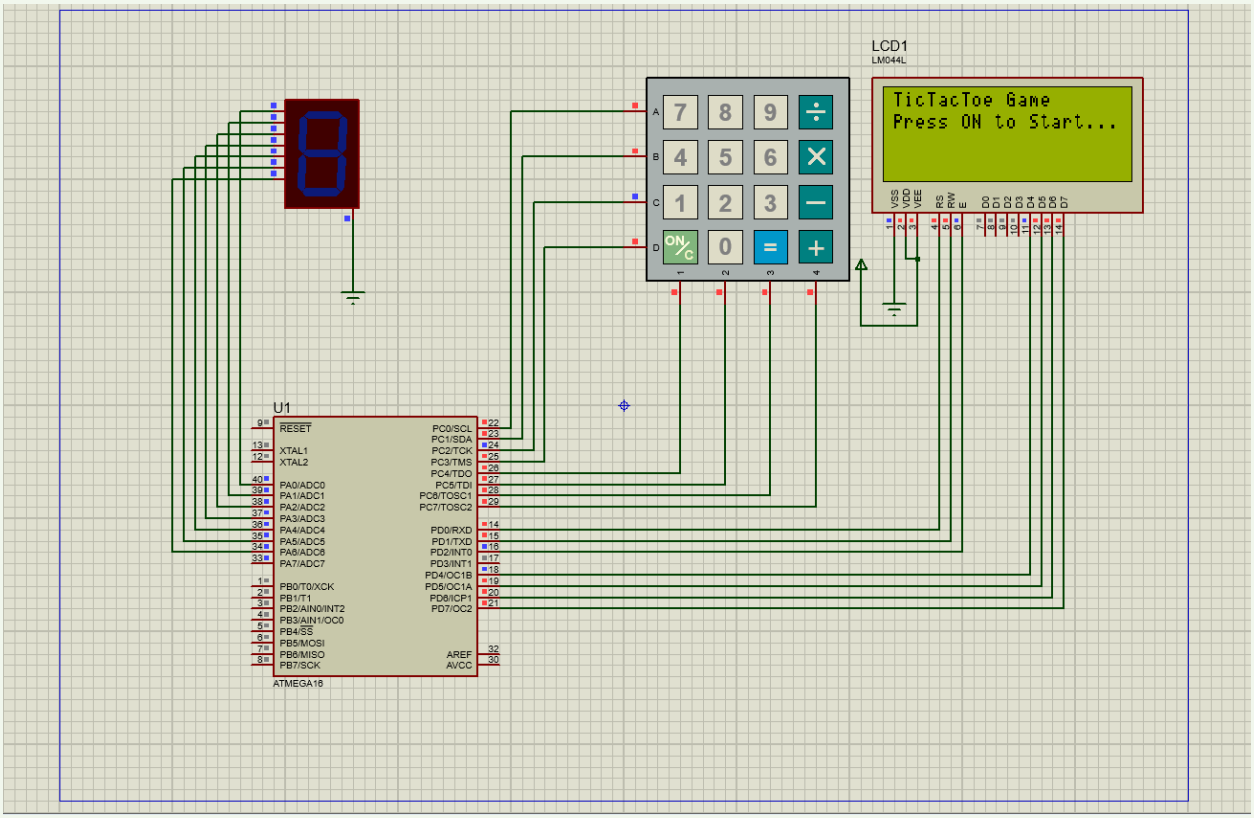
Tic-tac-toe game design project in proteus and code vision

Course: Microprocessor and assembly

Professor: Alireza Haji Eskandar

Servanaz Moeini - Farhad Norouzzadeh

Dooz, XO or tic-tac-toe game is an old but interesting game that here we have designed it by using ATmega16 microcontroller.

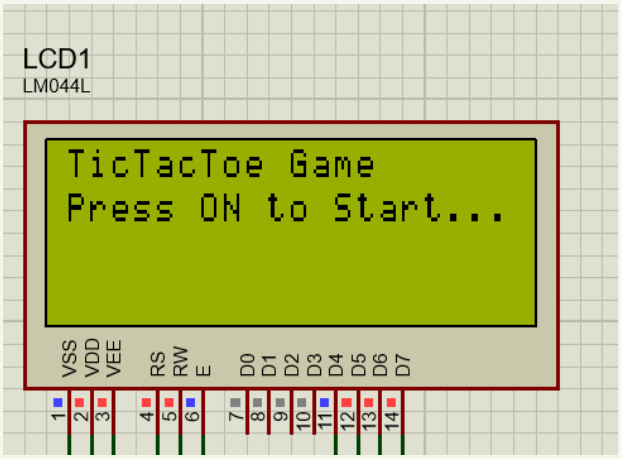


Components used in this project

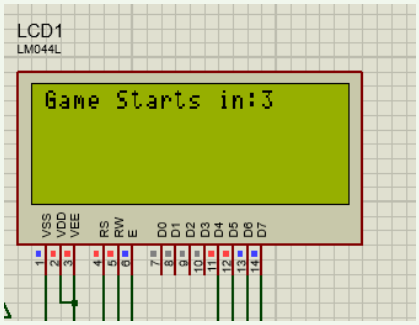
To design this game, we have used an ATmega 16 microcontroller, a calculator keypad, a seven segment, as well as the LMO44L screen, which has dimensions of 20x4.

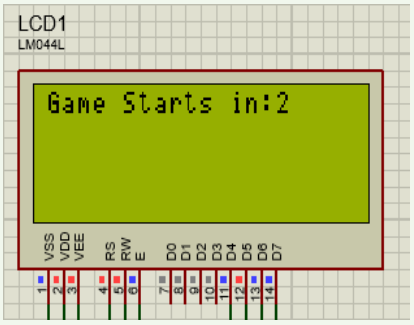
Game performance

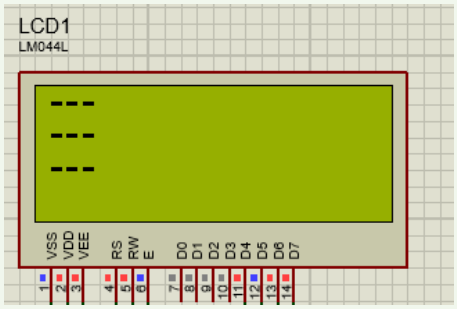
When starting the game, we have to press the ON key from the keyboard to allow us to play. Please note that other keys will not work to start the game, and it is only possible to start the game with the ON key.



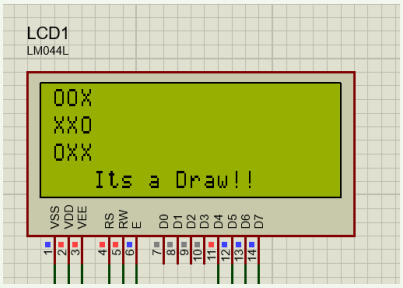
Then we come across the message "Game Starts In:" which is the countdown to start the game and after 3 seconds we enter the game page.

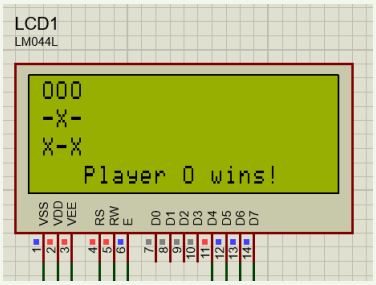


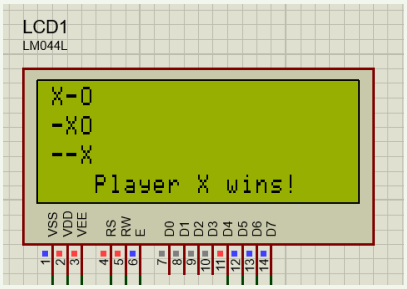




In this mode, the game starts. And two users "one X and one O" each by pressing the keys 1 to 9 can fill the game houses with X or O. And if each player succeeds in filling 3 houses in rows, columns or diagonals, they win the game.







Codes of the game

For the playing field, we have considered a 2-dimensional array called Game, in each house of which we have considered the values (+1) for the first player, (-1) for the second player, and (0) for the empty state of the house.

Note: The default value for houses is zero or empty.

For winning modes, an 8-element array called Win is defined, which includes 3 modes for row win, 3 modes for column win, and 2 modes for diagonal win.

Other global variables that are defined are:

Player: To determine the turn of the players

Count: counting the number of filled houses

State: To show the end of the game and its winner or the state of draw

If the value of State is equal to 1, it means that the first player has won, and if its value is -1 It means the second player is the winner. The value 0 means the game is in progress. Also, the value 2 means equal.

Txt: An array that holds the string “Game Starts In”

Row: Determines the value of "0 or 1" pull-up value for the desired keyboard.

```
// global variable
int game[3][3];
int win[8];
int player, count, state, r, c, k;
unsigned char row[4]={0xFE, 0xFD, 0xFB, 0xF7};
char txt[20];
unsigned char num[2]={0x06, 0x5B};
```

init() function:

This function is for initializing games global variables such as Game and Win

Also, the task of clearing the screen and seconds counter "to start the game" is also defined and called in this function.

Additionally, this function changes the value of the seven segments number to 1 "meaning it's player #1's turn".

```
void init()
{
   player=0;
   count=0;
   state=0;
   strcpy(txt, "");
   for(r=0; r<3; r++)
      for(c=0; c<3; c++)
         game[r][c]=0;                    // initialing game array
   for(r=0; r<8; r++)
      win[r]=0;                          // initialing win array
   r=0;
   c=0;
   lcd_clear();
   lcd_puts("TicTacToe Game\nPress ON to Start...");
```

```
    do{
       keypad();
       k=r * 4 + c;
    } while(k != 12);
    lcd_clear();
    for(r=3; r>0; r--)
    {
       sprintf(txt, "Game Starts in:%d", r);
       lcd_puts(txt);
       delay_ms(100);
       lcd_clear();
    }
    PORTA=num[player];
}
```

keypad() function:

This function, which is also used in all of the functions in the game, has the task of identifying pressed keys.

The main function of this method is that it constantly changes the pull-ups of the keyboard rows "in order from top to bottom". And “while loops” at the end of this function are written so that if the user holds

down the key for a while, the game will remain at the same stage until the user releases the key.

```
void keypad()
{
   while(1)
   {
      for(r=0; r<4; r++)
      {
         c=4;
         PORTC=row[r];
         DDRC=0x0F;
         if(PINC.4 == 0)
            c=0;
         if(PINC.5 == 0)
            c=1;
         if(PINC.6 == 0)
            c=2;
         if(PINC.7 == 0)
            c=3;
         if(!(c == 4))
         {
            while(PINC.4 == 0);
```

```
            while(PINC.5 == 0);
            while(PINC.6 == 0);
            while(PINC.7 == 0);
            return;
          }
          delay_ms(5);
       }
    }
}
```

showBoard() function:

This function is responsible for displaying the Board or playing field, as well as filling it. So that if the user presses the first key, that row and column of the playing field "2D array Game" will be filled with the number +1 and the value Print "X" on the playing field. And if the second user presses the key, puts the value of -1 in the same row and column of the Game

Also it prints the value "O" in the field.

The parts of the game that have a value of zero are also filled with "-", which means that the cell is empty and each user can choose that cell when it’s their turn.

```
void showBoard()
{
   lcd_clear();
   for(r=0; r<3; r++)
      for(c=0; c<3; c++)
      {
         lcd_gotoxy(c, r);
         if(game[r][c] == 1)
            lcd_putchar('X');
         else if(game[r][c] == -1)
            lcd_putchar('O');
         else
            lcd_putchar('-');
      }
}
```

WinnerCheck() function:

In this function we fill the win array. This array has 8 cells, 3 of which are related to the row board, 3 cells are related to the column board, and the other two cells are related to the diagonal board. If the value of each element of the win array is equal to 3, it means that the first player has won the game, and if its value is equal to -3, it means that the second player has won, and if it is a value other than these two values, then no one has won the game.

```
void winnerCheck()
{
    for (r = 0; r < 8; r++)
        win[r] = 0;
    for (r = 0; r < 3; r++)
        for (c = 0; c < 3; c++)
        {
            win[r] += game[r][c];
            win[r + 3] += game[c][r];
            if (r == c)
                win[6] += game[r][c];
            if (r + c == 2)
                win[7] += game[r][c];
        }}
```

main() function:

```
void main(void)
{
   PORTC = 0xFF; // initial value
   DDRC = 0x0F;  // input / output
   PORTA = 0x00; // initial value
   DDRA = 0xFF;  // output
   lcd_init(20);
   init();
   showBoard();
   while (state == 0)
   {
      keypad();
      if (r != 3 && c != 3)
      {
         if (game[r][c] == 0)
         {
            if (player == 0)
               game[r][c] = 1;
            else
               game[r][c] = -1;
            player = !player;
            PORTA = num[player];
```

```
            count++;
        }
    }
    k = r * 4 + c;
    if (k == 13)
        init();
    showBoard();
    winnerCheck();
    for (r = 0; r < 8; r++)
        if (win[r] == 3)
            state = 1;
        else if (win[r] == -3)
            state = -1;
    if (count == 9 && state == 0)
        state = 2;
}
lcd_gotoxy(3, 3);

if (state == 1)
    lcd_puts("player X wins !!");

else if (state == -1)
    lcd_puts("player O wins !!");
```

```
else if (state == 2){
   lcd_gotoxy(9,3);
   lcd_puts("Draw");
}

while (1)
{
   keypad();
   k = r * 4 + c;
   if (k == 13)
      init();
}}
```