



Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México

Proyecto Final

Arquitectura Computacional I

Implementación en PCB de un Teclado Matricial 4 x 4

Integrantes:

Iván Cabrera Altamirano

455591

Profr. Jorge Alberto Ramírez Landa

Fecha de Entrega:

Martes 12 de Noviembre de 2002

Introducción

El proyecto final de la materia de Arquitectura Computacional I en esta ocasión consiste en la implementación de un teclado matricial 4 x 4 en PCB, teniendo como teclas desde 0 hasta F, obteniendo una salida en 4 bits, además es debemos programar nuestras funciones lógicas en Arreglos Genéricos Programables (GAL's).

Un GAL es un dispositivo programable que permite la implementación de funciones lógicas sin necesidad de alambrear nuestra función, consiste en una matriz de AND's y OR's que solo permite que cada función dependa como máximo de la sumatoria de 8 miniterminos. Se utilizan varios lenguajes de programación como por ejemplo: OPAL, PALASM u VHDL.

Un teclado matricial de 4 x 4 puede tener infinidad de usos, por eso es importante conocer como podemos nosotros controlar uno. Además de conocer los diferentes procesos a seguir para obtener los circuitos impresos (PCB).

Una aplicación común puede ser un teléfono, horno de microondas, o un teclado de computadora como los que comúnmente conocemos, ya que aún cuando físicamente son diferentes su funcionamiento interno es muy similar, simplemente cuando tengamos más teclas necesitaremos más bits para representar la información que estamos seleccionando.

En este caso, nosotros formaremos el teclado a partir de un arreglo 4 x 4 de mini-botones, lo cual es lo mismo a utilizar un teclado comercial.

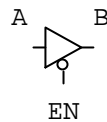
Implementación en PCB de un Teclado Matricial 4 x 4

El realizar un controlador para teclado matricial resulta interesante ya que la manera en la cual lo debemos conectar no es la forma convencional, pensaríamos en que solamente polarizándolo nos quedaría, ya que surge la pregunta de cómo poder saber que tecla estamos apretando, tendríamos que estar seleccionando todas nuestras líneas a la vez, lo cual de principio se vuelve imposible.

Si necesitamos seleccionar todas nuestras líneas a la vez lo que podemos hacer es utilizar un decodificador de 2 a 4, donde conectaremos a la entrada del decodificador un contador binario, por lo tanto al utilizar una frecuencia alta nos encontramos seleccionado las filas de una manera tan rápida que será posible pensarlo como si ocurriera a la vez.

Con respecto a las columnas estas las conectaremos a voltaje a través de unas resistencias, y a su vez estas se conectarán a compuertas NAND junto con las salidas de otro decodificador de 2 a 4, esta parte del circuito servirá para activar una señal que a partir de este momento la denominaremos FREEZE (FR) que servirá para detener el contador, y a su vez activar la parte del decodificación y visualización del display de 7 segmentos cuando la tecla presionada sea igual a la combinación de los 4 bits donde obtenemos nuestra salida.

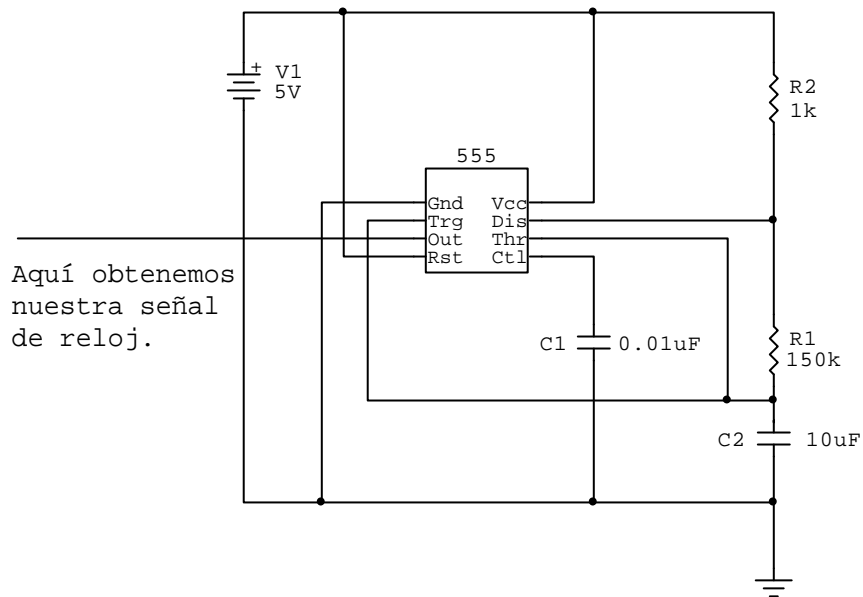
La manera en que activaremos el contador mediante la señal FR será utilizando un Buffer de 3 Estados, en este lo utilizaremos con su ENABLE (EN') de manera negada.



| A | EN | B |
|---|----|---|
| 0 | 0 | 0 |
| 0 | 1 | - |
| 1 | 0 | 1 |
| 1 | 1 | - |

Por lo tanto cuando $EN' = 0$ dejaremos pasar la señal de A hasta B. Y cuando sea $EN' = 1$, no tendremos señal en B. Por lo tanto un buffer de 3 estados se comporta como un switch que deja pasar la señal que tengamos a la entrada dependiendo de la señal de activación (EN). Un buffer de 3 estados comúnmente lo encontraremos en los integrados 74LS244 y 74LS125, teniendo como única diferencia que el 74LS125 tiene su señal de activación negada (EN'). Es nuestro caso utilizaremos el Buffer 3s para permitir o no el paso de la señal de reloj que activa a su vez al contador (74LS93).

En el caso de la simulación utilizamos CircuitMaker, donde utilizamos un pulser para generar nuestra señal de reloj, en la realidad estamos utilizando un 555 como Multivibrador Astable. La configuración que utilizamos es la siguiente:



Si queremos variar la frecuencia de oscilación de nuestro circuito deberemos variar R1, para tal propósito podemos utilizar un potenciómetro ya sea normal o de precisión.

En las salidas de nuestro contador colocamos el decodificador de hexadecimal a 7 segmentos para poder visualizar el número que obtenemos en nuestro display, además debemos de activar que solo se visualice cuando la señal FR lo permita, ya que de lo contrario nos mostraría todo el recorrido que sigue para llegar al número que tecleamos, lo cual no deseamos, ya que no sabemos la aplicación posterior que pueda tener nuestro circuito.

La sección del decodificador Hexadecimal a 7 segmentos, los 2 decodificadores 2 a 4, y la parte de selección de la señal FR lo implementaremos mediante Arreglos Genéricos Programables (GAL), con esto podremos ahorrar un poco de cableado y además poder programarlos de acuerdo a nuestras necesidades en específico.

Tenemos varias opciones para los programar los GAL's, como por ejemplo: OPAL, PALASM u VHDL. En esta ocasión utilizaremos OPAL ya que nos brinda la sencillez adecuada a nuestro proyecto.

Utilizaremos GAL's del tipo 20V8 – Se puede configurar como 20 Entradas con 8 salidas – V significa que puede ser utilizada para circuitos combinatoriales u secuenciales. Hay 16V8 y 22V10 entre otras.

Una restricción en un GAL es que una salida puede ser solamente la sumatoria de 8 términos, por lo tanto si nuestra función depende de más términos será necesario utilizar varias salidas, para este proyecto no fueron necesarios mas de 8 términos por salida.

Dentro del primer GAL programaremos dos decodificadores de 2 a 4. Por lo tanto ya estamos ocupando las 8 salidas. En nuestro segundo GAL programaremos el decodificador HEX7SEG2 donde con 4 entradas decodificaremos a 7 para manejar nuestro display de 7 segmentos. Además aprovechando utilizaremos otras 8 entradas y una salida para programar la señal FR.

Los códigos fuente (Extensión EQN) de los 2 GAL's son los siguientes:

DECO139.EQN

CHIP DECO139 GAL20V8

| | | | | | | | | | | | |
|-----|----|----|----|----|----|----|-----|-----|-----|-----|-----|
| ;1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| NC | NC | NC | NC | NC | NC | NC | B0I | B1I | A0I | A1I | GND |
| ;13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| NC | E | A3 | A2 | A1 | A0 | B3 | B2 | B1 | B0 | NC | VCC |

EQUATIONS

$A0 = /E * A1I + /E * A0I$
 $A1 = /E * /A0I + /E * A1I$
 $A2 = /E * /A1I + /E * A0I$
 $A3 = /E * /A0I + /E * /A1I$
 $B0 = /E * B1I + /E * B0I$
 $B1 = /E * /B0I + /E * B1I$
 $B2 = /E * /B1I + /E * B0I$
 $B3 = /E * /B0I + /E * /B1I$

HEX7SEG2.EQN

CHIP HEX7SEG GAL20V8

| | | | | | | | | | | | |
|-----|----|----|----|----|----|----|----|----|----|----|-----|
| ;1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| X1 | X2 | X3 | X4 | I1 | I2 | I3 | I4 | B2 | B1 | B0 | GND |
| ;13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| B3 | NC | B | A | G | F | C | D | E | FR | NC | VCC |

EQUATIONS

$A = /I2 * /I4 + /I1 * I3 + /I1 * I2 * I4 + I1 * /I2 * /I3 + I1 * /I4 + I2 * I3$
 $B = /I1 * /I3 * /I4 + /I1 * I3 * I4 + /I2 * /I4 + I1 * /I3 * I4 + /I2 * /I3$
 $C = /I1 * I2 + I1 * /I2 + /I3 * I4 + /I1 * /I3 + /I2 * I4$
 $D = I2 * /I3 * I4 + I1 * /I3 + /I1 * /I2 * /I4 + /I2 * I3 * I4 + I2 * I3 * /I4$
 $E = /I2 * /I4 + I3 * /I4 + I1 * I3 + I1 * I2$
 $F = /I3 * /I4 + /I1 * I2 + I1 * /I2 + I2 * I3$
 $G = /I2 * I3 + I1 * /I2 + I1 * I3 + I2 * /I3 * I4 + /I1 * I2 * /I4$
 $FR = /X4 * /B3 + /X3 * /B2 + /X2 * /B1 + /X1 * /B0$

Los esquemas generados por el compilador serían los siguientes:

| Chip diagram (DIP) DEC0139 | | | | Chip diagram (DIP) HEX7SEG | | | |
|-------------------------------|----|----|-----|-------------------------------|----|----|-----|
| NC | 1 | 24 | UCC | X1 | 1 | 24 | UCC |
| NC | 2 | 23 | NC | X2 | 2 | 23 | NC |
| NC | 3 | 22 | B0 | X3 | 3 | 22 | FR |
| NC | 4 | 21 | B1 | X4 | 4 | 21 | E |
| NC | 5 | 20 | B2 | I1 | 5 | 20 | D |
| NC | 6 | 19 | B3 | I2 | 6 | 19 | C |
| NC | 7 | 18 | A0 | I3 | 7 | 18 | F |
| B0I | 8 | 17 | A1 | I4 | 8 | 17 | G |
| B1I | 9 | 16 | A2 | B2 | 9 | 16 | A |
| A0I | 10 | 15 | A3 | B1 | 10 | 15 | B |
| A1I | 11 | 14 | E | B0 | 11 | 14 | NC |
| GND | 12 | 13 | NC | GND | 12 | 13 | B3 |

Ya que tenemos nuestros GAL's programados procedemos a armar nuestro prototipo en el protoboard siguiendo nuestro diseño esquemático y la programación de los GAL's.

Teniendo nuestro prototipo funcionando por completo ya podemos comenzar el diseño de lo que será el PCB (Printed Circuit Board). El procedimiento que seguimos para obtener nuestra placa para soldar los componentes fue el siguiente:

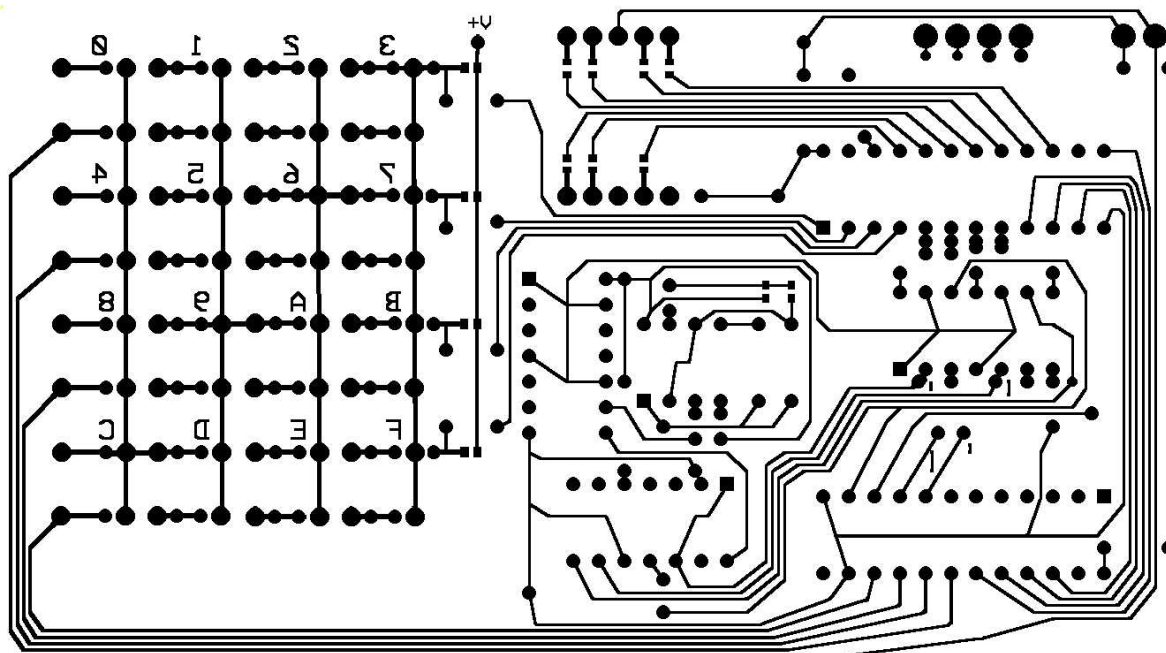
1° Diseñar nuestro diagrama con algún software, existe una gran variedad, por ejemplo: ExpressPCB, Protel, EAGLE, PCBDesign. En esta ocasión decidimos utilizar el ExpressPCB ya que tiene una interfaz fácil de entender aun cuando después tuvimos unos problemas que más adelante expondremos.

2° Ya que tenemos nuestro diseño por completo del esquemático debemos imprimirlo sobre papel herculene, debemos imprimir nuestro diseño de manera invertida (la mayoría de los programas lo hacen de forma automática), además debemos imprimirlo del lado opaco del papel herculene. Es importante que nuestra impresión sea en impresora láser.

3° Luego debemos limpiar nuestra placa de cobre, esto lo podemos lograr con al limpiar la placa con una lija de agua o con una fibra metálica. Cuando ya tengamos limpia la placa debemos transferir nuestro diseño a la placa, esto lo podemos hacer si calentamos el papel sobre la placa, cuando tengamos una temperatura adecuada, el toner comenzara a desprenderse del papel y se pegara a la placa.

4° Habiendo realizado la transferencia debemos corregir los errores que pudiéramos tener, esto lo podemos hacer con un plumón permanente de punta extra fina, o si es necesario lo podemos hacer con las plantillas que venden en tiendas de electrónica para corregir las pistas que pudieran tener defectos.

5° Cuando tengamos nuestra placa con el esquemático sin errores, podemos introducirlo a una bandeja que contenga cloruro férrico disuelto en agua, dependiendo de la concentración de nuestra solución será el tiempo que tarde el cloruro en disolver las partes del cobre que se encuentren sin cobertura ya sea de toner o plumón. Cuando esto suceda es suficiente con limpiar la placa y taladrar con brocas de 1/32" y 3/64".



ITEM CEM - Arquitectura Computacional I - Prof. Jorge Ramirez L - 422204-422201 - 11/05

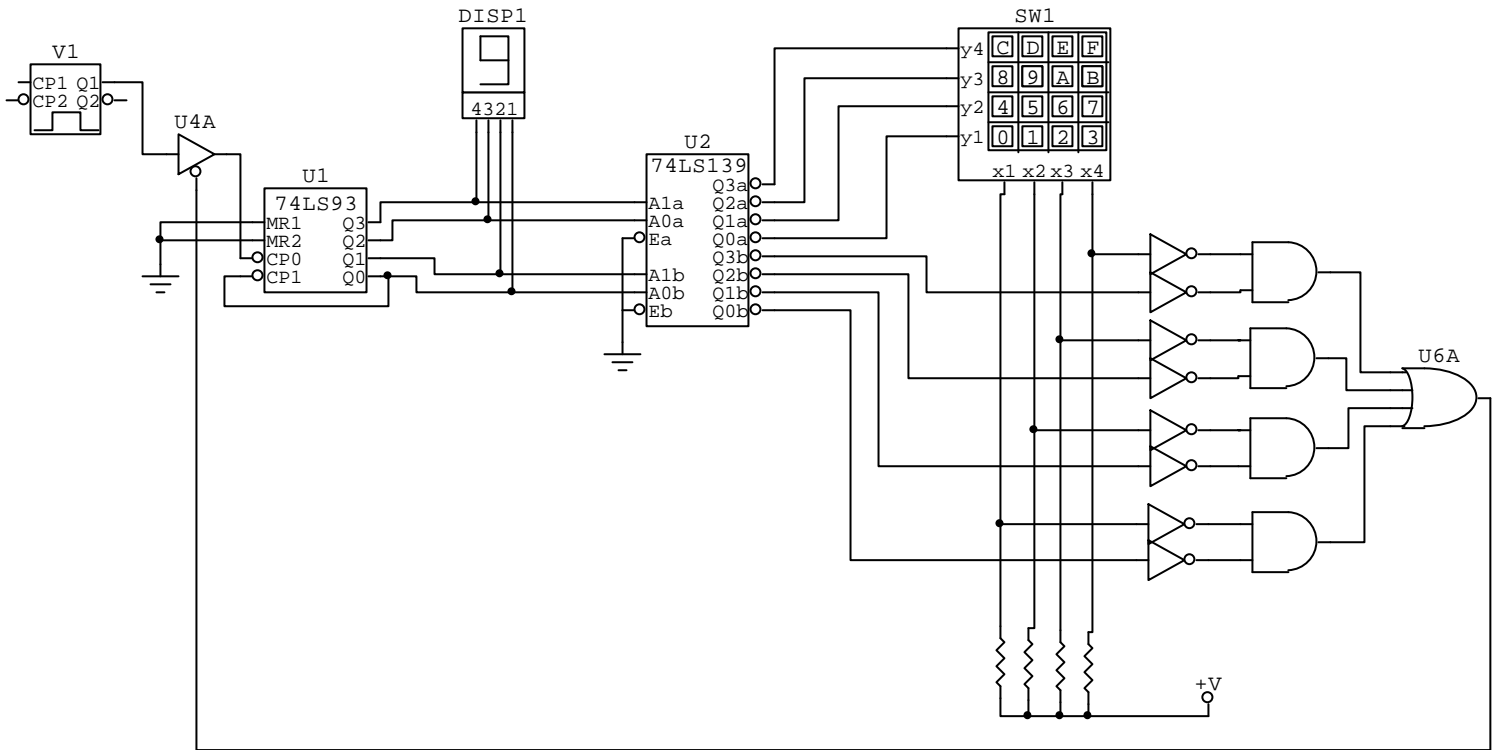
Llegando a este punto ya podemos comenzar a soldar nuestros componentes en la placa que previamente hemos preparado.

Lista de Materiales

| | |
|----|------------------------------|
| 2 | IC GAL20V8 |
| 2 | IC 74LS125 |
| 1 | IC 74LS93 |
| 1 | IC 555 |
| 7 | Resistencias 330Ω (SMD) |
| 5 | Resistencias 1KΩ (SMD) |
| 1 | Resistencia 100KΩ (SMD) |
| 16 | Push Buttons |
| 1 | Display de 7 Segmentos |
| 1 | Capacitor Electrolitico 10μF |
| 1 | Capacitor Ceramico 0.01μF |

Equipo y Herramienta:

- Cautín, Pinzas, etc.
- Impresora Láser,
- Multímetro, Generador de Funciones, Osciloscopio, Punta Lógica.
- Dremmel con Brocas de 1/32 y 3/64.
- Cloruro Férrico.



Fallas:

- A pesar de que el diseño en la PC con el ExpressPCB fue muy rápido y bueno, al momento de imprimirlo nos encontramos con que estábamos trabajando con una versión "Freeware" que al imprimir incluía un background de color gris que nos impedía transferirlo a la placa. Lo que hicimos fue hacer un "copy & paste" y modificar los pequeños errores con un editor de gráficos, solamente que es difícil coincidir con el tamaño real de los componentes, lo cual ocasiono un problema más adelante.
- Aparentemente ya estaba todo correcto, el problema siguiente se presentó al momento de transferir el esquema a la placa de circuito impreso, ya que cuando planchábamos la impresión, no medíamos el calor que aplicábamos y por consecuencia no quedaba, o calentábamos de más y el papel se quemaba o calentábamos poco y no se transfería por completo. Preferimos calentar un poco de menos, haciendo pruebas previas y obtuvimos una transferencia casi perfecta. Solucionando los defectos con pistas autoadheribles y un plumón permanente.
- Ya teníamos una buena transferencia y al momento de taladrar nos damos cuenta que se desajustó el tamaño de los pads donde van soldados los circuitos impresos. Por lo cual tendríamos que modificar la manera en que taladráramos. Este problema surge en el momento de hacer el "copy & paste" donde si bien algunas cosas si quedaron al tamaño adecuado los pads si sufrieron modificaciones lo cual nos ocasiono un problema posterior.
- Otro error fue en el diseño, al momento de no activar la salida de los 4 bits, solo cuando FR fuera 0, ya que si activábamos siempre la salida tendríamos resultados no deseados en el momento de aplicar nuestro teclado a cualquier dispositivo que lo requiriera, ya que nos entregaría todo el camino que el controlador sigue hasta el número que hemos presionado.

Conclusiones:

No siempre es conveniente querer realizar diseños lo más compactos ya que esto nos puede ocasionar problemas al momento de transferir el circuito y también en el momento de soldarlo ya que se llegan a unir las pistas y es difícil hacer las correcciones, además muchas veces ampliando el tamaño del diseño no es necesario utilizar tantos puentes, ya que en este diseño es muy común tener que hacer conexiones con el uso de puentes. Además es mejor hacer pistas más gruesas para evitar interrupciones en la continuidad de la pista.

Bibliografía:

Nelson, Victor and Nagle, Troy. Análisis y Diseño de Circuitos Lógicos Digitales. Prentice Hall. Primera Edición. México 1996.

Tocci, Ronald and Widmer, Neal. Digital Systems: Principles and Applications. Prentice Hall. Octava Edición. Columbus, Ohio 2001.

GAL20V8 Data Sheet. Lattice Semiconductor Corporation. <http://www.latticesemi.com>

74LS125 Data Sheet. QUAD 3-State Buffer. Fairchild Semiconductor Corporation. <http://www.fairchildsemi.com>

74LS93 Data Sheet. Decade Counter – Divide by eight – Motorola. <http://www.motorola.com/semiconductor/>

555 Data Sheet. Timer. National Semiconductor. <http://www.national.com>