Game using GLCD and numpad

Hello!

Group Members

Member 1 – Saqib Azim - 150070031

Member 2 – Saksham Tandon – 15D070003

Member 3 – Shreyas karande - 150070020

Member 4 – Aniruddh Kumar Yadiki - 150070056

Game Description

<u>Components used:</u>

1. Pt-51 board
2. 128*64 pixels Graphical LCD
3. Numpad
4. IC 7408 (for AND gates)
5. BreadBoard

<u>Gameplay</u>:

- 8*8 pixels sized blocks fall from the top of the screen at random.
- User controlled GUN which performs left or right motion and action of firing the bullet with the help of buttons on the numpad.
- The bullet which is continuous stream of 2*4 sized square boxes
- Numpad serves as the input medium for the user.

CODE IMPLEMENTATION

Interfacing of Numpad

CONFIGURING NUMPAD:

- 16 buttons which can take binary values(0 or 1) assembled in the form of a 4*4 square matrix.
- Equipped with a port consisting of 10 pins – 2 pins for connecting the supply(Vdd); 4 pins corresponding to 4 columns(c0,c1,c2,c3); 4 pins corresponding to 4 rows(r0,r1,r2,r3).
- Each of these 4 columns and 4 rows are internally connected to corresponding pins on the port.
- A specific "ROW" of buttons can be selected for operation by assigning a "HIGH" value to the pin corresponding to that particular row.
- In our project, we have chosen "ROW 2"(r2). The configuration of 'r2' for operation can be verified with the help of this image which shows a few lines of code(program) we wrote for the game.
- As can be seen in code, r3 pin is assigned an active low.This configures r3 for operation. To make other rows non-functional, we have connected the 3 pins corresponding to row 0, row1, row2 to Vdd line on the breadboard. ****(r3 in the image represents row2)****

```
25    sbit r3 = P0^3;
26    sbit c0 = P0^4;
27    sbit c1 = P0^5;
28    sbit c2 = P0^6;
29    sbit c3 = P0^7;
```

```
428   void main(){
429                   unsigned char page_number;
430                   r3 = 0;
431                   c0 = 1;
432                   c1 = 1;
433                   c2 = 1;
434                   c3 = 1;
435                   IEN0 = 0x81;
```

- The left and right movement of the gun and the firing of the bullet is controlled with the buttons of fuctional ROW3 (R3). ROW3 has 4 columns, i.e. 4 buttons.
- In our game, we have configured 3 of these 4 buttons as follows: c1,c2,c3 of r3 correspond to left movement of the gun, right movement of the gun, firing of the bullet respectively.
- When the user presses one of these 3 buttons, it serves as the input for our code and perfoms the corresponding action(left/right/fire).

➢ How does our code know that the button has been pressed?? How does it identify which button is the input and what action needs to be taken??

➢ How did we configure these 3 columns for their corresponding actions????

✓ We have used an EXTERNAL interrupt in the circuit and the program/code corresponding to it can be verified by the image of a part of our code on the next slide. We enabled an external interrupt in the main body by "IEN0 = 0× **80**".

✓ Whenever user presses one of those 3 buttons, the interrupt 0 is called which checks and identifies which column is operational at that moment.

✓ We used IC 7408 which consists of 4 AND gates.

✓ In the main body of our code, we have assigned a "HIGH" value to each of (c0,c1,c2,c3) and can be verified by part of our code on the last slide.

✓ Whenever user hits any button(c0,c1,c2,c3) of r3, the value corresponding to

Challenges in using interrupts:

- If the key was kept pressed, then the game movement stopped as the code was looping interrupt handler
- Therefore we are now checking the switches every time the block descends by 1 stage so that even if the switch is kept pressed it does not interfere with the game code

```
void numpad_Int(void) interrupt 0 {
        delay_ms(10);
        find_col();
        IE0_=0;
}

void find_col(){
        ACC = P0;
        ACC &= 0xf0;
        if(ACC != 0xF0){
                if(c0 == 0){
                        //P1 = 0x10;
                        numpad_key=0;
                }
                else if(c1 == 0){
                        //P1 = 0x20;
                        numpad_key=1;
                }
                else if(c2 == 0){
                        //P1 = 0x40;
                        numpad_key=2;
                }
                else if(c3 == 0){
                        //P1 = 0x80;
                        numpad_key=3;
                }
        }
        switch(numpad_key){
                case 0:
                        moveleft();
                        break;
                case 1:
                        moveright();
                        break;
                case 2:
                        //movebullet();
                        break;
                case 3:
                        break;
                default:
                        break;
        }
}
```

- After that, we perform an “AND” operation on the 4 values corresponding to 4 columns (c0,c1,c2,c3) of r3. If the final output is zero, our code identifies that the key has been pressed(OUTPUT CAN BE ZERO ONLY IF ATLEAST ONE OF c0,c1,c2,c3 BECOMES ZERO).
- To find which button has been pressed, we have used a variable named “numpad_key” in our code as can be seen in the image beside.
- Depending on the value “numpad_key” takes, the action of left/right movement or firing is performed by the core.


THANK YOU!!