4 2e1

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLAN

DISEÑO DE UN PROGRAMADOR DE MICROCONTROLADORES MC8748

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A

VICTOR HUGO ARROYO HERNANDEZ

Director de Tesis: ING. JOSE LUIS RIVERA LOPEZ

1990

V N A M

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

P R O L O G O .

La presente Tesis muestra en forma global diferentes arquitecturas para el diseño de circuitos microcontroladores, desde las más elementales, empleando circuitos Flip - Flop, pasando a hacer uso de memorias que almacenan instrucciones, hasta llegar al diseño utilizando un microcontrolador en un sólo encapsulado.

En el último capitulo se muestra el diseño de un programador de microcontroladores MC8748 de INTEL empleando una arquitectura basada en un microcontrolador semejante. Anexando en el Apéndice un listado de los programas propuestos para el funcionamiento del mismo.

Este programador puede ser útil para personas que cuenten con un programador de memorias y que deseen anexar a su equipo una interface para programar microcontroladores.

I.1. I N T R O D U C C I O N .

Desde hace tiempo se han construido máquinas que regulan su funcionamiento, es decir, una vez ajustadas éstas funcionan automáticamente; siendo sólo necesario proporcionarles material y energía, y lubricarlas.

Para muchos, la automatización comprende prácticamente todo adelanto tecnológico desde que el hombre creó herramientas por vez primera. Para otros sólo, se refiere a las formas más adelantadas y relativamente recientes. Todo depende del punto de vista de cada uno.

Una definición dice: "Automatización es el desplazamiento de cualquier trabajador mediante el uso de maquinaria". Prácticamente esto nos hace retroceder al invento de la rueda, porque ésta permitió al hombre sentarse y dejarse arrastrar en vez de caminar.

Definiéndola técnicamente: "Automatización es la sustitución con sistemas mecánicos, hidráulicos, neumáticos, electrónicos y eléctricos, de los órganos humanos de observación, decisión y

esfuerzo, a fin de incrementar la productividad, controlar la calidad y reducir el costo".

En algunos casos, el ingeniero y el especialista limitarán el uso de la palabra automatización a una estructura fundamental que requiere realimentación, de tal modo que un mecanismo discierna y corrija, o informe sobre los errores o cambios, a medida que ocurren.

La Automatización se aplica a diversas tareas en una gran variedad de formas:

- En un extremo se encuentran los mecanismos que trasladan automáticamente objetos y materiales de un lugar a otro, que cambian de forma estos objetos a medida que avanzan en su trayectoria, que se guían y mantienen por sí mismos y que corrigen su propio trabajo. Su función consiste en producir objetos físicos.

- Por otro lado, hay mecanismos conocidos como computadoras, éstas ejecutan con gran rapidez tareas lógicas y de toma de decisiones, ya sea rutinarias o complejas. Trabajan sobre valores intangibles, el hombre les suministra instrucciones sobre las

tareas que deben llevar a cabo, para analizar e interpretar datos complejos.

- Aunque estas dos divisiones de la Automatización estan diametralmente separadas, existe un área intermedia, que tiene igual o mayor importancia. En ellas se encuentran los sistemas mixtos, en los que las computadoras intervienen en las tareas de diseñar o controlar la producción de objetos materiales.

Refiriéndose a ésta última se encuentran los equipos conocidos como **"Controles Automáticos"**. Donde las máquinas electrónicas dirigen y controlan el funcionamiento de otras de un modo predeterminado. En esa forma es posible obtener una producción que se regula y corrige totalmente por sí misma. Intervienen en ella mecanismos perfeccionados que suministran controles sensoriales, de medición, de comparación de prueba, etc.

REALIMENTACION.

Un aspecto que es común en las formas de Automatización es la realimentación. Existe control de realimentación cuando la información sobre el resultado (cualquiera que sea la naturaleza que lo produce) se realimenta a una etapa anterior, de modo que influye en el procedimiento y por consiguiente cambia el resultado mismo.

I.2. CIRCUITOS SECUENCIALES

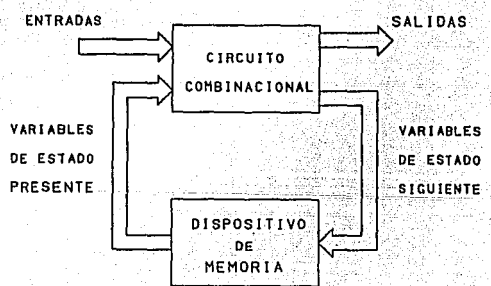
En algunos sistemas lógicos es necesario retener los valores de salida obtenidos, para que en combinación con los valores de entrada presentes se determinen las nuevas salidas, teniendo con esto una secuencia, por lo que a éstos se les conoce como: " SISTEMAS SECUENCIALES "; y tienen las siguientes caracteristi-- cas :

- Existe realimentación de las salidas del sistema a las entradas de éste.

- El sistema cuenta con la capacidad de almacenar información y así las señales de entrada y salida presentes son usadas para la determinación de la señal de salida posterior.

Un sistema secuencial es mostrado en la figura 1.

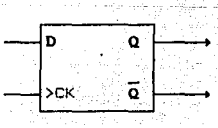
FIG. 1 SISTEMA SECUENCIAL.



El dispositivo de memoria puede ser implementado mediante circuitos especiales que pueden retener un estado binario indefinidamente (mientras que éste se encuentre polarizado). A estos circuitos se les conoce como " Flip-Flop " (F-F), que pueden ser del tipo asíncrono (aquéllos que varían conforme al cambio de sus entradas), o síncronos (cuya variación se efectúa de acuerdo a una señal de reloj).

Cada F-F será empleado como variable de estado para el sistema secuencial, conociéndose las salidas de estos como " Variables de Estado Presente " (V.E.P.), ya que es el estado actual en que se encuentra el sistema, y las entradas son conocidas como " Variables de Estado Siguiente " (V.E.S.).

En la figura 2 se muestra el símbolo lógico y su correspondiente tabla de transición de un F-F tipo " D ", con el que se trabajará posteriormente.



D	Q
0	0
1	1

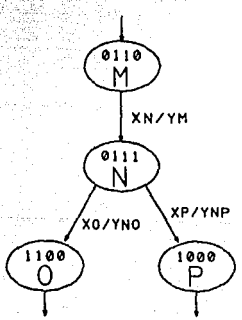
FIG. 2 FLIP-FLOP TIPO "D"

I.3. DIAGRAMAS DE ESTADO.

Las transiciones de los F-F's y la relación entre las V.E.P., las V.E.S., Entradas y Salidas involucradas en un sistema secuencial pueden ser analizadas empleando una técnica que se basa en el uso de " Diagramas de Estado ". Esta técnica, adecuada para el diseño, consiste en la representación gráfica de la secuencia condicional que se desea desarrollar; conociéndose a ésta como: " Diagrama de Estados " (D.E.).

La figura 3 representa la sección de un diagrama de estados con su respectiva información .

FIG. 3 DIAGRAMA DE ESTADO.



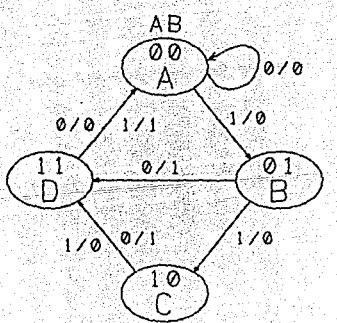
Donde cada estado es identificado por su V.E.P. (mediante un código binario) y etiquetado por una literal, la secuencia de transición entre estados es indicada por medio de líneas direccionadas junto a las cuales deberán de especificarse las condiciones de entrada 'X' a cada trancisión y las salidas 'Y' de cada estado.

De la fig. 3 'XN' es la condición a cumplir, para que ocurra la transición al estado 'N'. En el estado 'N' se tiene una bifurcación condicional, observándose una cuenta (de la variable de estado) al estado 'P' si se cumple que 'XP' sea verdadera o una carga al estado 'O' si 'XO' es verdadera; también se observa que la salida depende de la entrada.

I.4. DISEÑO DE CIRCUITOS SECUENCIALES.

Es conveniente en el diseño de circuitos secuenciales disponer la información del Diagrama de Estados en ' Mapas ' (de Karnaugh) para así realizar la simplificación del circuito combinacional (o emplear cualquier otro método de simplificación). La simplificación se realiza indicando, primeramente, en un Mapa las literales de los estados involucrados, conociendo a este Mapa con el nombre ' Mapa de Estado Presente ' ; por ejemplo, se desea el diseño de un circuito secuencial empleando F-F tipo 'D', teniendo el diagrama de estados mostrado en la figura 4, que describe su funcionamiento.

FIG. 4 DIAGRAMA DE ESTADO



Notar que las transiciones de A⟶B, B⟶C, C⟶D y D⟶A se efectúa una cuenta, mientras que en B⟶D se realiza una carga.

Su mapa de estado presente es:

A \ B	0	1
0	a	b
1	c	d

En el caso de no existir un estado definido para una combinación de las variables de estado, se indica colocando un asterisco '* ' en la respectiva casilla del mapa; queriendo decir con esto que no importa el valor que adquiera tal estado.

En otro mapa se indicará el estado al que se dirigirá, con su respectiva condición; a este mapa se le llama ' Mapa de Estado Siguiente ' o ' Mapa de Acción '. Para el ejemplo:

A \ B	0	1
0	1⟶b	1⟶c 0⟶d
1	d	a

Para el estado '00' si X = 1 se tendrá una transición al estado '01', y en caso contrario permanecerá en el mismo.

Para el estado '01' si X = 1 se dirige al estado '10', de lo contrario, es decir, si X = 0, al estado '11'.

Los estados '01' y '11' se transfieren incondicionalmente a los estados '10' y '00' repectivamente.

Adicionalmente se procede a llenar un Mapa de Acción para cada Variable de Estado.

Para la variable 'B'.

A \ B	0	1
0	X	$\bar{X}$
1	1	0

$$FB = X\,\bar{B} + \bar{X}\,X\,\bar{A} = X\,\bar{B}$$

Aquí se observa que para el estado '00' la variable B toma el valor de '1' al existir la transición y de '0' en caso de no ser así, por lo que su valor depende de 'X', para el estado '10' la variable B toma el valor de '1' sin importar el valor de 'X', similarmente para el estado '11' B tomará el valor de '0', mientras que para el estado '01' el valor de la variable B dependerá de $\bar{X}$.

Un análisis similar puede hacerse para la variable 'A'.

A \ B	0	1
0	0	1
1	1	0

$$FA = \bar{A}\,B + A\,\bar{B}$$

Es necesario además un mapa de salidas mediante el cual se obtendrá el circuito decodificador de las salidas.

Este mapa también se llena en función de la entrada.

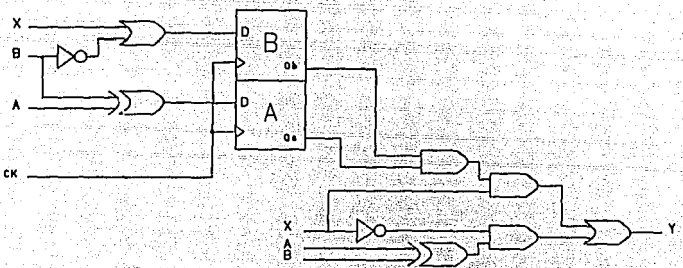
Mapa de Salida.

A \ B	0	1
0	0	X
1	X	X

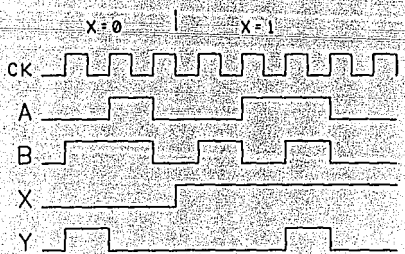
$$Y = A\,B\,X + A\,\bar{B}\,\bar{X} + \bar{A}\,B\,\bar{X}$$

La implementación electrónica se obtiene de los mapas de acción de cada variable, empleando tantos F-F's como variables de estado se encuentren involucradas quedando el sistema de la siguiente forma:

Diagrama eléctrico del circuito secuencial:



Su operación puede observarse mediante su diagrama de tiempos:



I.5. SISTEMAS DE CONTROL SECUENCIALES.

(AUTOMATAS)

Una de las más importantes aplicaciones de las técnicas digitales es el 'control', donde señales binarias son recibidas e interpretadas por un sistema digital y las salidas de control son generadas de acuerdo a la secuencia de aparición de las entradas.

Por lo que comúnmente son conocidos como 'Sistemas de Control Secuenciales'. En los que, para su diseño, resulta poco práctica la técnica empleada en el desarrollo de circuitos secuenciales, por lo que se usan arquitecturas de uso propuestas, configuradas con circuitos comerciales que realizan determinadas funciones preestablecidas. Evitando con esto el uso excesivo de circuitos integrados y de interconexiones externas.

Las arquitecturas de uso propuestas, que se trabajarán, se encuentran configuradas en base a los siguientes circuitos de mediana escala de integración.

- Multiplexores.
- Contadores.
- Registros de Corrimiento.

El multiplexor es un dispositivo que recibe información binaria de 2^n líneas y transmite una sola señal binaria, la única transmitida es seleccionada, por combinaciones binarias aplicadas en las 'n' líneas de selección, con que cuenta el multiplexor.

Un contador es un circuito secuencial que realiza un conteo binario determinado, mediante la aplicación de un tren de pulsos en su entrada de reloj.

Un registro es un grupo de Flip-Flop's y de compuertas lógicas que efectúa determinadas transiciones secuenciales de estado. La configuración básica de un 'Registro de Corrimiento' consiste en una cadena de Flip-Flop's conectados en cascada, es decir, la salida de cada Flip-Flop es conectada a las entradas del siguiente, estando todos activados por el mismo tren de pulsos de reloj.

I.6. MEMORIAS.

La 'memoria' es un dispositivo electrónico que puede almacenar estados binarios ('1' y '0'), conocidos como 'bit'.

Existen varios tipos de memorias, clasificadas en dos grupos:

■ ■ Las memorias de tipo 'no volátil', aquellas que retienen su información permanentemente después de ser programadas, sin la necesidad de estar polarizadas. A este grupo, pertenecen las cuatro memorias descritas a continuación:

■ Memoria de Solo Lectura (ROM 'Read Only Memory'). Estas son memorias que almacenan un conjunto fijo de información binaria, definida mediante la interconexión de enlaces internos que pueden ser de ruptura (abiertos) o fusionados.

Como su nombre lo indica, en estas memorias su información únicamente puede ser leída y no modificada; los datos son almacenados durante el proceso de fabricación, mediante una técnica conocida como 'máscara', la cual resulta costeable en el caso de fabricación a gran escala, con el mismo tipo de configu-- ración.

■ Por lo anterior se creó un nuevo tipo de ' ROM ' llamada 'Memoria Programable de Solo Lectura' (PROM 'Programmable Read Only Memory') la cual, al adquirirla tiene el valor de '1' en cada bit de todas las palabras contenidas, es decir, que todos los enlaces se encuentran fusionados; tales enlaces en la 'PROM' se rompen por medio de pulsos de corriente eléctrica, aplicados a través de las terminales de salida. Un enlace roto define un estado binario y uno no roto define el otro estado. El proceso de programación para ROM's y PROM's es irreversible, es decir, que el patrón dado es permanente y no puede alterarse.

■ Un tercer tipo, llamada EPROM' (Ereaser PROM), puede ser reestablecida a su configuración nativa, por exposición a luz ultravioleta, obteniendo un efecto de borrado.

■ Una innovación más reciente es la memoria 'EEPROM' (Electrical EPROM), que puede ser borrada por medios electrónicos.

■ ■ Las memorias de tipo volátil pierden la información almacenada en el transcurso de un período de tiempo, o cuando se le corta el suministro de energía.

■ La Memoria de Acceso Aleatorio (RAM 'Random Access Memory') es una memoria de lectura-escritura, se usa para almacenar datos, parámetros, variables y resultados intermedios, que necesiten renovación, y que estén sujetos a cambio.

Las 'RAM' pueden constar de un tipo de estructura de Flip-Flop's , a las que se les conoce como memorias estáticas o pueden estar conformadas por un tipo de estructura capacitiva, a las que se les conoce como memorias dinámicas.

Las memorias estáticas no pierden su información, salvo que no reciban un suministro de energía eléctrica constante, mientras que las memorias dinámicas, almacenan la información durante un lapso de tiempo, por lo que requieren de una señal de 'refresco' para mantener la información presente.

Existen las memorias 'Cuasiestáticas', que refrescan su información automáticamente, mediante un sistema contenido en el mismo encapsulado.

Tanto las memorias del tipo 'ROM' como las 'RAM' son de acceso aleatorio. Empero, la terminología común refiere a las 'ROM' como 'solo para leer' y a las 'RAM' como de 'acceso aleatorio'.

Una memoria consta de 'n' líneas de entrada y 'm' líneas de salida. Cada combinación de bits en las líneas de entrada se llaman 'Dirección', cada combinación de bits obtenidos en las líneas de salida se les conoce como 'Palabra'.

I.7. INTRODUCCION A LOS MICROPROCESADORES.

Físicamente, un microprocesador es un circuito integrado de alta escala de integración, comúnmente en un encapsulado de 40 patas (pin's). Es un circuito secuencial, activado por una o más señales de reloj, que cambia sus estados lógicos internos y sus señales de salida. Los pulsos de reloj son generados usualmente por circuitos externos adicionales. Cada cambio de 'estado' es determinado a su vez por el 'estado interno presente' y las señales de entrada aplicadas en ese instante. Esta descripción de su funcionamiento es similar a la de un flip_flop, radicando la diferencia en su complejidad, ya que existen muchas posibles combinaciones de las señales de entrada, muchos posibles estados internos y gran cantidad de posibles combinaciones de salidas.

La 'operación' del microprocesador, también conocido como Unidad Central de Procesamiento (C.P.U. del inglés), puede ser vista, sin embargo, en términos simples. Esencialmente es un conjunto de señales agrupadas para la repetición de un ciclo de estados internos. Este ciclo, 'ciclo de instrucción', siempre comprende los siguientes pasos:

i) Un conjunto de señales de entrada (una 'Instrucción') es leída ('Lach') a un registro interno ('Registro de Instrucción') del procesador.

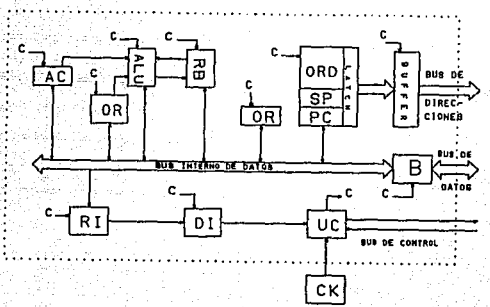
ii) El procesador realiza una secuencia de estados determinada por la combinación de bit's de la instrucción, y posiblemente involucrando la lectura de nuevas señales ('Datos') o la generación de señales de salida ('Resultados'). Esto se conoce como 'Ejecución' de la instrucción.

iii) Finalmente, un conjunto de señales de salida es generado ('la dirección de la instrucción siguiente'), que es usada por un dispositivo externo (normalmente de almacenamiento o 'memoria'), para determinar la siguiente instrucción que se pondrá a disposición del microprocesador.

I.7.1. ORGANIZACION INTERNA DE UN MICROPROCESADOR.

Un diagrama de bloques de la organización interna, en términos generales, es mostada en la figura 6.

FIG. 6 DIAGRAMA DE BLOQUES DE UN MICROPROCESADOR



Donde:

CK → Reloj.

Todas la operaciones del C.P.U. están activadas por sus señales de reloj; que pueden ser aplicadas por un circuito externo o generadas por una lógica interna, requiriendo únicamente de un cristal como referencia.

UC → Unidad de Control.

Las acciones de las C.P.U. deben estar sincronizadas. Por lo que, éste es un circuito secuencial síncrono.

Las señales de control internas son indicadas en la figura 25 por una 'C', estando distribuidas a todos los otros componentes internos de la C.P.U.

REGISTROS

La C.P.U. cuenta con dos tipos de registros; registros de propósito especial, aquéllos que tienen funciones definidas (contador de programa, registro de instrucción, etc.), y registros de propósito general, utilizados para el manejo de datos (acumulador, etc.).

PC → Contador de Programa.

El registro Contador de Programa contiene la dirección de localidad de memoria de donde la C.P.U. obtiene el código de la instrucción que deberá de procesar a continuación. Cada vez que se realiza un acceso a memoria para obtener un código de instrucción (ciclo 'Fetch') o un dato, el contenido del 'PC' se incrementa

en 1, tomando valores secuenciales, a excepción de que se ejecute una instrucción de salto, tomando entonces el valor de la dirección de la localidad de donde se efectúa el salto.

RI → Registro de Instrucción.

Se emplea para recibir el código de la instrucción que la C.P.U. obtiene durante el ciclo 'Fetch', y éste a su vez alimenta al Decodificador de instrucción.

DI → Decodificador de Instruccion.

El Decodificador de Instrucción es en principio un circuito combinacional, que no difiere de otros decodificadores. Sus salidas son usadas por la Unidad de Control para determinar el curso de ejecución del código contenido en el Registo de Instrucción.

SP → Apuntador de Stack.

Es una pila de almacenamiento de direcciones, indispensable para el uso de subrutinas e interrupciones. Basándose en la lógica:

'El primero en entrar, es el último en salir'.

ORD → Otros Registros de Direcciones.

Estos registros pueden ser usados para retener la dirección donde se realizará una lectura o escritura de datos. Por ejemplo; un 'Registro de Indice' está asociado con la operación de incremento y decremento, y puede ser usado para tener accesos a localidades de almacenamiento concecutivas.

OR → Otros Registros.

Aquí se incluyen los registros empleados para almacenar momentaneamente datos requeridos en una operación. Su número varía según el microprocesador que se esté considerando.

ALU → Unidad Aritmética y Lógica.

Esta unidad es la que realiza el trabajo de procesamiento. Recibe datos y efectúa con ellos operaciones aritméticas, lógicas, de comparación, corrimiento, entre otras.

AC → Acumulador.

El acumulador es el registro principal del microprocesador, ya que generalmente se opera con el contenido de éste registro. En

las operaciones que realiza la Unidad Aritmética Lógica, el acumulador contiene uno de los operandos, y los resultados de las operaciones obtenidas por la 'ALU' se almacenan aquí.

RB → Registro de Banderas.

Este registro almacena información generada en función de los resultados obtenidos en las operaciones que realiza la 'ALU'. Donde cada bit es una bandera, teniendo entre las más comunes: la bandera de cero, de paridad, de signo, acarreo, etc.

Latch.

Frecuentemente, la información se debe conservar durante cierto tiempo mientras espera ser leída. En situaciones como ésta se utilizan dispositivos conocidos como 'latch'. El 'latch' es un circuito (tal como los Flip-Flops) empleado para almacenar estados lógicos.

BUS.

Un 'Bus' (elemento de transporte) consiste en un grupo de líneas por las cuales se transfiere información de un registro a otro. Realizando tal transferencia siempre entre dos dispositivos,

mientras los otros que se encuentran conectados, deberán de comportarse como si no existieran.

El propósito fundamental de emplear el concepto de 'Bus', es reducir el número de líneas de conexión requeridas para la transferencia de información.

Su clasificación dependerá del tipo de información a transmitir:

1) Bus de Datos.
2) Bus de Direcciones.
3) Bus de Control.

B → Buffer.

El 'Buffer' es un circuito de acoplamiento. Un uso importante se da en los puertos de entrada y salida de datos, aquí los 'Buffers' sirven para aislar señales, enviadas por un dispositivo a través del 'Bus de Datos' al microprocesador, hasta que éste último las requiera, o viceversa.

Con el uso de 'Buffers' de tercer estado, se incrementa el número de entradas que se pueden conectar a los 'Bus' del

microprocesador, ya que mientras éste no indique lo contrario, el puerto permanecerá en tercer estado, en otras palabras, electricamente aislado del sistema.

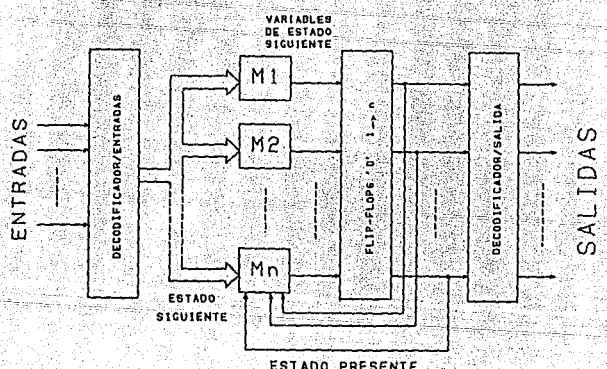
El 'Bus de Datos' se utiliza para enviar y recibir datos, por los que el 'Buffer' empleado debe de permitir el flujo de información en ambos sentidos, siendo formado por dos 'Buffers' de tercer estado, y un inversor en las líneas habilitadoras, para así asegurar que sólo uno de ellos trabaja a la vez.

II. DISEÑO DE CONTROLADORES EMPLEANDO MULTIPLEXORES.

Un grupo de multiplexores puede ser usado para implementar un sistema de control. Para lo cual se definirá una configuración general del sistema. La técnica empleada es conocida como 'Técnica de Implementación con Multiplexor Directamente Direccionado (MDD)'.

En la figura 7 se muestra un diagrama de bloques de un control implementado con multiplexores:

FIG.7 DIAGRAMA DE BLOQUES DE UN MICROCONTROLADOR



Como se observa la variable de estado siguiente (V.E.S.) se encuentra en función del estado presente (E.P.) de la máquina y de la palabra de entrada. Así, los Multiplexores decodifican el E.P. de la máquina y seleccionan la apropiada variable de entrada que a su vez determina el estado siguiente de la máquina.

Para definir los pasos requeridos en el diseño de controladores con multiplexores se empleará la arquitectura de trabajo propuesta en la figura 8:

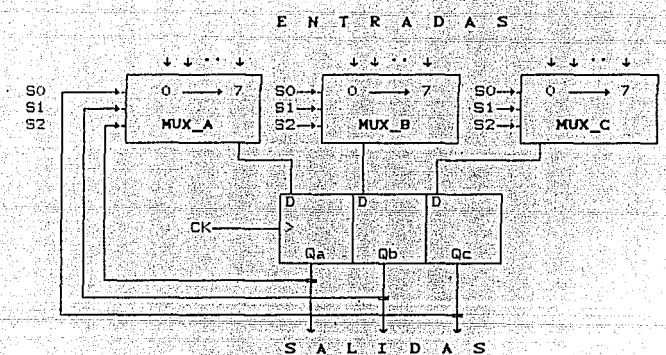
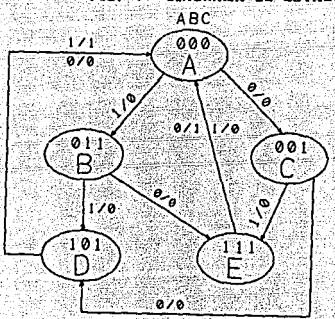


FIG. 8 ARQUITECTURA EN BASE A MULTIPLEXORES

y con auxilio de los ejemplos siguientes:

I.- Se tiene una máquina cuyo funcionamiento se representa por medio del siguiente diagrama de estados, figura 9. Diseñar un sistema para su control.

FIG. 9 DIAGRAMA DE ESTADO



X/Y → ENTRADA / SALIDA

El primer paso es llenar el mapa de estados presente. Para el ejemplo se tienen 3 variables de estado, por lo que el número de estados es 8.

Qa Qc \ Qb	00	01	11	10
0	a	*	*	*
1	c	b	e	d

Los estados no definidos son identificados por medio de un '*' indicando con esto que 'no importa' que valor tome.

Después se procede a llenar el mapa de estado siguiente. Que nos indica que camino se seguirá dependiendo de la Entrada.

Qa

Qc Qb

	00	01	11	10
0	1→b 0→c	*	*	*
1	0→d 1→e	1→d 0→e	a	a

De aquí se puede obtener el M.E.S. para cada variable.

Para A

Qa

Qc Qb

	00	01	11	10
0	0	0	0	0
1	1	1	0	0

Para B

Qa

Qc Qb

	00	01	11	10
0	X	0	0	0
1	X	X	0	0

Para C

Qa

Qc Qb

	00	01	11	10
0	1	0	0	0
1	1	1	0	0

A los estados no definidos se les asigna el valor de '0' para que la maquina se remita al inicio de la secuencia, en caso de presentarse alguno de estos estados.

Para obtener el circuito decodificador se llena el mapa de salidas, analizando las salidas del diagrama de estados.

Qa

Qc Qb

	00	01	11	10
0	0→0 1→0	*	*	*
1	0→0 1→0	0→0 0→0	0→1 1→0	0→0 1→1

por lo tanto:

Qa

Qc Qb

	00	01	11	10
0	0	*	*	*
1	0	0	X	X

la función de salida es:

$$y = AB\bar{X} + A\bar{B}X$$

o

$$y = A (B + X)$$

Por último se procede a indicar las entradas en la arquitectura propuesta de los multiplexores, con ayuda de los Diagramas de Estado de las Variables de Estado. Y se realiza la implementación electrónica.

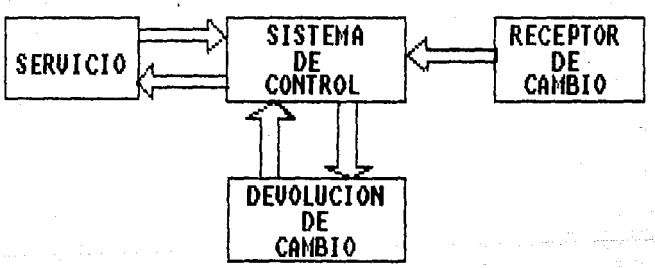
Las entradas aplicadas en los multiplexores son:

ENT.	Mux A	Mux B	Mux C
0	0	X	1
1	1	X	1
2	0	0	0
3	1	$\bar{X}$	1
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0

II.- Diseñar un dispositivo de control para una máquina despachadora. Que debe de efectuar las siguientes actividades:

1) Aceptación de monedas.
2) Devolución de cambio (en caso de que exista).
3) Dar servicio.

Para lo cual se cuenta con tres dispositivos, mostrados a continuación:



Con las especificaciones siguientes:

* RECEPTOR DE MONEDAS (RM).

1.- Entrada de una sola moneda a la vez.

2.- Detección electrónica de monedas.

3.- Detección garantizada de monedas de 5, 10, 25 y 50.

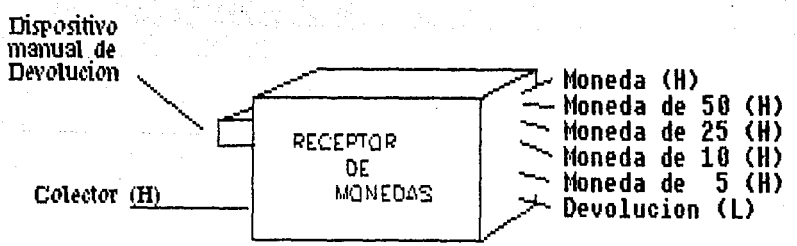
4.- Rechazo automático de monedas inválidas.

5.- Entradas y salidas compatibles con TTL.

6.- Mecanismo de captación de moneda y retiro manual.

7.- Prevención de saturación de monedas.

Diagramas de señales de entrada y salida.



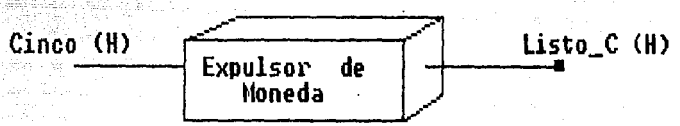
* DISPOSITIVO DE CAMBIO (DC).

1.- Sistema de eyección (electrónica) de monedas de 5.

2.- Salida ' L_C ' (Listo_Cambio), para indicar cuando se puede iniciar una nueva secuencia de eyección.

3.- Carga automática de 50 monedas de 5 para reserva de cambio.

DIAGRAMA DE SEÑALES.

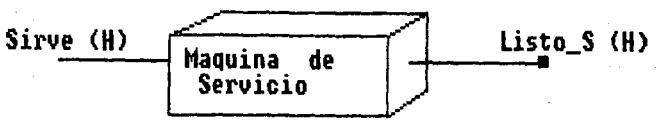


* MAQUINA DE SERVICIO (MS).

1.- Señal de servicio compatible con TTL.

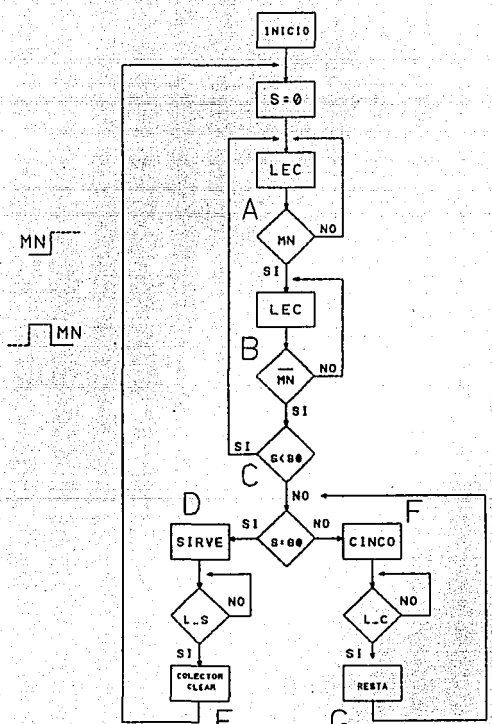
2.- Línea de estado ' L_S ' (Listo_Servicio).

DIAGRAMAS DE SEÑALES.



Por medio del siguiente diagrama de flujo se describen las actividades a realizar por el sistema de control.

FIG. 10 DIAGRAMA DE FLUJO



Una vez entendido el funcionamiento que deberá desempeñar el controlador se procede a seleccionar la técnica para el diseño.

Los pasos propuestos por la técnica " M.D.D. " son:

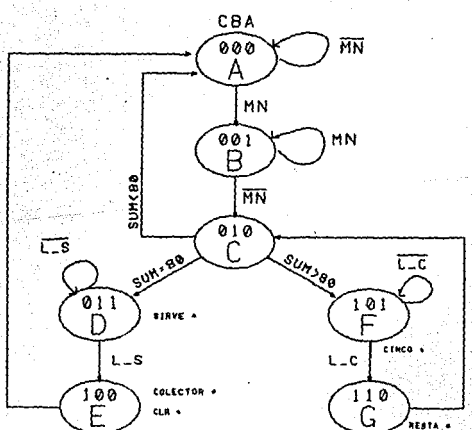
1.- Mapa de estado presente (del diagrama de estado presente).

2.- Mapa de estado siguiente.

3.- Mapa de las salidas.

4.- Implementación electrónica considerando las entradas dadas por el Mapa de Estado Siguiente.

FIG. 11 DIAGRAMA DE ESTADO DE LA MAQUINA EXPENDEDORA



+ Mapa de Estado Presente.

núm. de variables 3.

núm. de estados 8.

Qa \ Qb Qc	00	01	11	10
0	a	c	g	e
1	b	d	*	f

+ Mapa de Estado Siguiente:

Qa \ Qb Qc	00	01	11	10
0	MN→b	< →a = →d > →f	c	a
1	$\overline{MN}$→c	L_S →e	*	L_C →g

+ Mapa de Estado Siguiente para cada variable.

Para A

Qa \ Qb Qc	00	01	11	10
0	MN	>	0	0
1	MN	$\overline{L\_S}$	0	$\overline{L\_C}$

Para B

Qa \ Qb Qc	00	01	11	10
0	0	=	1	0
1	$\overline{MN}$	$\overline{L\_S}$	0	$\overline{L\_C}$

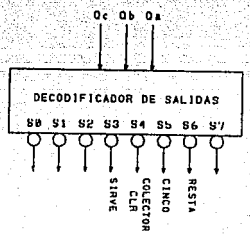
Para C

Qa \ Qb Qc	00	01	11	10
0	0	$\overline{<}$	0	0
1	0	$\overline{L\_S}$	0	1

+ Implementación Electrónica.

Las salidas son activadas por medio de un decodificador.

FIG. 12 ETAPA DE SALIDA



Las entradas aplicadas en los multiplexores son:

ENT.	Mux A	Mux B	Mux C
0	0	0	MN
1	0	$\overline{MN}$	MN
2	$\overline{<}$	=	>
3	L_S	$\overline{L\_G}$	$\overline{L\_S}$
4	0	0	0
5	1	L_C	$\overline{L\_C}$
6	0	1	0
7	0	0	0

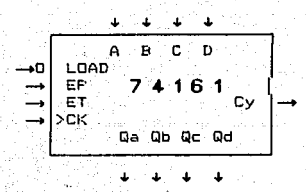
III. DISEÑO DE CONTROLADORES EMPLEANDO CONTADORES.

La implementación será en base a un contador síncrono de cuatro bit's con las siguientes características:

- CLR síncrono o asíncrono.
- Cuenta interna hacia adelante.
- Señal de habilitación de carga bit en paralelo.
- Señal de control de cuentas ('EP' y 'ET').

Su símbolo lógico se muestra en la figura 13.

FIG. 13 SIMBOLO LOGICO DE UN CONTADOR



La tabla de Acción:

EP	CARGA	ACCION
0	0	HOLD (preserva el estado)
0	1	CARGA
1	0	CUENTA
1	1	CARGA

Para el análisis se empleará la la arquitectura de trabajo mostrada en la figura 14.

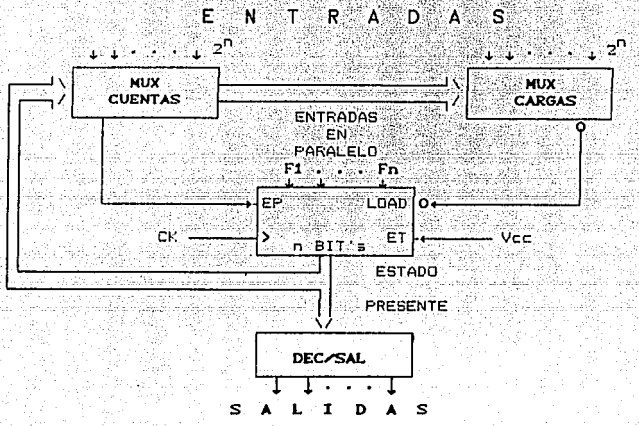


FIG. 14 ARQUITECTURA EN BASE A CONTADORES

La metodología de trabajo es:

1.- Elegir estados (en el D.E. procurando usar 'cuentas' o 'cargas')

2.- Mapa de Estado Presente.

3.- Mapa de Acción (Estado Siguiente).

4.- Mapa de Control de Modo.

5.- Mapa de Cargas en paralelo.

6.- Implementación electrónica.

Empleando el ejemplo I del capítulo anterior, para ejemplificar el análisis de diseño mediante esta técnica.

Definición de las acciones a realizar (instrucciones).

Brinco Condicional (BC)

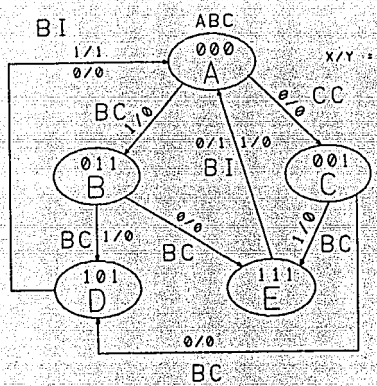
Brinco Incondicional (BI)

Cuenta Condicional (CC)

Cuenta Incondicional (CI)

+ Indicación de las acciones en el Diagrama de Estados, como lo muestra la figura 15.

FIG. 15 DIAGRAMA DE ESTADO



+ El Mapa de Estados Presentes.

Qa Qb / Qc	00	01	11	10
0	a	*	*	*
1	c	b	e	d

+ Mapa de Estado Siguiente (Mapa de Acción)

Qc \ Qa Qb	00	01	11	10
0	CC→c BC→b	*	*	*
1	BC→e BC→d	BC→e BC→d	BI→a	BI→a

+ Mapas de control de modo (definen la acción a realizar).

Considerando la tabla de acción.

Mapa de Cuenta (EP).

Qc \ Qa Qb	00	01	11	10
0	X	0	0	0
1	0	0	0	0

Mapa de Carga (Carga).

Qc \ Qa Qb	00	01	11	10
0	X	1	1	1
1	1	1	1	1

+ Mapas de cargas en paralelo:

Para Fa

Qc \ Qa Qb	00	01	11	10
0	0	0	0	0
1	1	1	0	0

$$Fa = \overline{Qa}\ Qc$$

Para Fb

Qc \ Qa Qb	00	01	11	10
0	X	0	0	0
1	X	X	0	0

$$Fb = Qa\ Qb\ X + Qa\ Qb\ Qc\ \overline{X}$$

Para Fc

Qc \ Qa Qb	00	01	11	10
0	1	0	0	0
1	1	1	0	0

$$Fc = \overline{Qb}\ (\ \overline{Qa} + Qc\ )$$

+ **Implementación:**

Señales Aplicadas a las entradas de los Multiplexores.

ENT	Mux. Cuenta	Mux. Carga
0	$\bar{X}$	X
1	0	1
2	0	1
3	0	1
4	0	1
5	0	1
6	0	1
7	0	1

El diagrama eléctrico de las entradas se muestra en la figura 16.

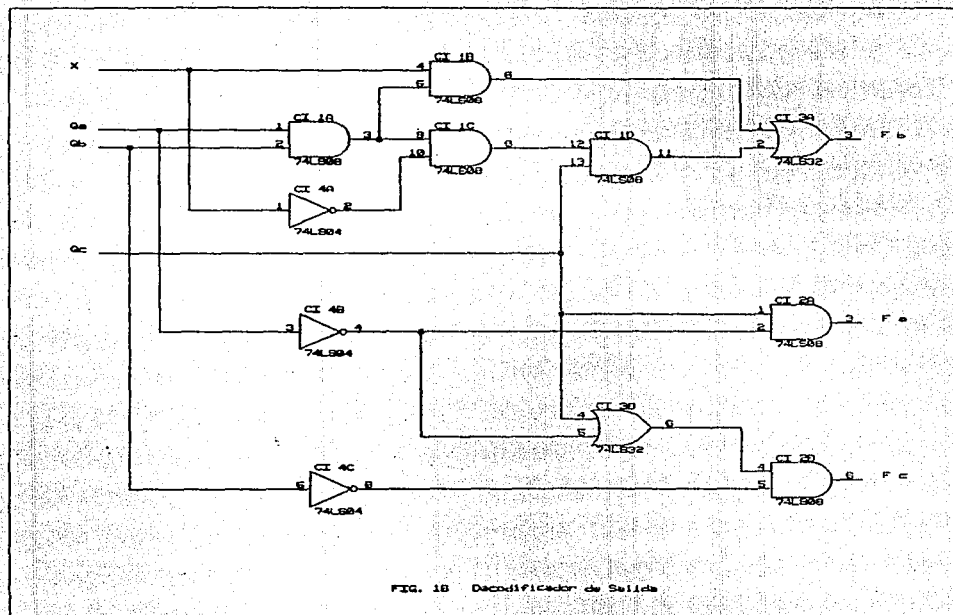
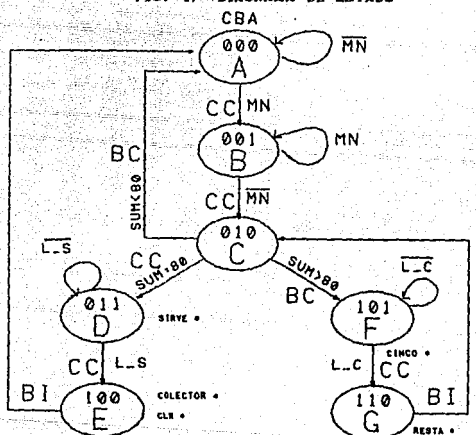


FIG. 18 Decodificador de Salida

Aplicando al ejemplo de la máquina expendedora.

+ El Diagrama de Estados indicando las instrucciones resulta de la forma mostrada en la figura 17.

FIG. 17 DIAGRAMA DE ESTADO



+ Mapa de Estado Presente.

Qa \ Qb Qc	00	01	11	10
0	a	c	g	e
1	b	d	*	f

+ Mapa de Acción.

Qa \ Qb Qc	00	01	11	10
0	CC/ MN →c	CC/= BC/>; BC/<	BI →c	BI →a
1	CC/_ MN →c	CC/ L_S →e	*	CC/ L_C →g

+ Mapas de Control de Modo. Tomando en cuenta la tabla de acción.

Mapa de Cuenta (EP).

Qa \ Qb Qc	00	01	11	10
0	MN	=	0	0
1	$\overline{MN}$	L_S	*	L_C

Mapa de Cargas.

Qa \ Qb Qc	00	01	11	10
0	0	≠	1	1
1	0	0	*	0

* Mapas de cargas en paralelo:

Para Fa

Qa \ Qb Qc	00	01	11	10
0	0	>	0	0
1	0	0	0	0

$$Fa = \overline{Qa}\ Qb\ \overline{Qc}\ (\ SUM>80\ )$$

Para Fb

Qa \ Qb Qc	00	01	11	10
0	0	0	1	0
1	0	0	0	0

$$Fb = Qc\ Qb\ \overline{Qa}$$

Para Fc

Qa \ Qb Qc	00	01	11	10
0	0	>	0	0
1	0	0	0	0

$$Fc = Fa$$

+ Implementación:

Señales Aplicadas a las entradas de los Multiplexores.

ENT	Mux. Cuenta	Mux. Carga
0	MN	0
1	MN	0
2	=	=
3	L_S	0
4	0	1
5	L_C	0
6	0	1
7	*	*

El diagrama eléctrico de las entradas se muestra en la figura 18.

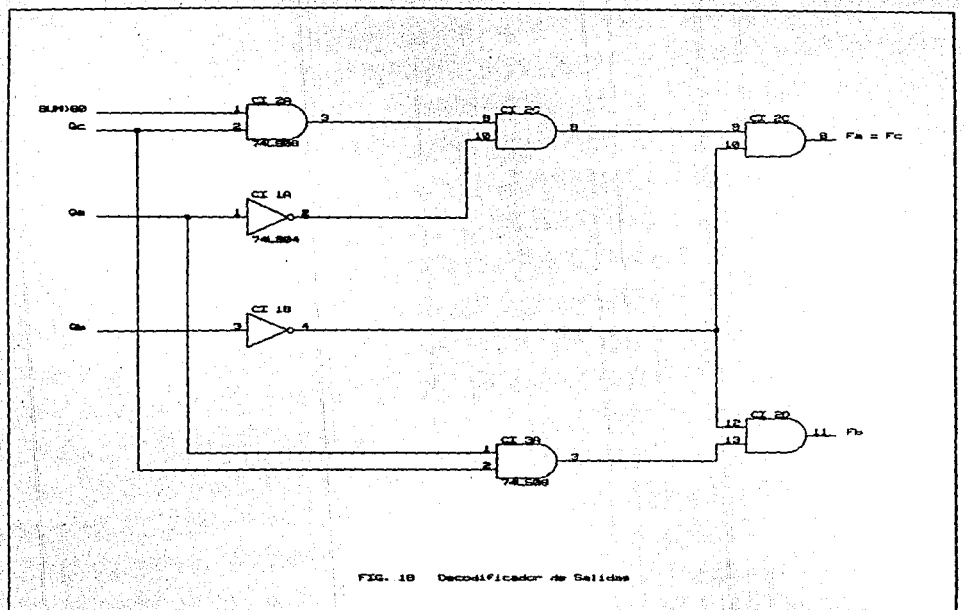
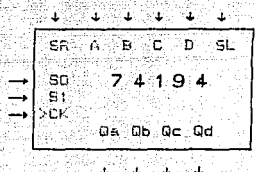


FIG. 18 Decodificador de Salidas

IV. DISEÑO DE CONTROLADORES EMPLEANDO REGISTROS DE CORRIMIENTO

Este sistema está basado en un registro de corrimiento de 4 bits, cuyo símbolo lógico se muestra en la figura 19:

FIG. 19 SIMBOLO LOGICO DE UN REGISTRO DE CORRIMIENTO

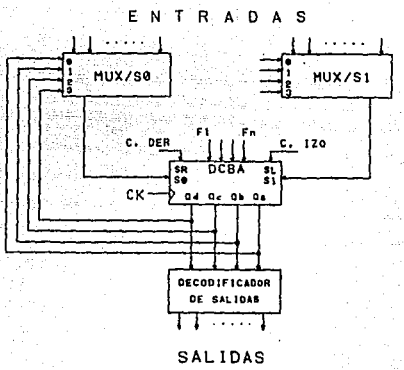


Su tabla de acción:

S0	S1	ACCION
0	0	Conserva el estado
0	1	Corrimiento/Derecha
1	0	Corrimiento/Izquierda
1	1	Carga en paralelo

La arquitectura de trabajo propuesta se muestra en la figura 20:

FIG. 20 ARQUITECTURA EN BASE A REGISTROS DE CORRIMIENTO



Instrucciones del Controlador.

CCD0 → Corrimiento/Derecha Condicionado (0 → 'SR')

CCD1 → Corrimiento/Derecha Condicionado (1 → 'SR')

CID0 → Corrimiento/Derecha Incondicionado (0 → 'SR')

CID1 → Corrimiento/Derecha Incondicionado (1 → 'SR')

CCI0 → Corrimiento/Izquierda Condicionado (0 → 'SL')

CCI1 → Corrimiento/Izquierda Condicionado (1 → 'SL')

CII0 → Corrimiento/Izquierda Incondicionado (0 → 'SL')

CII1 → Corrimiento/Izquierda Incondicionado (1 → 'SL')

BC → Brinco Condicional

BI → Brinco Condicional

Pasos a seguir en el diseño de controladores implementados con Registro de Corrimiento:

1.- Seleccionar las instrucciones a emplear para cada estado.

2.- Mapa de Estado Presente.

3.- Mapa de Acción.

4.- Mapa de Control de Modo.

5.- Mapa de Cargas en Paralelo.

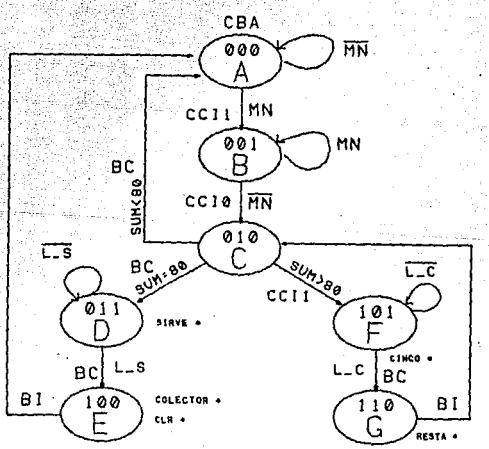
6.- Mapas para 'SR' y 'SL'.

7.- Implementación electrónica.

Aplicando estos pasos al ejemplo de la máquina expendedora.

+ Selección de instrucciones.

FIG. 21 DIAGRAMA DE ESTADO



Nota.- Para la selección de las instrucciones es necesario considerar la cuatro variables, ya que en los corrimientos puede afectar. Aún cuando en los análisis siguientes no intervenga.

+ Mapa de Estado Presente.

Qa \ QcQb	00	01	11	10
0	a	c	g	e
1	b	d	*	f

+ Mapa de Acción.

Qa \ QcQb	00	01	11	10
0	CCI1 MN	BC/< BC/= BC/>	BI →c	BI →a
1	CCI0 $\overline{MN}$	BC/ L_S →e	*	BC/ L_C →g

+ Mapas de Control de Modo. Tomando en cuenta la tabla de acción.

Mapa para 'S0'.

Qa \ QcQb	00	01	11	10
0	MN	1	1	1
1	$\overline{MN}$	L_S	*	L_C

Mapa para 'S1'.

Qa \ Qb	00	01	11	10
0	0	$\overline{\quad}$ >	1	1
1	0	L_S	*	L_C

+ Mapas de cargas en paralelo:

Para Fa

Qa \ Qb	00	01	11	10
0	*	=	0	0
1	*	0	*	0

$$Fa = \overline{Qc}\ \overline{Qa}\ (\ SUM=80\ )$$

Para Fb

Qa \ Qb	00	01	11	10
0	*	=	1	0
1	*	0	*	1

$$Fb = Qb\ \overline{Qa}\ (\ SUM=80\ ) + \overline{Qb}\ Qa$$

Para Fc

Qa \ Qb	00	01	11	10
0	*	0	0	0
1	*	1	*	1

$$Fc = Qa \left(\overline{Qb} + \overline{Qc} \right)$$

Nota.- Para el estado 'c'

SUM=80 → 011

Para Fd

$$Fd = 0$$

+ Mapas para las entradas 'SR' y 'SL'.

Para 'SL'

Qa \ Qb	00	01	11	10
0	1	1	*	*
1	0	*	*	*

$$F_{SL} = \overline{Qa}$$

Para 'SR'

No existe corrimiento a la derecha. Por lo tanto, no afecta lo que se tenga a la entrada.

+ Implementación:

Señales Aplicadas a las entradas de los Multiplexores.

ENT	Mux. S0	Mux. S1
0	MN	0
1	$\overline{MN}$	0
2	1	$\overline{S}$
3	L_S	L_S
4	1	1
5	L_C	L_C
6	1	1
7	*	*

El diagrama eléctrico de las entradas se muestra en la figura 22.

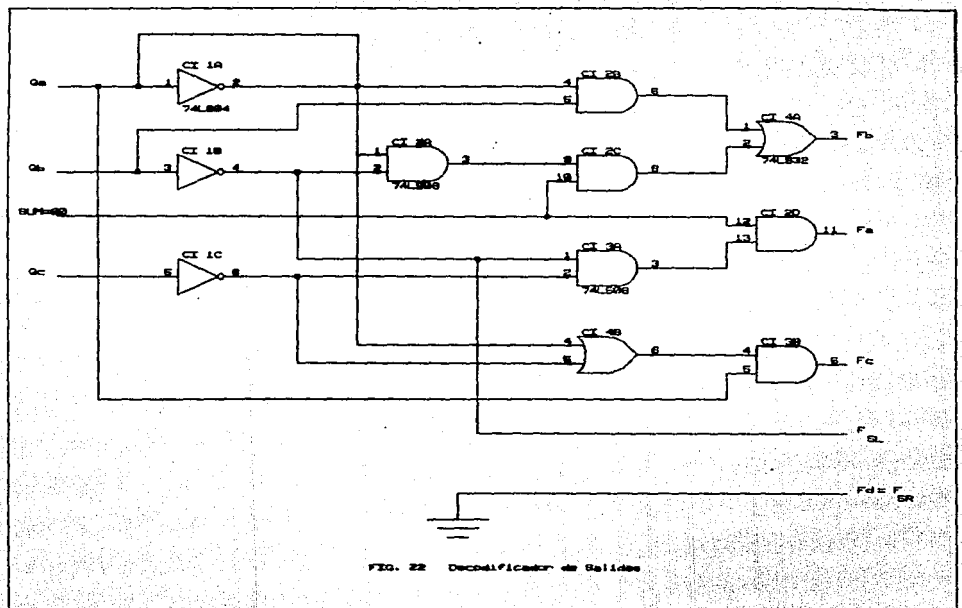


FIG. 22 Decodificador de Salidas

Ejemplo III.

Diseño de un sistema de control para un juego de luces secuenciales, que se active con las siguientes entradas y realice la operación indicada.

Variable de Entrada	Salida OOOO OOOO
Direccional Derecha (DD)	——→
Direccional Izquierda (DI)	←——
Reversa (REV)	←—→
Freno (FRENO)	←——→
Luz Intermitente (INTER)	o—o o—o

Empleando un controlador con registro de corrimiento.

* Para el presente diseño se consideran únicamente cuatro de las salidas, puesto que el estudio de las otras es similar.

Su Diagrama de Estado es:

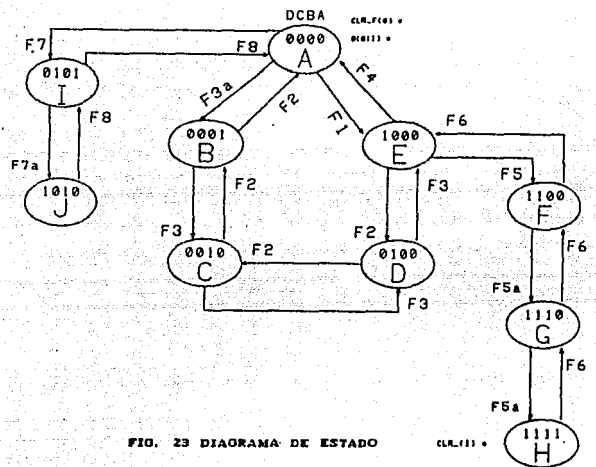


FIG. 23 DIAGRAMA DE ESTADO

Donde:

F1 = DD + (REV)$\bar{B}$ + (FRENO)$\overline{CLR\_F}$ ('CCD1')

F2 = DD + (REV)$\bar{B}$ ('CCD0')

F3 = DI + (REV)B ('CCI0')

$F3_a$ = DI + (REV)B ('CCI1')

F4 = DI + (REV)B + CLR_F ('CCI0')

F5 = (FRENO)$\overline{CLR\_F}$ ('BC')

$F5_a$ = $\overline{CLR\_F}$ ('CCD1')

F6 = CLR_F ('CCI0')

F7 = INTER ('BC')

$F7_a$ = INTER ('CCI0')

F8 = ('CID0')

F9 = $\overline{INTER}$ ('BC')

+ Mapa de Estado Presente:

Qd
Qb Qc
Qa

	00	01	11	10
00	a	d	f	e
01	b	i	*	*
11	*	*	h	*
10	c	*	g	j

+ Mapa de Acción:

Qb Qa \ Qd Qc	00	01	11	10
00	A	D	F	E
01	B	I	*	*
11	*	*	H	*
10	C	*	G	J

EDO.-	INS.	(CONDICION)	ESTADO SIGUIENTE
A →	CCI1	(DI+(REV)B)	→ b
	CCD1	(DD+(REV)$\overline{B}$+(FRENO)$\overline{CLR\_F}$)	→ e
	BC	(INTER)	→ i
B →	CCD0	(DD+(REV)$\overline{B}$)	→ a
	CCI0	(DI+(REV)B)	→ c
C →	CCD0	(DD+(REV)B)	→ b
	CCI0	(DI+(REV)$\overline{B}$)	→ d
D →	CCD0	(DD+(REV)$\overline{B}$)	→ c
	CCI0	(DI+(REV)B)	→ e

EDO. - INS. (CONDICION)	ESTADO SIGUIENTE
E → CCD0 (DD+(REV)$\overline{B}$)	→ d
CCI0 (DI+(REV)B+CLR_F)	→ a
BC ((FRENO)$\overline{CLR\_F}$)	→ f
F → CCD1 ($\overline{CLR\_F}$)	→ g
CCI0 (CLR_F)	→ e
G → CCD1 ($\overline{CLR\_F}$)	→ h
CCI0 (CLR_F)	→ f
H → CCI0 (CLR_F)	→ g
I → BC ($\overline{INTER}$)	→ a
CCI0 (INTER)	→ j
J → CID0	→ i

+ Mapas de Control de Modo:

Considerando la tabla de acción:

S0	S1	ACCION
0	0	Conserva el estado
0	1	Corrimiento/Derecha
1	0	Corrimiento/Izquierda
1	1	Carga en paralelo

EDO.	SO	S1
A	DI+(REV)B+INTER+(FRENO)CLR_F	DD+(REV)$\overline{B}$+INTER
B	DI+(REV)B	DD+(REV)$\overline{B}$
C	DI+(REV)B	DD+(REV)$\overline{B}$
D	DI+(REV)B	DD+(REV)$\overline{B}$
E	DI+(REV)B+FRENO+CLR_F	DD+(REV)$\overline{B}$+(FRENO)$\overline{CLR\_F}$
F	CLR_F	$\overline{CLR\_F}$
G	CLR_F	$\overline{CLR\_F}$
H	CLR_F	'0'
I	'1'	INTER
J	'0'	'1'

+ Mapas de Carga en Paralelo.

'A'

Qa \ Qb Qc	00	01	11	10
00	1	*	*	0
01	*	0	*	*
11	*	*	*	*
10	*	*	*	*

Qd

$Fa = \overline{Qd}\ \overline{Qc}$

B

Qd
Qb Qc
Qa

Qb Qa \ Qd Qc	00	01	11	10
00	0	*	*	0
01	*	0	*	*
11	*	*	*	*
10	*	*	*	*

Fb = 0

C

Qd
Qb Qc
Qa

Qb Qa \ Qd Qc	00	01	11	10
00	1	*	*	1
01	*	0	*	*
11	*	*	*	*
10	*	*	*	*

$Fc = \overline{Qc}$

D

Qd
Qb Qc
Qa

Qb Qa \ Qd Qc	00	01	11	10
00	0	*	*	1
01	*	0	*	*
11	*	*	*	*
10	*	*	*	*

Fd = Qd

+ Mapa de ' SR ' y ' SL '.

Qd
Qb Qc
Qa

	00	01	11	10
00	1	0	1	0
01	0	*	*	*
11	*	*	*	*
10	0	*	1	0

$$F_{SR} = \overline{Qa}\ \overline{Qb}\ \overline{Qd}\ \overline{Qc} + Qd\ Qc$$

Qd
Qb Qc
Qa

	00	01	11	10
00	1	0	0	0
01	0	0	*	*
11	*	*	0	*
10	0	*	0	*

$$F_{SL} = \overline{Qa}\ \overline{Qb}\ \overline{Qd}\ \overline{Qc}$$

V. CONTROLADORES MICROPROGRAMABLES

CONTROLADOR PROGRAMABLE.-

Es aquél que utiliza algún dispositivo de almacenamiento para contener instrucciones reconocidas por un sistema. Si el sistema por utilizar contiene lenguajes de alto nivel y/o simuladores se tiene la posibilidad de solucionar problemas más complejos.

El dispositivo de almacenamiento (memoria) permitirá la escritura de programas secuenciales, cuyas instrucciones (en un controlador universal) básicamente son:

a) Ejecución de la siguiente instrucción (condicional o incondicional).

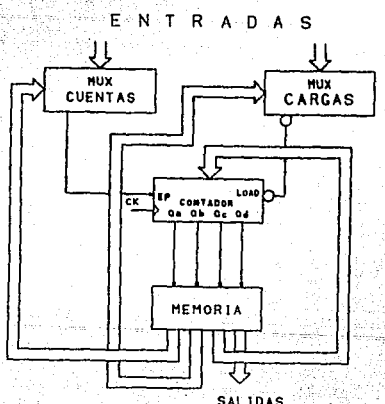
b) Brinco hacia una localidad de memoria específica (condicional o incondicional).

c) Desición entre la ejecución de la siguiente instrucción o brinco a una localidad específica.

El diseño de controladores microprogramables se analizará mediante una arquitectura básica, que únicamente permitirá efectuar una cuenta (incremento en 1 del contador de programa 'PC', Programm Counter) y un brinco (asignación de un nuevo valor al 'PC').

La arquitectura de un controlador microprogramable se muestra en la figura 24.

FIG. 24 DIAGRAMA DE BLOQUES DE UN CONTROLADOR MICROPROGRAMABLE



Es necesario un formato de las microinstrucciones válidas en ésta arquitectura.

FORMATO DE LAS MICROINSTRUCCIONES A UTILIZAR

Cuenta Condicional → ' **CC** ' (variables) , < salidas >

Cuenta Incondicional → ' **CI** ' < salidas >

Brinco Condicional → ' **BC** ' (variables) [dir] , < salidas >

Brinco Incondicional → ' **BI** ' [dir] < salidas >

Cuenta/Brinco Condicional → ' **C/B C** ' (variable_cta) ,
(variable_bco) [dir] ,
< salidas >

Los pasos a seguir en el diseño son:

1.- Identificar en el diagrama de estados las instrucciones respectivas.

2.- Elaborar el listado del microprograma.

3.- Asignación de variables (a la entrada de los multiplexores) en base al listado del paso anterior.

4.- Tabla del microprograma que será almacenado en la memoria.

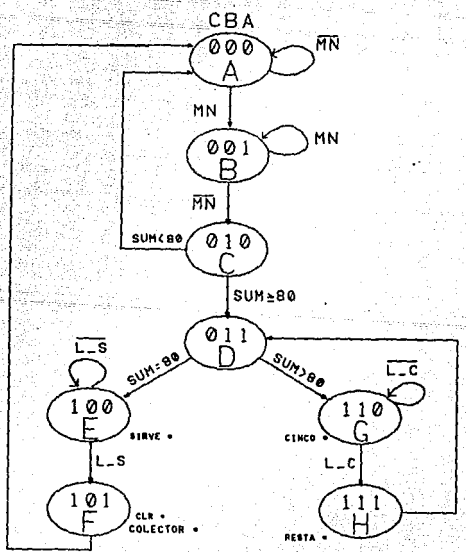
Empleando el ejemplo de la máquina expendedora:

+ Diagrama de Estados.

(Identificando las instrucciones a usar).

La figura 25 muestra la reconfiguración realizada para eliminar dos direcciones distintas de brinco en un sólo estado.

FIG. 25 DIAGRAMA DE ESTADO



+ Listado del Microprograma.

DIRECCION	EDO.	INSTRUCCION
00	a	CC (MN), < >
01	b	CC ($\overline{MN}$), < >
02	c	C/B C ($\overline{SUM<80}$),(SUM<80)Ia],< >
03	d	C/B C (SUM=80),(SUM>80)Ig],< >
04	e	CC (L_C), < sirve >
05	f	BI [a], < colector, CLR >
06	g	CC (L_C), < cinco >
07	h	BI [d], < resta >

+ La asignación de variables en las entradas de los multiplexores.

ENT.	MUX_CUENTA	MUX_CARGA
0	'1'(Vcc)	'1'(Vcc)
1	'0'(0 V)	'0'(0 V)
2	MN	SUM<80
3	$\overline{MN}$	SUM>80
4	$\overline{SUM<80}$	-
5	SUM=80	-
6	L_S	-
7	L_C	-

+ Tabla de Microcódigo.

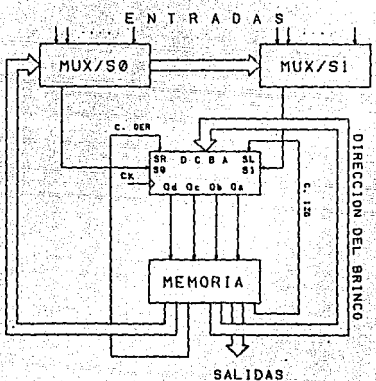
Formato de la 'palabra interna' de la memoria:

DIR	DIR MUX CTA	DIR MUX CGA	DIR DEL BRINCO	SALIDAS
BIT /	012	345	678	
00	010	001	000	00000
01	011	001	000	00000
02	100	010	000	00000
03	101	011	110	00000
04	110	001	000	10000
05	001	000	000	01100
06	111	001	000	00010
07	001	000	011	00001

(9) sirve
(10) colector
(11) CLR
(12) cinco
(13) resta

Aplicando esta técnica al ejemplo de las luces secuenciales, la figura 26 muestra la estructura de trabajo propuesta, empleando memorias y registros de corrimiento.

FIG. 26 ARQUITECTURA PROPUESTA



Considerando ésta arquitectura, se requiere de nuevas instrucciones válidas, cuyo formato está definido de la siguiente manera:

INSTRUCCION < variables > (dir (si existe brinco)), < salidas >

Las instrucciones son las ya definidas, para los sistemas de control configurados en base a registros de corrimiento. Pero considerando además las combinaciones entre ellas, creando instrucciones condicionales. Encontrandose éstas identificadas por una diagonal '/', que indica las posibles acciones. por ejemplo:

La instrución:

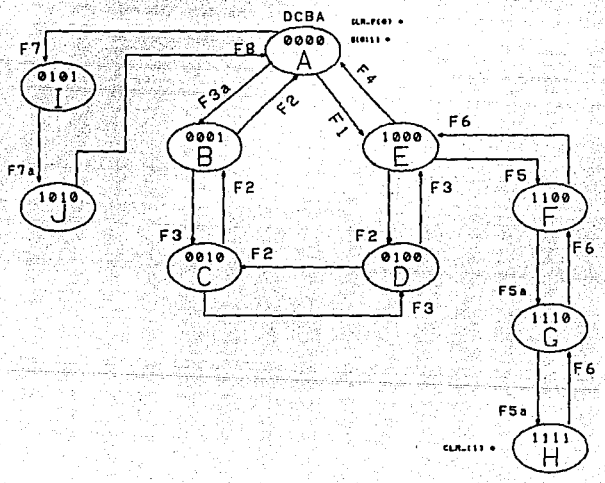
' CC I/D 1/0 /BC (x) , (y) , < > '

indica, que se realizará un corrimiento hacia la Izquierda, hacia la Derecha o un Brinco Condicional, dependiendo de los valores que tomen las variables de condición 'x' y 'y', considerando la tabla de acción del registro de corrimiento, e introduciendo un '1' por las entradas 'SL' o un '0' por 'SR' según sea el caso

DESARROLLO:

+ La identificación de las instrucciones en el Diagrama de Estados se muestra en la figura 27.

FIG. 27 DIAGRAMA DE ESTADO



Donde:

VARIABLES	INSTRUCCION
$F1 = DD + (REV)\overline{B} + (FRENO)\overline{CLR\_F}$	(CCD1)
$F2 = DD + (REV)\overline{B}$	(CCDO)
$F3 = DI + (REV)B$	(CCIO)
$F3_a = DI + (REV)B$	(CCI1)
$F4 = DI + (REV)B + CLR\_F$	(CCIO)
$F5 = (FRENO)\overline{CLR\_F}$	(BC)
$F5_a = \overline{CLR\_F}$	(CCD1)
$F6 = CLR\_F$	(CCIO)
$F7 = INTER$	(BC)
$F7_a = --$	(CIIO)
$F8 = --$	(BI)

+ Listado del microprograma.

EDO.	INSTRUCCION	
a	CC I/D 1 /BC	(DI+(REV)B+INTER), (DD+(REV)$\overline{B}$+INTER+FRENO)[i], < B($\uparrow_{\downarrow}$),CLR_F($\downarrow$) >
b	CC I/D 0	(DI+(REV)B),(DD+(REV)$\overline{B}$), < >
c	CC I/D 0	(DI+(REV)B),(DD+(REV)$\overline{B}$), < >
d	CC I/D 0	(DI+(REV)B),(DD+(REV)$\overline{B}$), < >
e	CC I/D 0 /BC	(DI+(REV)B+CLR_F+(FRENO)$\overline{CLR\_F}$), (DD+(REV)B+(FRENO)$\overline{CLR\_F}$)[f], < >
f	CC I/D 0/1	(CLR_F),($\overline{CLR\_F}$), < >
g	CC I/D 0/1	(CLR_F),($\overline{CLR\_F}$), < >
h	CCIO	(CLR_F),< $\overline{CLR\_F}$($\uparrow$) >
i	CIIO	[j], < >
j	BI	[a], < >

IMPLEMENTACION:

+ Asignación de entrada a los multiplexores.

ENT.	MULTIPLEXOR	
	'S0'	'S1'
0	DI+(REV)B+INTER	DD+(REV)$\overline{B}$+INTER+FRENO
1	DI+(REV)B	DD+(REV)$\overline{B}$
2	DI+(REV)B+CLR_F+(FRENO)CLR_F	DD+(REV)$\overline{B}$+(FRENO)$\overline{CLR\_F}$
3	CLR_F	$\overline{CLR\_F}$
4	CLR_F	0
5	1	0
6	1	1
7	-	-

+ Las salidas corresponden a las líneas de dirección.

Los valores de las salidas deberán ser retenidos por un elemento de memoria.

+ Tabla de Microcódigo.

Formato de la 'palabra interna' de la memoria:

DIR	DIR MUX CTA	DIR MUX CGA	SALIDAS
BIT	012	3 4	5678
00	000	1 1	0101
01	001	0 0	0000
02	001	0 0	0000
03	110	0 0	0000
04	001	0 0	0000
05	101	0 0	0000
06	110	0 0	0000
07	101	0 0	0000
08	010	0 0	1100
09	110	0 0	0000
10	110	0 0	0000
11	110	0 0	0000
12	011	0 1	0000
13	110	0 0	0000
14	011	0 1	0000
15	100	0 0	0000

VI. DISEÑO DEL PROGRAMADOR DEL MC8748

VI.1. REQUERIMIENTOS.

El Programador de MC8748 deberá:

- Coordinar la lectura de los datos suministrados mediante un teclado hexadecimal y la decodificación de los mismos a siete segmentos para ser desplegados a un display de tres dígitos.
- Estos datos serán interpretados como la longitud del programa en bytes.
- Controlar el direccionamiento y la lectura del código de instrucción contenido en una memoria (MCM2716).
- Generación de los trenes de pulsos requeridos para la programación de la memoria del MC8748.

Para el control del programador con los requerimientos anteriores resultan ineficientes los controladores implementados con una arquitectura como las propuestas anteriormente, a causa de la cantidad de variables de entrada y salida que se requiere operar, sin mencionar la gran variedad de estados que deberá trabajar. Considerando esto y analizando las características de los microprocesadores, se propone una arquitectura de trabajo implementada mediante un microcontrolador de éste tipo, y en especial de MC8748, ya que se trata de un sistema completamente integrado en un encapsulado, que simplifica considerablemente su utilización al eliminar gran cantidad de conexiones externas que se requerirían de no contener su memoria y reloj integrado.

VI.2 CARACTERISTICA DEL MICROCONTROLADOR MC8748.

Esta computadora de 8 bits incluye 70 instrucciones, 21 para saltos condicionales, 27 líneas de entrada y salida, un reloj interno, 1 k-byte de memoria EPROM, una memoria RAM de 64 bytes, ambas con capacidad de expansión.

El MC8748 es empleado con el propósito de desarrollo, es decir, para el diseño, mientras que para la producción en serie se emplea el MC-8048, que difiere en la memoria ROM que sustituye a la EPROM del primero.

La configuración de las patas del MC-8748 se muestra en la figura 28.

FIG. 28 CONFIGURACION DEL MC8748

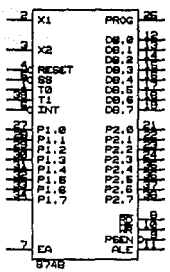
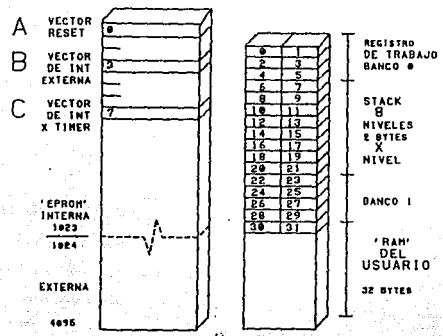


FIG 29 MAPA DE MEMORIA PARA PROGRAMA 'EPROM' Y DE DATOS 'RAM'.

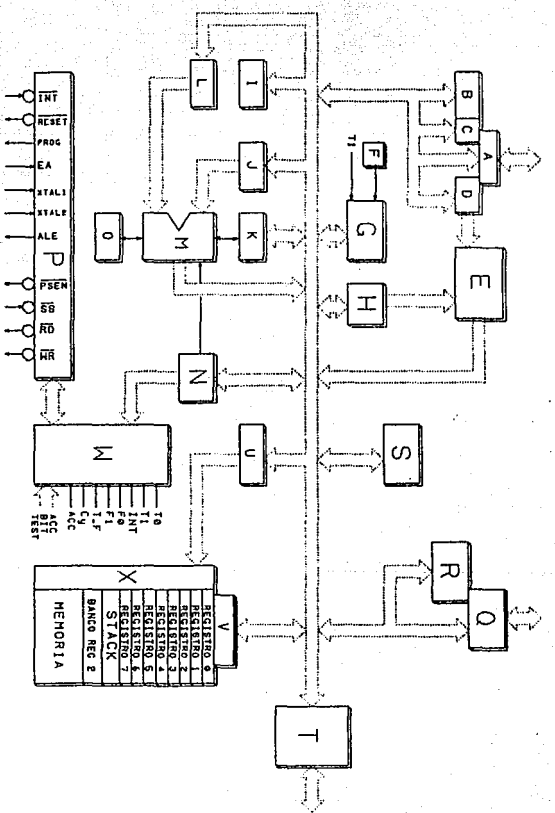


A → Cuando de activa la línea $\overline{\text{RESET}}$ el 8748 realiza el primer FETCH en la dirección '0000'.

B → Al activarse la señal $\overline{\text{INT}}$ del 8748 se genera una llamada a la subrutina que comienza en la dirección '0003', es decir, la primera instrucción de la subrutina a ejecutarse deberá colocarse aquí (generalmente se escribe un JMP) y el final de la subrutina debe ser una instrucción del tipo RETR o RET.

C → Cuando existe un overflow (sobre flujo) de TIMER se genera una llamada a subrutina en la dirección '0007' (debiendo también ser la última instrucción de la subrutina un RETR o RET.

FIG. 30 DIAGRAMA DE BLOQUES



VHAH

A ≡ BUFFER PUERTO 2.

B ≡ PUERTO 2 (PARTE BAJA) Y EXPANSION DE E/S

C ≡ PUERTO 2 (PARTE ALTA)

D ≡ CONTADOR DE PROGRAMA (PARTE ALTA)

E ≡ "EPROM" 1K X 8

F ≡ OSCILADOR

G ≡ TIMER

H ≡ CONTADOR DE PROGRAMA (PARTE BAJA)

I ≡ ACUMULADOR

J ≡ REGISTRO TEMPORAL

K ≡ REGISTRO DE BANDERAS

L ≡ LATCH DE ACUMULADOR

M ≡ UNIDAD ARITMETICA LOGICA

N ≡ REGISTRO DE INSTRUCCIONES Y DECODIFICADOR

O ≡ AJUSTE DIRECCIONAL

P ≡ UNIDAD DE CONTROL

Q ≡ BUFFER - BUS

R ≡ LATCH DEL BUS

S ≡ REGISTRO DE ESTADO

T ≡ PUERTO 1 - BUS - BUFFER - LATCH

U ≡ REGISTRO DE DIRECCIONES "RAM"

V ≡ MULTIPLEXOR

W ≡ LOGICA DE SALTO CONDICIONAL

X ≡ DECODIFICADOR

VI.2.1. DESCRIPCION DE LAS LINEAS DE ENTRADA/SALIDA.

PUERTO 1 ($P_{10} \rightarrow P_{17}$) Y PUERTO 2 ($P_{20} \rightarrow P_{27}$).

Características:

* Ambos tienen características idénticas.
* Cuando operan como salidas éstas son del tipo "Latch".
* Como puertos de entrada no son tipo "Latch" y por tanto las entradas deben estar presentes hasta que se efectue la lectura de datos.
* Las entradas deben ser "TTL" y las salidas manejan una sola carga también de este tipo.
* Los datos pueden ser "Mascarables" con instrucciones ANL y ORL.

BUS ($DB_0 \rightarrow DB_7$).

Características:

* Es un puerto bidireccional de 8 bits asociado a los habilitadores de entrada y salida RD y WR respectivamente.
* Si la característica bidireccional no se necesita, el BUS se puede utilizar como puerto de salida tipo "Latch" o puerto de entrada "No Latch".
* Las líneas del puerto son de tercer estado.
* El modo tipo "Latch" (INS, OUTL) es empleado cuando el BUS no se usa como puerto de expansión. Las instrucciones OUTL y MOVX pueden ser mezcladas si es necesario. Empero, una salida

previamente "Latcheada" será destruida por la ejecución de una instrucción MOVX y el BUS tomará un estado de alta impedancía. Por lo que el uso de MOVX después de un OUTL para poner el BUS en alta impedancía es necesario antes de ejecutar una INS para leer una palabra externa.

Como puerto de salida → OUTL → genera el pulso WR.

Como puerto de entrada → INS → genera el pulso RD.

Cuando no se lee ni se escribe el puerto BUS permanece en estado de alta impedancía.

Entradas TEST e INT.

Son tres pines que pueden usarse como variables condicionales con instrucciones del tipo JMP. Permiten realizar saltos sin tener que hacer una lectura de puerto.

VI.2.2. INTERRUPCIONES.

Una interrupción se inicia al aplicar un nivel "0" en el pin INT (si y sólo si se encuentra habilitado el modo de interrupciones).

El sistema de interrupciones es de un sólo nivel, ello implica que si durante la ejecución de una interrupción aparece otra petición del mismo tipo se ignora hasta que se ejecute la primera.

Lo mismo sucede si la segunda petición es interna (por el overflow del TIMER).

En caso de que una interrupción por TIMER ocurra al mismo tiempo que una externa, se le dará prioridad a ésta última. La interrupción por TIMER es tipo "latch" y por tanto se le dará servicio posteriormente.

Instrucción **EN I** → activa interrupciones.

Instrucción **DIS I** → desactiva interrupciones.

El pin INT también puede supervisarse con la instrucción **JNI**

Si no se utilizan interrupciones INT puede emplearse como entrada tipo T0 y T1.

VI.2.3. TIMER.

Es un contador de 8 bits, con el que se pueden generar (tiempos de espera) loops de tiempo sin sobrecargar al procesador.

Como contador, el TIMER puede inicializarse con:

MOV T, A ; T ← A

MOV A, T ; A ← T

El contenido del contador no se afecta por RESET, sólo por instrucción.

Existen dos medios para detener la operación del contador:

STOP TCNT ; por instrucción

RESET ; físicamente

y dos formas para arrancar el contador:

```
STRT      ; por instrucción, incrementa la
            cuenta cada 32 ciclos.
STRT CNT  ; por instrucción. El contador
            se incrementa con cada transi-
            ción ALTO-BAJO del pin T1.
```

La interrupción por TIMER puede habilitarse o deshabilitarse respectivamente con las instrucciones siguientes:

```
EN  TCNTI ;
DIS TCNTI ;
```

VI.2.4. PC Y STACK.

EL PC utiliza sólo 10 bits para direccionar 1024 bytes del "EPROM" interno, los dos bits más significativos del PC, permiten realizar "FETCHS" A "EPROMs" externos.

El Stack se accesa a través de un apuntador "Stack Pointer" de 3 bits. Durante interrupciones o llamadas a subrutinas, el valor del PC se guarda en un par de registros del Stack (que contienen 16 bytes, es decir, hasta 8 anidaciones de subrutinas).

El Stack inicia en las localidades 8 y 9 de la "RAM". En caso de sobreflujo se reescribirá en éstas localidades.

VI.2.5. REGISTRO DE ESTADO (PROGRAM STATUS WORD "PSW").

Es una palabra de 8 bits que puede ser cargada a y del acumulador.

Los tres primeros bits menos significativos conforman el Stack Pointer, por lo que es afectado por las instrucciones CALL y RETR, y al efectuarse una interrupción.

Los bits del PSW se definen:

0 → 2 ≡ Stack Pointer.

3 ≡ No usado.

4 ≡ Selector de Bancos de Registros.

0 → Banco 0

1 → Banco 1

5 ≡ Puede ser accesada por el usuario, trabaja con la instrucción de salto condicional JF0.

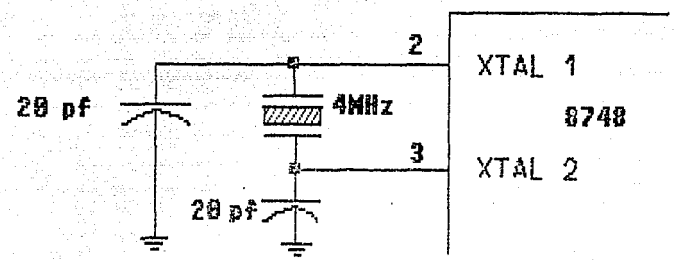
6 ≡ Carry auxiliar.

7 ≡ Carry (Bit de acarreo).

VI.2.6. RELOJ INTERNO.

El 8748 cuenta con un reloj interno que sólo necesita una referencia de frecuencia (Cristal, Inductor o Reloj Externo) para activar dicho circuito resonante de alta ganancia, con un rango de frecuencia de 1 a 6 MHz.

Configuración Básica del Reloj.



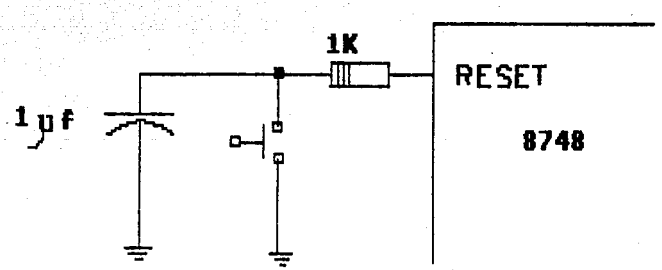
VI.2.7. RESET.

Permite la reinicialización del procesador mediante el Pin del mismo nombre, activo en 0 que debe permanecer por lo menos 30 mseg. a tierra. después que la fuente de poder esté dentro del rango de tolerancia. Si el sistema ya se encuentra energizado sólo necesita de 5 ciclos de máquina.

Efectos del RESET.

- PC ← 00000000
- SP ← 000
- Selecciona Banco 0 de registros.
- Selecciona Banco 0 de memoria.
- BUS en alta impedancia (excepto cuando EA = 5v).
- Puertos 1 y 2 se fijan en modo de entrada.
- Deshabilita interrupciones externas y de TIMER.
- Detiene el contador de TIMER.
- Bandera de Sobreflujo de TIMER a 0.
- F0 y F1 en 0.
- Deshabilita la salida del reloj por T0.

Configuración Básica del RESET.



VI.3. PROGRAMACION DEL MC8748.

El proceso de programación consiste en:

- Activar el modo de programación.
- Aplicar la dirección (que será almacenada en modo latch) del dato a programar,
- Aplicar el dato en el BUS, y
- Los pulsos para grabar el dato.

Los pin's empleados son:

XTAL1 ⟶ Reloj de 3 - 4 MHz.

XTAL2 ↗

RESET ⟶ Inicialización y almacenamiento de la dirección (0 Vcc).

T0 ⟶ Selecciona el Modo de Programación (0 Vcc).

EA ⟶ Activa el Modo Programa (Vcc - 18.5 V)

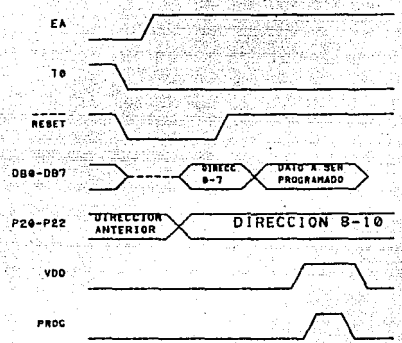
BUS ⟶ Puerto de entrada para la Dirección y el Dato.

P_{20}-P_{22} ⟶ Puerto de la entrada de la Dirección (2 BMS).

VDD ⟶ Fuente suplementaria (5.25 - 21.5 V).

PROG ⟶ Pin donde se aplica el pulso de Programación (Vcc - 18.5 V).

FIG. 31 DIAGRAMA DE TIEMPOS.



Corrientes que deberán soportar las fuentes de poder:

I_{DD} = 20 mA

I_{PROG} = 1 mA

I_{EA} = 1 mA

NOTA.- Un intento para programar un MC8748 mal configurado ocasionaría severos daños.

La indicación de una correcta configuración es la presencia de la señal de reloj ALE. La ausencia de ésta, puede emplearse para deshabilitar el programador.

Configuración de la base del integrado antes de ser insertado éste:

V_{DD} = 5 V

RESET = 0 V

T0 = 5 V

EA = 5 V

P_{10} - P_{11} = 0 V

Señal de Reloj - funcionando

Antes de retirar el MC8748, el programador deberá estar en un estado semejante al anterior, a excepción del pin EA que deberá tener 18 V aplicado.

VI.4. ARQUITECTURA Y PROGRAMAS PROPUESTOS.

El Programador de Microcontroladores consta de una etapa de circuiteria y otra de instrucciones propiamente dicho. Tales instrucciones constituyen rutinas que conforman el programa central de proceso, cada rutina tiene una actividad especifica y por lo general controla una etapa de la circuiteria.

VI.4.1. DESCRIPCION DEL PROGRAMA 'PROGMIC'.

El programa inicialmente habilita las interrupciones externas.

Tiene una instrucción de salto en el Vector Reset a la rutina INICIO .

Una instrucción de salto a la rutina INTERR en el Vector de Interrupción Externa.

Define la tabla de código a siete segmentos.

Invoca la rutina LIMPIA.

Alternadamente realiza un llamado a las rutinas DESP y SENSA.

VI.4.2. RUTINA 'LIMPIA'.

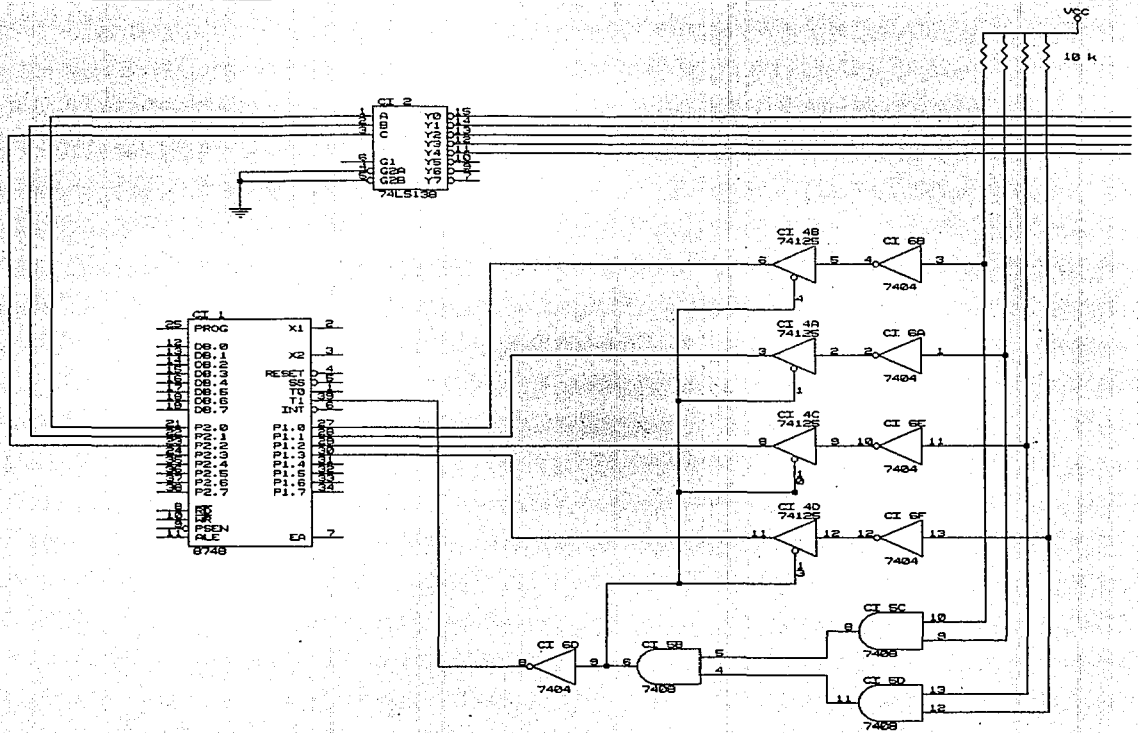
Se encarga de inicializar las localidades 30H, 32H y 34H de la 'RAM' con el valor de 3FH, que corresponde al código "0" para siete segmentos, aquí se almacenarán los correspondientes códigos

a ser desplegados y que fueron suministrados mediante el Teclado, los valores en hexadecimal correspondientes se almacenarán en las localidades 31H, 33H y 35H respectivamente, por lo que LIMPIA les asigna el valor de 0.

VI.4.3. TECLADO

El sistema realizara una transmision de datos uno a uno, es decir, el dato contenido en la localidad de memoria cero de la MCM2716 sera grabado en la correspondiente localidad del MC8748 y asi sucesivamente. La cantidad de localidades de memoria cuya informacion se copiara (la longitud del Programa en Byte's) sera indicada por medio del teclado hexadecimal del tipo matriz.

El circuito electrico para su implementacion es el mostrado en la fig. 32.


Vcc
10 k
CI 2
74LS138
CI 1
8748
PROG
RESET
EA
PSEN
ALE
CI 4B
74126
CI 6B
7404
CI 4A
74126
CI 6A
7404
CI 4C
74126
CI 6C
7404
CI 4D
74126
CI 6F
7404
CI 6E
7404
CI 5B
7408
CI 5C
7408
CI 5D
7408

VI.4.4. RUTINA 'SENSA'.

Genera una cuenta binaria por las tres líneas menos significativas del Puerto 2, que generan un barrido de una señal baja ("0") a las líneas del teclado. por medio del Multiplexor 74138.

Al ser pulsada una tecla se envía una señal alta ("1") al pin T1. Cuando el MC8748 del sistema detecta un "1" en el su pin T1 invoca la rutina DETEC.

VI.4.5. RUTINA 'DETEC'.

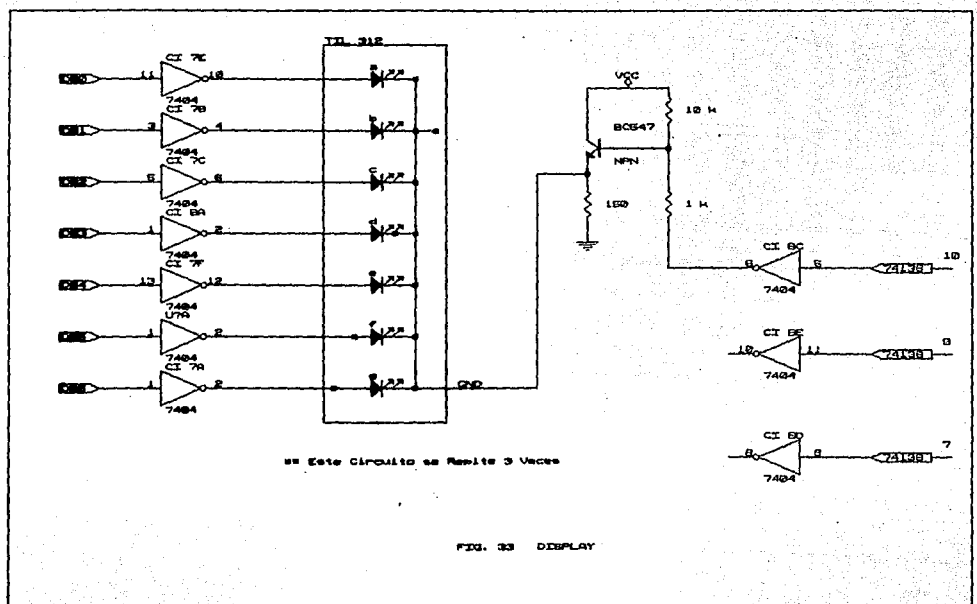
Inmediatamente de ser invocada "lee" el codigo presente en el Puerto 1, y detiene el proceso hasta que la tecla sea soltada.

Con el valor previamente enviado por el Puerto 2 y el "leido" por el Puerto 1 se genera su correspondiente valor hexadecimal y su codigo a siete segmentos, para ser ambos almacenados.

VI.4.6. DISPLAY.

Para poder visualizar la cantidad suministrada, el sistema envia los 3 codigos almacenados con anterioridad, a traves del BUS de Datos a tres Displays que son activados en sincronia con los datos.

El proceso es controlado por la rutina DESP. El diagrama electrico es el que muestra la fig. 33.



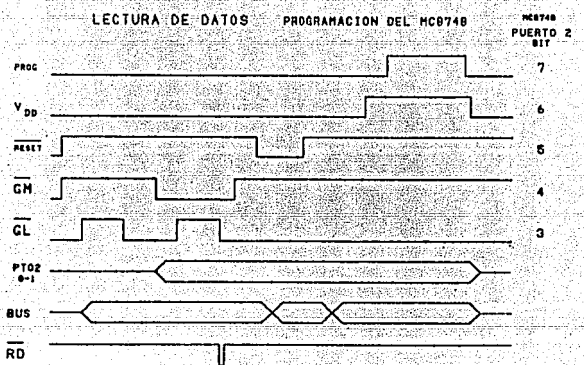
** Este Circuito se Repite 3 Veces

FIG. 33 DISPLAY

VI.4.7. RUTINA 'INTERR'.

Con los valores de las localidades 33H y 35H de la RAM se genera un numero Hexadecimal de dos dígitos, y en combinacion con el de la localidad 31H se tiene una cifra de tres dígitos hexadecimales (la cantidad de Bytes a grabar) con la que se indica el numero de veces que se repite el ciclo Lectura-Graba, cuyo Diagrama de Tiempos se da en la fig. 34.

FIG. 34 DIAGRAMA DE TIEMPOS.



Los diagramas de la etapa de Lectura y Programación del Sistema se muestran en las figuras 35 y 36.

Como se puede observar, el BUS de Datos del MC8748 transmite y recibe las Direcciones y Datos que intervienen en el proceso.

Los listados de las rutinas empleadas por el sistema se dan en el Apéndice I.

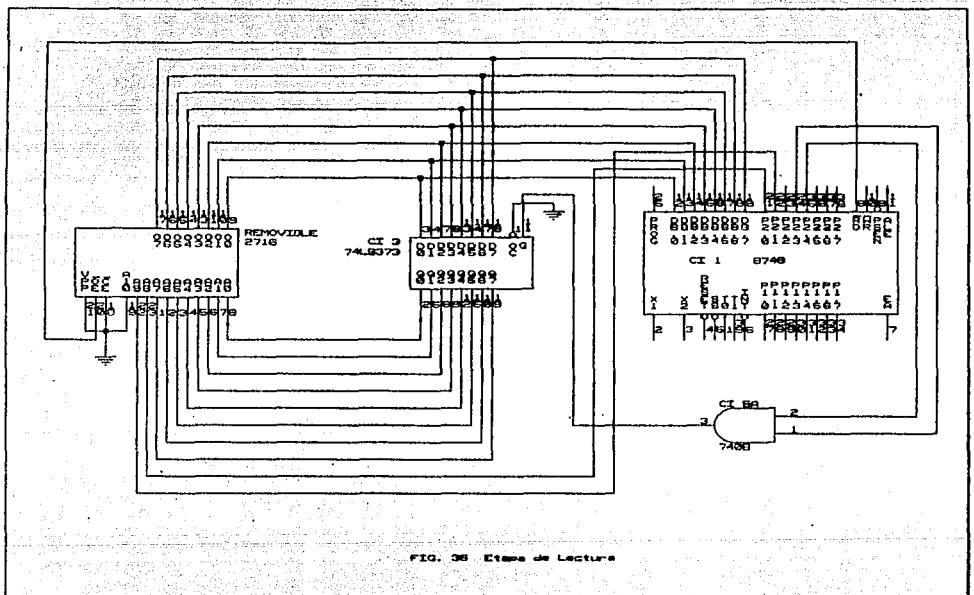


FIG. 36 Etapa de Lectura

FIG. 38 Etapa de Programacion

CONCLUSIONES Y PERSPECTIVAS.

Es de notar que el uso de arquitecturas pre-definidas, cuyo funcionamiento se encuentra determinado por ciertos parámetros, representan un ahorro de tiempo y trabajo para el diseñador, puesto que no existe la necesidad de volver a comenzar un trabajo de diseño desde cero. Teniendo en cuenta que es necesario solamente determinar los parámetros que regirán su operación la labor se reduce considerablemente.

Para poder contar con un módulo de tales características se debe de considerar que su operación será de propósito general, pudiendo considerarlo como una función de usuario hablando en términos de programas escritos para computadora.

Es recomendable, durante un diseño de este tipo, tener en mente la idea 'Trabajar para no trabajar'.

Las arquitectura más conveniente a usar depende de su aplicación, en otras palabras, se tiene que considerar la cantidad de variables que intervienen en el proceso así como los diversos estados del sistema.

B I B L I O G R A F I A .

- FUNDAMENTALS OF DIGITAL SYSTEMS DESIGN

  V. Thomas Rhyne

  PRENTICE HALL E. E. S.

  1973

- FUNDAMENTALS OF LIGIC DESIGN

  Charles N. Roth

  PRENTICE HALL E. E. S.

  1979

- SISTEMAS DIGITALES BASADOS EN MICROPROCESADOR

  James W. Gault - Russell L. Pimmel

  Mc. GRAW HILL

  1985

- AN INTRODUCTION TO COMPUTER LOGIC

  H. Trog Nagle Jr. - B. D. Carroll

  PRENTICE HALL E. E. S.

  1975

- LOGICA DIGITAL Y DISEÑO DE COMPUTADORAS

  Morris Mano

  PRENTICE HALL

  1979

- CIRCUITOS LOGICOS Y SISTEMAS DE MICROCOMPUTADORAS

Claude A. Wiatrowski - Charles H. House

LIMUSA

1987

- 16 BIT MICROPROCESSOR ARCHITECTURE

Terry Dollhoff

PRENTICE HALL COMPANY

1974

- COMPUTER ORGANIZATION HARDWRE SOFTWARE

G. W. Gorsline

PRENTICE HALL

1980

- DIGITAL DESING WHIT STANDARD MSI AND LSI

Thomas R. Blakeslee

J. WILEY - INTERSCIENCE

1979

- MICROCONTROLLER HANDBOOK

INTEL - 1985

- MICROPROCESSOR PROGRAMING & SOFTWARE DEVELOMPENT

F. G. Duncan

PRENTICE HALL INTERNATIONAL

1979

A

P

E

N

D

I

C

E

```
*-------------------------------------------------------------------
*
*  Programa ..... PROGMIC
*
*  Disenado por... Arroyo Henandez Victor Hugo.
*
*  Fecha ......... FEBRERO 1989.
*
*  Descripcion ... Codificado en Lenguaje ensamblador para ser
*                  ejecutado por un MC8748.
*
*                    Lee  codigo  de  una  memoria  MCM2716  y  lo
*                       transmite  a  la  memoria  EPROM  de  un
*                    microcontrolador MC8748.
*
* -------------------------------------------------------------------
                EN      I
                JMP     INICIO
                JMP     INTERR
                NOP
                NOP
                NOP
                NOP
                NOP
INICIO:         MOV     R0,#20H       ;DEFINE CODIGO PARA 7 SEGMENTOS
                MOV     @R0,#3FH
                INC     R0
                MOV     @R0,#06H
                INC     R0
                MOV     @R0,#5BH
                INC     R0
                MOV     @R0,#4FH
                INC     R0
                MOV     @R0,#66H
                INC     R0
                MOV     @R0,#6DH
                INC     R0
                MOV     @R0,#7DH
                INC     R0
                MOV     @R0,#07H
                INC     R0
                MOV     @R0,#7FH
                INC     R0
                MOV     @R0,#67H
                INC     R0
                MOV     @R0,#77H
                INC     R0
                MOV     @R0,#7CH
                INC     R0
                MOV     @R0,#39H
```

```
                INC     R0
                MOV     @R0,#5EH
                INC     R0
                MOV     @R0,#79H
                INC     R0
                MOV     @R0,#71H
                MOV     R2,#3FH
                CALL    LIMPIA
REST:           MOV     R4,#06H         :CONTROLA TECLADO Y DISPLAY
PRIM:           MOV     A,R4
                JZ      REST
                CALL    DESP
                MOV     R1,#2EH
                MOV     A,R4
                ADD     A,R1
                MOV     R1,A
                CALL    SENSA
                JMP     PRIM
```

```
*-----------------------------------------------------------------
*
*  Rutina ........ DESP:
*
*  Diseñado por... Arroyo Henandez Victor Hugo.
*
*  Fecha ......... FEBRERO 1989.
*
*  Descripcion ... Realiza el barrido de la señal a los Dispay's
*                  desplegando los correspondientes codigos de
*                  siete segmentos de los digitos suministrados
*                  por el operador y contenidos en las localidades
*                  30H, 32H y 34H de la memoria RAM del MC8748.
*
*-----------------------------------------------------------------
DESP:           MOV     R6,#30H         :DESPLIEGUE DE DATOS
                MOV     R7,#03H
CICLO:          DEC     R7
                MOV     A,R7
                ADD     A,#05H
                OUTL    P2,A
                MOV     A,R7
                RL      A
                ADD     A,R6
                MOV     R0,A
                MOV     A,@R0
                OUTL    BUS,A
                MOV     A,#0AFH
RETAR:          DEC     A
                JNZ     RETAR
                MOV     A,R7
                JNZ     CICLO
                RET
```

```
*-----------------------------------------------------------------
*
*  Rutina ..... SENSA:
*
*  Diseñado por... Arroyo Henandez Victor Hugo.
*
*  Fecha ..... FEBRERO 1989.
*
*  Descripcion ... Monitorea el teclado hexadecimal  tipo  matriz
*                  e indica cuando una tecla es pulsada.
*
* -----------------------------------------------------------------
SENSA:          MOV     A,#04H          ;DETECTA SI ALGUNA TECLA
CONT:           DEC     A               ;ES ACTIVADA
                MOV     R7,A
                OUTL    P2,A
                JT1     DETEC
                JNZ     CONT
                RET
```

```
*-------------------------------------------------------------------
*
*  Rutina ........ DETEC:
*
*  Diseñado por... Arroyo Hernández Victor Hugo.
*
*  Fecha ......... FEBRERO 1989.
*
*  Descripción ... Se invoca cuando una tecla es sensada.
*                  Decodifica el correspondiente valor de la
*                tecla, determina el codigo a siete segmentos y
*                actualiza las localidades 30H a la 34H de la
*                memoria RAM del MC8748.
*
*-------------------------------------------------------------------
DETEC:          IN      A,P1
                ANL     A,#0FH
RETEN:          JTI     RETEN
                MOV     R5,#00H         ;DETERMINA LA SEGUNDA
CONT2:          RRC     A               ;COORDENADA DE
                INC     R5              ;LA TECLA ACTIVADA
                JNC     CONT2
                DEC     R5
                MOV     A,R7            ;DETERMINA EL CODIGO
                ANL     A,#03H          ;QUE CORRESPONDE
                CLR     C               ;A LA TECLA ACTIVADA
                RLC     A
                RLC     A
                ADD     A,R5
                MOV     R6,A
                ADD     A,#20H
                MOV     R0,A
                MOV     A,@R0
                MOV     @R1,A
                MOV     A,R6
                INC     R1
                MOV     @R1,A
                DEC     R4
                DEC     R4
                RET
```

```
*-------------------------------------------------------------
*
*  Rutina ........ LIMPIA:
*
*  Diseñado por... Arroyo Henandez Victor Hugo.
*
*  Fecha ......... FEBRERO 1989.
*
*  Descripcion ... Inicializa con los valores  correspondientes
*                  las localidades de memoria 30H  a la  34H  de
*                   la  memoria  RAM  del  MC8748.  Que  son  las
*                  localidades reservadas para almacenar los datos
*                  leidos del teclado.
*
* -------------------------------------------------------------
LIMPIA:         MOV     R0,#2FH         ;LIMPIA LAS LOCALIDADES DONDE SE
                MOV     A,#03H          ;ALMACENA EL PRIMER DATO
REP2:           DEC     A
                MOV     R3,A
                INC     R0
                MOV     A,R2
                MOV     @R0,A
                INC     R0
                MOV     @R0,#00H
                MOV     A,R3
                JNZ     REP2
                RET
```

```
*-------------------------------------------------------------------
*
*  Rutina ........ INTERR:
*
*  Diseñado por... Arroyo Henandez Victor Hugo.
*
*  Fecha ......... FEBRERO 1989.
*
*  Descripcion ... Genera los diagramas de tiempos para la
*                  lectura de la memoria MCM2716y la programacion
*                  del MC8748.
*
*-------------------------------------------------------------------
INTERR:     MOV     R1,#31H
            MOV     A,@R1           ;DETERMINA EL NUMERO DE
            MOV     R5,A            ;ITERACIONES CON LOS TRES
            MOV     R1,#33H         ;DIGITOS LEIDOS DEL
            MOV     A,@R1           ;TECLADO
            RL      A
            RL      A
            RL      A
            RL      A
            MOV     R4,A
            MOV     R1,#35H
            MOV     A,@R1
            ORL     A,R4
            MOV     R4,A
            MOV     R6,A
            INC     R5
MONI1:      MOV     A,R6
            MOV     R4,A            ;GENERA EL LAS SEÑALES
            INC     R4              ;NECESARIAS PARA LA
            DEC     R5              ;LECTURA DE LA MEMORIA
MONI:       DEC     R4              ;MCM2716
            MOV     A,#37H
            OUTL    P2,A
            CALL    TIEMP
            MOV     A,R4
            OUTL    BUS,A
            ORL     P2,#3FH
            CALL    TIEMP
            ANL     P2,#37H
            CALL    TIEMP
            MOV     A,R5
            ORL     A,#30H
            OUTL    P2,A
```

```
ANL     P2,#27H         ;GENERA LA SEÑALES PARA
CALL    TIEMP           ;GRABAR LOS DATOS AL
ORL     P2,#28H         ;MC8748
CALL    TIEMP
IN      A,P1
MOV     R2,A
ANL     P2,#27H
CALL    TIEMP
ORL     P2,#30H
CALL    TIEMP
ANL     P2,#17H
CALL    TIEMP
MOV     A,R4
OUTL    BUS,A
CALL    TIEMP
ORL     P2,#30H
CALL    TIEMP
MOV     A,R2
OUTL    BUS,A
CALL    TIEMP
ORL     P2,#70H
CALL    TIEMP
ORL     P2,#0F0H
CALL    TIEMP50
ANL     P2,#77H
ANL     P2,#37H
MOV     A,R4
JNZ     MONI
MOV     A,R5
JNZ     MONI1
RETR
```

```
*-----------------------------------------------------------------
*
*  Rutina ..... TIEMP:  Y   TIEMP50:
*                       ,
*  Diseñado por... Arroyo Henandez Victor Hugo.
*
*  Fecha ..... FEBRERO 1989.
*                ,
*  Descripcion ... TIEMP: Genera los tiempos de espera entre
*                  cambios de estado, y
*                  TIEMP50 Genera un tiempo de espera de 50 mseg.
*
* ----------------------------------------------------------------
TIEMP:          MOV     A,#0FFH
CICLO2:         DEC     A
                JNZ     CICLO2
                RET
TIEMP50:        MOV     R7,#13H
CIEXT           DEC     R7
                MOV     A,#0FFH
LENT2:          DEC     A
                JNZ     LENT2
                MOV     A,R7
                JNZ     CIEXT
                RET
```

MARZO - 1990

ANALISIS DE COSTOS

PROGRAMADOR DE MICROCONTROLADORES MCS748

D E S C R I P C I O N	CANTIDAD DE UNIDADES	COSTO UNITARIO	COSTO TOTAL
C.I. 8748	1	* 50,295	50,295
C.I. 7404	3	1,218	3,654
C.I. 7408	1	1,167	1,167
C.I. 74125	1	1,435	1,435
C.I. 74138	1	1,030	1,030
C.I. 74373	1	1,410	1,410
Regulador de Voltaje 7805	3	2,000	6,000
Display TIL-312	3	3,262	9,786
Transistor BC547	6	650	3,900
Oscilador de Cuarzo 4 MHz.	1	2,800	2,800
Capacitor 1 mic. F	1	770	770
Capacitor 22 pF	2	130	260
Resistencia 150 ohm 1/4 watt	3	75	225
Resistencia 220 ohm 1/4 wat	3	75	225
Resistencia 1 K ohm 1/4 watt	4	75	300
Resistencia 10 K ohm 1/4 watt	6	75	450
Diodo Zener 15v a 1 watt	3	1,000	3,000
Teclado Hexadecimal de Matriz	1	33,000	33,000
	Costo Total		122,707

* El Tipo de Cambio Vigente para el Calculo es 1 Dollar por 2,7500 M.N.

** Los Importes aqui Presentados son una Media de los Precios en el Mercado

*** Aqui es Necesario Adicionar los Costos por Implementacion en Circuito Impreso