**http://www.cvr.espol.edu.ec/research/cfrs/2004/images/fiec.gif**

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FIEC**

**Laboratorio de Microcontroladores**

**Proyecto**

**Lectura de valor analógico de un potenciómetro y Tema Libre:**

**Temporizador programable en segundos**

**Alumno/a**

**Hans Alulima Fuentes**

**Paralelo: # 3**

**Proyecto: # 7**

**Fecha de presentación:**

**23 01 2012**

**2011 – 2° TÉRMINO**

**OBJETIVOS:**

* Utilizar las técnicas y herramientas aprendidas.
* Realizar una aplicación para los Microcontroladores usando programación en lenguaje C e implementarlo a nivel de hardware.
* Experimentar con los diferentes procesos desde el diseño hasta la implementación final, y enfrentarse con los diferentes problemas cuando se usa una placa soldada por uno mismo.

**1.- Enunciado del proyecto**

El PROGRAMA1 consiste en hacer uso del módulo ADC que posee el microcontrolador. Lee el valor analógico del potenciómetro conectado en el PORTA y muestra su equivalente digital DECIMAL: unidades y decenas en los dos DISPLAYs conectados en los puertos C y D respectivamente; y su valor BINARIO en los 8 LEDs conectados al PORTB.

El PROGRAMA2 consiste en la implementación de un temporizador programable en segundos haciendo uso de 4 botoneras para el ajuste y mostrando el valor en los dos DISPLAYs de 7 segmentos. Si se presiona la BOTONERA1 se incrementan los segundos en pasos de uno en uno (unidades), si se presiona la BOTONERA2 se incrementan los segundos en pasos de 10 en 10 (decenas), si se presiona la BOTONERA3 se decrementan los segundos en pasos de 1 en 1 y finalmente si se presiona la BOTONERA4 se inicia el temporizador previamente ajustado. El cual decrementa de uno en uno cada segundo el valor hasta llegar a cero, en ese mismo instante parpadea los LEDs en el PORTB para indicar que ha finalizado la temporización.

****

**2.- Diagrama de Bloques**



****

**3.- Diagrama de Flujo funcional del Programa principal y de las subrutinas**

***Programa Principal:***



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

Se escriben los parámetros de configuración y inicializan las variables y los puertos a ser utilizados. Se muestran en los puertos C y D las iniciales del estudiante. Quedamos encerrador en un lazo infinito que pregunta por la BOTONERA1 o la BOTONERA2.

Si se presiona la BOTONERA1, se ejecuta el PROGRAMA1: Se configura el modulo ADC y se leen los Datos Analógicos desde el PIN0 del PORTA. Una vez obtenido el valor digital se lo divide para 10.23 para obtener un rango de 0 a 99. Seguido separamos el número en unidades y decenas y Finalmente mostramos el valor binario en el PORTB y el valor Digital Decimal en los PUERTOS C y D. Usando un arreglo de códigos de 7 segmentos para poder mostrar el número correctamente.

Si se presiona la BOTONERA2, se ejecuta el PROGRAMA2: Se inicializan las variables y puertos a usar; y quedamos dentro de un lazo infinito también, en que se espera por la presión de alguna de las 4 botoneras a usar. Si presiona la BOTONERA1 se incrementa el valor del temporizador en unidades. Si se presiona la BOTONERA2 se incrementa el valor del temporizador en decenas. Si se presiona la BOTONERA3 se decrementa el valor del temporizador de 1 en 1 y finalmente si se presiona la BOTONERA4 se inicializa el conteo del temporizador, quedándonos en un lazo infinito en que se va decrementando el valor del tiempo total cada segundo en pasos por unidad. Al finalizar se hacen parpadear los LEDs del PORTB y se regresa al estado en el que nos permite ajustar el tiempo nuevamente.

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

int dec,uni,decimal,dato,aux;

int const disp[10] = {0x3F, 0x06, 0x5B, 0x4F, //Se declaran como constantes

0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F}; //Codigo Hexa para cada nuemro

int const id[2] = {0x77, 0x38};

int espejo(int temp1) //Funcion Para invertir Bits de una variable

{ int temp2=0;

if ((temp1&0x01)==0x01)

temp2=temp2+128;

if ((temp1&0x02)==0x02)

temp2=temp2+64;

if ((temp1&0x04)==0x04)

temp2=temp2+32;

if ((temp1&0x08)==0x08)

temp2=temp2+16;

if ((temp1&0x10)==0x10)

temp2=temp2+8;

if ((temp1&0x20)==0x20)

temp2=temp2+4;

if ((temp1&0x40)==0x40)

temp2=temp2+2;

if ((temp1&0x80)==0x80)

temp2=temp2+1;

return temp2;

}

void p1()

{

ADC\_Init(); //inicializo el ADC

while(1)

{

dato=ADC\_Read(0); //entrada analogica 0

aux=(dato/10.23); //division perfecta para marcar rango

if(aux<=99)

{

dec=aux/10; //division para extraer decimal del dato anterior

uni=aux%10; //modulo (residuo de la division) para extraer unidades

}

if(aux==100)

{

dec=0;

uni=0;

}

PORTB=espejo(aux); // muestro el valor binario en el PORTB

PORTC=disp[uni]; // muestro valor unidaades en el PORTC.

PORTD=disp[dec]; // muestro valor decenas en el PORTD

}

}

void p2()

{

int contu=0, contd=0, contt=0; //declaracion de variables

do{ //se enceran displays

PORTC=disp[0];

PORTD=disp[0];

}while(PORTA.F2); // se valida que suelte la resistencia

while(1)

{

contt=(contd\*10)+contu;

if(PORTA.F1) // si presiona la botonera PORTA.pin1

{

contu++; // incremento unidades

if(contu==10) // valido que cuando pase de 10 se incremente decenas y unidades

{ contu=0;

contd++;

if(contd==10)

{ contu=0;

contd=0;

}

}

Delay\_ms(300);

}

if(PORTA.F2) //si presiona la botonera PORTA.Pin2

{

contd++; // se incrementas decenas

if (contd==10) // se valida el paso por 9

{ contd=0;

contu=0;

}

Delay\_ms(300);

}

if(PORTA.F3) //si presiona la botonera PORTA.pin3

{

contt--; // se decrementa el contador total.

if(contt==0) // se valida al contador

contt=0;

contu=contt%10;

contd=contt/10;

Delay\_ms(300);

}

if(PORTA.F4) // si presiona la PORTA.Pin4

{

while(contt>0) // mientras no sea cero lo decremeta

{

Delay\_ms(1000);

contt--;

contu=contt%10; //se las separa nuevamente y se los

contd=contt/10; // digitos en unidades y decenas

PORTC=disp[contu]; //se muestran unidades en el PORTC

PORTD=disp[contd]; // se muestran decenas en el PORTD

}

PORTB=0xFF; // se hace parpadear todos los ledd

Delay\_ms(500);

PORTB=0x00;

Delay\_ms(500);

PORTB=0xFF;

Delay\_ms(500);

PORTB=0x00;

Delay\_ms(500);

PORTB=0xFF;

Delay\_ms(500);

PORTB=0x00;

Delay\_ms(500);

}

PORTC=disp[contu]; // Se muestran los valores/ tiempo total en march

PORTD=disp[contd];

}

}

void main()

{

ANSEL=0x01;

ANSELH = 0; // Configure other AN pins as digital I/O

C1ON\_bit = 0; // Disable comparators

C2ON\_bit = 0;

TRISA = 0xFF; // PORTA como entrada

TRISC = 0; // PORTC como salida

TRISB = 0; // PORTB como salida

TRISD = 0; // PORTD como salida

PORTB=0; // inicializado en 0 el PORTB

PORTC=id[1]; // iniciales

PORTD=id[0]; // iniciales

while(1)

{

if(RA1\_bit=1) //prueba el bit 1 del PORTA

p1();

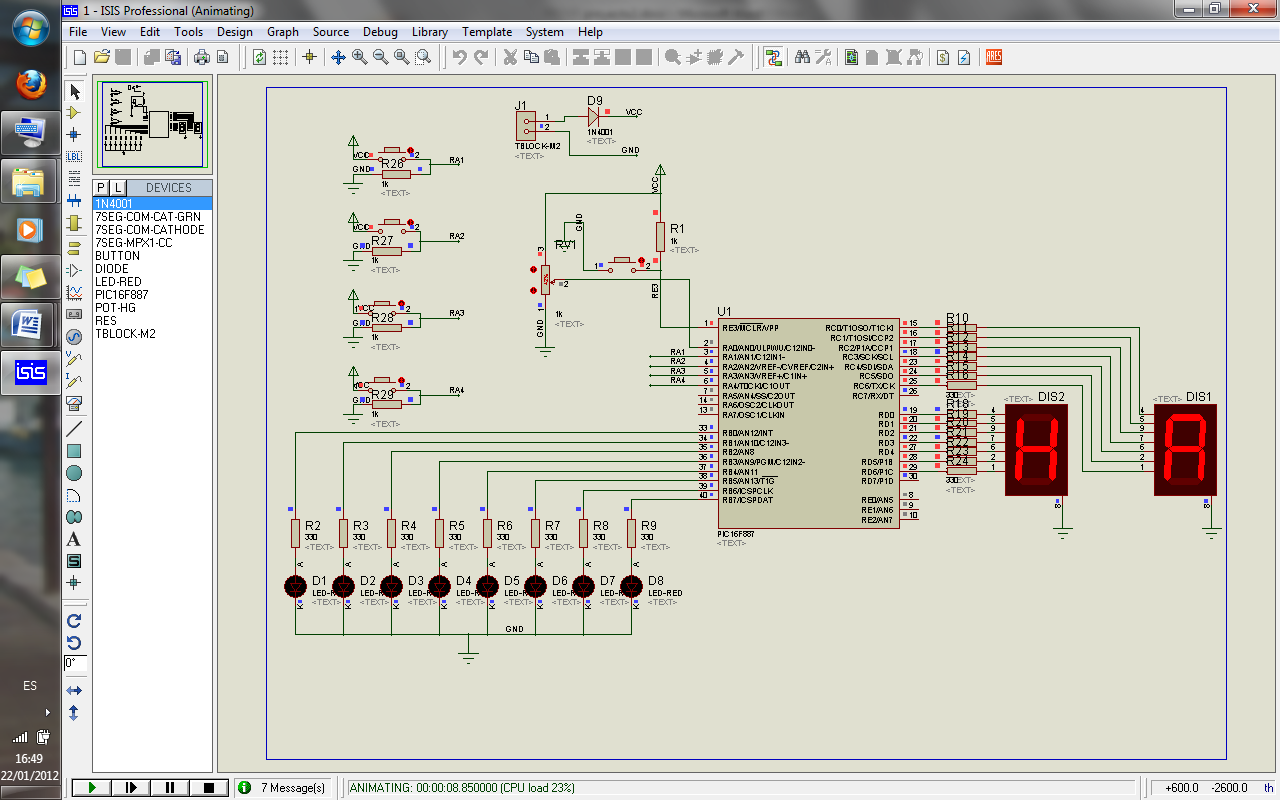
if(PORTA.F2=1) //prueba el bit 2 del PORTA

p2();

}

}

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

****

****

**7.- Conclusiones**

* El primer proyecto ya nos dejo con muchos conocimientos acerca de las placas y el microcontrolador PIC 16F887 y ahora hemos reforzado aquellos conocimientos y los hemos mejorado y encontrado mejores aplicaciones y usos del PIC y así poder pensar en formas de poder usarlo para la vida cotidiana.
* El aprendizaje en Lenguaje C en el programa MikroC resultó más comprensible que el lenguaje assembler, además permite interactuar mas con el PIC16F887, con este programa además se pudo simular en PROTEUS y poder observar lo que puede suceder en la vida real con un circuito antes de probarlo.
* Con respecto a la placa diremos q fue fácil la soldadura de los elementos no tuvimos ningún problema con la soldadura de los elementos en realidad fue fácil y requirió poco tiempo y ahí se vio nuestra habilidad para soldar desarrollada gracias a los proyectos implementados anteriormente, por ultimo es importante decir que utilizamos todo lo visto en las clases teóricas así como en prácticas anteriores.

**8.- Recomendaciones**

* Es recomendable después de tener el código en el MikroC con su respectivo proyecto, revisar paso a paso el código y para eso es de mucha utilidad la herramienta, que se asemeja a la ventana Special Function Registers en mplab, que nos muestra el cambio de los valores de los bits a lo largo del programa, y nos muestra en rojo cuando un bit a cambiado de valor, esto nos ayuda a comprender el comportamiento de los Registros del microcontrolador.
* Reducir el número de líneas es bueno para un mejor uso en la memoria del PIC y evitar un desperdicio de la misma, además se evita redundancias que podrían confundir a otro programador que vea el informe del proyecto.
* Se recomienda a la hora de implementar en PROTEUS tener mucho cuidado y sobre todo escoger bien los elementos porque una elección incorrecta representaría una pérdida de tiempo así de la misma manera a la hora de soldar, para detectar cualquier error es recomendable tener un milímetro para chequear el comportamiento de nuestro circuito midiendo continuidad.