****

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FIEC**

**Laboratorio de Microcontroladores**

**Proyecto Segundo Parcial**

**Óhmetro y Opcional**

**Alumno:**

**Alfonso Israel Guevara Gallardo**

**Paralelo: # 4**

**Grupo: # 1**

**Fecha de presentación:**

**23 de agosto del 2012**

**2012 – I TÉRMINO**

**1.- Enunciado del proyecto**

**Ejercicio1**

En el primer Ejercicio se uso un convertidor análogo digital tiene como entrada un nivel de voltaje (valor analógico) y produce en su salida un número binario de n bits proporcional al nivel de la entrada (valor digital). Los convertidores de señal análoga a digital abrevian ADC o A/D.

Uno de los parámetros que definen al A/D es la resolución como la mínima variación de voltaje en la entrada que produce cambio del valor digital en la salida. Por ejemplo un convertidor de 10 bits tiene un total de 210 valores (1024 valores de 0 a 1023).

**Ejercicio2**

En este ejercicio se hizo uso de las librerías de sonido mikroC para PIC proporciona una biblioteca de sonidos para suministrar a los usuarios con las rutinas necesarias para la señalización de sonido en sus aplicaciones. Generación de sonido necesita hardware adicional, como piezo-altavoz (ejemplo de piezo-altavoz interfaz se da en el esquema en la parte inferior de esta página).

**2.- Diagrama de Bloques**



**3.- Diagrama de Flujo funcional del Programa principal**



**4.- Descripción del algoritmo o estrategia utilizada.**

Ejercicio 1

* Limpiar pantalla
* Asignar bits de ADC a variable
* Multiplicar y dividir por el factor de la resistencia en este caso 1000 de 1 k
* Mostrar cero si valor de ADC es cero
* Mostrar valor en decimal con algoritmo de divisiones sucesivas y modulo para 10.
* Convertir numero decimal a binario
* Mostrar número binario.

Ejercicio 2

* Configurar PORTA como entrada
* Configurar RE1 como salida para Speaker
* Inicializar PORTE con función sound\_Init().
* Ejecutar sonido dependiendo de la tecla presionada en PORTA

**5.- Listado del programa fuente en lenguaje ensamblador con comentarios en las líneas de código que considere fundamentales**

/\*

\* Nombre del Proyecto:

proyecto2.c

\* Nombre del Autor:

(c) Mikroelektronika, 2009.

\* Description:

(Explicación del ejercicio)

\* Test configuration:

MCU: PIC16F887

Oscillator: HS, 08.0000 MHz

SW: mikroC PRO for PIC

\* NOTES:

\*/

/\*

\* Nombre del Proyecto:

P9a\_pwm.c

\* Nombre del Autor:

(c) Mikroelektronika, 2009.

\* Description:

(Explicación del ejercicio)

\* Test configuration:

MCU: PIC16F887

Oscillator: HS, 08.0000 MHz

SW: mikroC PRO for PIC

\* NOTES:

\*/

//Declaracion de variables

unsigned short kp;

int opcion,i,tecla\_sound;

int n;

int tomenu;

unsigned long value;

char aux\_value;

unsigned char d1,d2,d3,d4,\*text;

int but,random,iter,tam=0;

char crh;

char CPU[20],jug1[20];

unsigned short keypadPort at PORTD;

char \*string;

sbit LCD\_RS at RB4\_bit;

sbit LCD\_EN at RB5\_bit;

sbit LCD\_D4 at RB0\_bit;

sbit LCD\_D5 at RB1\_bit;

sbit LCD\_D6 at RB2\_bit;

sbit LCD\_D7 at RB3\_bit;

sbit LCD\_RS\_Direction at TRISB4\_bit;

sbit LCD\_EN\_Direction at TRISB5\_bit;

sbit LCD\_D4\_Direction at TRISB0\_bit;

sbit LCD\_D5\_Direction at TRISB1\_bit;

sbit LCD\_D6\_Direction at TRISB2\_bit;

sbit LCD\_D7\_Direction at TRISB3\_bit;

// End LCD module connections

//Definicion de procedimientos y funciones

void menu();

unsigned short Codigo\_Teclado(int codigo);

void Tone1();

void Tone2();

void Tone3();

void ToneA();

void ToneC();

void ToneE();

//programa principal

void main() {

start:

TRISB=0;

PORTB=0xFF;

TRISB=0xFF;

ANSEL=0;

ANSELH=0;

ANSEL = 0b00100000;

TRISE = 0b00000001;

TRISC = 0;

Keypad\_Init(); // Initialize Keypad

Lcd\_Init(); // Initialize Lcd

Lcd\_Cmd(\_LCD\_CLEAR); // Clear display

Lcd\_Cmd(\_LCD\_CURSOR\_OFF); // Cursor off

kp = 0;

opcion=0,tomenu=0;

tam=1;

do{

menu();

opcion=Keypad\_Key\_Press();

// kp=Codigo\_Teclado(aux);

Delay\_ms(50);

if( opcion!=1 && opcion!=2 && opcion!=0 ) {

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out(1, 1,"ELIJA OPCION!!!");

Lcd\_Out(2, 1," INTEMTE DE NUEVO");

Delay\_ms(2000);

}

// Lcd\_Chr(1, 1,kp);

}while (opcion!=1 && opcion!=2);

switch (opcion){

case 1: {

Lcd\_Cmd(\_LCD\_CLEAR);

do{

value = adc\_read(5);

value=value\*1000/1022;

n=value;

if(n==0)

Lcd\_Out(3,10,"000000000");

for(i=0;n>0;i++){

Lcd\_chr(3,20-i,48+(n%2));

n=n/2;

if(n==0)

Lcd\_chr(3,19-i,48+n);

Delay\_ms(15);

}

Lcd\_Out(3, 1,"Binario: ");

d1=value/1000;

d2=(value/100)%10; //"%" este operador devuelve el módulo de la //división

d3=(value/10)%10;

d4=value%10;

Lcd\_Out(1, 1,"Lectura en ohmios!");

Lcd\_Out(2, 1,"Decimal: ");

Lcd\_chr(2,10,48+d1); // se imprime la lectura

Lcd\_chr\_CP(48+d2); // 48 es el codigo ASCI del numero '0'

Lcd\_chr\_CP(48+d3);

Lcd\_chr\_CP(48+d4);

Lcd\_Out\_Cp("[Ohms]");

tomenu=Keypad\_Key\_Press();

if(tomenu==12)

goto start;

//

} while (1);

} break;

case 2:{

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out\_Cp(" Modo Piano ");

while (1) { // Wait for button to be released

TRISA = 0xF8; // Configure RB7..RB3 as input

TRISE = 0b00000011; // Configure RD3 as output

Sound\_Init(&PORTE, 1);

if (Button(&PORTA,7,1,1)) // RB7 plays Tone1

Tone1();

while (PORTA & 0x80); // Wait for button to be released

if (Button(&PORTA,6,1,1)) // RB6 plays Tone2

Tone2();

while (PORTA & 0x40); // Wait for button to be released

if (Button(&PORTA,5,1,1)) // RB5 plays Tone3

Tone3();

while (PORTA & 0x20); // Wait for button to be released

if (Button(&PORTA,4,1,1)) // RB4 plays Melody2

ToneA();

while (PORTA & 0x10); // Wait for button to be released

if (Button(&PORTA,3,1,1)) // RB3 plays Melody

ToneC();

while (PORTA & 0x08);

}

}

}

}

void menu(){

Lcd\_Out(1, 1," PROYECTO MICC # 2");

Lcd\_Out(2, 1,"\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*");

Lcd\_Out(3, 1,"1.OHMETRO");

Lcd\_Out(4, 1,"2.JUEGO");

}

unsigned short Codigo\_Teclado(int codigo){

switch (codigo) {

case 1: kp = 49; break; // 1 // Uncomment this block for keypad4x4

case 2: kp = 50; break; // 2

case 3: kp = 51; break; // 3

case 4: kp = 45; break; // /

case 5: kp = 52; break; // 4

case 6: kp = 53; break; // 5

case 7: kp = 54; break; // 6

case 8: kp = 42; break; // \*

case 9: kp = 55; break; // 7

case 10: kp = 56; break; // 8

case 11: kp = 57; break; // 9

case 12: kp = 47; break; // -

case 13: kp = 42; break; // \*

case 14: kp = 48; break; // 0

case 15: kp = 61; break; // =

case 16: kp = 43; break; // +

}

return kp;

}//

void Tone1() {

Sound\_Play(659, 500); // Frequency = 659Hz, duration = 250ms

}

void Tone2() {

Sound\_Play(698, 500); // Frequency = 698Hz, duration = 250ms

}

void Tone3() {

Sound\_Play(784, 500); // Frequency = 784Hz, duration = 250ms

}

void ToneA() {

Sound\_Play( 880, 500);

}

void ToneC() {

Sound\_Play(1046,500);

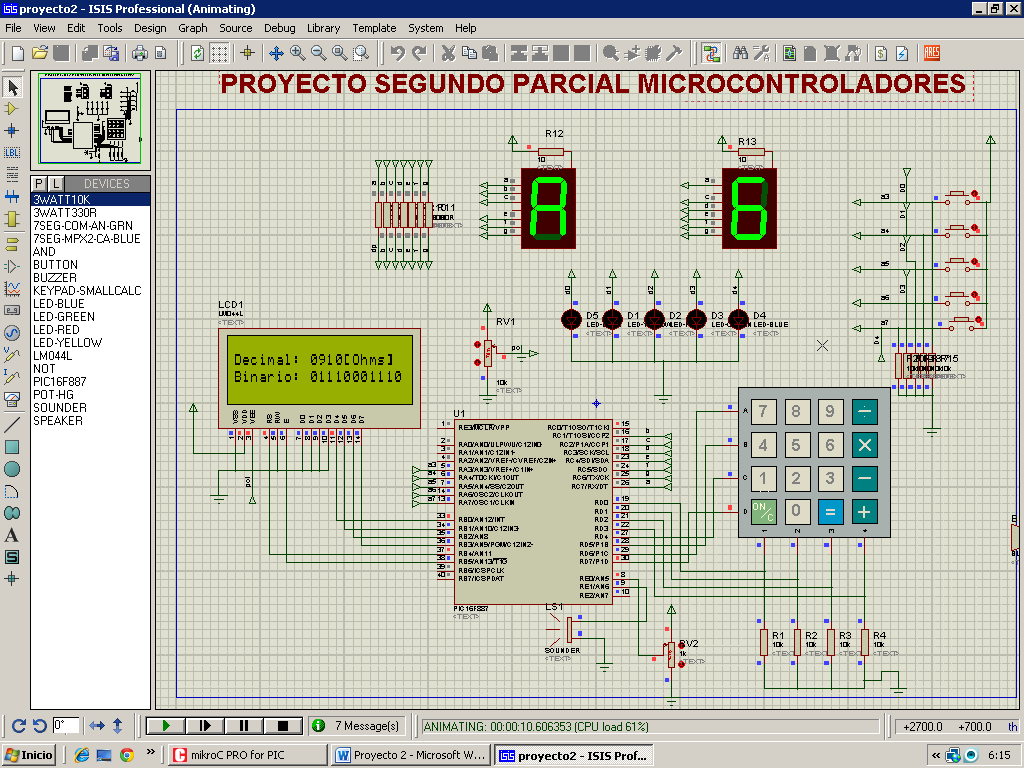
}

void ToneE() {

Sound\_Play(1318, 50);

}

**6.- Copia impresa del circuito armado en PROTEUS para la simulación en el momento de su ejecución**





**7.- Conclusiones**

* El uso de pantallas LCD y teclado se hace mucho más fácil con ayuda de las funciones de vienen en las librerías de C para pics. Existe una función para cada necesidad ya sea inicializar, mostrar datos, comandos, encender, apagar, desplazar, entre otras. Teniendo esta facilidad se hace indispensable consultar las variables que retorna para poderlas declarar al inicio del programa.
* El texto que podemos escribir en el LCD es mucho más grande del que se aparenta ya que solo 16 caracteres se muestran en pantalla, pero la capacidad de albergar texto es mucho más grande, consideremos a este texto como una cinta que se mueve a medida que efectuamos los desplazamientos. Lo que aclara ciertas dudas que pudieron haber surgido en la práctica.
* Cuando escogimos el keypad, que se uso en las simulaciones de la práctica, pudimos observar que no concordaban ciertas teclas con la programación interna. Esto es irrelevante ya que únicamente debernos cambiar la programación y adecuarla a los elementos del simulador, ya que como es un programa en C se puede modificar sin afectar el resto de la programación, además para que el PIC detecte que una tecla fue presionada es necesario poner un retardo.

**8.- Recomendaciones**

* Es importante seguir estrictamente las conexiones mostradas para obtener una simulación acertada según el programa que se haya cargado al microcontrolador. Por ello, observe con cuidado cuales son entradas, salidas tanto de la pantalla LCD como del teclado.
* La instrucción WHILE(1) es recomendable utilizar cada vez que se requiera de un lazo infinito, es parecido cuando en lenguaje ensamblador utilizábamos el GOTO al que en ocasiones utilizábamos para volver a una misma etiqueta.
* Si no conocemos el tiempo que va a durar el retardo debemos tener cuidado con los valores que colocamos ya que si son demasiado grandes podemos quedarnos atrapados en un lazo que puede demorar mucho tiempo y la visibilidad al presentarlo será demasiado lenta.