****



**FACULTAD INGENIERIA**

**de**

Ing. Sistemas Computacionales Hardware

**MICROPROCESADORES**

**Proyecto Final**

**Videojuego**

**ENRIQUE GERARDO HERNANDEZ VEGA**

**Elías Marín Chaparro 205043**

**Javier Grijalva Ortega 197558**

**Emmanuel Tovar Estrada 181167**

**OBJETIVO**

El objetivo del proyecto final, es poner en práctica todos los conocimientos obtenidos durante el curso de la materia de Microprocesadores. Principalmente el uso de teclado e interrupciones, usando una pantalla LCD 16x2, y programando en C.

**MARCO TEÓRICO**

La Pantalla LCD es uno de los periféricos más empleados para la presentación de mensajes, variables y casi cualquier información proveniente de un microcontrolador. Gracias a su flexibilidad, buena visibilidad y precio reducido se ha convertido en el estándar de visualización más utilizado con los microcontroladores.

Las pantallas LCD que incluyen un microcontrolador interno como el HITACHI 44780 o compatible pueden:

                 ·  Presentar caracteres ASCII y griegos.

                 ·  Recibir y mostrar caracteres personalizados.

                 ·  Desplazamiento de caracteres (Scroll) tanto a la derecha como a la izquierda.

                 ·  Cambiar el aspecto y el movimiento del cursor.

                 ·  Direccionar de manera simple la posición de visualización de los caracteres.

**TERMINALES DE CONEXIÓN:**

Los terminales de conexión de las pantallas LCD de caracteres han sido estandarizados, siendo generalmente compatibles pin a pin con lo mostrado en la tabla:

|  |  |  |  |
| --- | --- | --- | --- |
| **Terminal** | **Nombre** | **Función** | **Descripción** |
| **1** | Vss | Energía | Referencia 0 V. GND |
| **2** | Vdd | Energía | +5 V DC |
| **3** | Vee | Ajuste Contraste | Variable de 0 a 5 V |
| **4** | RS | Comando | Selección de Dato/Comando |
| **5** | R/W | Comando | Control de Lectura/Escritura |
| **6** | E | Comando | Habilitación |
| **7** | D0 | E/S | DATO LSB |
| **8** | D1 | E/S | DATO |
| **9** | D2 | E/S | DATO |
| **10** | D3 | E/S | DATO |
| **11** | D4 | E/S | DATO |
| **12** | D5 | E/S | DATO |
| **13** | D6 | E/S | DATO |
| **14** | D7 | E/S | DATO MSB |

**CGRAM** (Character Generator RAM): Memoria volátil  de 64 bytes que permite almacenar hasta 8 caracteres personalizados para ser mostrados en la pantalla.

**SCRRAM** (Display Data RAM): Memoria volátil de 80 bytes, en la cual, se almacenan los caracteres que se van a mostrar en la pantalla.

**CGROM** (Character Generator ROM): Memoria no volátil, en la cual, se almacenan los caracteres predefinidos que soportan la pantalla y su controlador.

[**CREACIÓN DE NUEVOS CARACTERES EN LCDS ALFANUMÉRICOS MEDIANTE CGRAM.**](http://www.zaxione.com/index.php?option=com_content&view=article&id=34:creacion-de-nuevos-caracteres-en-lcds-alfanumericos&catid=9:picbasic-pro&Itemid=4)

Estos LCDs se componen de una matriz de 5 píxeles horizontales por 7 verticales por carácter. En su memoria **CGRAM** puede almacenar hasta 8 nuevos caracteres.

En el ejemplo tenemos una matriz que nos ayudara a calcular los Bytes necesarios para crear un carácter o símbolo nuevo. Como ejemplo, en rojo están algunos puntos de la matriz simulando una flecha.

o o **o**  o o

o **o o o** o

**o** o **o**  o **o**

o o **o**  o o

o o **o** o o

o o **o** o o

o o o o o

Tenemos una numeración que se repite en cada fila horizontal.  
Esta numeración es necesaria para poder hacer el cálculo y pasar de píxeles a Bytes.

16 + 8 + **4** + 2 + 1 = **4**

16 + **8** + **4** + **2** + 1 = **14**

**16**  + 8 + **4** + 2 + **1** = **21**

16 + 8 + **4** + 2 + 1 = **4**

16 + 8 + **4** + 2 + 1 = **4**

16 + 8 + **4** + 2 + 1 = **4**

16 + 8 + 4 + 2 + 1 = **0**

Si sumamos los números marcados en rojo de cada fila horizontal tendremos 7 números decimales necesarios para generar un nuevo carácter.  
Cada número de cada línea se envía al LCD a una dirección de memoria para así formar nuestro nuevo carácter.

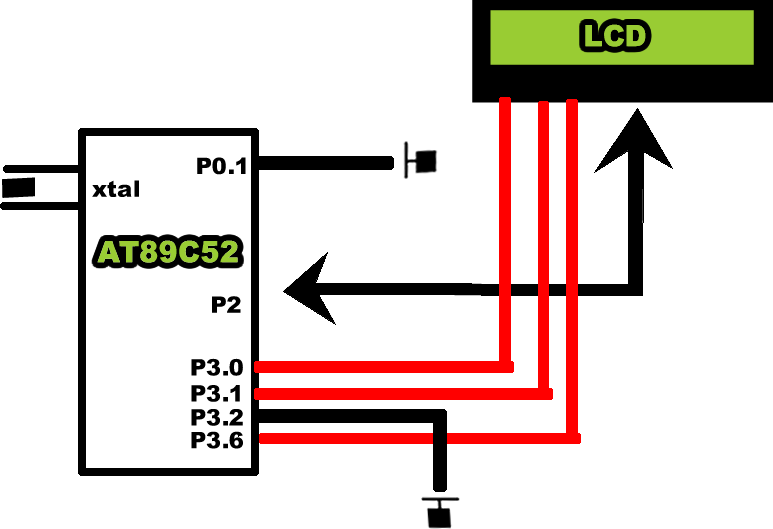
**DESARROLLO**

El videojuego consiste en disparar a un enemigo que está en movimiento parpadeando. El disparo se realizara mediante un push button conectado al puerto 0.1.

El envió de datos y comandos se realizara por el puerto 2 del microcontrolador. Otro push button estará conectado en el puerto 3.2 INT 0 para el manejo de interrupciones externas, este tendrá como fin manejar una pausa momentánea.

Rs, Rw y En estarán controlados mediante los pines de los puertos 3.1,3.2 y 3.6 respectivamente.

**DIAGRAMA BLOQUES**



**DIAGRAMA DE FLUJO**

Inicio

Inicializamos las variables a usar, los puertos P3.0, P3.1 y P3.6 se declaran como variables tipo sbit para Rs, Rw y En respectivamente.

Declaramos la función delay, se utilizara para retardos que permiten la visualización de los gráficos en el LCD.

Declaramos la función lcd\_comando la cual utilizaremos para enviar instrucciones al lcd.

Declaramos la función lcd\_dato, la cual enviara datos al LCD. Después declaramos la función lcd\_ini para inicializar el LCD.

Declaramos la función enemigo, esta se encargara del movimiento del grafico enemigo en el lcd.

Declaramos la función IntExt para el manejo de las interrupciones, tendrá la finalidad de meter una pequeña pausa en el juego.

Declaramos la función gráficos, en ella generaremos los caracteres personalizados mediante cgram. 8 bytes por cada carácter.

En la función principal activamos el uso de interrupciones mediante ITO,EA y EX0, para usar INT 0. Enviamos un 0 al puerto 0.1. donde se encuentra el push button.

Llamamos a las funciones lcd\_ini, y gráficos. Dentro de un ciclo infinito llamamos a la función enemigo.

Cuando se presione el push button, se hará un break para salir de la función enemigo, y seguir con un ciclo dentro del while del main.

En este ciclo se imprimirá el disparo, cuando las direcciones del disparo y del enemigo coincidan se imprimirá una explosión en el LCD.

Se regresa al ciclo infinito

**CONCLUSIONES**

Con la elaboración de este proyecto extendimos el conocimiento en el uso de la pantalla LCD específicamente en CGRAM. Nos permitió poner en práctica todo lo aprendido durante el curso.

**BIBLIOGRAFIA**

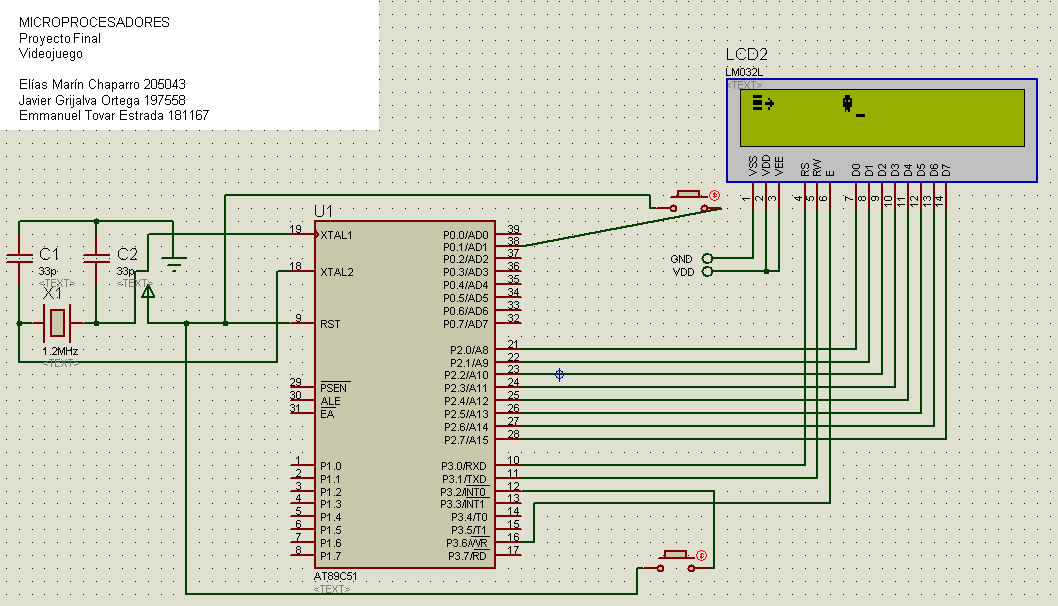
<http://www.zaxione.com/index.php?option=com_content&view=article&id=34:creacion-de-nuevos-caracteres-en-lcds-alfanumericos&catid=9:picbasic-pro&Itemid=4>

Tutorial LCD.pdf

Manual 8051.pdf

**ANEXOS**

**DIAGRAMA ELECTRICO**

****

**CÓDIGO**

**#include<reg51.h> //librería del microcontrolador AT89C51**

**sfr lcd\_P2=0xA0; //dirección puerto 2 en el micro**

**sbit rs=P3^0; //declaramos Rs en el puerto 3.0**

**sbit rw=P3^1; //declaramos Rw en el puerto 3.1**

**sbit en=P3^6; //declaramos en en el puerto 3.6**

**sbit puerto=P0^1; //el puerto 0.1 lo asignamos a la variable puerto**

**int e,dec;**

**dispdir=0x80; //valor de posición del disparo**

**void delay(unsigned int contador) //Función del retraso**

**{**

**int i,j;**

**for(i=0;i<contador;i++) //Ciclo para generar el retraso**

**for(j=0;j<1275;j++);**

**}**

**void lcd\_comando(unsigned char comm) //Función para enviar instrucciones al lcd**

**{**

**lcd\_P2=comm; //enviamos al P2 la instrucción**

**en=1; //Activamos el enabled**

**rs=0; // Ponemos el Rs en 0 para la instrucción**

**rw=0; //Rw en 0 para escritura**

**delay(1); //retraso**

**en=0; //Desactivamos el enabled**

**}**

**void lcd\_dato(unsigned char disp) //función para enviar datos al lcd**

**{**

**lcd\_P2=disp;**

**en=1; //Activamos el enabled**

**rs=1; // Ponemos el Rs en 1 para dato**

**rw=0; //Rw en 0 para escritura**

**delay(1); //retraso**

**en=0; //Desactivamos el enabled**

**}**

**void lcd\_ini() //función para inicializar el lcd**

**{**

**lcd\_comando(0x38); //Declaramos q sea de 8bits, 2 lineas , 5x7 puntos**

**delay(200); //retardo**

**lcd\_comando(0x0F); //cursor parpadeando**

**delay(200); //retardo**

**}**

**void enemigo() //función del enemigo**

**{**

**e=9; //variable para el for**

**dec=0x89; //posición inicial del enemigo en el lcd**

**dispdir=0x80; //posición inicial del disparo en el lcd**

**for(e;e>1;e--)**

**{**

**lcd\_comando(64); //posición de memoria del LCD donde está el enemigo**

**lcd\_comando(dec); //posicion del enemigo en el lcd**

**lcd\_dato(0); //identificador del conjunto de 8 bits**

**delay(5); //retardo**

**dec=dec-0x01; //decrementamos la posición del enemigo**

**if(puerto) //condición para verificar si se activo el push button**

**{**

**break;**

**}**

**lcd\_comando(72); //posición de memoria del LCD donde está el clear**

**lcd\_comando(dec+0x01); //posicion clear en el lcd**

**lcd\_dato(1); //identificador del conjunto de 8 bits**

**delay(5); //retardo**

**}**

**}**

**void IntExt()interrupt 0 //función de la interrupción**

**{**

**delay(5); //retardo**

**}**

**void graficos() //función de graficos**

**{**

**lcd\_comando(64); //posición de memoria del LCD donde está el enemigo**

**lcd\_dato(14); //línea 1 de la matriz de 5x7 puntos**

**lcd\_dato(31); //línea 2 de la matriz de 5x7 puntos**

**lcd\_dato(21); //línea 3 de la matriz de 5x7 puntos**

**lcd\_dato(31); //línea 4 de la matriz de 5x7 puntos**

**lcd\_dato(31); //línea 5 de la matriz de 5x7 puntos**

**lcd\_dato(14); //línea 6 de la matriz de 5x7 puntos**

**lcd\_dato(10); //línea 7 de la matriz de 5x7 puntos**

**lcd\_dato(0);**

**lcd\_comando(0x89); //posicion del enemigo en el lcd**

**lcd\_dato(0); //identificador del conjunto de 8 bits**

**lcd\_comando(72); //posición de memoria del LCD donde está el clear**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_comando(0x81); //posicion del clear en el lcd**

**lcd\_dato(1); //identificador del conjunto de 8 bits**

**lcd\_comando(80); //posición de memoria del LCD donde está el disparo**

**lcd\_dato(0);**

**lcd\_dato(4);**

**lcd\_dato(2);**

**lcd\_dato(31);**

**lcd\_dato(2);**

**lcd\_dato(4);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_comando(0x81); //posicion del disparo en el lcd**

**lcd\_dato(2); //identificador del conjunto de 8 bits**

**lcd\_comando(88); //posición de memoria del LCD donde está la nave**

**lcd\_dato(31);**

**lcd\_dato(0);**

**lcd\_dato(31);**

**lcd\_dato(31);**

**lcd\_dato(0);**

**lcd\_dato(31);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_comando(0x80); //posicion de la nave en el lcd**

**lcd\_dato(3); //identificador del conjunto de 8 bits**

**lcd\_comando(96); //posición de memoria del LCD donde está la explosion**

**lcd\_dato(0);**

**lcd\_dato(4);**

**lcd\_dato(14);**

**lcd\_dato(31);**

**lcd\_dato(14);**

**lcd\_dato(4);**

**lcd\_dato(0);**

**lcd\_dato(0);**

**lcd\_comando(0x90); //posicion de la explosión en el lcd**

**lcd\_dato(4); //identificador del conjunto de 8 bits**

**}**

**void main()**

**{**

**//interrupciones para INT 0**

**IT0=1;**

**EA=1;**

**EX0=1;**

**puerto=0; //ponemos en 0 el puerto 0.1**

**lcd\_ini(); //se inicializa el lcd**

**graficos(); //se cargan los gráficos**

**while(1){ //ciclo infinito**

**enemigo(); //corre la función del enemigo**

**if (puerto){ //verificacion para ver si se el puerto esta en 1 (activacion del push boton)**

**for(e;e>1;e--) //ciclo que mantiene los valores manejados en el ciclo**

**{ //for de la función del enemigo**

**dispdir=dispdir+0x01; //se incrementa la posición del disparo**

**lcd\_comando(80); //imprimimos el disparo**

**lcd\_comando(dispdir);**

**lcd\_dato(2);**

**delay(5);**

**lcd\_comando(72); //clear**

**lcd\_comando(dec+0x01);**

**lcd\_dato(1);**

**delay(5);**

**lcd\_comando(64); //imprimimos el enemigo**

**lcd\_comando(dec);**

**lcd\_dato(0);**

**delay(5);**

**if (dispdir==dec) //comparamos la posición del disparo y el enemigo**

**{ //si son iguales imprimimos la explosion**

**lcd\_comando(54); //imprimimos explosion**

**lcd\_comando(dispdir);**

**lcd\_dato(4);**

**delay(50);**

**lcd\_comando(72); //clear**

**lcd\_comando(dispdir);**

**lcd\_dato(1);**

**delay(5);**

**break; //terminamos el ciclo for**

**}**

**dec=dec-0x01; //decrementamos el disparo**

**lcd\_comando(72); //clear**

**lcd\_comando(dispdir);**

**lcd\_dato(1);**

**delay(5);**

**lcd\_comando(72); //clear**

**lcd\_comando(dec+0x01);**

**lcd\_dato(1);**

**delay(5);**

**}**

**}**

**}**

**}**