CAR GAME

Using

LCD Interfacing with External Interrupt

By:

- B.Prashanth Yadav (20134164)
- B.Yaswanth (20134150)

Group: CS-A1

Abstract

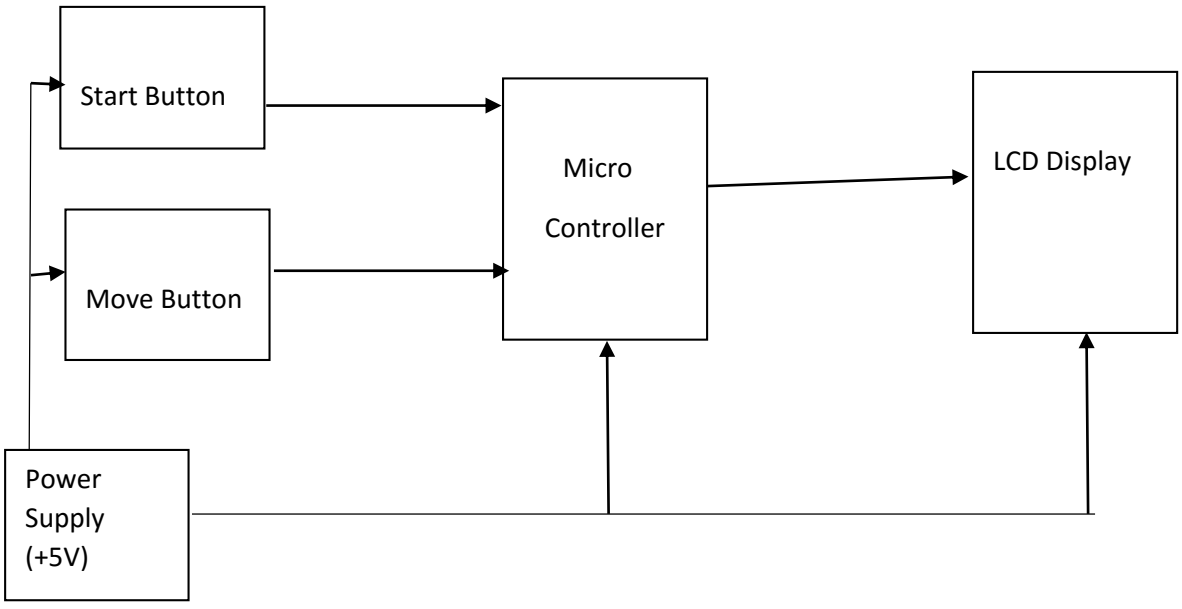
This project "*Car game on LCD Display*" is a game which is controlled with two buttons. One button is to start the game. There is a *car(X)* present at the left corner of the 16x2 LCD display. It can be moved up and down using the second button. A series of *obstacles(0)* and *points(1)* arrives the car starting from the right corner of the display. The player has to collect the points and escape the obstacles by moving the car up and down. For each point collected by the player, his score gets incremented by 1. The game ends when the player touches any obstacle and total score is displayed on the screen.

Technical Details

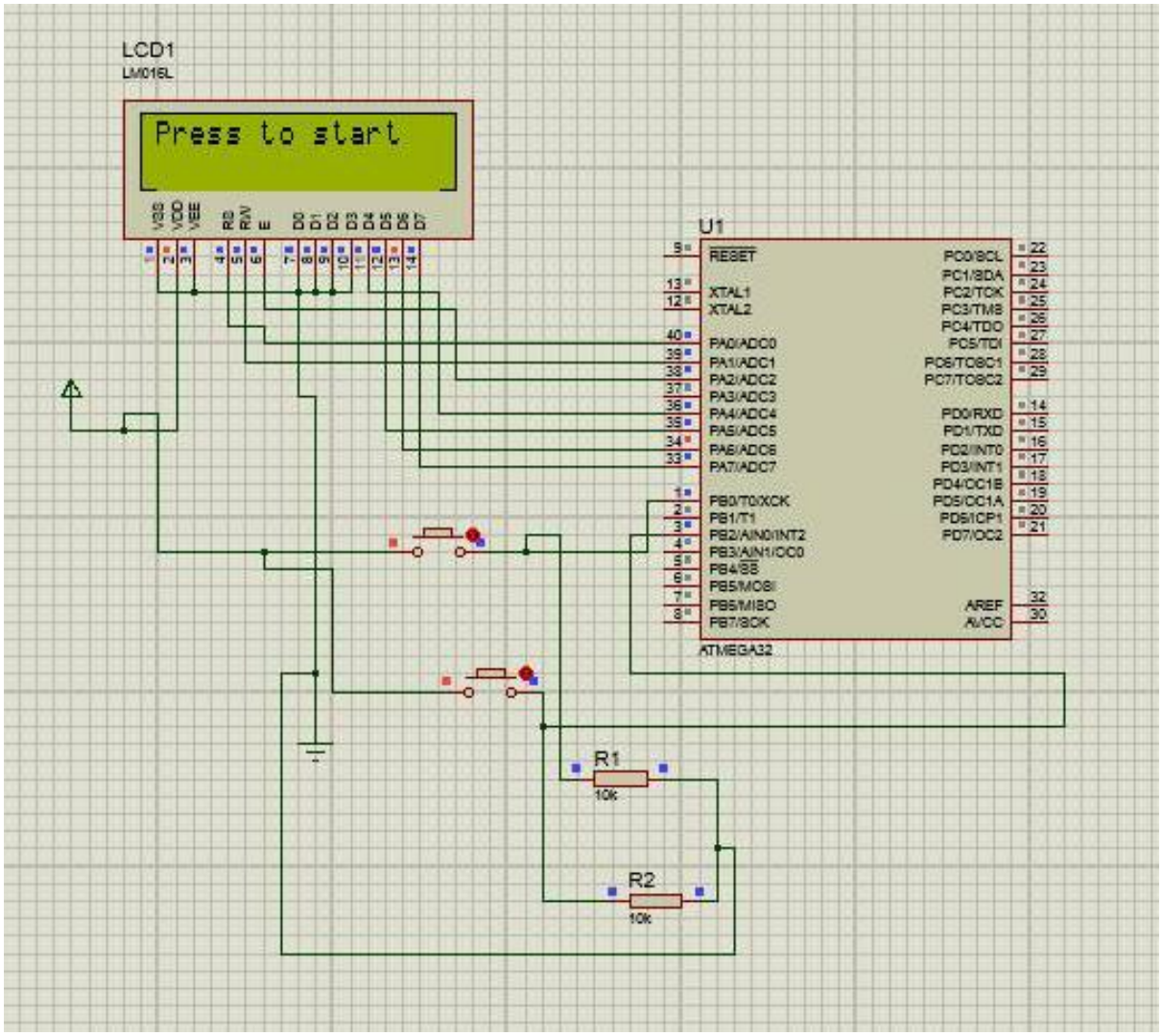
Circuit Components:

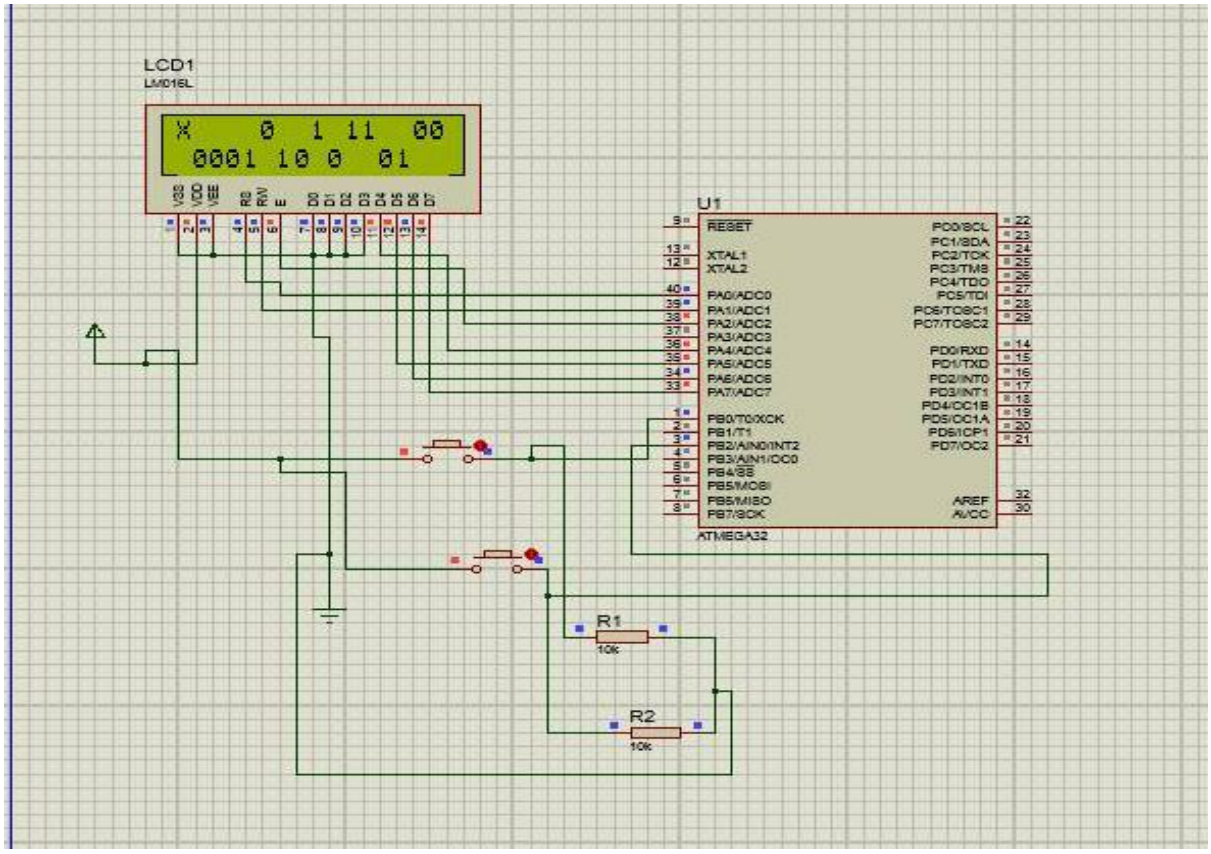
1. Atmega32 Micro-controller
2. 16x2 LCD Display
3. Buttons (2)
4. Bread board

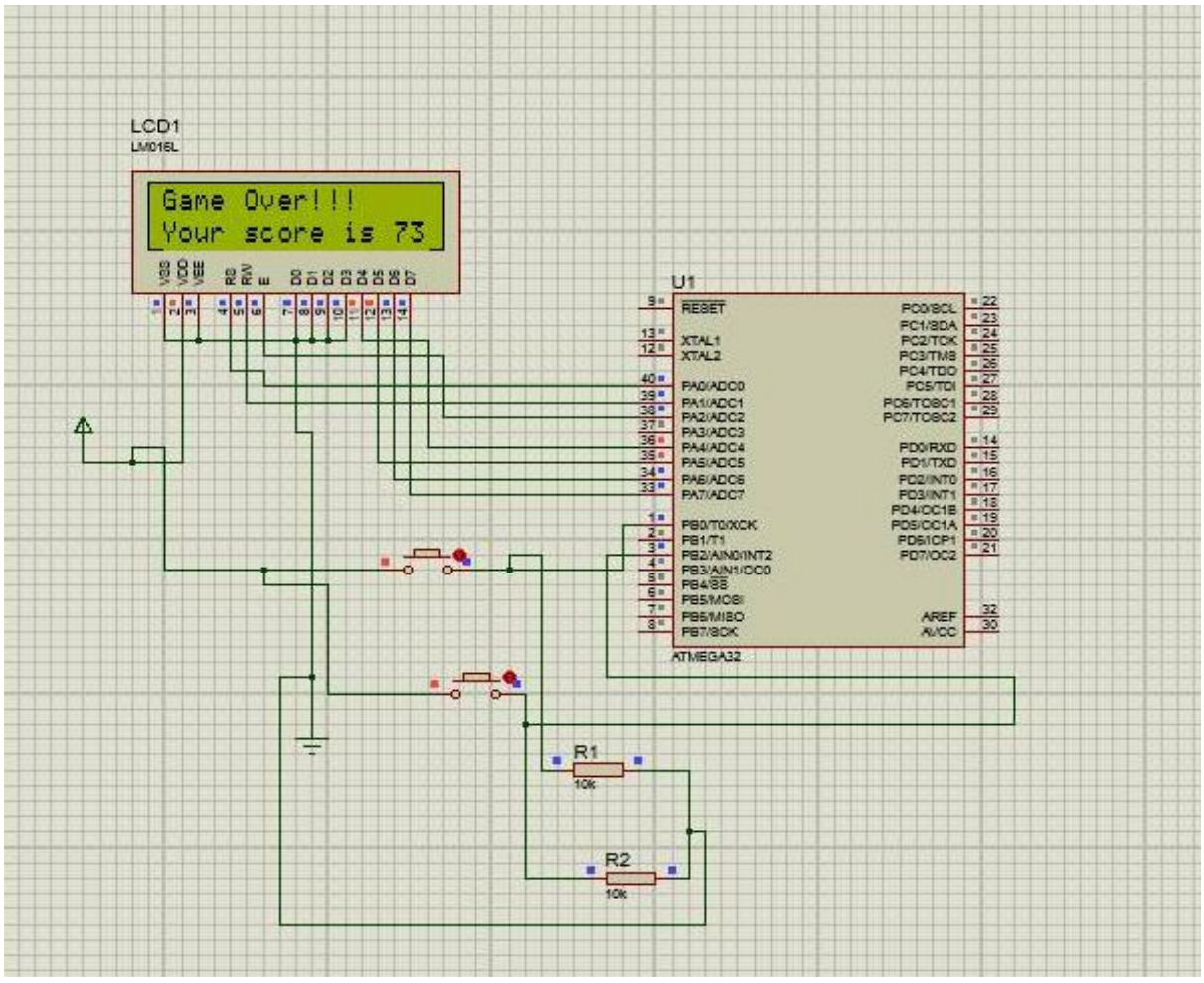
Block Diagram:



Circuit Diagram:

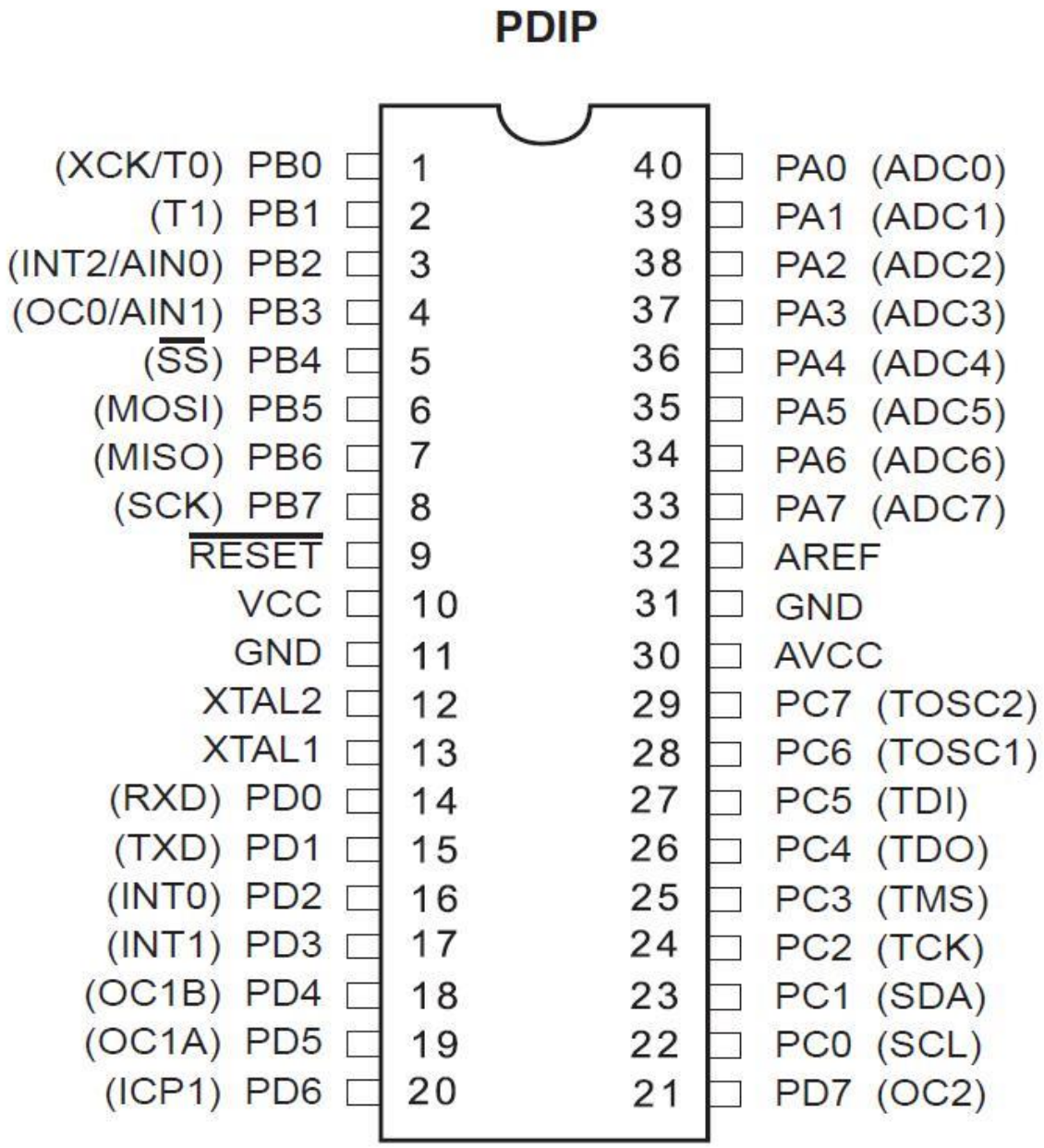



LCD1
LM016L
X 0 1 11 00
0001 10 0 01
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA32
R1
10k
R2
10k

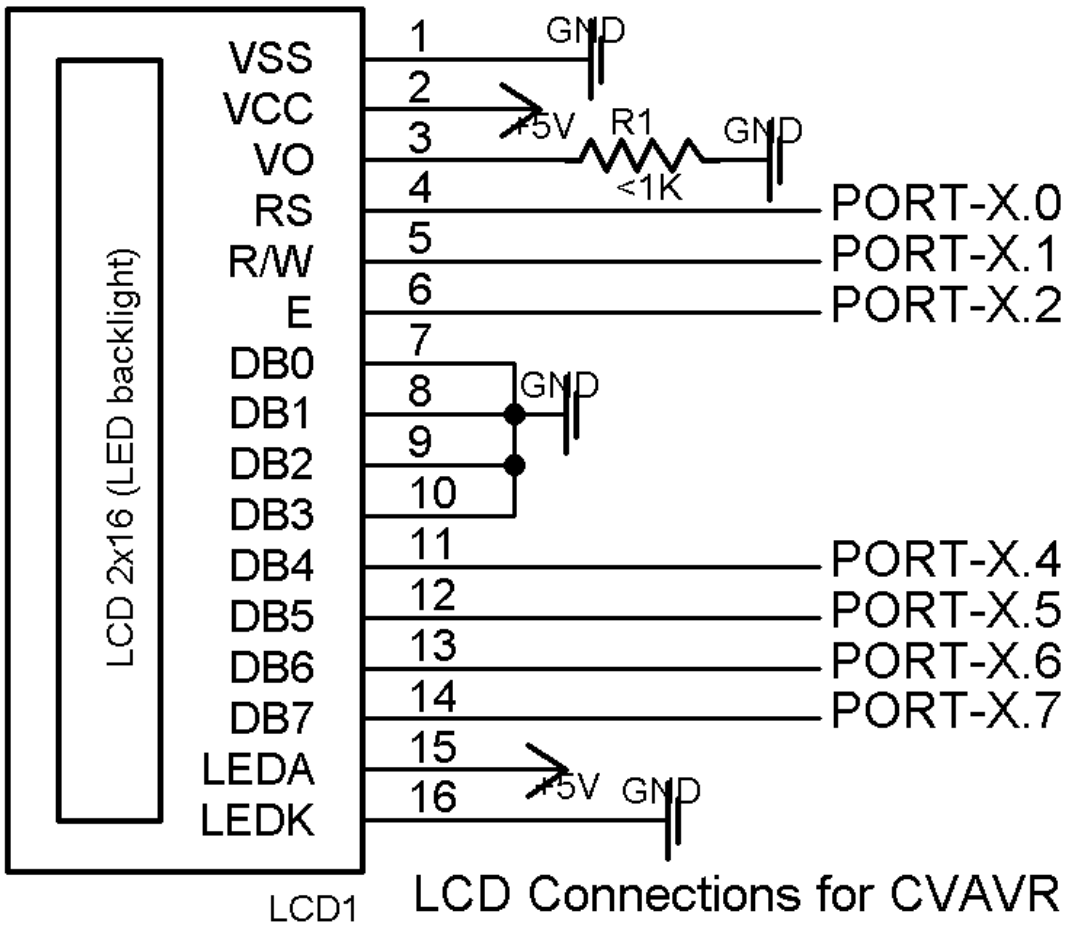

LCD1
LM016L
Game Over!!!
Your score is 73
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA32
R1
10k
R2
10k

Pin Configurations:

Atmega32:



LCD Display:



LCD Connections for CVAVR

Description:

The whole setup consists of a micro controller, LCD Display and two buttons to operate. Let us discuss the pin diagrams of these components.

PIN Configuration:

Atmega32 has 40 pins, which are divided into 4 ports i.e, PORTA, PORTB, PORTC, PORTD and some other pins including RESET, GND, VCC, AVCC. We discuss only the pins required for this project. Each PORTX has 8 pins from PORTX.0 to PORTX.7. These ports are used as I/O ports. There are also several other functions of those ports but are not required in the case of LCD Interfacing.

16x2 LCD display has 16 pins where 8 of them are PORTX's and the rest are either GND or VCC.

Each button is connected to a VCC and GND along with a 10K resistor.

Procedure:

- We select any port, let us say PORTA. The PORTA.y of Atmega32 are connected to PORTX.y of LCD respectively.
- And all connections of GND and VCC are made according to the PIN diagram for both Atmega32, LCD display and buttons.
- Now the functionalities should be fed into the micro controller. For this we use "*CVAVR".* It provides a GUI to describe the PIN behaviour of all the components and allows us to write the functionality code in high level languages like C, Java etc.
- This code is to be converted to machine code to be understood by the Micro controller.
- This also can be done by CVAVR . It converts our ".c" file into ".hex" file, which can be fed into micro controller.
- To make a connection between your computer and micro controller we need a device called Programmer.
- The GND and VCC can also be provided by Programmer or if already the program is fed into the controller, we can supply power using an adapter.
- The connections are successfully completed now, and the game can be played.

How to play:

- Now after providing the power, the LCD glows with the Start screen. Press start button to start the game. Then the player can be moved using the other button.
- The other button to move is the Interrupt button. It sends interrupt to the process printing the 0, 1 array to the screen and moves the player, then the first process continues.
- When the player collides with an Obstacle represented by 0, the game is over and if he gets collided with 1, he gets a point.
- The code representing the game is below.

Code:

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Professional
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Car game with lcd display
Version :
Date    : 01-11-2015
Author  : B.Prashanth (20134164) and B.Yaswanth (20134150)
Company :
Comments:


Chip type               : ATmega32
Program type            : Application
AVR Core Clock frequency: 8.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 512
*****************************************************/

#include <mega32.h>
#include <delay.h>
#include <stdlib.h>
// Alphanumeric LCD Module functions
#include <alcd.h>
int s,a[16][2],y,i,j,z,counter,score,speed=150;
char c[10];
```

```
void main(void);
// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
// Place your code here

}

// External Interrupt 2 service routine
interrupt [EXT_INT2] void ext_int2_isr(void)
{
    s++;
  delay_ms(100);
  if(s==2)
    s=0;
}

// Declare your global variables here
void init()
{
    for(i=1;i<16;i++)
    for(j=0;j<=1;j++)
        a[i][j]=0;
    s=y=i=j=z=counter=score=0;
    speed=150;
}
void player(void)
{
      lcd_gotoxy(0,s);
      lcd_putchar('X');
      lcd_gotoxy(0,1-s);
      lcd_putchar(' ');
}
void finish()
```

```
{
    lcd_clear();
    lcd_puts("Game Over!!!");
    lcd_gotoxy(0,1);
    itoa(score,c);
    lcd_puts("Your score is ");
    lcd_puts(c);
    while(1)
    {
        if(PINB.0==1)
            {
                delay_ms(1000);
                init();
                main();
            }
    }
}
// Declare your global variables here
void obstacle(void)
{
   if(a[1][s]==1)
   {
        finish();
   }
   if(a[1][1-s]==2)
   {
        finish();
   }
   if((a[1][0]==2) || (a[1][1]==2)){
    score++;
    if(score>1)
          if(score%2==0)
                speed--;
   }
```

```
    z++;
    z=(z+rand()%2)%2;
    y=(rand()+z)%4;
    if(y==1)
    {
        a[15][z]=1;
        a[15][1-z]=0;
    }
    if(y==0)
    {
        a[15][z]=0;
        a[15][1-z]=1;
    }
    if(y==3)
    {
        a[15][z]=0;
        a[15][1-z]=2;
    }
    if(y==2)
    {
        a[15][z]=2;
        a[15][1-z]=0;
    }

    for(i=1;i<15;i++)
    {
         a[i][0]=a[i+1][0];
         a[i][1]=a[i+1][1];
    }
}
void print(void)
{
    for(i=15;i>=1;i--){
        for(j=0;j<2;j++){
```

```
            lcd_gotoxy(i,j);
            if(a[i][j]==1){
                lcd_putchar('0');
            }
            else if(a[i][j]==2)
                lcd_putchar('1');

            else
                lcd_putchar(' ');
        }
    }
}

void main(void)
{
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;
```

```
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=0xFF
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer1 Stopped
// Mode: Normal top=0xFFFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
```

```
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: On
// INT1 Mode: Falling Edge
// INT2: On
// INT2 Mode: Rising Edge
GICR|=0xA0;
MCUCR=0x08;
MCUCSR=0x40;
GIFR=0xA0;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// USART disabled
UCSRB=0x00;

// Analog Comparator initialization
```

```
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// Alphanumeric LCD initialization
// Connections specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTA Bit 0
// RD - PORTA Bit 1
// EN - PORTA Bit 2
// D4 - PORTA Bit 4
// D5 - PORTA Bit 5
// D6 - PORTA Bit 6
// D7 - PORTA Bit 7
// Characters/line: 8
lcd_init(16);

// Global enable interrupts
#asm("sei")
 lcd_puts("Press to start");
while (1)
```

```
      {
                  if(PINB.0==1)
    {
        delay_ms(50);
        while (1)
        {
            counter++;
            print();
            player();
            if(counter%speed==0)
                  {
                     obstacle();
                     counter=0;
                  }
        }
    }
  }
}
```