Nombre de la asignatura: Laboratorio de Programación de Avanzada

Nombre de la práctica: Manejo de entradas y salidas digitales

Integrantes:
- Zuñiga Fragoso Diego Joel
- Manríquez Navarro Daniela del Carmen

Número de práctica: 1

Total de horas: 2 Hrs.

Objetivos

- Configurar los pines del microcontrolador para operar como entradas simples o salida simples (SISO), al igual puertos como entradas o salidas.
- Desarrollar un programa que recibirá señales de entradas y a partir de estas, manipular acciones visuales (leds) del microcontrolador mediante conocimientos de programación básica.
- Comprender la implementación de la jerarquía de tareas en la programación mediante la prioridad en las lecturas de las entradas.

Descripción de la práctica

El estudiante debe ser capaz de configurar adecuadamente el microcontrolador para el uso de señales de entrada, estas a su vez controlaran: "*la selección*", "*paro*" e "*inicio*" de cuatro distintas secuencias de salida del microcontrolador (uC), que se visualizaran mediante diodos emisores de luz (led). Debido a que las secuencias se ejecutaran de manera cíclica por tiempo indefinido, el alumno deberá contemplar la cuestión de la jerarquía en la lectura de entradas, con la finalidad de dar prioridad al botón de *paro* y que ejecute la acción de manera inmediata sin importar la secuencia que se ejecute, así como solo iniciar una secuencia mediante el botón de *Inicio* sin importar si los botones de selección son alterados mientras se corre una secuencia, una vez este activada una secuencia, esta podrá cambiarse por otra mediante el paro de la misma y dar inicio a la nueva secuencia.

Consideraciones para la elaboración

- Las secuencias de salida serán determinadas por el alumno.
- La programación de la prioridad debe ser en código C, sin el uso de etiquetas ni interrupciones.

- Los botones deben ejecutar acciones de maneras inmediatas, sin necesidad de ser presionados por mucho tiempo o múltiples ocasiones.

Marco teórico

Las entradas y salidas digitales, o para ser más general, la habilidad para monitorear y controlar hardware directamente, es la principal característica de los microcontroladores. Como consecuencia prácticamente todos los microcontroladores tiene al menos 1-2 pines de I / O digital. Los pines de I / O están generalmente agrupados en puertos de 8 pines. Los pines pueden ser de solo entrada, de solo salida, o -más comúnmente- bidireccional, es decir, capacidad para ser entrada y salida. Si leemos el nivel de voltaje de un pin con un multímetro (con respecto a GND), veremos un voltaje análogo, sin embargo, el microcontrolador lo digitaliza por mapeo a uno de dos estados, 0 lógico o 1 lógico.

La funcionalidad de la entrada digital es usada siempre que la señal monitoreada deba ser interpretada digitalmente, esto es, cuando solo cambien entre dos valores “alto” (correspondiente al 1 lógico) y “bajo” (correspondiente al 0 lógico). Siempre que la señal deba ser monitoreada como “alto” y “bajo” dependerá de su nivel de voltaje, el cual estará determinado en las especificaciones del microcontrolador.

La funcionalidad de la salida digital es usada para ajustar los pines de salida a niveles de voltaje dados. Los niveles correspondientes a “alto” y “bajo” son especificados también por las especificaciones del microcontrolador.

Debemos tomar en cuenta que los pines de salida son más críticos que los pines de entrada en sentido de que dependen fuertemente de una protección de corriente externa. Después de todo, se puede conectar una salida a GND, ajustar la salida a 1, y hacer un corto circuito. Aunque los microcontroladores pueden soportar un corto circuito por un tiempo breve (generalmente menos de un segundo), un corto puede eventualmente destruir el microcontrolador.

Proteus fue desarrollado por Labcenter Electronics, es un software con el cual se puede generar fácilmente un esquemático, desarrollar en PCB y simular microprocesadores. Es un entorno diseñado para la realización completa de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño, simulación, depuración, y construcción. Es una herramienta para la elaboración avanzada de esquemas electrónicos, que incorpora una librería de más de 6000 modelos de dispositivos digitales y analógicos.

EQUIPO Y MATERIALES

- Software para programación
- Software de simulación
- Programador de microcontrolador
- Microcontrolador
- 4 push buttons
- 8 leds

- 12 resistencias de 220 o 330 ohms

DESARROLLO DE LA PRÁCTICA

1. Escribir el programa para el microcontrolador de la Figura 1, considerando:
   a) Seleccionar el uC adecuado para la práctica, configuración necesaria para operar el uC y frecuencia de reloj.
   b) La selección de los pines de entrada y salida quedará a consideración del alumno.
   c) La secuencia de salida que se encuentre activa podrá ser detenida en cualquier momento, es decir, sin importar que la secuencia haya o no terminado.
   d) La secuencia solo inicia después de un paro y al presionar el botón de inicio.
2. Desarrollar la simulación del circuito (de acuerdo al diagrama de bloques, Figura 1), tomando en cuenta los siguiente:
   a) Se tendrán dos botones que servirán como señales de entrada para la selección de la secuencia de salida (los casos se representan en binario), un botón de inicio y un botón de paro.
   b) Ocho LEDs conectados a las salidas seleccionadas del microcontrolador servirán para visualizar las secuencias de salida.
3. Realizar la conexión del circuito y programar el microcontrolador.

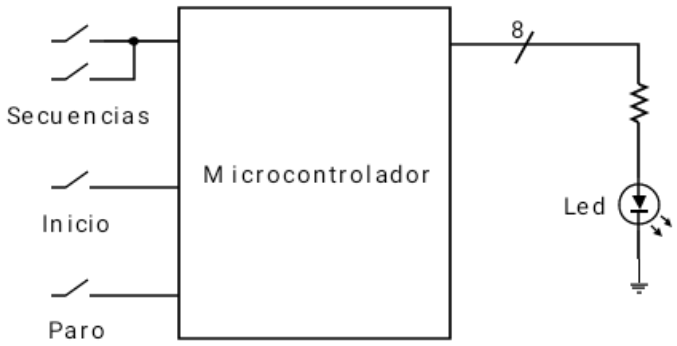

Figura 1. Diagrama de bloques para el circuito.

Resultados de la práctica

Descripción del código realizado
Inicializamos la librería, los fusibles y la frecuencia. #include <18f4550.h>

```
#fuses INTRC, NOWDT, NOPROTECT, NOLVP, CPUDIV1, PLL1, NOMCLR // Fusibles
#use delay(clock = 8M)

void main()
{
  int16 i = 0; // Los enteros tienen 1 byte le ponemos int16 para evitar sobrecarga
  int B = 0;

  while(true)
  {

   If para detectar el botón de encendido y modificar la bandera para activar secuencia.

    if(input(pin_d2))
    {
     // Dejamos atrapado el código si se mantiene presionado el botón. Para evitar errores
      while(input(pin_d2));
      B = 1;
    }
    else{}

   Ya que fue presionado el botón de inicio, estos IF`s detectan la combinación de botones de cada secuencia.

   Primer Boton presionado:

    if(input(pin_d0) && !input(pin_d1) && B)
    {
      for(i = 1; B ; i *= 2) // For de la secuencia, se repite dependiendo de la bandera
      {
        if(input(pin_d3))       // Espera el botón de apagado en cada momento de la secuencia
```

```
        {
          while(input(pin_d3));
          B = 0;    // Desactiva la bandera del ciclo de la secuencia
        }

        if(B)  // Si no se presionó "Apagado" anteriormente repite la secuencia en cada ciclo
        {
          if(i > 128)
            i = 1;

          output_b(i);
          delay_ms(200);
        }
        else{}
      }
      output_b(0);  // Al presionar apagado y salir del ciclo, apaga todos los leds
    }
    else{}
```

Segundo Boton presionado (Tiene el mismo funcionamiento que el primero)

```
    if(!input(pin_d0) && input(pin_d1) && B)
    {
      for(i = 128; B ; i /= 2)
      {
        if(input(pin_d3))
        {
          while(input(pin_d2));
          B = 0;
        }
```

```
      if(B)
      {
        if(i < 1)
          i = 128;

        output_b(i);
        delay_ms(200);
      }
      else{}
    }
    output_b(0);
}
else{}
```

Ningun Boton presionado (Tiene el mismo funcionamiento que el primero):

```
if(!input(pin_d0) && !input(pin_d1) && B)
{
  for(i = 1; B ; i++)
  {
    if(input(pin_d3))
    {
      while(input(pin_d3));
      B = 0;
    }

    if(B)
    {
      output_b(i);
      delay_ms(100);

      if(i >= 256)
```

```
        i = 0;
      }
      else{}
    }
    output_b(0);
  }
  else{}
```

Ambos Botones presionados (Tiene el mismo funcionamiento que el primero):

```
  if(input(pin_d0) && input(pin_d1) && B)
  {
    for(i = 256; B ; i--)
    {
      if(input(pin_d3))
      {
        while(input(pin_d3));
        B = 0;
      }

      if(B)
      {
        output_b(i);
        delay_ms(100);

        if(i <= 1)
          i = 256;
      }
      else{}
    }
    output_b(0);
  }
```

```
    else{}
  }
}
```

Desarrollo del código:

Partiendo del código que creamos en clase, este solo detectaba qué combinación de botones estaba siendo recibida y ejecutaba la secuencia correspondiente. Por lo tanto, tuvimos que realizar varias modificaciones para asegurarnos de que los nuevos 2 botones funcionaran correctamente.

Para darle más prioridad al botón de encendido, creamos un condicional "IF" al inicio del código que verifica si el botón de encendido ya ha sido presionado utilizando una bandera.

Una vez que establecimos esta jerarquía, necesitamos que comience alguna secuencia, ahí tuvimos que cambiar la lógica del código. Anteriormente, el patrón cambiaba inmediatamente después de alterar la combinación de botones (esto ocurría debido a una condición en el ciclo "for"). Dado que queríamos que el cambio ocurriera después de presionar el botón de apagado, reorganizamos el código. Ahora, ingresamos al patrón mediante un "if" que verifica inmediatamente después de presionar el botón de encendido en qué patrón debería entrar. Dentro de este "if", establecemos una condición en el ciclo "for" que verifica la bandera de encendido/apagado en cada ciclo de la secuencia. Esto nos permite detectar si se presiona el botón de apagado durante cada ciclo del patrón y, en caso afirmativo, salir del ciclo y del "IF", entonces por el "while(true)" el programa esperara nuevamente la presión del botón de encendido.

Conclusiones de la práctica

Joel Zuñiga: Esta practica me ayudo mucho a comprender mas a fondo como se comporta el código con la variante de que tenemos un ciclo infinito corriendo todo el tiempo, y como puedo ocupar eso a mí favor, además de comprender de la misma forma como se comportan los booleanos de los botones y como mandar una secuencia en los LED’s.

Daniela Manriquez:

En está práctica pude comprender de mejor manera a utilizar las entradas y salidas, asi como enteder como es que se comporta el ciclo infinito y usarlo de mejor manera para poder parar y seguir con las secuencias.

Conclusiones del equipo:

Esta práctica a pesar de no ser tan compleja, nos ayudo a familiarizarnos mas con los programas como “Proteus” y “Pic-C compiler”, así como las funciones de la librería del microcontrolador.

Bibliografía

García E. (2008). Compilador C CCS y simulador PROTEUS para microcontroladores PIC: Alfa Omega.

Grindling G. & Weiss B. (2007). Introduction to Microcontrollers. Vienna University of Technology.