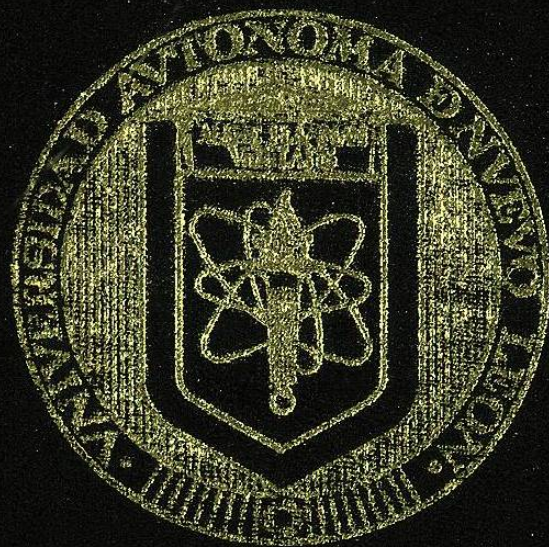


UNIVERSIDAD AUTONOMA DE NUEVO LEON  
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA  
DIVISION DE ESTUDIOS DE POST-GRADO



DISEÑO DE UN CONTROLADOR LOGICO  
PROGRAMABLE PARA PROPOSITOS EDUCATIVOS

# TESIS

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS DE LA  
INGENIERIA ELECTRICA CON ESPECIALIDAD EN  
ELECTRONICA

PRESENTA

JOSE MANUEL CERVANTES VIRAMONTES

MONTERREY, NUEVO LEON

JUNIO DE 1996.

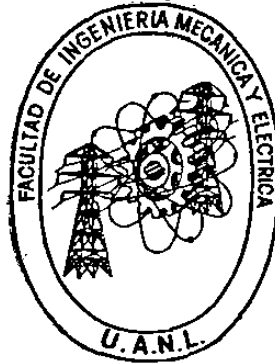


TM  
Z5853  
.M2  
FIME  
1996  
C4



1020115471

**UNIVERSIDAD AUTONOMA DE NUEVO LEON  
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA  
DIVISION DE ESTUDIOS DE POST-GRADO**



**DISEÑO DE UN CONTROLADOR LOGICO PROGRAMABLE  
PARA PROPOSITOS EDUCATIVOS**

**TESIS**

**EN OPCION AL GRADO DE MAESTRO EN CIENCIAS DE LA  
INGENIERIA ELECTRICA CON ESPECIALIDAD EN  
ELECTRONICA**

**PRESENTA**

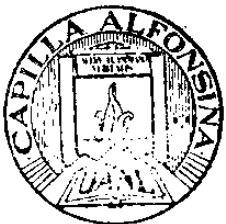
**JOSE MANUEL CERVANTES VIRAMONTES**

**MONTERREY, NUEVO LEON.**

**JUNIO DE 1996.**

0117-19760

TM  
Z5853  
.M2  
FIME  
1996  
C4



FONDO TESIS

UNIVERSIDAD AUTONOMA DE NUEVO LEON  
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA  
SUBDIRECCION DE POSTGRADO

Los miembros del comité de tesis recomendamos que la tesis **“Diseño de un controlador lógico programable para propósitos educativos”** realizada por el **Ing. José Manuel Cervantes Viramontes** sea aceptada para su defensa como opción al grado de **Maestro en Ciencias de la Ingeniería Eléctrica con especialidad en Electrónica**

El Comité de Tesis



Asesor

M.C. Luis M. Camacho Velázquez



Coasesor

M.C. César Elizondo González



Coasesor

M.C. Félix González Estrada



V. B.

M.C. Roberto Villarreal Garza  
Subdirección de Postgrado

San Nicolás de los Garza, N.L. a 14 de junio de 1996

## AGRADECIMIENTOS.-

Deseo hacer patente mi agradecimiento al programa SUPERA de ANUIES por el apoyo y estímulo que me brindaron para la realización del presente trabajo.

Externo un reconocimiento a la empresa MOTOROLA por la capacitación que me impartieron y por la donación de material y equipo, en especial a David Gámez y Chet Freda.

También deseo externar mi gratitud a las siguientes personas e instituciones:  
A la Universidad Autónoma de Zacatecas quién me dio mi formación básica, en especial a mis compañeros de trabajo los profesores de la Facultad de Ingeniería con los cuales he convivido durante veinte años.

A la Universidad Autónoma de Nuevo León; en especial a los profesores de la Maestría en Ciencias de la Ingeniería Eléctrica por haber compartido sus conocimientos conmigo a M.C. Luis Camacho, M.C. Cesar Elizondo y M.C. Ronald López.

A mi esposa e hijos por brindarme su comprensión, por el tiempo que no he podido dedicarles en este último año.

**DISEÑO DE UN CONTROLADOR LOGICO PROGRAMABLE  
PARA PROPOSITOS EDUCATIVOS**

	Pag.
i.- INTRODUCCION.	10
<b>I.-INTRODUCCION A LOS CONTROLADORES LOGICOS PROGRAMABLES.</b>	<b>11</b>
1.1.- Unidad de procesamiento o C.P.U.	11
1.2.- Módulos de entrada.	13
1.3.- Módulos de salida.	15
1.4.- Fuente de alimentación.	17
1.5.- Dispositivo de programación.	17
<b>II.- DESCRIPCION DEL MICROCONTROLADOR.</b>	<b>18</b>
2.1.- Introducción.	18
2.2.- Descripción de las terminales del microcontrolador.	21
2.3.- Registros internos del CPU del microcontrolador.	26
2.4.- Modos de direccionamiento.	29
2.5.- Mapa de memoria.	30
2.6.- Interrupciones y vectores de interrupción.	33
2.7.- Registros especiales del microcontrolador.	33
<b>III.- DESCRIPCION DEL PROYECTO.</b>	<b>36</b>
3.1.- Unidad central de procesamiento.	36
3.2.- Sección de entrada y salida.	49
3.3.- Módulos simuladores.	53
3.4.- Dispositivo de programación.	55
<b>IV.- DIAGRAMA ELECTRONICO.</b>	<b>60</b>
<b>V.- PROGRAMACION.</b>	<b>65</b>
5.1.- Sintaxis del ensamblador.	66
5.2.- Conjunto de instrucciones.	70
5.3.- Ejemplos de aplicación.	77



## LISTA DE FIGURAS

	Pag.
Fig. 1.1.- Diagrama a bloques de un PLC.	12
Fig. 1.2.- Interruptores discretos.	13
Fig. 1.3.- Interfase de entrada CA/CD.	14
Fig. 1.4.- Interfase de salida para una carga de CA.	15
Fig. 1.5.- Interfase de salida para una carga de CD.	16
Fig. 1.6.- Elementos de salida.	16
Fig. 1.7.- Unidades de programación.	17
Fig. 2.1.- Número de parte de los miembros de la familia del 68HC11.	19
Fig. 2.2.- Asignación de terminales de las dos presentaciones del 68HC11A8.	20
Fig. 2.3.- Diagrama a bloques del 68HC11A8.	21
Fig. 2.4.- Registros del CPU del microcontrolador.	26
Fig. 2.5.- Mapa de memoria del 68HC11A8.	31
Fig. 3.1.- Conexiones básicas para el modo de operación expandido.	37
Fig. 3.2.- Relaciones de tiempo del microcontrolador.	38
Fig. 3.3.- Oscilador de cristal.	41
Fig. 3.4.- Circuito de restablecimiento.	42
Fig. 3.5.- Conexiones de la memoria RAM y EEPROM.	44
Fig. 3.6.- Decodificador de direcciones para los puertos de entrada y salida.	46
Fig. 3.7.- Mapa de memoria del sistema.	48
Fig. 3.8.- Módulo de entrada digital.	49
Fig. 3.9.- Módulo de salidas digitales.	50
Fig. 3.10.- Módulo de entradas analógicas.	52
Fig. 3.11.- Módulo simulador de entradas.	53
Fig. 3.12.- Módulo simulador de salidas.	54
Fig. 3.13.- Conexión del circuito MAX232.	55
Fig. 3.14.- Conexiones del MAX232 al DB25.	57
Fig. 3.15.- Interfase de comunicación serie (SCI) del microcontrolador.	57
Fig. 4.1.- Diagrama completo del CPU del PLC.	61
Fig. 4.2.- Entradas digitales del PLC.	62
Fig. 4.3.- Salidas digitales del PLC.	63
Fig. 4.4.- Tarjeta de entradas analógicas del PLC.	64

## LISTA DE TABLAS

	Pag.
Tabla 2.1.- Miembros de la familia del 68HC11.	18
Tabla 2.2.- Resumen de los modos de operación.	23
Tabla 2.3.- Funciones de las señales de los puertos dependiendo del modo de operación.	25
Tabla 2.4.- Direcciones de los vectores de restablecimiento.	27
Tabla 2.5.- Localización de los vectores de interrupción del 68HC11.	32
Tabla 2.6.- Registros especiales y asignación de bits de control.	34
Tabla 3.1.- Asignación de las terminales para el conector DB25 en el estándar RS232.	56

## GLOSARIO

ALU = UNIDAD ARITMETICA Y LOGICA.  
CA = CORRIENTE ALTERNA.  
CCR = REGISTRO DE CODIGO DE CONDICION.  
CD = CORRIENTE DIRECTA.  
COP = VIGILANTE DE COMPUTADORA FUNCIONANDO ADECUADAMENTE.  
CPU = UNIDAD CENTRAL DE PROCESAMIENTO.  
DIP = ENCAPSULADO DE DOBLE LINEA.  
EPROM = MEMORIA PROGRAMABLE Y BORRABLE (CON LUZ ULTRAVIOLETA).  
EEPROM = MEMORIA PROGRAMABLE Y BORRABLE ELECTRICAMENTE.  
FLS = INTERRUPTOR DE FLUJO.  
FS = INTERRUPTOR DE NIVEL.  
I/O = ENTRADA / SALIDA.  
IX = REGISTRO INDICE X.  
IY = REGISTRO INDICE Y.  
Kb = KILOBYTES.  
LS = INTERRUPTOR DE LIMITE.  
PB = BOTON DE EMPUJE.  
PC = CONTADOR DEL PROGRAMA.  
PLC = CONTROLADOR LOGICO PROGRAMABLE.  
PLCC = ENCAPSULADO DE PLASTICO SIN SOLDADURA.  
PS = INTERRUPTOR DE PRESION.  
RAM = MEMORIA DE ACCESO ALEATORIO (MEMORIA VOLATIL).  
ROM = MEMORIA DE SOLO LECTURA (NO VOLATIL).  
RW o R/W = HABILITA LA LECTURA (ALTA) O LA ESCRITURA (BAJA).  
SCI = INTERFASE DE COMUNICACION SERIE ASINCRONA.  
SP = APUNTADOR DE PILA.  
SPI = INTERFASE PERIFERICA SERIE SINCRONA.  
TS = INTERRUPTOR DE TEMPERATURA.  
VCD = VOLTS DE CORRIENTE DIRECTA.  
VCA = VOLTS DE CORRIENTE ALTERNA.

## BIBLIOGRAFIA.-

Motorola Technical Training MC68HC11 Course Notes, Motorola Semiconductor Products Sector, Phoenix Arizona, 1992.

68HC11 Reference Manual, Motorola , 1991.

68HC11 Technical Data, Motorola,1993.

Memory Data, Motorola, 1990.

Fast and LS TTL Data, Motorola, 1993.

PCbug11 User's Manual, Motorola, 1991.

68HC11 Programming Reference Guide, Motorola, 1993.

IASM Integrated Assembler 3.0 User's Manual, P&E Microcomputer Systems, 1992.

Lipovski Jack, Single and Multiple Chip Microcomputer Interfacing, Prentice Hall.

Martin Fred, Introduction to 6811 Programming, Massachusetts Institute of Technology, 1994.

Martin Fred and Oberoi Pankaj, The 6.270 Robot Builder's Guide, Massachusetts Institute of Technology, 1994.

Porras / Montanero, Automatas programables, Mc Graw Hill,1992.

Valvano Jonathan, Microprocessor Applications and Organization Lab Manual, Electrical and Computer Engineering University of Texas at Austin, 1995.

Valvano Jonathan, Microprocessor Applications and Organization Course Notes, University of Texas at Austin, 1992.

## INTRODUCCION:

Los controladores lógicos programables se encuentran actualmente en la mayoría de las industrias, para la automatización de casi todos los procesos.

Normalmente esta automatización se lleva a cabo haciendo un estudio de los PLCs disponibles en el mercado y seleccionando el más adecuado, luego se efectúa la programación de acuerdo con las características del proceso.

El presente trabajo no pretende competir con un PLC comercial, únicamente explica los elementos internos que forman un controlador lógico programable, con la intención de que sean más fácil de entender las características y funcionamiento de un PLC comercial, se trata de un sistema que se va a usar solamente con propósitos educativos.

Se diseñó un sistema que sea capaz de realizar algunas de las funciones más importantes de un controlador lógico programable. El sistema tiene 32 líneas de entrada digital, 32 entradas analógicas y 32 salidas digitales. Cuenta con 8 Kbytes de memoria RAM y 8 Kbytes de memoria EEPROM.

A diferencia de un PLC comercial, el presente sistema debe ser programado en lenguaje ensamblador, de tal manera que quien pretenda programarlo deberá estudiar el conjunto de instrucciones del microcontrolador utilizado, que en este caso es el 68HC11 de Motorola. Mediante el uso del compilador adecuado podría ser posible programarlo mediante lenguaje C. Adicionalmente haciendo algunas modificaciones podría usarse el C interactivo.

El sistema podría ser usado en los laboratorios de las escuelas, para que los estudiantes de electrónica conozcan el interior de un controlador lógico programable. También podría utilizarse para practicar la programación en lenguaje ensamblador.

En el capítulo I se explican en forma general los componentes que forman un controlador lógico programable. En el capítulo II se describen las características del microcontrolador 68HC11. En el capítulo III se explica con todo detalle el diseño de cada uno de los componentes usados en el sistema. El capítulo IV contiene el diagrama electrónico completo. En el último capítulo se explica como se usan las instrucciones del microcontrolador para poder programarlo, y se incluyen algunos ejemplos de aplicación.

# CAPÍTULO 1

## INTRODUCCION A LOS CONTROLADORES LOGICOS PROGRAMABLES

Un controlador lógico programable conocido como PLC es un aparato electrónico construido a partir de un microprocesador o microcontrolador que se utiliza en la automatización de procesos industriales. Cuenta con una memoria capaz de almacenar programas escritos por el usuario y susceptibles de modificación, para controlar una gran cantidad de equipos a través de las unidades de entrada y salida.

Como se muestra en la figura 1.1 los elementos que forman un PLC son:

- a) Unidad de procesamiento o CPU.
- b) Módulos de entrada.
- c) Módulos de salida.
- d) Fuente de alimentación.
- e) Dispositivo de programación.

### a) Unidad de procesamiento o CPU.

La principal función del procesador es la de controlar y dirigir las actividades del sistema completo, las cuales lleva a cabo interpretando y ejecutando un conjunto de programas almacenados en memoria.

El CPU puede tener más de un microprocesador o microcontrolador para ejecutar las tareas o comunicaciones del sistema.

El procesador ejecuta operaciones lógicas o aritméticas sobre una variable de entrada y determina el estado siguiente de la variable de salida. Realiza una supervisión permanente de la operación del sistema. Muestra secuencialmente cada una de las entradas, evalúa el programa y actualiza cada salida, para luego repetir el ciclo.

La memoria generalmente se encuentra contenida en el CPU. Se pueden diferenciar dos tipos de memoria:

-La memoria de ejecución, la cual contiene los programas permanentes, es no volátil por estar implementada con ROM.

-La memoria de aplicación en la cual se almacenan los programas del usuario y los datos que están cambiando, es implementada con memoria volátil RAM.



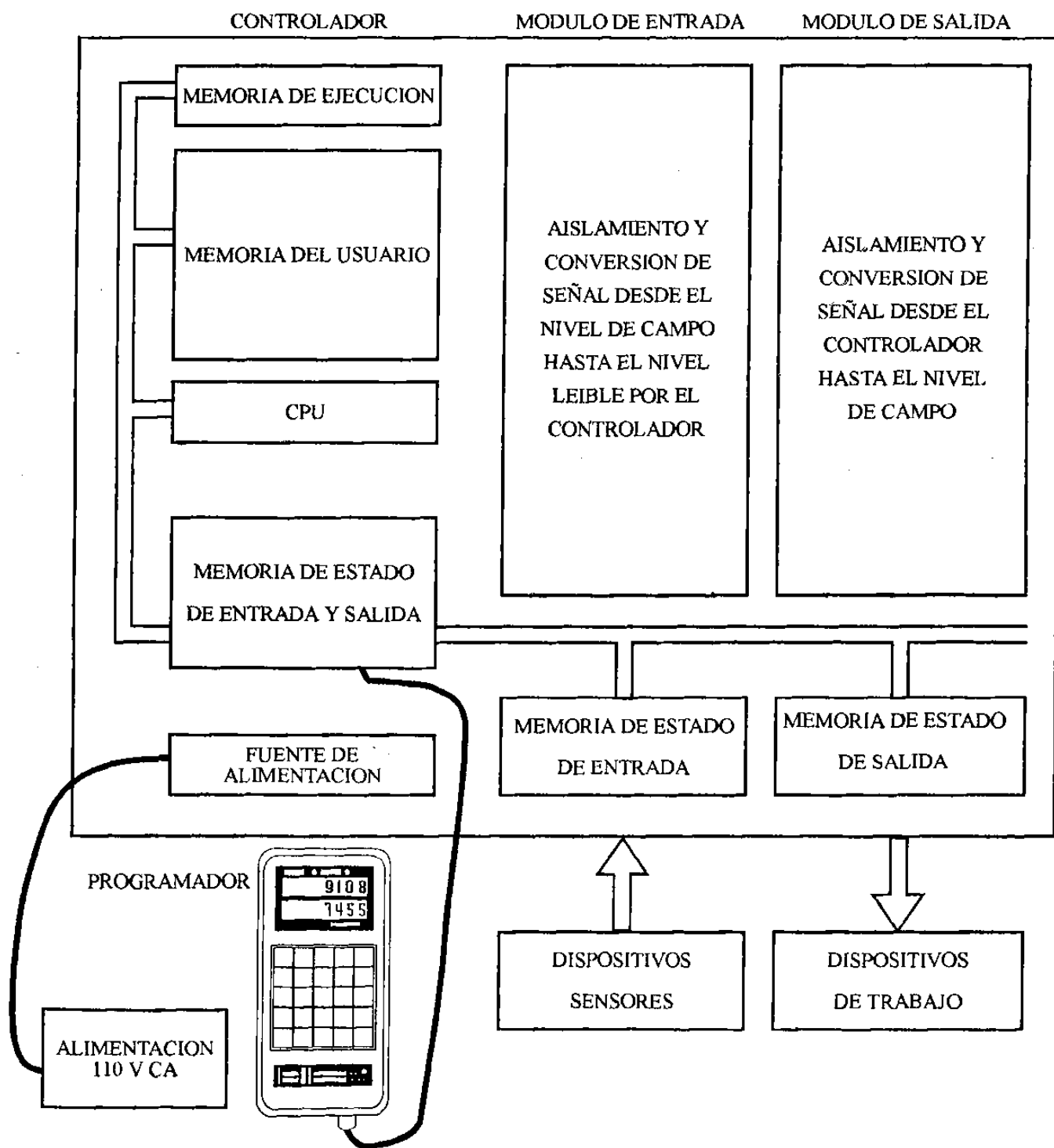


Fig. 1.1 Diagrama a bloques de un PLC.

## b) Módulos de entrada.

Es la interconexión entre el procesador y los dispositivos de entrada. Los módulos de entrada examinan el estado de los interruptores y demás dispositivos de entrada y convierten esa información a una señal que pueda ser interpretada por el procesador. También sirven para proteger al procesador de altos voltajes peligrosos, rebotes de señal, picos de voltaje y ruido eléctrico de las fuentes de alimentación.

La clase más común de interfase de entrada es la digital o tipo discreta la cual recibe únicamente señales abierto/cerrado equivalente a una acción de conmutación.

La figura 1.2 muestra algunos dispositivos de entrada discretos.

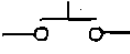
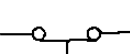
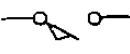
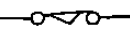
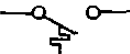
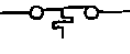

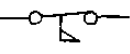
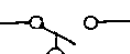
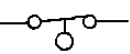
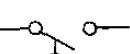
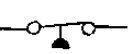
TIPO DE ELEMENTO	CONTACTO NORMALMENTE ABIERTO	CONTACTO NORMALMENTE CERRADO
BOTON DE EMPUJE (PB)		
INTERRUPTOR DE LIMITE (LS)		
INTERRUPTOR DE TEMPERATURA (TS)		
INTERRUPTOR DE FLUJO (FLS)		
INTERRUPTOR DE NIVEL (FS)		
INTERRUPTOR DE PRESION (PS)		

Fig. 1.2 Interruptores discretos .

Los módulos de entrada se deben seleccionar de acuerdo con las características de las señales que se van a monitorear ya que estas pueden ser analógicas o digitales. La figura 1.3 muestra una interfase de entrada CA/CD, estas varían dependiendo del fabricante pero en general tienen el mismo principio de funcionamiento. La sección de potencia convierte el voltaje de entrada de CA a una señal de nivel lógico de CD por medio de un puente rectificador, luego se filtra y se pasa a un circuito de detección de nivel, para finalmente llegar al procesador a través de una etapa de aislamiento eléctrico producido con un optoacoplador.

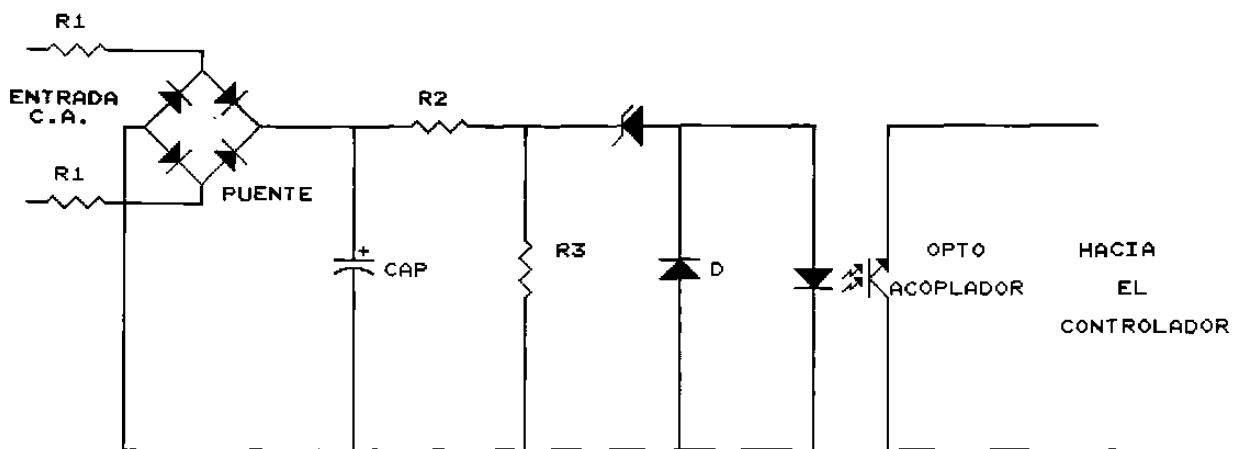


Fig. 1.3 Interfase de entrada CA/CD.

### c) Módulos de salida.

Es la interconexión entre el procesador y los dispositivos de salida. Además de proporcionar una amplificación de voltaje o corriente a la señal, sirven para aislar eléctricamente la etapa de alta potencia de la sección de lógica sensitiva, mediante el uso de optoacopladores o transformadores de pulsos. Los módulos de salida proveen energía a los elementos externos si se usa la configuración de fuente (source), y si se usa la configuración de sumidero (sink), la corriente fluye de la carga hacia el módulo de salida.

La figura 1.4 muestra una interfase de salida para CA, donde se puede notar el aislamiento eléctrico proporcionado por el optoacoplador y la etapa de potencia conseguida por medio de un triac.

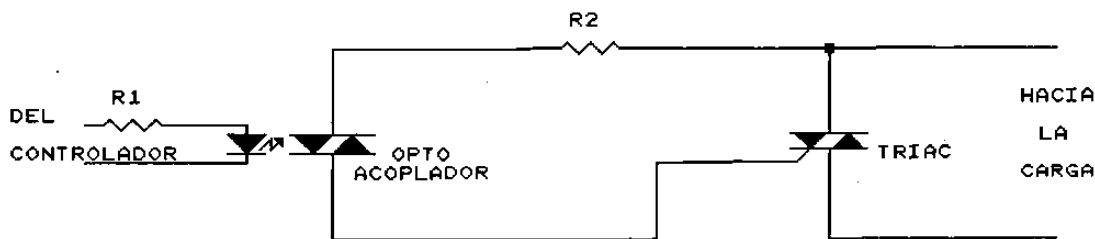


Fig. 1.4 Interfase de salida para una carga de CA.

La figura 1.5 muestra una interfase de salida para CD, puede notarse nuevamente el aislamiento proporcionado por el optoacoplador y la etapa de potencia que usa un transistor de potencia para conmutar la carga.

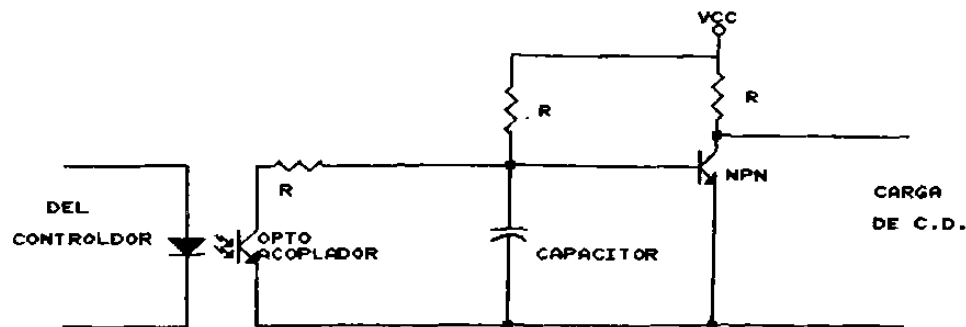


Fig. 1.5 Interfase de salida para una carga de CD

Las interfases de salida de tipo discreto son las más comúnmente usadas en los PLC. Los dispositivos controlados son de naturaleza discreta o digital que pueden tener uno de los dos estados encendido/apagado.

La figura 1.6 muestra algunos elementos de salida.



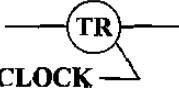
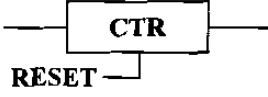

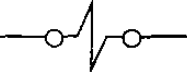


1.- BOBINA	
2.- BOBINA LANCHEADA	
3.- BOBINA DE TIEMPO	
4.- CONTADOR	
5.- MOTOR	
6.- SOLENOIDE	
7.- CALENTADOR	
8.- LAMPARA	

Figura 1.6 Elementos de salida.

#### d) Fuente de alimentación.

La función de la fuente es la de proporcionar voltaje de corriente directa bien regulado y protección a los otros componentes del sistema, normalmente el voltaje de entrada a la fuente es de 120 o 240 V C.A., aunque en algunos casos puede ser 24 V C.D.

En caso de falla de la alimentación el PLC debe tener una batería de respaldo que energice la memoria para evitar la pérdida de datos y programas que estén almacenados en la RAM.

#### e) Unidad de programación.

La unidad de programación es un aparato generalmente externo que se conecta al PLC cuando se desea modificar o transferir programas. El programador usualmente permite la entrada de un programa ya sea en forma de diagrama de escalera o en programación estructurada. Dependiendo del tipo de PLC se podrá usar un programador manual, una terminal o una computadora personal.

Algunas terminales de programación nos permiten además:

- Rastrear las salidas a través de la lógica hasta las entradas que las afectan.
- Comparar los valores contenidos en los registros con valores de contadores y temporizadores externos.
- Localizar posibles fallas analizando el flujo de la señal a través del sistema.

La figura 1.7 muestra las unidades de programación.

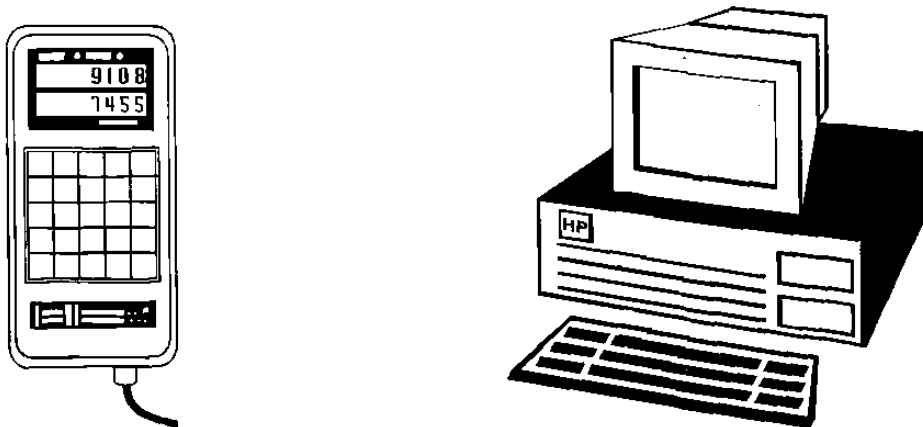


Fig. 1.7.- Unidades de programación.



## CAPÍTULO 2

### DESCRIPCION DEL MICROCONTROLADOR 68HC11

#### 2.1 .-Introducción.

El 68HC11 es un microcontrolador avanzado de 8 bits con capacidades periféricas integradas dentro del mismo circuito.

Dependiendo de la versión del microcontrolador se pueden tener dentro del mismo circuito integrado hasta 24K de memoria ROM o hasta 24K de EPROM, hasta 1K de RAM, y una determinada cantidad de EEPROM. Ver la tabla 2.1

NUMERO DE PARTE	EPROM	ROM	EEPROM	RAM	CONFIG
68HC11A8	---	---	512	256	\$0F
68HC11A1	---	---	512	256	\$0D
68HC11A0	---	---	---	256	\$0C
68HC811A8	---	---	8K+512	256	\$0F
68HC11E9	---	12K	512	512	\$0F
68HC11E1	---	---	512	512	\$0D
68HC11E0	---	---	---	512	\$0C
68HC811E2	---	---	2K	256	\$FF
68HC711E9	12K	---	512	512	\$0F
68HC11D3	---	4K	---	192	N/A
68HC711E9	4K	---	---	192	N/A
68HC11F1	---	---	512	1K	\$FF
68HC11K4	---	24K	640	768	\$FF
68HC711K4	24K	---	640	768	\$FF
68HC11L6	---	16K	512	512	\$0F
68HC711L6	16K	---	512	512	\$0F

Tabla 2.1 Miembros de la familia del 68HC11.

Las principales funciones periféricas están provistas dentro del circuito. Se incluye un convertidor análogo/digital de ocho canales con ocho bits de resolución. Una interfase de comunicaciones serial asíncrona (SCI) y una interfase periférica serial síncrona (SPI). El principal sistema temporizador de carrera libre de 16 bits tiene tres líneas de captura de entrada, cinco líneas de comparación de salida y una función de interrupción de tiempo real. Un subsistema acumulador de pulsos de ocho bits puede contar eventos externos o medir periodos externos de tiempo.

Un circuito de automonitoreo está integrado para proteger contra errores del sistema. Un sistema vigilante de computadora funcionando adecuadamente (COP) protege contra fallas de programación. Un sistema de monitoreo de reloj genera un restablecimiento del sistema en caso de que el reloj se detenga o funcione demasiado lento. Un circuito de detección de código ilegal provee una interrupción no disimulable en caso de que se detecte un código no válido.

Hay dos modos de bajo consumo de potencia disponibles los cuales son controlados por programación, el modo de espera (wait) y el modo de paro (stop).

La figura 2.1 muestra como se compone el número de parte de los miembros de la familia del 68HC11.

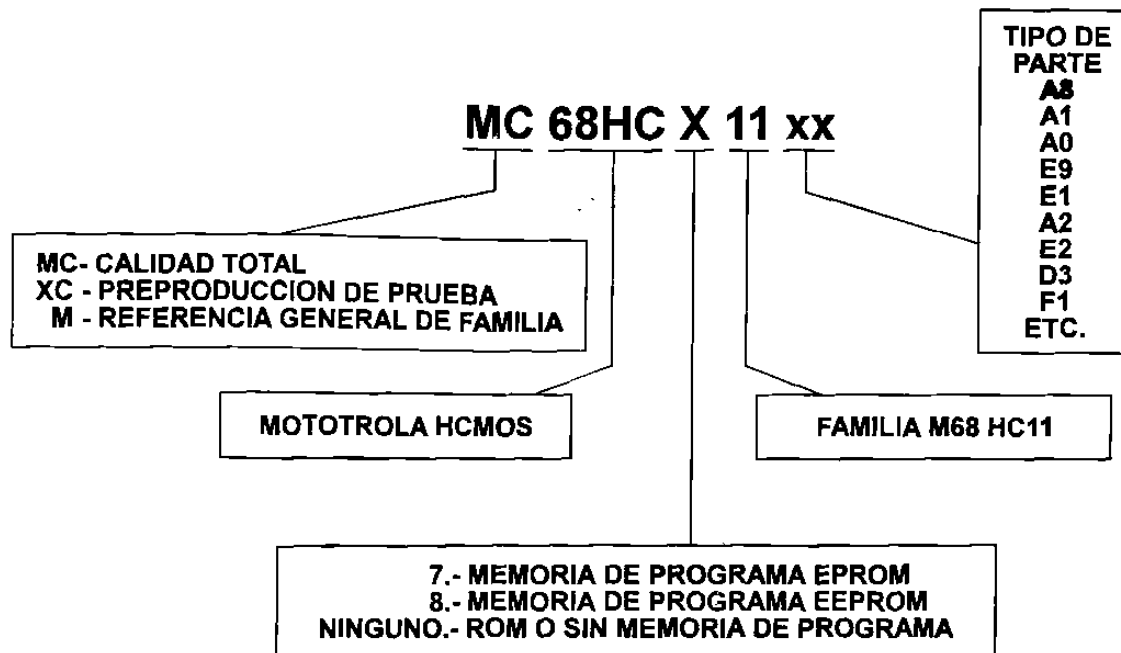


Fig. 2.1 Número de parte de los miembros de la familia del 68HC11.

La figura 2.2 muestra las dos presentaciones del 68HC11A8, la presentación de 52 terminales en encapsulado tipo PLCC y la presentación DIP de 48 terminales, la única diferencia entre ambas es que la versión de 48 terminales solo tiene cuatro líneas de entrada del puerto E y por lo tanto el convertidor A/D es de cuatro canales.

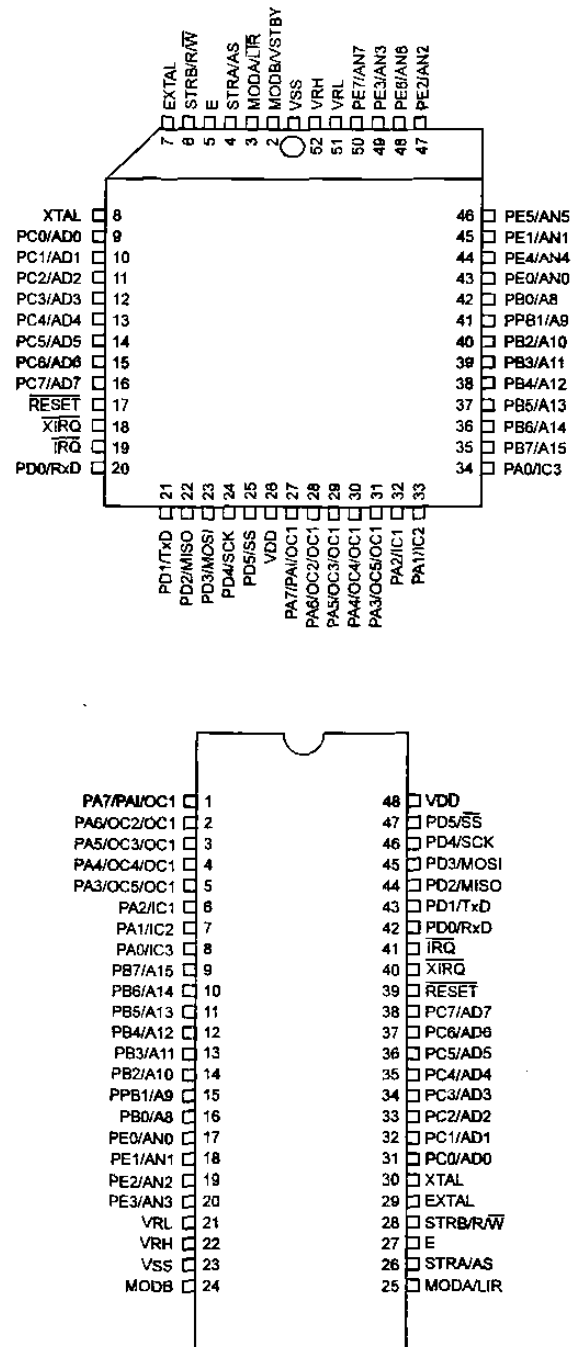


Fig 2.2 Asignación de terminales de las dos presentaciones del 68HC11A8.

## 2.2.-Descripción de las terminales del microcontrolador.

La figura 2.3 muestra el diagrama a bloques del 68HC11A8 en el cual se pueden apreciar todos los recursos que tiene el microcontrolador integrados en el mismo circuito.

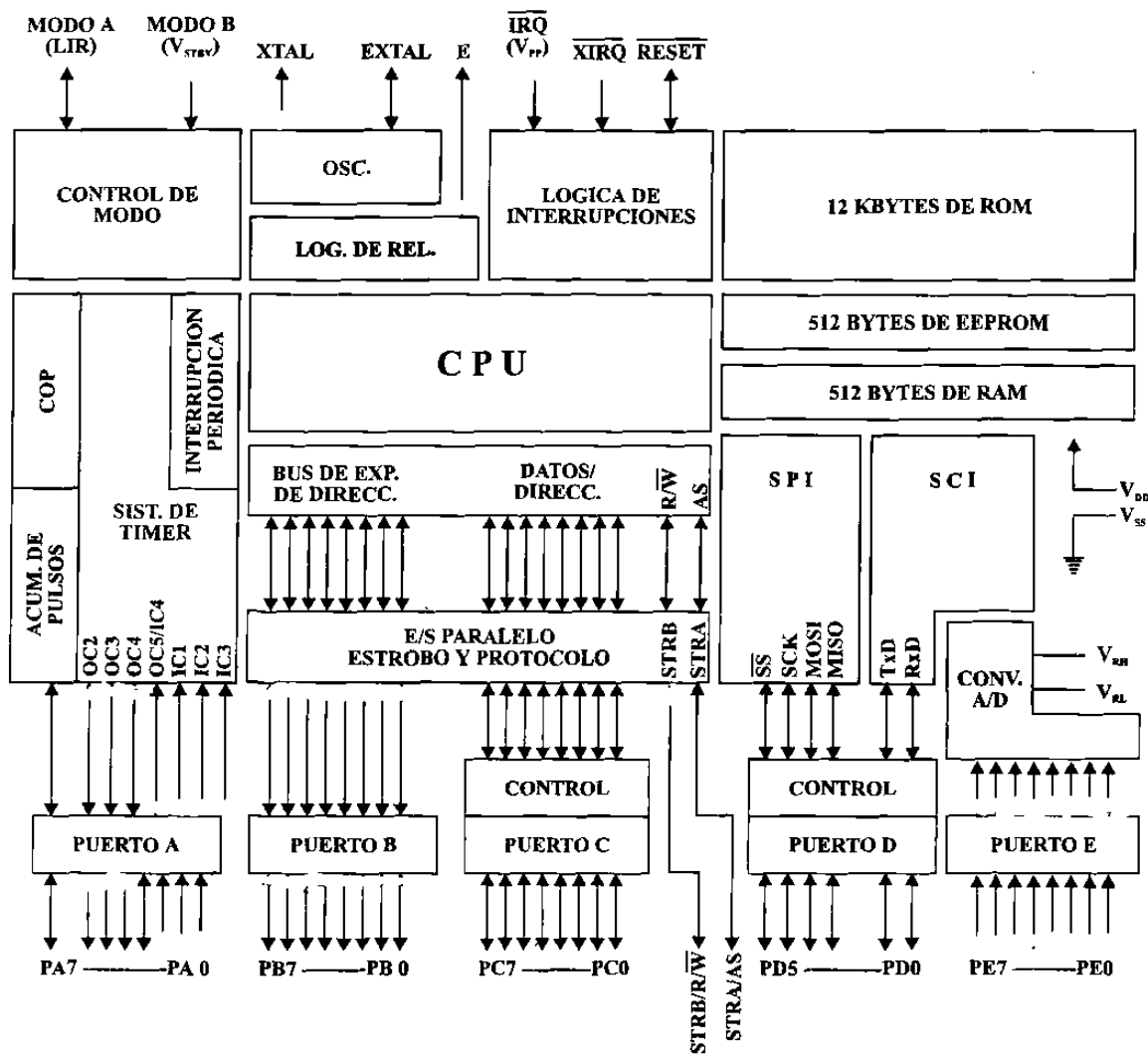


Fig.2.3 Diagrama a bloques del 68HC11A8.

### 2.2.1.-Terminales de alimentación Vdd y Vss.

La energía es suministrada al sistema a través de estas terminales. Vdd se conecta a +5 Vcd y Vss se conecta a tierra, se recomienda colocar un capacitor entre ambas terminales para aterrizar las frecuencias altas que se pudieran presentar.

### 2.2.2.-Terminal de reinicialización (RESET').

Es una señal bidireccional de control, que actúa como entrada para inicializar al MCU a un estado conocido de arranque. También actúa como salida para indicar que ha sido detectada una falla ya sea en el monitoreo del reloj o en el circuito vigilante.

### 2.2.3.-Manejador del cristal y entrada de reloj externo (XTAL y EXTAL).

Estas dos terminales proveen la interfase para un cristal o para un reloj compatible CMOS, para controlar la circuitería del generador de reloj interno. La frecuencia aplicada a estas dos terminales es cuatro veces mayor que la frecuencia deseada del reloj E.

### 2.2.4.-Salida de reloj (E).

Es la conexión con el exterior del reloj E generado internamente. La señal E es usada como una referencia de tiempo. La frecuencia en E es cuatro veces menor que la del cristal conectado en XTAL y EXTAL. Cuando la señal de salida de reloj es baja un proceso interno se está realizando; cuando es alta, se están accediendo los datos.

### 2.2.5.-Petición de interrupciones (IRQ').

La terminal IRQ provee un medio de aplicar peticiones de interrupción asíncronas al MCU.

### 2.2.6.-Interrupción no disimulable XIRQ'/Vpp).

Esta terminal provee un medio de aplicar peticiones de interrupción no disimulables después de la inicialización de restablecimiento. Durante el restablecimiento, el bit X del CCR se pone en alto y cualquier interrupción es disimulada hasta que el programa las habilite.

Vpp es el voltaje de programación nominal de 12 volts necesario para la programación de los circuitos que contengan EPROM interno.

### 2.2.7.-Entradas de modo de operación (MODA /LIR y MODB/Vstby).

Durante el restablecimiento MODA y MODB seleccionan uno de los cuatro modos de operación (ver tabla 2.2).

MODB	MODA	MODO DE OPERACION	RBOOT	SMOD	MDA	IRV
1	0	C.I. UNICO	0	0	0	0
1	1	EXPANDIDO	0	0	1	0
0	0	BOOTSTRAP	1	1	0	1
0	1	DE PRUEBA	0	1	1	1

Tabla 2.2 Resumen de los modos de operación.

Después que el modo de operación ha sido seleccionado, la terminal de registro de carga de instrucción LIR proporciona una salida que indica que la ejecución de una instrucción se ha iniciado. Se requieren varios ciclos de reloj E para ejecutar cada instrucción sin embargo LIR solo permanece baja durante el primer ciclo (ciclo de búsqueda). Esta salida se proporciona para ayuda durante la depuración de los programas.

La terminal Vstby se usa para alimentar a la memoria RAM con un voltaje de respaldo en caso que falle la alimentación principal.

#### 2.2.8.-Voltajes de referencia (Vrl y Vrh).

Estas dos entradas proporcionan los voltajes de referencia para la circuitería del convertidor A/D. Vrl es la referencia baja típicamente 0 volts. Vrh es la referencia alta. Para una adecuada operación del convertidor A/D, Vrh debe ser al menos 3 V mayor que Vrl; y ambos deben estar entre Vss y Vdd.

#### 2.2.9.- STRA/AS.

Esta terminal realiza dos funciones diferentes, dependiendo el modo de operación. En el modo de chip sencillo, STRA realiza un reconocimiento de entrada (handshake). En el modo expandido multicanalizado AS proporciona un reconocimiento de dirección.

#### 2.2.10.-STRB(R/W).

La terminal STRB actua como un reconocimiento de salida en el modo de chip sencillo. En el modo expandido R/W indica la dirección de transferencia del canal de datos externo; un nivel bajo indica escritura y un nivel alto indica que se está realizando una lectura de datos.



### 2.2.11.-Puerto A.

En todos los modos de operación , el puerto A puede ser configurado para tres capturas de entrada y cuatro comparaciones de salida. La otra terminal se puede usar como la cuarta captura de entrada o la quinta comparación de salida. Cualquier línea de este puerto que no sea usada para las funciones descritas anteriormente puede ser usada como entrada o salida de propósito general. La tabla 2.3 muestra las funciones que realizan las terminales de cada uno de los puertos.

### 2.2.12.-Puerto B.

Durante el modo de operación de chip sencillo todas las líneas del puerto B son terminales de salida de propósito general. En el modo de operación expandido todas las líneas actúan como la parte alta del canal de direcciones.

### 2.2.13.-Puerto C.

Durante la operación de chip sencillo las líneas del puerto C actúan como terminales de entrada o salida de propósito general. Las entradas del puerto C pueden adicionalmente ser capturadas en un registro alterno PORTCL aplicando una transición en la entrada STRA.

En el modo expandido las líneas del puerto C actúan como señales multicanalizadas de direcciones/datos. Durante la porción de direcciones del ciclo, los bits (7..0) del canal de direcciones son las salidas (7..0) del puerto C. Durante la porción de datos del ciclo se convierten en las líneas bidireccionales de datos. La dirección de los datos en el puerto C está dada por el nivel de la línea R/W-.

### 2.2.14.-Puerto D.

Todas las seis líneas del puerto D pueden ser usadas como señales de entrada o salida de propósito general. Estas líneas sirven también para los sistemas de comunicación en serie cuando estos están habilitados.

PD0 es la entrada de recepción de datos (RXD) del SCI.

PD1 es la salida de transmisión de datos (TXD) del SCI:

Las otras cuatro terminales se usan para el SPI: PD2 es la entrada MISO. PD3 es la salida MOSI. PD4 es el reloj serieSCK. PD5 es la entrada de selección de esclavo SS.

### 2.2.15.-Puerto E.

El puerto E puede ser usado como un puerto de entrada de propósito general o como las entradas del convertidor A/D cuando este sea habilitado.

BITS DEL PUERTO	SINGLE CHIP Y BOOTSTRAP	EXPANDIDO Y ESPECIAL DE PRUEBA
PA0		PA0/IC3
PA1		PA1/IC2
PA2		PA2/IC1
PA3		PA3/OC5/IC4/OC1
PA4		PA4/OC4/OC1
PA5		PA5/OC3/OC1
PA6		PA6/OC2/OC1
PA7		PA7/PA1/OC1
PB0	PB0	ADDR8
PB1	PB1	ADDR9
PB2	PB2	ADDR10
PB3	PB3	ADDR11
PB4	PB4	ADDR12
PB5	PB5	ADDR13
PB6	PB6	ADDR14
PB7	PB7	ADDR15
PC0	PC0	ADDR0/DATO0
PC1	PC1	ADDR1/DATO1
PC2	PC2	ADDR2/DATO2
PC3	PC3	ADDR3/DATO3
PC4	PC4	ADDR4/DATO4
PC5	PC5	ADDR5/DATO5
PC6	PC6	ADDR6/DATO6
PC7	PC7	ADDR7/DATO7
PD0		PD0/RxD
PD1		PD1/TxD
PD2		PD2/MISO
PD3		PD3/MOSI
PD4		PD4/SCK
PD5		PD5/SS'
-	STRA	AS
-	STRB	R/W'
PE0		PE0/AN0
PE1		PE1/AN1
PE2		PE2/AN2
PE3		PE3/AN3
PE4		PE4/AN4
PE5		PE5/AN5
PE6		PE6/AN6
PE7		PE7/AN7

Tabla 2.3.- Funciones de las señales de los puertos dependiendo del modo de operación.

### 2.3.-Registros internos del CPU del microcontrolador.

El microcontrolador tiene siete registros que no se pueden direccionar como si fueran localidades de memoria por ser parte integral de su CPU, estos registros se muestran en la figura 2.4 y se explicarán en los siguientes párrafos.

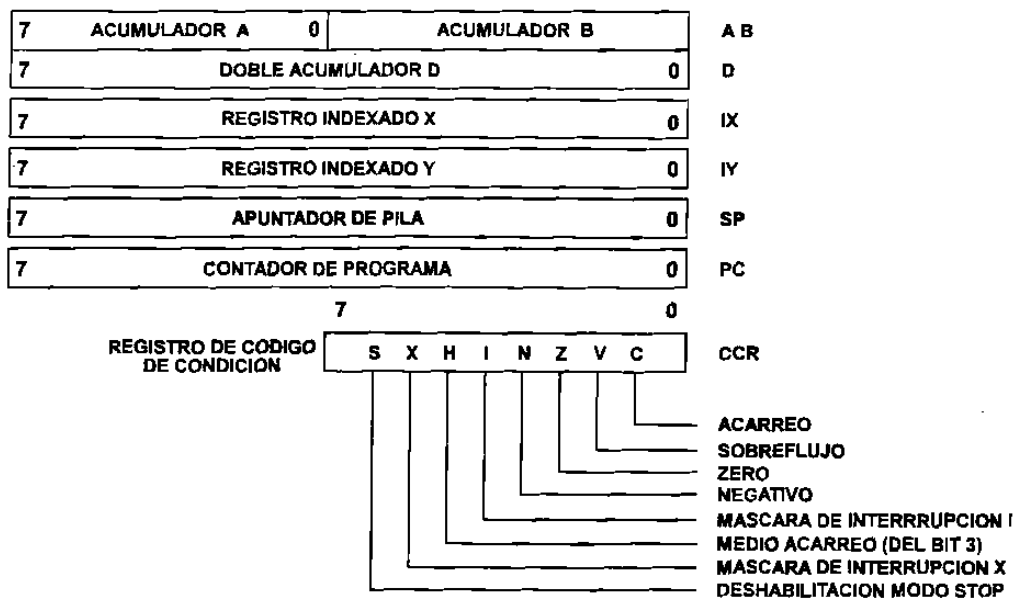


Fig. 2.4 Registros del CPU del microcontrolador.

#### 2.3.1.-Acumuladores A, B y D.

Los acumuladores A y B son registros de ocho bits de propósito general que retienen los operandos y resultados de los cálculos aritméticos o manipulación de datos. Para algunas instrucciones estos dos acumuladores son tratados como un solo registro de 16 bits (doble octeto) llamado acumulador D.

#### 2.3.2.-Registro índice X (IX).

El registro índice X proporciona un valor indexado de 16 bits que puede ser sumado a una compensación de ocho bits dada en una instrucción para crear la dirección efectiva. El registro IX también puede ser usado como un contador o como un registro de almacenamiento temporal.

#### 2.3.3.-Registro índice Y (IY).

El registro IY realiza un procedimiento de modo indexado similar al del registro IX. Sin embargo la mayoría de las instrucciones que usan el registro IY requieren un octeto extra de código de máquina y un ciclo extra de tiempo de ejecución.

#### 2.3.4.-Puntero de pila (SP).

La pila puede ser ubicada en cualquier parte del espacio de direccionamiento y puede ser de cualquier tamaño de acuerdo a la memoria disponible del sistema. Normalmente el SP es inicializado por uno de las primeras instrucciones en el programa de aplicación. La pila es configurada como una estructura de datos que crece de la memoria alta hacia la memoria baja. Cada vez que un nuevo dato sea colocado en la pila, el SP es decrementado. Cada vez que se extraiga un dato de la pila, el SP es incrementado. En todo momento el SP contiene la dirección de la siguiente localidad libre dentro de la pila.

#### 2.3.5.-Contador del programa (PC).

El contador del programa es un registro de 16 bits que contiene la dirección de la siguiente instrucción que será ejecutada. Después del restablecimiento, el contador del programa es inicializado de uno de seis posibles vectores, dependiendo del modo de operación y de la causa del restablecimiento. La tabla 2.4 muestra las seis direcciones de estos vectores.

	RESET o POR	Falla de Reloj	Vigilante COP
Normal	\$FFFE, F	\$FFFC, D	\$FFFA, B
Boot o Prueba	\$BFFE, F	\$BFFC, D	\$BFFA, B

Tabla 2.4 Direcciones de los vectores de restablecimiento

#### 2.3.6.-Registro de código de condición (CCR).

El CCR tiene cinco indicadores de código de condición (C,V,Z,N, y H), dos bits encubridores de interrupción (IRQ y XIRQ), y un bit deshabilitador de paro (S). El CCR es automáticamente actualizado por la mayoría de las instrucciones. Por ejemplo la instrucción "cargar el acumulador A (LDAA)" y la instrucción "almacenar el acumulador A (STAA)" automáticamente borran o ponen en alto las banderas N, Z, y V del CCR.

##### 2.3.6.1.-Acarreo/Préstamo (C).

El bit C se coloca en nivel alto si la unidad aritmética y lógica (ALU) realiza un acarreo o préstamo durante una operación aritmética. El bit C también actúa como una

bandera de error para las operaciones de multiplicación y división. Las instrucciones de corrimiento y rotación operan a través del acarreo para facilitar corrimientos de operandos de varias palabras.

#### 2.3.6.2.-Sobreflujo (V).

El bit de sobreflujo será alto si una operación causa un sobreflujo aritmético. De otra manera el bit será bajo.

#### 2.3.6.3.-Cero (Z).

Si el resultado de una operación aritmética, lógica o manipulación de datos es cero el bit Z es puesto alto. De otra manera será bajo. Las instrucciones de comparación realizan una sustracción interna implícita y el CCR incluyendo Z reflejan el resultado de esa operación.

#### 2.3.6.4.-Negativo (N).

El bit N será alto si el resultado de una operación aritmética, lógica o manipulación de datos es negativo (MSB=1). De otra manera el bit N es cero. Una forma rápida de saber si el contenido de una localidad de memoria tiene el bit mas significativo alto es cargarla en el acumulador y verificar el bit N.

#### 2.3.6.5.- Máscara de interrupción (I).

La máscara I de solicitud de interrupciones IRQ es una máscara global que deshabilita todas las fuentes de interrupción disimulables. Mientras que el bit I permanezca alto, las interrupciones permanecen pendientes, pero la operación del CPU continua sin detenerse hasta que el bit I sea bajo. Después del restablecimiento, el bit I es colocado en nivel alto por omisión y solamente puede ser borrado por una instrucción dentro del programa.

#### 2.3.6.6.-Medio acarreo (H).

EL bit H es puesto en alto cuando ocurre un acarreo entre los bits 3 y 4 de la unidad aritmética y lógica durante una instrucción ADD, ABA o ADC. De otra manera el bit H es bajo. El medio acarreo es usado durante las operaciones con números BCD.

#### 2.3.6.7.-Máscara de interrupción X (X).

La máscara de interrupción X deshabilita las interrupciones desde la terminal XIRQ. Después del restablecimiento el bit X es colocado en alto por omisión y solamente puede ser borrado por una instrucción dentro del programa.

#### 2.3.6.8.-Deshabilitación de paro (S).

Colocando en nivel alto el bit S evita que la instrucción STOP coloque al microcontrolador en el modo de bajo consumo de potencia de paro. Si la instrucción de STOP es encontrada mientras el bit S se encuentra en alto esta es tratada como una no-operación (NOP), y el programa continua con la siguiente instrucción. El bit S es colocado en alto durante el restablecimiento de tal manera que la instrucción STOP está deshabilitada por omisión.

### 2.4.-Modos de direccionamiento.

En el microcontrolador existen seis modos de direccionamiento: inmediato, directo, extendido, indexado, inherente, y relativo.

#### 2.4.1.-Inmediato.

En el modo de direccionamiento inmediato el argumento está contenido en el byte o bytes que sigue inmediatamente después del código de operación. El número de bytes que siguen al código de operación debe corresponder con el tamaño del registro o localidad de memoria con que se va a operar. Por ejemplo:

**LDAA #\$90** Significa que el valor hexadecimal \$90 está siendo cargado al acumulador A. Donde el símbolo # indica que el número que sigue es el que se va a cargar. Y el símbolo \$ quiere decir que el número que viene a continuación es un número hexadecimal.

#### 2.4.2.-Direccionamiento directo.

En el modo de direccionamiento directo, el byte bajo de la dirección donde se encuentra el dato está contenida en el byte que sigue al código de operación, y la parte alta de la dirección se asume que es cero. Este tipo de direccionamiento solo puede acceder a la página cero de la memoria, es decir a las localidades desde \$00 hasta \$FF. En la mayoría de los casos esta area de 256 bytes se reserva para datos que van a ser accedados muy frecuentemente.

#### 2.4.3.-Direccionamiento extendido.



En el modo de direccionamiento extendido, la dirección efectiva del argumento está contenida en los dos bytes que siguen al código de operación. Por ejemplo:

LDAA \$1004 Significa que el dato que está en la dirección \$1004 se va a cargar en el acumulador A.

#### 2.4.4.-Direccionamiento indexado.

En el modo de direccionamiento indexado, un valor de compensación de 8 bits sin signo contenido en la instrucción será sumado al valor contenido en el registro índice (IX o IY). La suma será la dirección efectiva donde se encuentra el dato. Este modo de direccionamiento permite referirnos a cualquier localidad de memoria en el espacio de 64 Kb. Por ejemplo supongamos que el registro X tiene el valor \$1000. Entonces la instrucción:

LDAA 0,X significa cargar el acumulador A con el dato que se encuentra en la localidad \$1000.

Y la instrucción:

LDAA 5,X cargará el acumulador A con el dato contenido en la localidad \$1005.

El valor de compensación es el que aparece antes de la coma y solamente pueden agregarse valores positivos o cero, es decir valores de 0 a 255 decimal.

#### 2.4.5.-Direccionamiento inherente.

En el modo de direccionamiento inherente o implícito toda la información necesaria para ejecutar la instrucción está contenida en el código de operación. Las operaciones que usan solamente el registro índice o los acumuladores, y las instrucciones de control que no tienen argumentos se incluyen en este tipo de direccionamiento. Por ejemplo, TBA transfiere el contenido del registro B al registro A.

#### 2.4.6.-Direccionamiento relativo.

El modo de direccionamiento relativo es usado unicamente en instrucciones de salto. Si la condición de salto es cierta, un valor de compensación de ocho bits con signo incluido en la instrucción se agrega al contenido del contador del programa para formar la dirección efectiva de salto. De otra manera el programa continua con la siguiente instrucción. Los valores de compensación deben estar comprendidos en el intervalo de -128 hasta +127 pudiendo hacer el salto hacia adelante o atrás de la cadena de instrucción.

### 2.5.-Mapa de memoria.

El modo de operación determina el mapa de memoria y si será posible acceder direcciones externas. La figura 2.5 muestra el mapa de memoria del microcontrolador MC68HC11E9. Las localidades de memoria para los recursos internos son las mismas para ambos modos, el de chip sencillo y el expandido. Los bits de control en el registro CONFIG permiten deshabilitar la memoria EPROM o la EEPROM.

La memoria RAM se mapea a la dirección \$0000 después del restablecimiento, pero puede trasladarse a otra dirección que tenga múltiplo de 4kb (\$x000) escribiendo el valor apropiado en el registro INIT. El bloque de 64 registros especiales se coloca después del restablecimiento en la dirección \$1000 pero también en caso necesario es posible trasladarlos a cualquier otra dirección que sea múltiplo de 4kb (\$x000) colocando el valor adecuado en el registro INIT. Si la RAM y los 64 registros se colocan en la misma dirección, los primeros 64 bytes de RAM serán inaccesibles. Por otro lado si la RAM y el ROM están en la misma dirección, la RAM tiene prioridad.

El programa cargador de arranque (bootloader) está contenido en el ROM interno de bootstrap. Este ROM que aparece en el espacio de memoria interno en las localidades \$BF00-\$BFFF, es habilitado solamente si el sistema es restablecido en el modo especial de bootstrap.

Las últimas localidades de memoria desde la dirección \$FFC0 hasta la \$FFFF son reservadas para los vectores de interrupción.

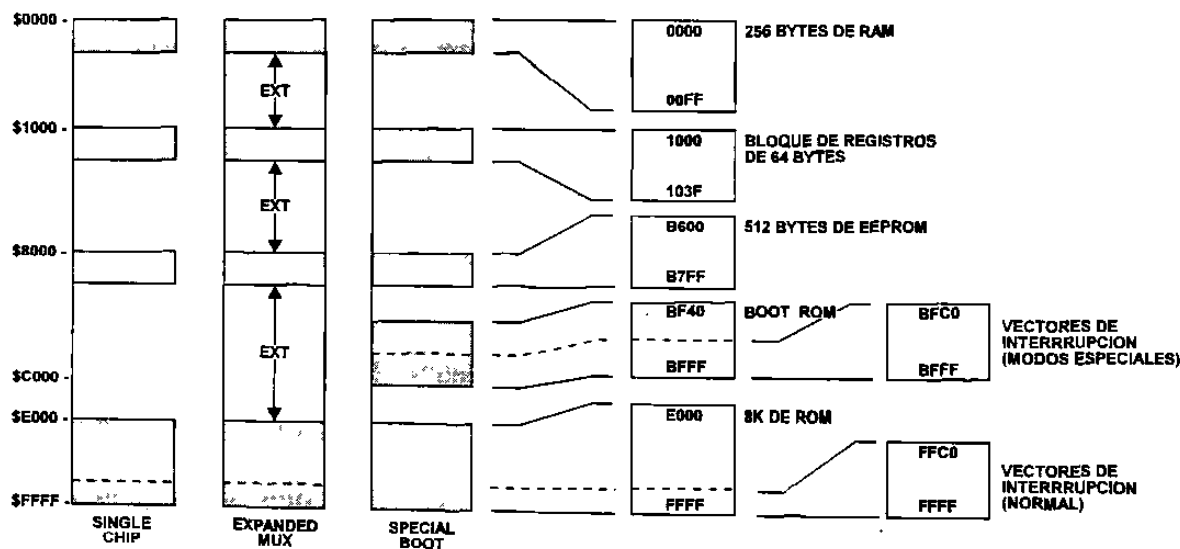


Figura 2.5.-Mapa de memoria del microcontrolador 68HC11E9.

DIRECCION	PROPOSITO
\$FFC0	RESERVADO
\$FFC2	RESERVADO
\$FFC4	RESERVADO
\$FFC6	RESERVADO
\$FFC8	RESERVADO
\$FFCA	RESERVADO
\$FFCC	RESERVADO
\$FFCE	RESERVADO
\$FFD0	RESERVADO
\$FFD2	RESERVADO
\$FFD4	RESERVADO
\$FFD6	SCI DEL SISTEMA SERIE
\$FFD8	SPI TRANSFERENCIA SERIE COMPLETA
\$FFDA	ENTRADA DEL ACUMULADOR DE PULSOS
\$FFDC	SOBREFLUJO DEL ACUMULADOR DE PULSOS
\$FFDE	SOBREFLUJO DEL TEMPORIZADOR
\$FFE0	ENTRADA DE CAPTURA 4/ SALIDA DEL COMPARADOR 5
\$FFE2	SALIDA DEL COMPARADOR 4 (TOC4)
\$FFE4	SALIDA DEL COMPARADOR 3 (TOC3)
\$FFE6	SALIDA DEL COMPARADOR 2 (TOC2)
\$FFE8	SALIDA DEL COMPARADOR 1 (TOC1)
\$FFEA	ENTRADA PARA CAPTURA 3 (TIC3)
\$FFEC	ENTRADA PARA CAPTURA 2 (TIC2)
\$FFEE	ENTRADA PARA CAPTURA 1 (TIC1)
\$FFF0	INTERRUPCCION EN TIEMPO REAL (RTI)
\$FFF2	IRQ' NIVEL BAJO EN PIN EXTERNO (IRQ)
\$FFF4	XIRQ' PSEUDO INTERRUPCION NO DISIMULABLE (XIRQ)
\$FFF6	INTERRUPCION POR SOFTWARE (SWI)
\$FFF8	TRAMPA DEL CODIGO DE OPERACION ILEGAL
\$FFFA	FALLA DEL VIGILANTE (COP)
\$FFFC	COP FALLO DEL MONITOREO DEL RELOJ
\$FFFE	RESET DEL SISTEMA (RESET)

Tabla 2.5 Localización de los vectores de interrupción del 68HC11.

## 2.6.-Interrupciones y vectores de interrupción.

Las rutinas de interrupción son un tipo de programas que se ejecutan cuando un determinado tipo de evento especial ha ocurrido, como por ejemplo: la recepción de datos por la línea serie, el disparo de un temporizador o la existencia de una transición en una línea de captura de entrada.

Cuando una interrupción ocurre, el microcontrolador detiene el proceso que está realizando, almacena el contenido de los registros en la pila y atiende la interrupción; primero localiza el código asociado al vector de interrupción y carga dicho vector en el contador del programa, de esa manera el programa salta a la dirección especificada por el vector. Cada interrupción es asociada con un código que determina la respuesta a la misma.

Si se presiona el botón de reinicialización, el contador del programa se cargará con el dato contenido en las dos últimas localidades de la memoria \$FFFE y \$FFFF ya que el vector de reinicialización está localizado en esas dos localidades. Este vector generalmente se apunta al inicio del programa principal. Por ejemplo si nuestro programa inicia en la dirección \$F800 se deberá grabar ese número en las dos últimas localidades de la memoria.

La tabla 2.5 presenta la ubicación de todos los vectores de interrupción.

## 2.7.- Registros especiales del microcontrolador.

Existen 64 registros especiales los cuales controlan los recursos internos del microcontrolador, estos registros se localizan a partir de la dirección \$1000 después de la reinicialización. La tabla 2.6 muestra los 64 registros y la asignación de los bits de control de cada uno de ellos.

Loc.	BIT 7	BIT 6	BIT 5	BIT4	BIT3	BIT 2	BIT 1	BIT 0	Registro
\$1000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PUERTO A
\$1001									RESERVADO
\$1002	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	PIOC
\$1003	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PUERTO C
\$1004	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PUERTO B
\$1005	PCL7	PCL6	PCL5	PCL4	PCL3	PCL2	PCL1	PCL0	PUERTO CL
\$1006									RESERVADO
\$1007	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
\$1008	0	0	PD5	PD4	PD3	PD2	PD1	PD0	PUERTO D
\$1009	0	0	DDR5	DDR4	DDR3	DDR2	DDR1	DDR0	DDRD
\$100A	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	PUERTO E
\$100B	FOC1	FOC2	FOC3	FOC4	FOC5	0	0	0	CFORC
\$100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3	0	0	0	OCIM
\$100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3	0	0	0	OCID
\$100E	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TCNT(alto)
\$100F	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TCNT(bajo)
\$1010	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TIC1(alto)
\$1011	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TIC1(bajo)
\$1012	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TIC2(alto)
\$1013	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TIC2(bajo)
\$1014	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TIC3(alto)
\$1015	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TIC3(bajo)
\$1016	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TOC1(alto)
\$1017	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TOC1(bajo)
\$1018	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TOC2(alto)
\$1019	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TOC2(bajo)
\$101A	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TOC3(alto)
\$101B	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TOC3(bajo)
\$101C	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TOC4(alto)
\$101D	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TOC4(bajo)
\$101E	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	TI4O5(alto)
\$101F	BIT 7	BIY 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT1	BIT 0	TI4O5(bajo)
\$1020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1
\$1021	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2
\$1022	OC1I	OC2I	OC3I	OC4I	I4O5I	IC1I	IC2I	IC3I	TMASK1

Tabla 2.6 Registros especiales y asignación de bits de control (1 de 2)

Loc.	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Registro
\$1023	OC1F	OC2F	OC3F	OC4F	I4O5F	IC1F	IC2F	IC3F	TFLG1
\$1024	TOI	RTII	PAOVI	PAII	0	0	PR1	PR0	TMSK2
\$1025	TOF	RT1F	PAOVF	PAIF	0	0	0	0	TFLG2
\$1026	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0	PACTL
\$1027	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	PACNT
\$1028	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
\$1029	SPIF	WCOL	0	MODF	0	0	0	0	SPSR
\$102A	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	SPDR
\$102B	TCLR	0	SCPI	SCP0	RCKB	SCR2	CSR1	SCR0	BAUD
\$102C	R8	T8	0	M	WAKE	0	0	0	SCCR1
\$102D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2
\$102E	TDRE	TC	RDRF	IDLE	OR	NF	FE	0	SCSR
\$102F	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0	SCDR
\$1030	CCF	0	SCAN	MULT	CD	CC	CB	CA	ADCTL
\$1031	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	ADR1
\$1032	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	ADR2
\$1033	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	ADR3
\$1034	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	ADR4
\$1035	0	0	0	PTCON	BPRT3	BPRT2	BPRT1	BPRT0	BPROT
\$1036-8									RES.
\$1039	ADPU	CSEL	IRQE	DLY	CME	0	CR1	CR0	OPTION
\$103A	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	COPRST
\$103B	ODD	EVEN	0	BYTE	ROW	ERASE	EELAT	EEPGM	PPROG
\$103C	RBOOT	SMOD	MDA	IRV	PSEL3	PSEL2	PSEL1	PSEL0	HPRIO
\$103D	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0	INIT
\$103E	TILOP	0	OCCR	CBYP	DISR	FCM	FCOP	TCON	TEST1
\$103F	EE3	EE2	EE1	EE0	1	NOCOP	1	EEON	CONFIG

Tabla 2.6 Registros especiales y asignación de bits de control (2 de 2)

## CAPÍTULO 3

### DESCRIPCIÓN DEL PROYECTO.

Se diseñó un sistema que pueda realizar algunas de las funciones más importantes de un controlador lógico programable. El sistema tiene 32 líneas de entrada digital, 32 entradas analógicas y 32 salidas digitales que mediante el uso de tarjetas de potencia pueden controlar cargas de corriente alterna. La diferencia de este sistema con uno comercial, es que debe ser programado en lenguaje ensamblador y por lo tanto quien pretenda programarlo deberá aprender a manejar todas las instrucciones del microcontrolador que se está usando, que en este caso es el 68HC11 de motorola. La intención es que pudiera utilizarse en los laboratorios de las escuelas, para que los estudiantes de electrónica aprendan como puede ser el interior de un controlador lógico programable, y a la vez practiquen a programar en lenguaje ensamblador. Si se usa el compilador adecuado el sistema puede también ser programado en lenguaje C. Adicionalmente con algunas modificaciones podría usarse el C interactivo.

#### 3.1.-Unidad Central de Procesamiento o CPU.

El CPU del controlador lógico programable está formado por el microcontrolador que es el que ejecuta las instrucciones, y la memoria en la cual esas instrucciones y otros datos son almacenados. La figura 3.1 muestra las conexiones básicas para el modo de operación expandido.

Cuando se desea accesar una localidad de memoria (para lectura o escritura), las líneas del puerto C actúan como líneas de direcciones durante la primera mitad del ciclo, transmitiendo los ocho bits de menor peso de la dirección, luego se convierten en líneas de datos durante la segunda mitad del ciclo, ya sea transmitiendo un octeto de datos para el ciclo de escritura o recibiendo un octeto de datos para el ciclo de lectura. El circuito 74HC373 se utiliza para atrapar y almacenar la parte baja del canal de direcciones. La salida AS (address strobe) le indica al registro 373 el momento adecuado de capturar los valores de la dirección.

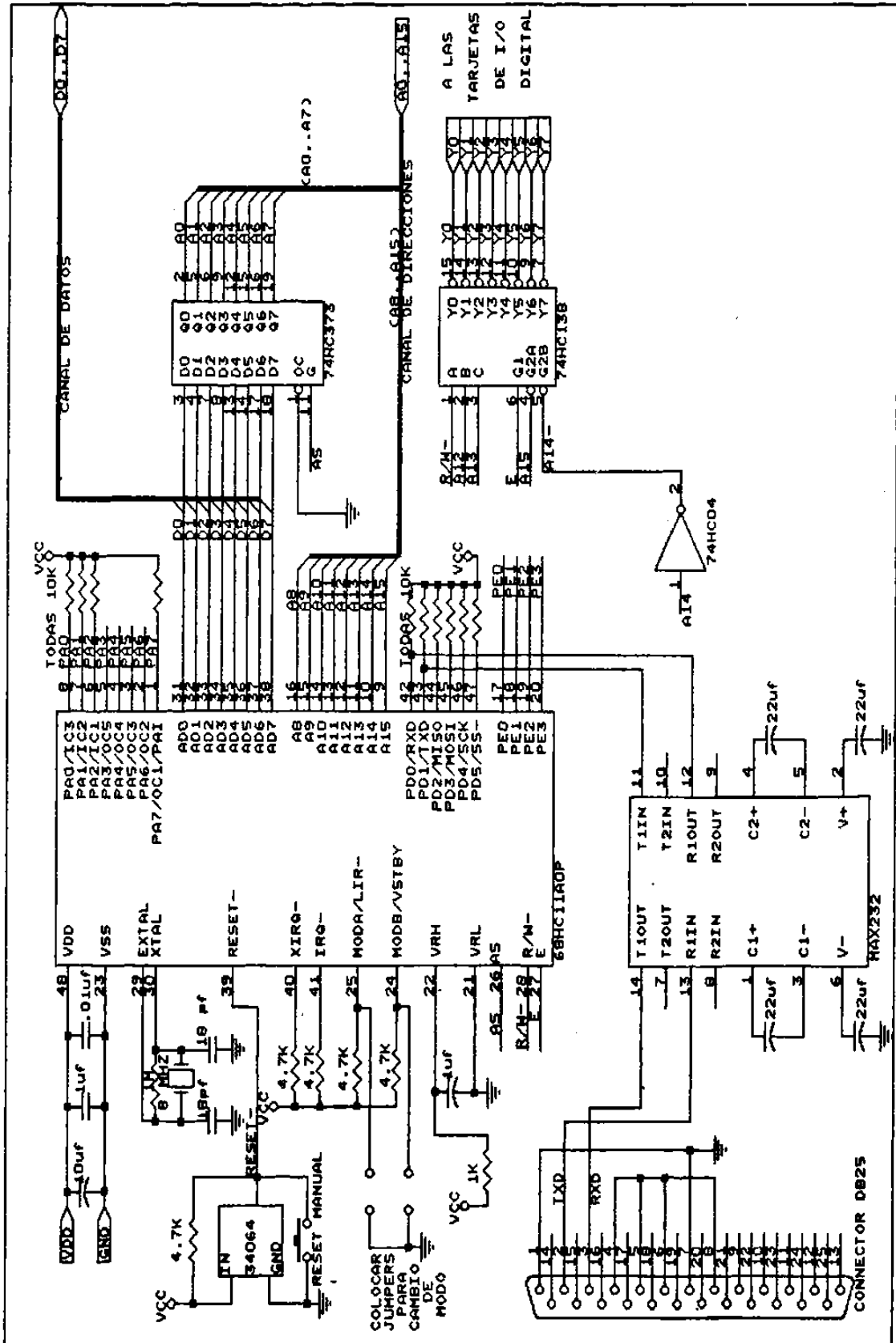
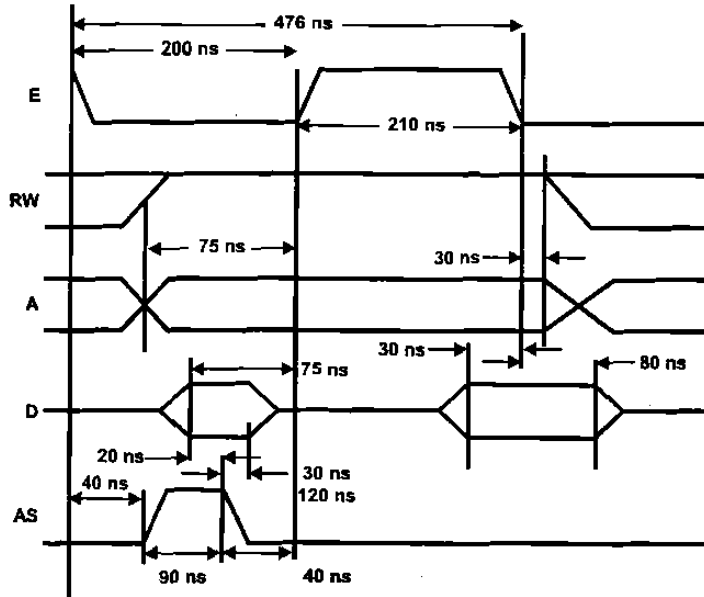


Fig 3.1 Conexiones básicas para el modo de operación expandido.

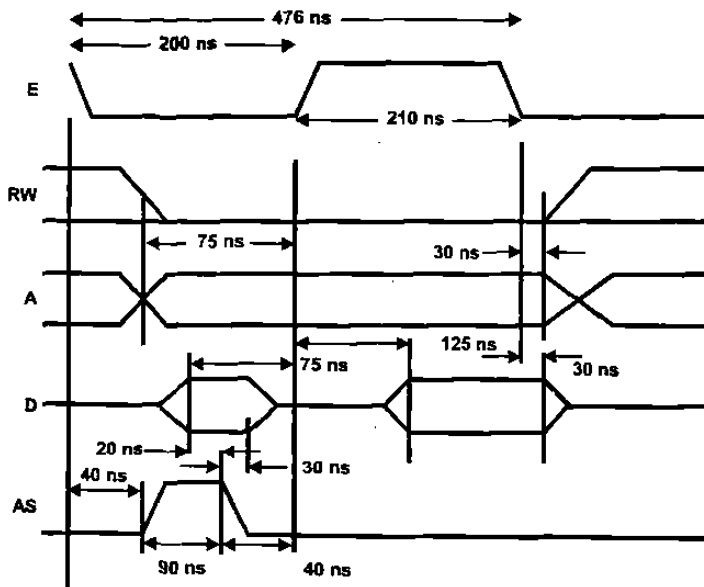


### 3.1.1.-Relaciones de tiempo para los canales del 68HC11.

La figura 3.2 muestra las relaciones de tiempo aproximadas para los canales del microcontrolador operando en el modo expandido.



a) Ciclo de lectura.



b) Ciclo de escritura.

Fig 3.2 Relaciones de tiempo del microcontrolador 68HC11.

En la parte superior de la figura 3.2 se muestra el reloj E también llamado habilitación (Enable), se trata de una señal cuadrada con un periodo de aproximadamente 476 nanosegundos. Un ciclo de memoria comienza y termina cuando el reloj E pasa de alto a bajo. En la parte inferior del diagrama se muestra la señal AS (address strobe) que es la encargada de capturar la parte baja del canal de direcciones de las señales multicanalizadas del puerto C; este pulso de 90 nanosegundos ocurre 40 nanosegundos después de que E baje. Se muestran también las líneas del canal de datos D, del canal de direcciones A, y la señal de lectura/ escritura RW.

El canal de direcciones es indeterminado hasta 75 nanosegundos después de que E suba, los 8 bits de menor peso de la dirección deben permanecer estables 20 nanosegundos antes y 30 nanosegundos después de que AS baje.

En un ciclo de lectura, la señal RW permanece alta durante el ciclo, y el microcontrolador espera datos válidos en el canal de datos por 30 nanosegundos antes y 80 nanosegundos después del flanco de bajada de E. Si alguna línea cambia en este intervalo, el microcontrolador podría interpretar esa línea como alta o baja indistintamente. La memoria es la responsable de proporcionar datos válidos y estables durante ese periodo.

En un ciclo de escritura, la señal RW permanece baja al tiempo que la dirección se encuentra estable y puede subir 30 nanosegundos después de iniciado el siguiente ciclo. Los datos que serán escritos, deberán ser colocados en el canal de datos y se garantizan definidos 125 nanosegundos después del flanco de subida de E, y permanecerán estables 30 nanosegundos después de que E baje. Estas señales están disponibles para usarse para controlar la memoria.

Debe apreciarse sin embargo que la señal RW no puede ser considerada como una señal de regulación de tiempo, no se puede depender de ella para satisfacer tiempos de retención y establecimiento. Similarmente las señales de direcciones no están precisamente alineadas con el ciclo de memoria. Las señales de control se pueden obtener mediante el OR de la señal RW con el reloj E invertido, o mediante el AND de las líneas de direcciones con el reloj E.

### 3.1.2.-Desvío de señales de alta frecuencia a tierra en las terminales de alimentación.

Aunque el 68HC11 es un microcontrolador de tecnología CMOS, existen transiciones muy rápidas en muchas de sus terminales. Dependiendo de la carga de esas señales rápidas, habrá significativas demandas de corriente a la fuente de alimentación de corta duración. Se deberá tener un cuidado especial de proporcionar un desvío adecuado a la fuente de alimentación (power supply bypassing).

Un sistema expandido típico deberá incluir un capacitor de 1 microfaradio y uno de 0.01 microfaradios conectados entre las terminales de alimentación. Ambos capacitores deberán estar lo mas cercano posible físicamente al microcontrolador y deberán tener muy buenas características de alta frecuencia, evitándose los de disco de cerámica. El capacitor de 1 microfaradio proporciona carga para la conmutación del canal a través de un camino de muy baja impedancia. Sin este capacitor de desvío, habría caídas muy altas de voltaje debido a las demandas muy altas aunque de muy corta duración de corriente, originadas por la conmutación de un nivel a otro de varias terminales del microcontrolador simultáneamente. El capacitor separado de 0.01 microfaradios se requiere porque el otro capacitor más grande de 1 microfaradio no es tan eficiente para eliminar el ruido de baja energía y alta frecuencia.

Estas son solo recomendaciones generales. Algunos sistemas que trabajen con baja carga pueden operar adecuadamente con un solo capacitor de 0.1 microfaradios conectado lo más cerca posible de las terminales de alimentación; mientras que otros sistemas más grandes trabajando en el modo expandido pueden requerir medidas más elaboradas de desvío de señales a tierra.

Es mucho más sencillo y menos costoso colocar un sistema adecuado de desvío como una medida preventiva que localizar y corregir un problema de ruido en un sistema marginal.

### 3.1.3.-Conexión de las terminales de referencia del convertidor A/D.

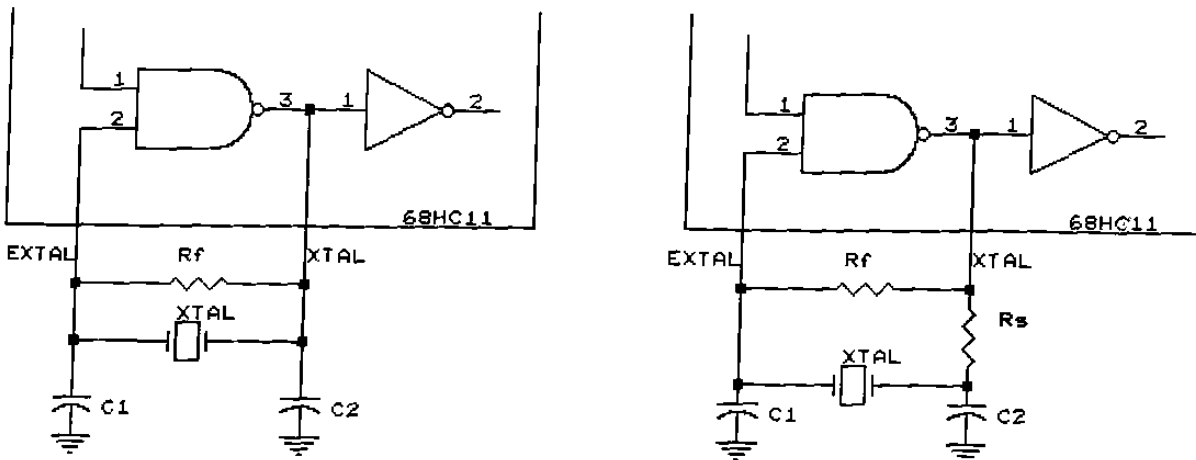
Las terminales  $V_{rh}$  y  $V_{rl}$  proporcionan los voltajes de referencia para la circuitería del convertidor A/D. Se recomienda conectar estas terminales a  $V_{dd}$  y  $V_{ss}$  a través de un circuito de filtro pasa bajos para aislar el ruido en la fuente de alimentación lógica, de las mediciones analógicas relativamente sensibles. Una referencia de voltaje precisa de bajo ruido podría usarse alternativamente. Debe haber una diferencia de potencial de al menos 2.5 volts entre las terminales  $V_{rh}$  y  $V_{rl}$  para una adecuada precisión.

### 3.1.4.- Circuito oscilador de cristal resonante en paralelo.

La figura 3.3 muestra los componentes internos y externos para el oscilador de cristal resonante en paralelo conocido como oscilador Pierce. La figura a) muestra las conexiones para alta frecuencia que se usan para cristales de más de 1 Mhz, y la figura b) se usa para frecuencias bajas o menores a 1 Mhz. La resistencia  $R_f$ , proporciona una corriente directa de polarización a la entrada para que la compuerta NAND opere en su región lineal. En diseños de baja frecuencia,  $R_s$  y  $C_2$  proporcionan un corrimiento de fase;  $R_s$  también limita

la potencia dentro del cristal, lo cual es importante para muchos pequeños cristales diseñados para muy bajos niveles (típicamente de 1 microwatt máximo).

En aplicaciones de alta frecuencia, la impedancia de salida de la compuerta NAND, combinada con la menor impedancia de C1 y C2, produce el mismo efecto que  $R_s$  en los diseños de baja frecuencia.



a) Conexiones para alta frecuencia.

b) Conexiones para baja frecuencia.

Fig. 3.3 Oscilador de cristal.

Los valores exactos para los componentes externos son función de los parámetros de procesamiento de la oblea, capacitancia del encapsulado, capacitancia e impedancia del circuito impreso, capacitancia del receptáculo del circuito integrado, voltaje de operación, tecnología del cristal, y frecuencia. Los valores típicos son:

$R_f = 1 \text{ Mohm} - 20 \text{ Mohms}$  Los valores mayores son sensibles a la humedad; los valores más pequeños reducen la ganancia.

$C_1 = 5 \text{ pf} - 25 \text{ pf}$  El valor generalmente es fijo.

$C_2 = 5 \text{ pf} - 25 \text{ pf}$  El valor puede ser variado para ajustar la frecuencia.

En la mayoría de las aplicaciones de alta frecuencia, los valores de C1 y C2 son iguales. En los diseños de baja frecuencia, es deseable que C1 sea menor que C2, lo cual proporciona un mayor voltaje a la entrada EXTAL debido a una transformación de impedancia.

En el diseño del PLC se ha usado un cristal de 8 Mhz por lo tanto la conexión que se utiliza es la de alta frecuencia. Los valores de los capacitores C1 y C2 son de 18 pf y el valor de la resistencia Rf es de 10 Mohms.

### 3.1.5.- Circuito de restablecimiento (RESET).

Los sistemas basados en el microcontrolador 68HC11 deben incluir un control automático del restablecimiento que lleve la terminal RESET al nivel bajo cuando la alimentación cae más allá de los límites permitidos, para evitar que el contenido del EEPROM interno se contamine; aun en sistemas que no usen el EEPROM interno, debe recordarse que el registro CONFIG contiene celdas de memoria EEPROM y debe ser protegido.

Un circuito sencillo y económico de inhibición de bajo voltaje (LVI) se construye a partir del MC34064, el dispositivo se conecta a Vdd, Vss y al botón de restablecimiento manual, una resistencia de 4.7 Kohms conectada entre Vdd y la terminal RESET es el otro componente requerido. La figura 3.4 muestra el circuito de restablecimiento utilizado.

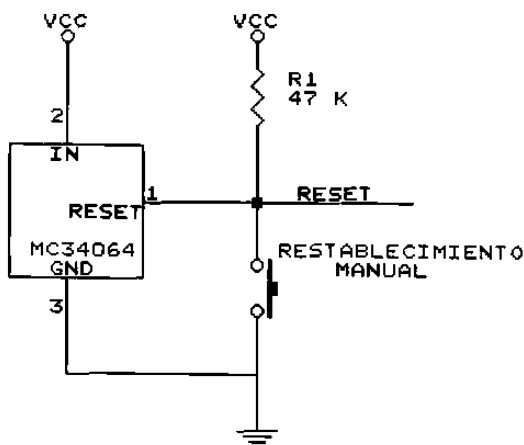


Fig. 3.4 Circuito de restablecimiento.

### 3.1.5.- Conexión de las terminales de interrupción IRQ y XIRQ.

Como estas terminales son activas en nivel bajo y en este caso no se van a utilizar, se desactivan conectándolas a Vdd a través de una resistencia de 4.7 Kohms cada una de ellas.

### 3.1.6.-Terminales de selección de modo MODA y MODB.

El modo de operación expandido se logra conectando estas dos terminales a nivel alto, lo cual se realiza conectando cada una de ellas a Vdd a través de resistencias de 4.7 Kohms. De esta manera será muy sencillo cambiar de modo de operación, simplemente se conecta a tierra el extremo de la resistencia que va unido al microcontrolador por medio de un jumper.

### 3.1.7.-Terminación de entradas no usadas.

Debido a que el microcontrolador es un dispositivo CMOS, las entradas no utilizadas deben ser conectadas a nivel alto a través de resistencias de 10 kohms para asegurar una operación adecuada y confiable. Si alguna entrada no es terminada, esta puede oscilar o flotar a un nivel medio de alimentación, lo cual puede causar una demanda extra de potencia. La oscilación puede resultar en acoplamiento de ruido de la fuente. En antiguos circuitos CMOS las enormes corrientes causadas por una entrada flotante podrían causar la destrucción del circuito integrado. En el microcontrolador es poco probable la destrucción del circuito, pero sigue siendo importante terminar las entradas no usadas para evitar oscilación, ruido, y demanda de corriente adicional.

Las terminales que no se usan y que son solamente entradas podrían conectarse entre si y luego a un punto de terminación común; sin embargo este método aunque es el más económico y toma menos espacio, no se recomienda, porque resulta muy difícil separar y usar una de esas entradas si se requiere posteriormente.

Las entradas del puerto E tienen una configuración diferente, y no es tan importante terminar las entradas no usadas de este puerto.

### 3.1.8.-Memoria del sistema.

El sistema cuenta con 8 kbytes de memoria RAM mediante el uso de un circuito 6264, los cuales se mapean a partir de la dirección C000 a la DFFF. Tiene también 8 kbytes de memoria EEPROM usando un circuito 2864, los cuales se mapean a partir de la dirección \$C000 a la \$FFFF.

#### 3.1.8.1.-Memoria RAM

La figura 3.5 muestra las conexiones de la memoria RAM, nótese que la memoria tiene 13 líneas de direcciones, las 8 direcciones de menor peso están conectadas a las salidas del 74HC373 y las 5 de mayor peso están conectadas directamente a las líneas de direcciones correspondientes del microcontrolador. Las 8 líneas de datos se conectan directamente al canal de datos del microcontrolador.

La línea WE' de la memoria RAM cuando está en nivel bajo habilita la escritura, y en nivel alto habilita la lectura, lo cual coincide con la operación de la línea R/W- del microcontrolador, por lo tanto esas dos líneas se conectan entre si directamente.

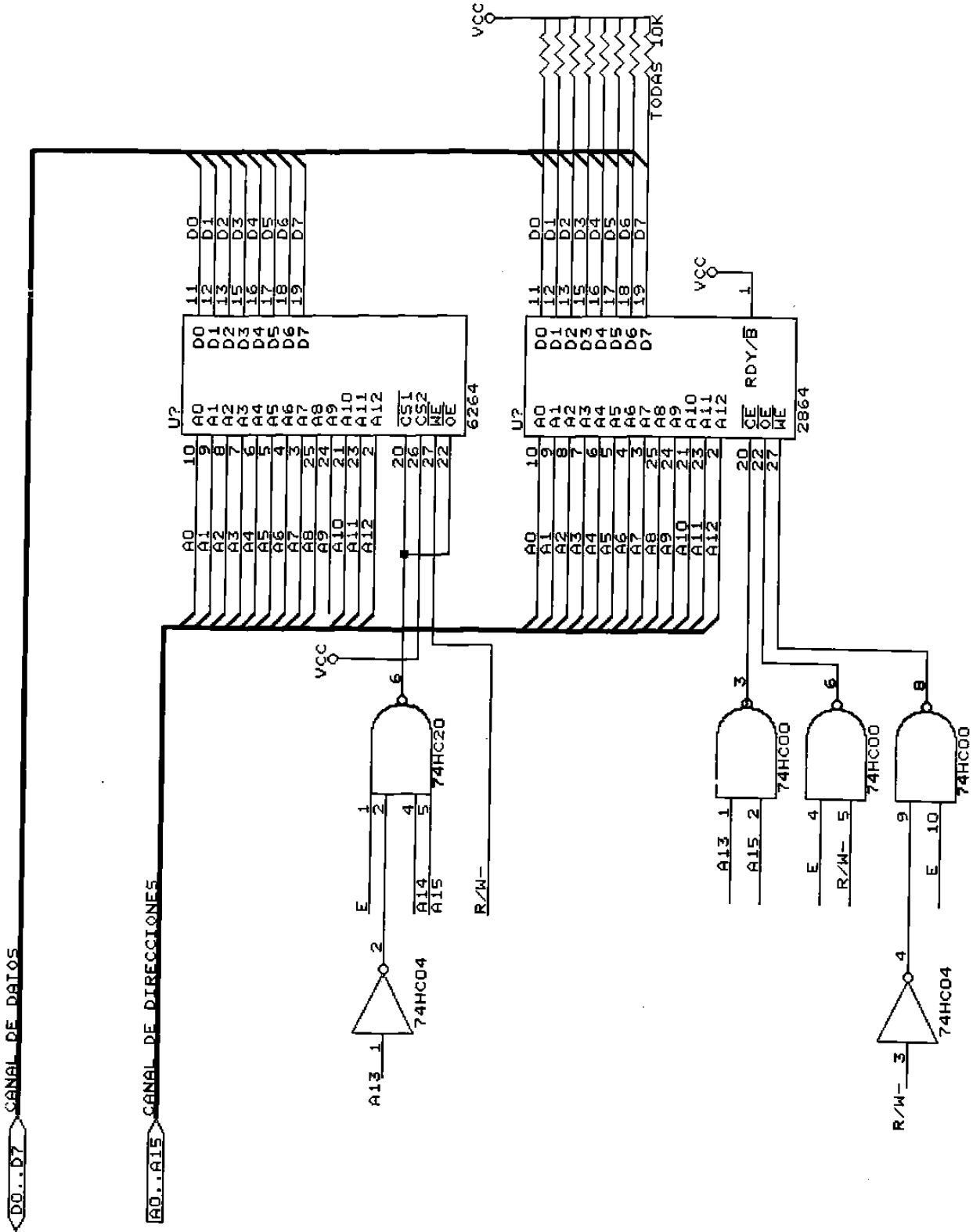


Fig 3.5 Conexión de la memoria RAM y EEPROM.

La línea OE' de la memoria se activa en nivel bajo, y sirve para habilitar la salida de la memoria por lo tanto se conecta junto con la línea CS1' que también es bajo activa a la salida de una compuerta NAND. Las entradas a esta compuerta son: Las líneas de direcciones A15 y A14 que cuando sean altas activarán la memoria, la línea A13 que se pasa por un inversor para que active la memoria cuando sea baja, y la señal de reloj E que es alta únicamente en la segunda mitad del ciclo de reloj. De esa manera la RAM queda ubicada entre las direcciones C000 y DFFF del mapa de memoria.

La línea CS2 es alto activa y se encuentra conectada permanentemente a Vcc para que esté siempre habilitada.

### 3.1.8.2.-Memoria EEPROM.

La parte inferior de la figura 3.5 muestra las conexiones de la memoria EEPROM 2864, las líneas de datos y direcciones se encuentran conectadas exactamente a los mismos puntos que la RAM.

La terminal OE' que es bajo activa, sirve para habilitar la salida de la memoria, se encuentra conectada a la salida de una compuerta NAND, la cual se va a activar cuando la señal de reloj sea alta y la línea R/W' sea alta, es decir solamente cuando se realice una lectura de la memoria.

La terminal WE' normalmente estará en nivel alto cuando se estén leyendo datos de la memoria, solamente pasará a nivel bajo cuando se vaya a grabar información, es decir cuando la línea R/W- sea baja y el reloj E sea alto.

La entrada CE' es también bajo activa, se encuentra conectada a la salida de una compuerta NAND cuyas entradas son las líneas de direcciones A13 y A15 que cuando sean ambas altas habilitarán la memoria. De esta manera la memoria EEPROM quedará ubicada entre las direcciones E000 y la FFFF. Los vectores de interrupción se localizan en el modo expandido entre las direcciones FFC0 y FFFF por lo tanto serán grabados en la memoria EEPROM.

### 3.1.9.-Uso del 74HC138 para habilitar los puertos de entrada y salida.

La figura 3.6 muestra un 74HC138, el cual es comunmente usado para decodificar las direcciones y habilitar circuitos en diferentes posiciones del espacio de direccionamiento.



Este circuito es un decodificador de 3 a 8 líneas: un número binario de tres dígitos (las entradas de selección) causa que una de sus ocho salidas sea seleccionada (las salidas de control). EL circuito tiene además tres líneas de entrada de control, las cuales deben ser habilitadas simultáneamente para que el circuito funcione.

Las salidas del 74HC138 controlan los módulos de entrada y salida, determinando si se activa un circuito de salida 74HC374 o un circuito de entrada 74HC244.

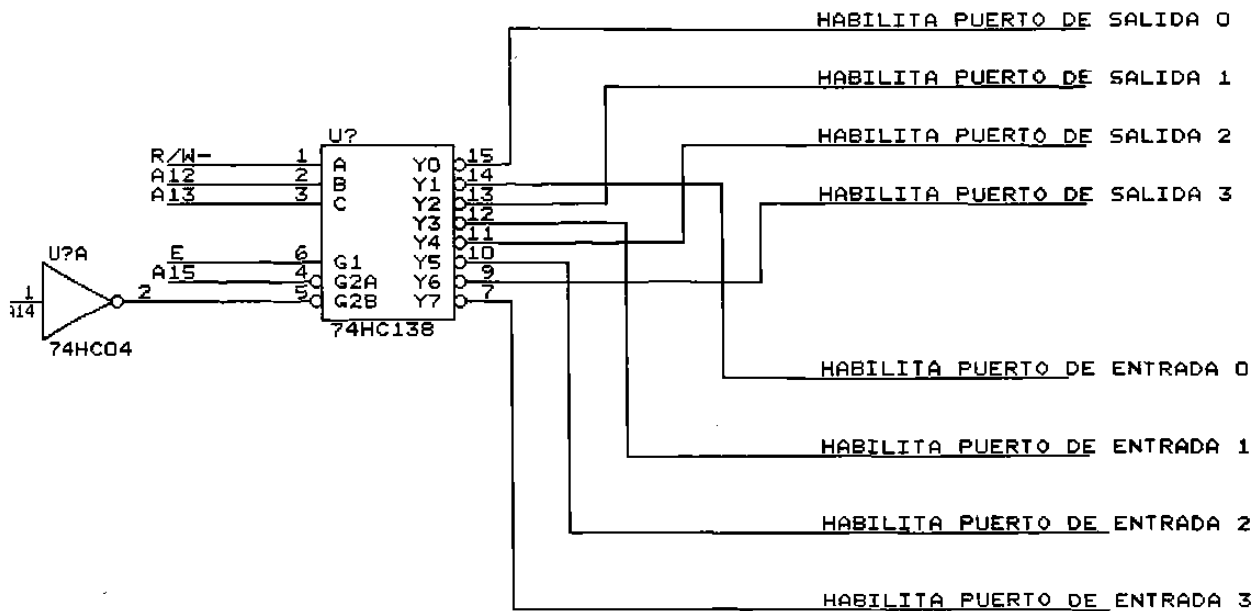


Fig. 3.6 Decodificador de direcciones para los puertos de entrada y salida.

Como se muestra en la figura 3.6, la línea A15 está conectada a una de las entradas bajo activas de habilitación G2A del decodificador, por lo tanto cuando sea baja activará el circuito (ya que si A15 es alta activará la memoria como se explicó antes). De esta manera no hay posibilidad de que la memoria y los puertos sean activados simultáneamente.

La otra entrada de habilitación bajo activa G2B del 74HC138 está conectada a la línea A14 negada, lo cual implica que para que el decodificador se active A14 debe ser alta para que al pasar por el inversor proporcione un nivel bajo.

La última entrada de habilitación del decodificador es alto activa y se conecta a la línea de reloj E.

Se concluye que cuando A15 sea cero y A14 sea alta, la señal de reloj E activará el 74HC138 en el momento adecuado para que se realice la habilitación de un puerto de entrada o salida.

Suponiendo que el decodificador haya sido habilitado, las entradas de selección C, B, y A determinan cual dispositivo conectado a las salidas del decodificador será activado. La línea A13 está conectada a la entrada de mayor peso C, la línea A12 se conecta con B, y la señal RW se conecta a la entrada de menor peso A, esta línea es alta para una lectura y baja para un ciclo de escritura. Suponiendo que A13 y A12 sean ambas altas y se esté realizando una lectura por lo que R/W- será alta, entonces las entradas CBA recibirán el número 111, y la salida Y7 será activada. Esta salida está conectada al circuito de entrada digital 74HC244, por lo que este se activará y colocará un byte de datos en el canal.

Si por otro lado se está realizando un ciclo de escritura, la línea RW será baja, y si nuevamente las líneas de direcciones A13 y A12 son ambas altas, entonces las entradas CBA recibirán el número 110, por lo que se activará la salida Y6. Esta salida está conectada al circuito de salida 74HC374, por lo que se exhibirá un byte de datos por ese puerto.

Nótese como ambos puertos, el de entrada y el de salida tienen la misma dirección en las líneas A13 y A12, únicamente la línea R/W- selecciona cual de los dos se activa.

Observe también como las líneas de A0 a la A11 no tienen efecto en la selección de los puertos, de tal manera que cada uno de ellos ocupa un bloque de 4 Kbytes en el mapa de memoria.

### 3.1.10.- Mapa de memoria del sistema.

La figura 3.7 muestra el mapa de memoria que ha sido utilizado por el sistema. Los 8 Kbytes de EEPROM han sido colocados en la parte más alta de la memoria, desde la dirección \$E000 a la \$FFFF, pero aparecen nuevamente entre la dirección A000 y BFFF.

Los 8 Kbytes de RAM se encuentran entre las direcciones C000 y DFFF.

Los 4 puertos de entrada o salida digital han sido colocados en bloques de 4 Kbytes cada uno de ellos, en las localidades que inician en \$4000, \$5000, \$6000 y \$7000.

Existe una pequeña cantidad de memoria RAM que se encuentra dentro del microcontrolador. Esta memoria consiste en 256 bytes localizados al inicio del espacio de direccionamiento, desde la \$00 a la \$FF.

El microcontrolador también cuenta con un banco de 64 registros internos para funciones especiales, ubicados de la dirección \$1000 a la \$103F.

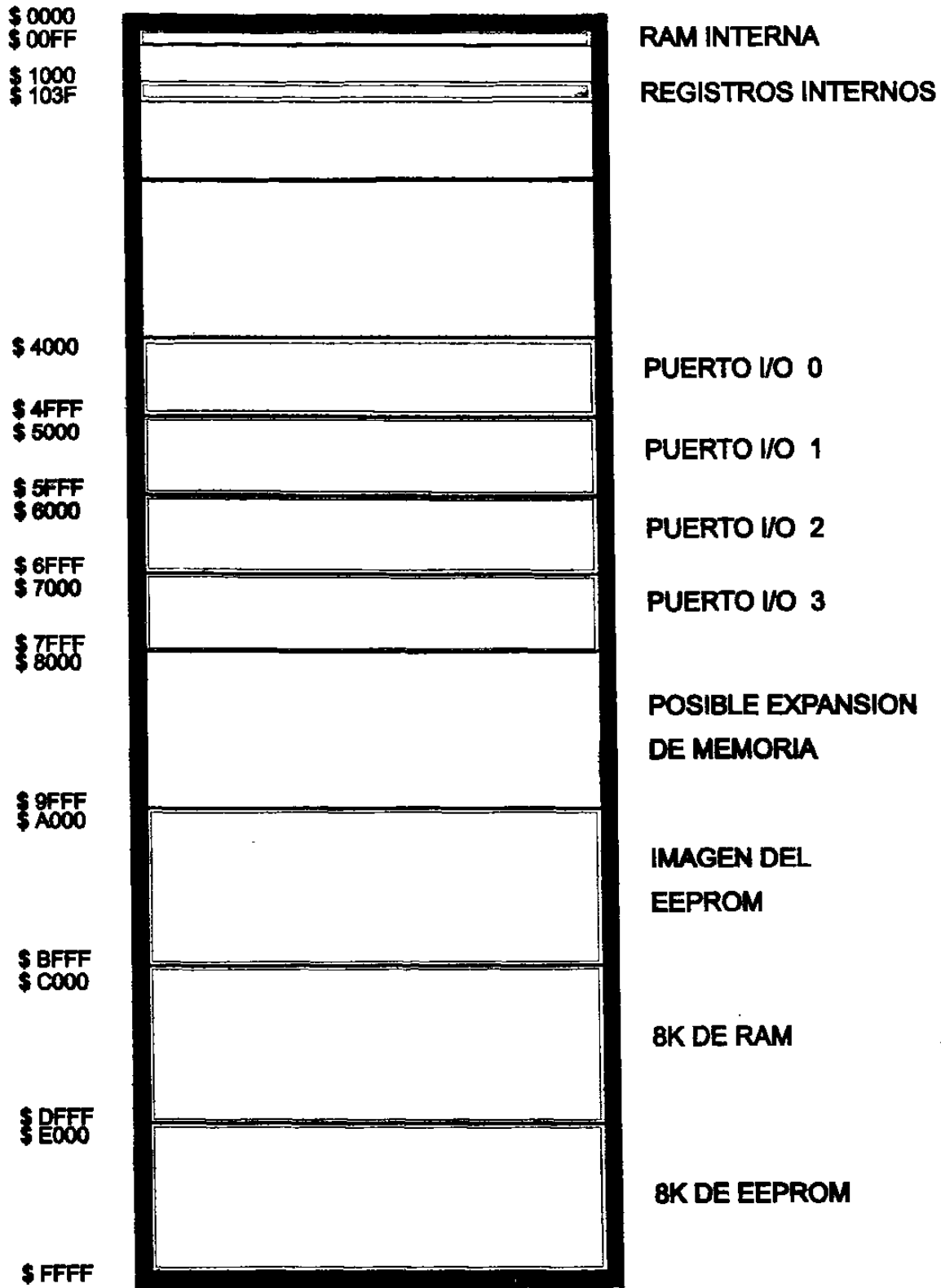


Fig 3.7.- Mapa de memoria del sistema.

### 3.2 Sección de entrada y salida.

#### 3.2.1.-Entradas digitales.

La figura 3.8 muestra un módulo de entrada digital, el circuito 74HC244 se utiliza para atrapar una palabra de ocho bits que viene de los sensores de entrada, para transferirla al canal de datos cuando haya sido seleccionado.

El circuito 74HC244 tiene dos mitades las cuales pueden ser habilitadas por separado, la línea de habilitación que proviene del 74HC138 está conectada a ambas entradas, de tal manera que ambas mitades son habilitadas simultáneamente.

Cada una de las entradas se debe conectar a una resistencia de 47 kohms, para que los valores que se obtengan por omisión sean niveles altos, y de esa manera no dejar entradas flotando.

En el sistema se pueden usar 4 módulos de entrada digital que se conectarán a cada una de las salidas impares del 74HC138: Y1, Y3, Y5, y Y7 respectivamente, para ser habilitados. De esta manera se puede tener un total de 32 entradas digitales.

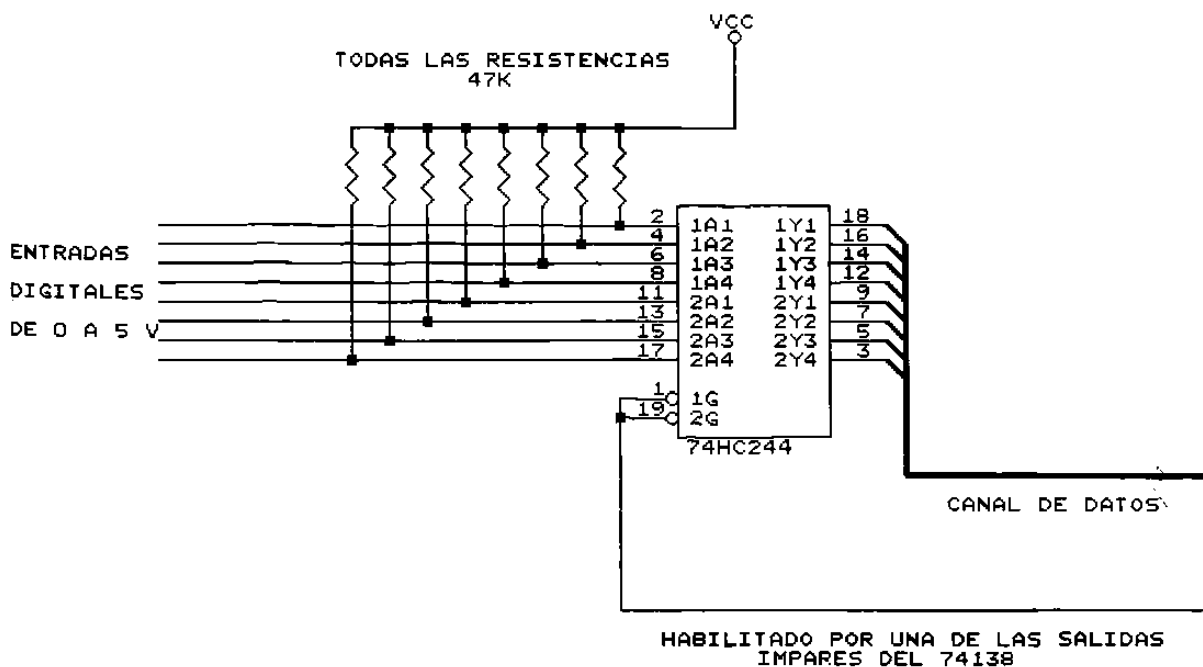


Fig 3.8 Módulo de entrada digital.

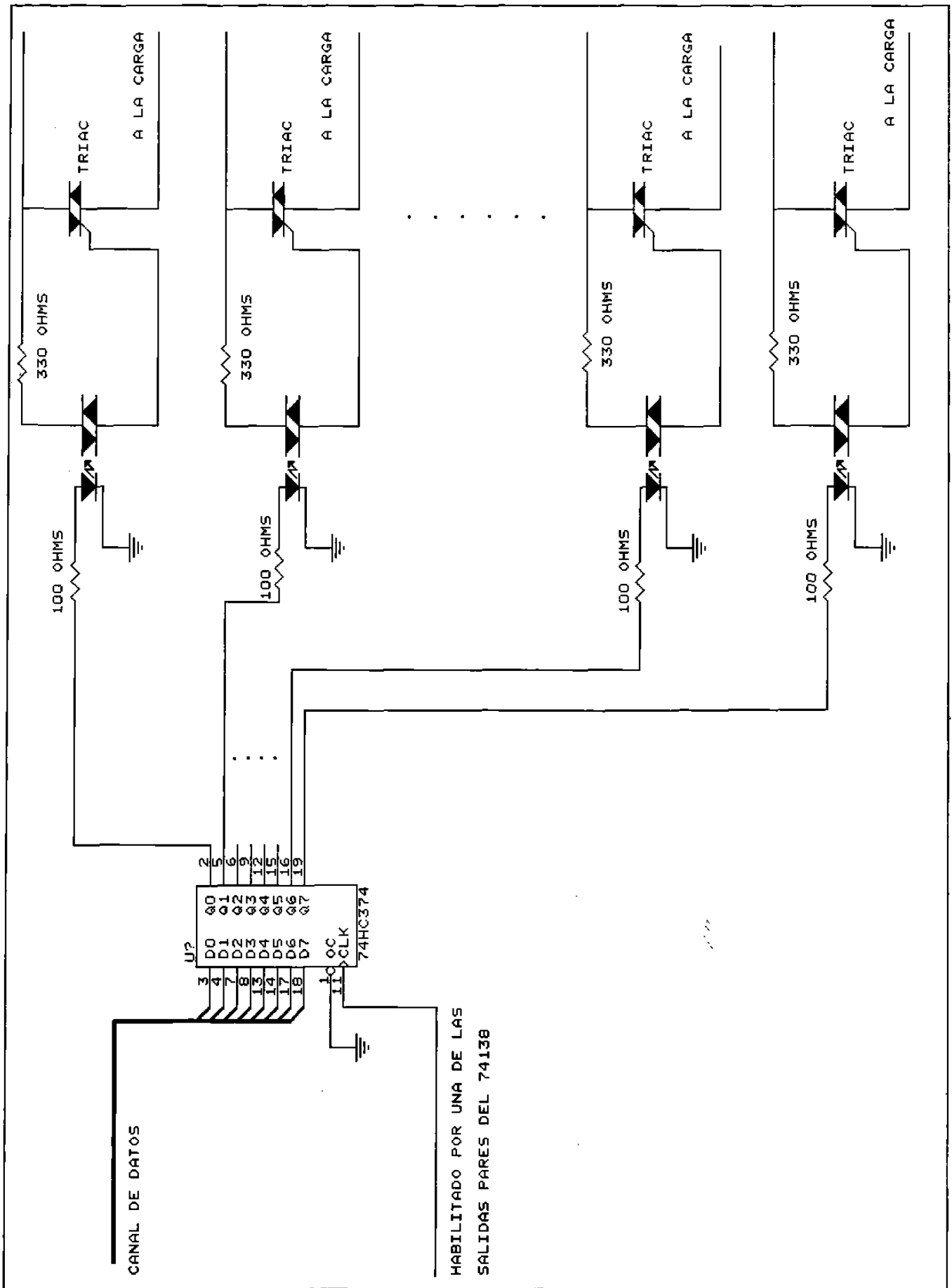


Fig 3.9 Módulo de salidas digitales.

### 3.2.2.-Salidas digitales.

La figura 3.9 muestra un módulo de salidas digitales, el cual está formado por una sección lógica que utiliza un 74HC374 para atrapar el valor presente en el canal de datos, cuando se active su entrada de reloj por medio del 74HC138. Después de la sección lógica sigue una etapa de acoplamiento óptico por medio del optoacoplador MOC3011. Por último está la etapa de potencia, que utiliza el triac 2N8344 para manejar cargas de hasta 15 amper.

El sistema es capaz de manejar cuatro módulos de salidas digitales haciendo un total de hasta 32 salidas.

### 3.2.3.-Entradas analógicas.

El microcontrolador cuenta con un convertidor A/D integrado para realizar la conversión análogo-digital. En esta operación, un voltaje comprendido entre 0 y 5 volts es linealmente convertido a un valor binario de 8 bits.

La figura 3.10 muestra un módulo de entradas analógicas, el cual usa un multicanalizador analógico 4051 para expandir la cantidad de entradas analógicas que puede recibir el microcontrolador. El 4051 tiene ocho entradas y una salida; dependiendo del valor de las tres entradas de selección, una de las ocho entradas será conectada hacia la salida, (en realidad el 4051 es bidireccional pero será usado solamente como multicanalizador con ocho entradas y una salida). El multicanalizador usa como entradas de selección las líneas D3, D4, y D5 del puerto D. Las entradas del multicanalizador vienen de los sensores de entrada, la salida del 4051 se conecta a una de las entradas del puerto E.

Como el microcontrolador tiene 4 entradas analógicas y cada multicanalizador aumenta 8 veces cada entrada, se podrían conectar un total de 32 entradas analógicas.

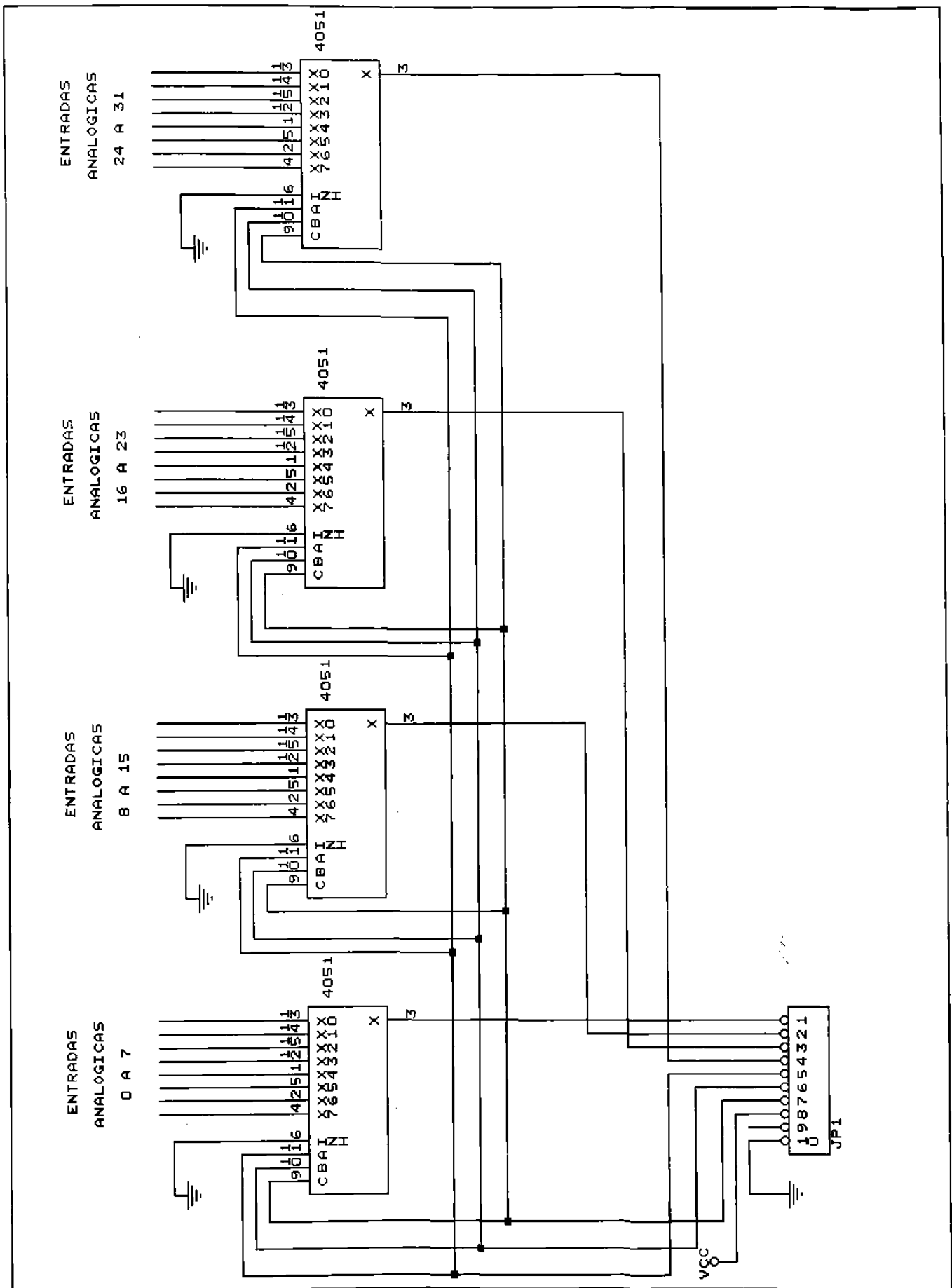


Fig 3.10 Módulo de entradas analógicas.

### 3.3.-Módulos simuladores.

#### 3.3.1.-Módulo simulador de entradas.

El módulo simulador de entradas consta del 74HC244, y ocho interruptores que pueden aplicar al sistema niveles altos o bajos dependiendo de la posición en que sean colocados. La figura 3.11 muestra la conexión del módulo simulador de entradas.

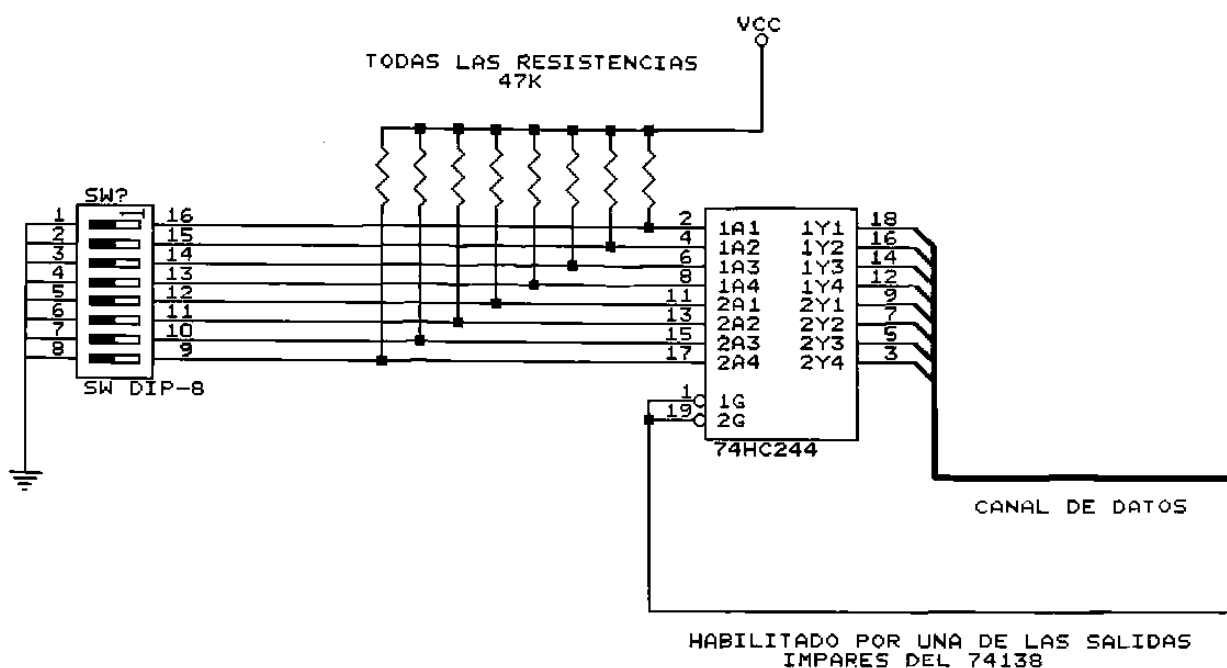


Fig 3.11 Módulo simulador de entradas.

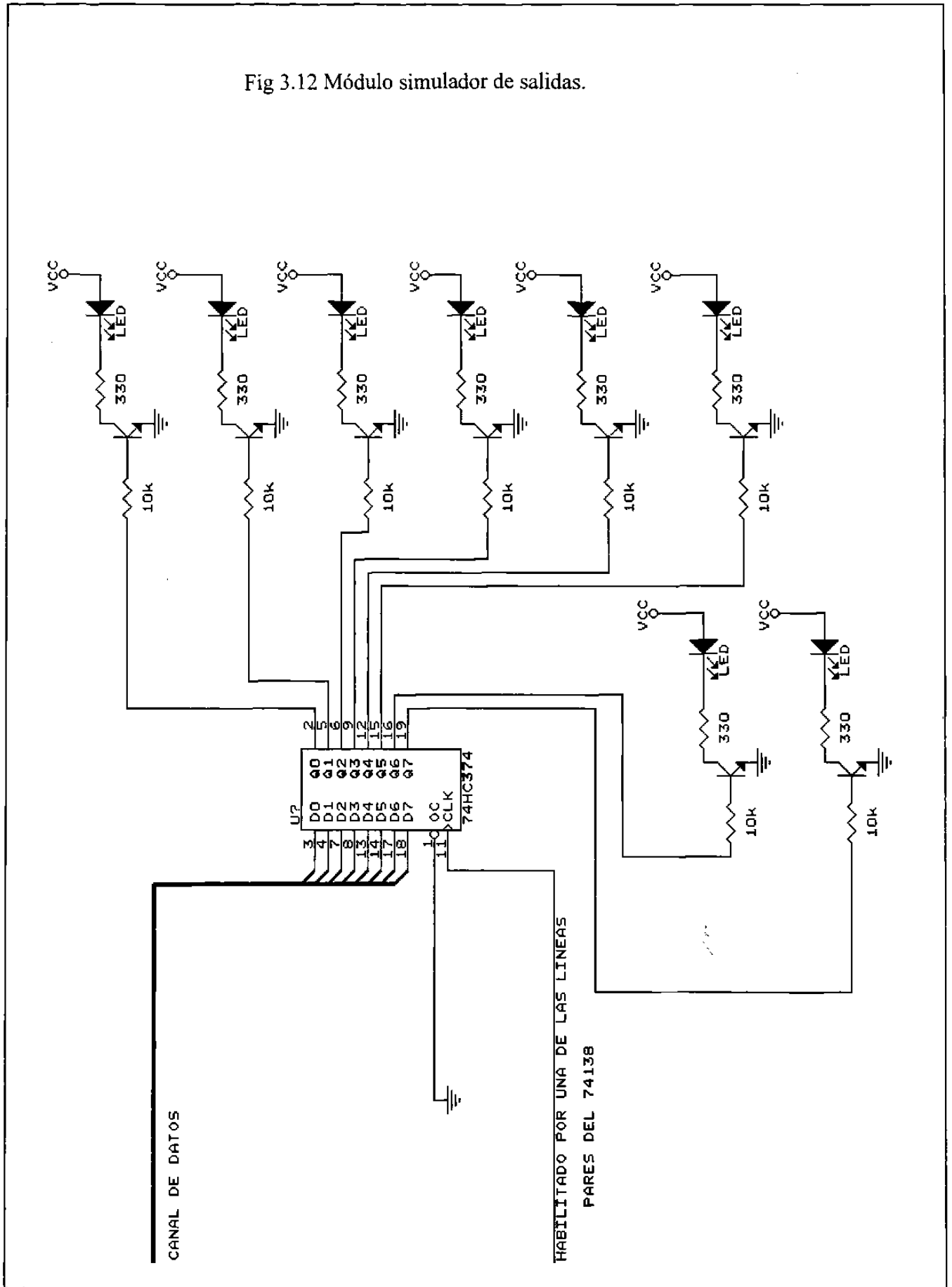
#### 3.3.2.-Módulo simulador de salidas.

El módulo simulador de salidas consta del circuito 74HC374, y un led para visualizar el nivel presente en dicha salida, un transistor 2N2222 trabajando en corte y saturación se encargará de proporcionar la corriente necesaria para que el LED encienda cuando la salida sea alta.

La figura 3.12 muestra las conexiones del módulo simulador de salidas.



Fig 3.12 Módulo simulador de salidas.



### 3.4.-Dispositivo de programación.

La computadora será el dispositivo de programación, la cual se comunicará a través del puerto serie con el sistema de comunicaciones serial (SCI) del microcontrolador.

#### 3.4.1.-Protocolo de comunicación RS232.

Se usará el protocolo RS232 el cual utiliza un sistema de tres líneas. En el sistema RS232, un cero lógico es indicado por un nivel de +15 volts con respecto a tierra, y el uno lógico se representa por una señal de -15 volts.

El microcontrolador cuenta con la circuitería para generar las formas de onda compatibles con los sistemas RS232, pero requiere de circuitería externa para cumplir con los niveles de voltaje, el circuito MAX232 logrará la interconexión adecuada.

La fig 3.13 muestra un circuito MAX232, fabricado por la compañía MAXIM, el cual contiene en su interior dos transmisores y dos receptores, y un circuito de bombeo de carga para producir los +10 y -10 volts requeridos por el transmisor a partir de la alimentación de 5 volts. Los únicos componentes externos requeridos son 4 capacitores de 22 microfaradios.

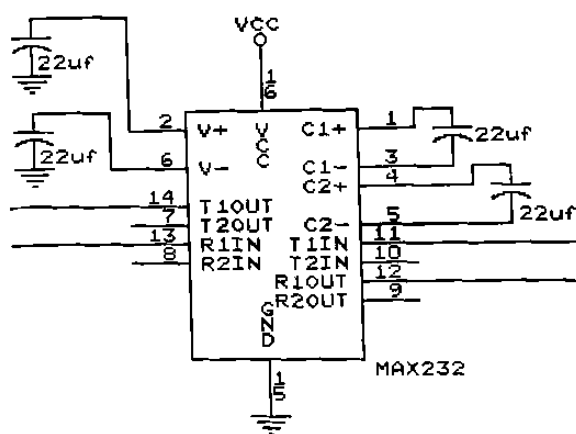


Fig. 3.13.- Conexión del circuito MAX232.

La interfase estándar RS232 también especifica los conectores y la asignación de terminales, la tabla 3.1 muestra la asignación de terminales para el conector DB25:

Tabla 3.1.-Asignación de terminales para el conector DB25, en el estándar RS232.

TERMINAL	NOMBRE	FUNCION
1	Tierra de protección ( FG)	Conecta el blindaje del cable a tierra, para proteger contra campos eléctricos, es diferente de la tierra de la señal.
2	Dato transmitido (TD)	Dato enviado de la computadora a la terminal.
3	Dato recibido (RD)	Dato enviado de la terminal a la computadora.
4	Solicitud para enviar (RTS)	Habilita los circuitos de transmisión (duplex completo).
5	Borrar para transmitir (CTS)	Respuesta a la solicitud para transmitir; significa que la transmisión puede proceder; cuando está alta indica que la circuitería de transmisión está trabajando.
6	Listo el conjunto de datos (DSR)	Especifica que la entrada o salida puede ocurrir, significa que la circuitería no se está usando para otros modos de operación, por lo tanto se puede usar para transmitir o recibir.
7	Tierra de la señal (SG)	Referencia de potencial común para todas las líneas.
8	Detección de portadora de datos (DCD)	Una buena señal está siendo recibida.
20	Listo el dato de la terminal (DTR)	Es una señal que indica el estado de la terminal.

La figura 3.14 muestra las conexiones del MAX232 con el conector DB25 y con las entradas del microcontrolador.

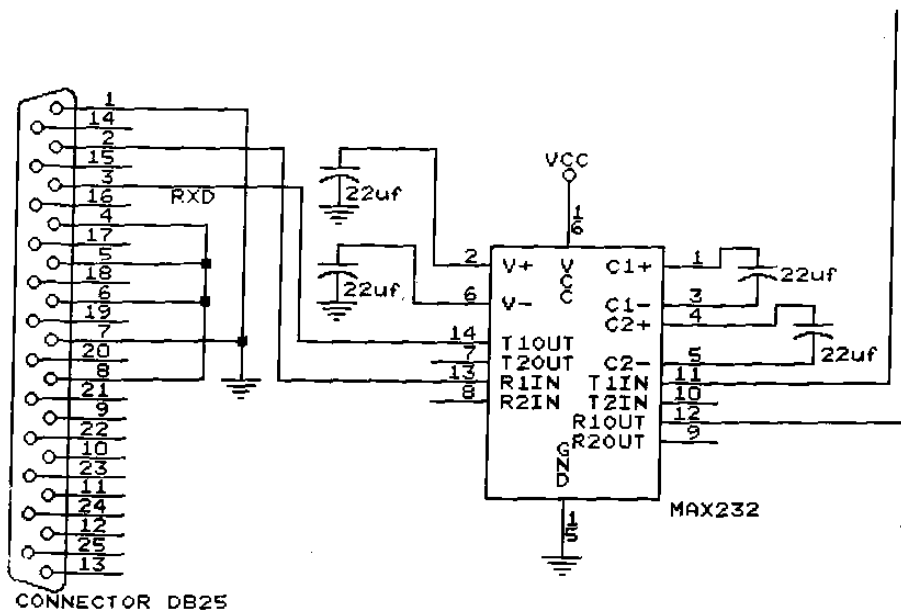


Fig 3.14.-Conexiones del MAX232 al DB25.

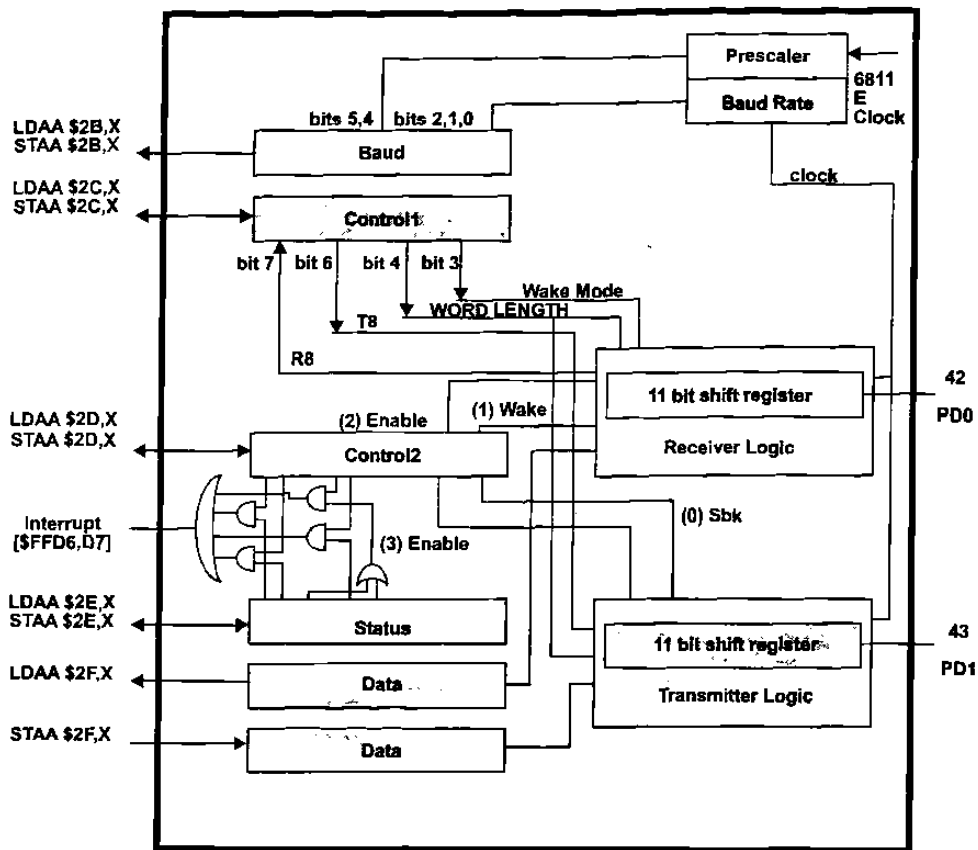


Fig 3.15.- Interfase de comunicación serie (SCI) del microcontrolador.

el sistema se desborde (overruns); es decir, una palabra debe ser movida del registro de corrimiento de entrada antes que la palabra anteriormente introducida sea leída del registro de datos.

El bit 2 bandera de ruido (NF) se pone en alto cuando el receptor detecta una transición corta positiva o negativa en la terminal de entrada de recepción, con una duración inferior a la de un bit completo. El bit 1 (FE) se pone en alto cuando se detecte un error en el marco; es decir, se espera un bit de paro y se recibe un nivel bajo. El bit 0 siempre es bajo. Para borrar el registro de estado (SCSR) se hace una lectura a éste y después a los registros de datos.

Los 4 bits de mayor peso del registro de control 2 (SCCR2), ubicado en la dirección \$102D, contienen habilitaciones de interrupción para los 5 bits más pesados del registro de estados (SCSR). El bit 3 (TE) habilita el transmisor, y el bit 2 (RE) habilita el receptor si son altos. Si el bit 0 es alto (SBK=1) se manda una pausa (break), que consiste en un nivel bajo continuo; y si hacemos (SBK=0) se detiene la transmisión de la pausa, pero hasta que se concluya el marco completo.

El siguiente procedimiento de inicialización sirve para ajustar el SCI a una velocidad de 1200 Baud y 8 bits de datos.

```

INIC  LDAA  #$33      SE AJUSTA LA VELOCIDAD A 1200 BAUD
      STAA  $102B
      CLR   $102C      SE SELECCIONAN 8 BITS DE DATOS
      LDAB  #$0C      SE HABILITAN TRANSMISOR Y RECEPTOR
      STAB  $102D

```

La siguiente rutina sirve para transmitir un byte del acumulador A.

```

TRANS BRCLR  $102E $80 TRANS  ESPERA HASTA QUE TDRE=1
      STAA   $102F           MANDA LOS DATOS

```

La rutina para recibir una palabra en el acumulador A es la siguiente:

```

REC   BRCLR  $102E $20 REC     ESPERA HASTA QUE RDRF=1
      LDAA   $102F           OBTENER LOS DATOS

```

## **CAPÍTULO 4**

### **DIAGRAMA ELECTRONICO**

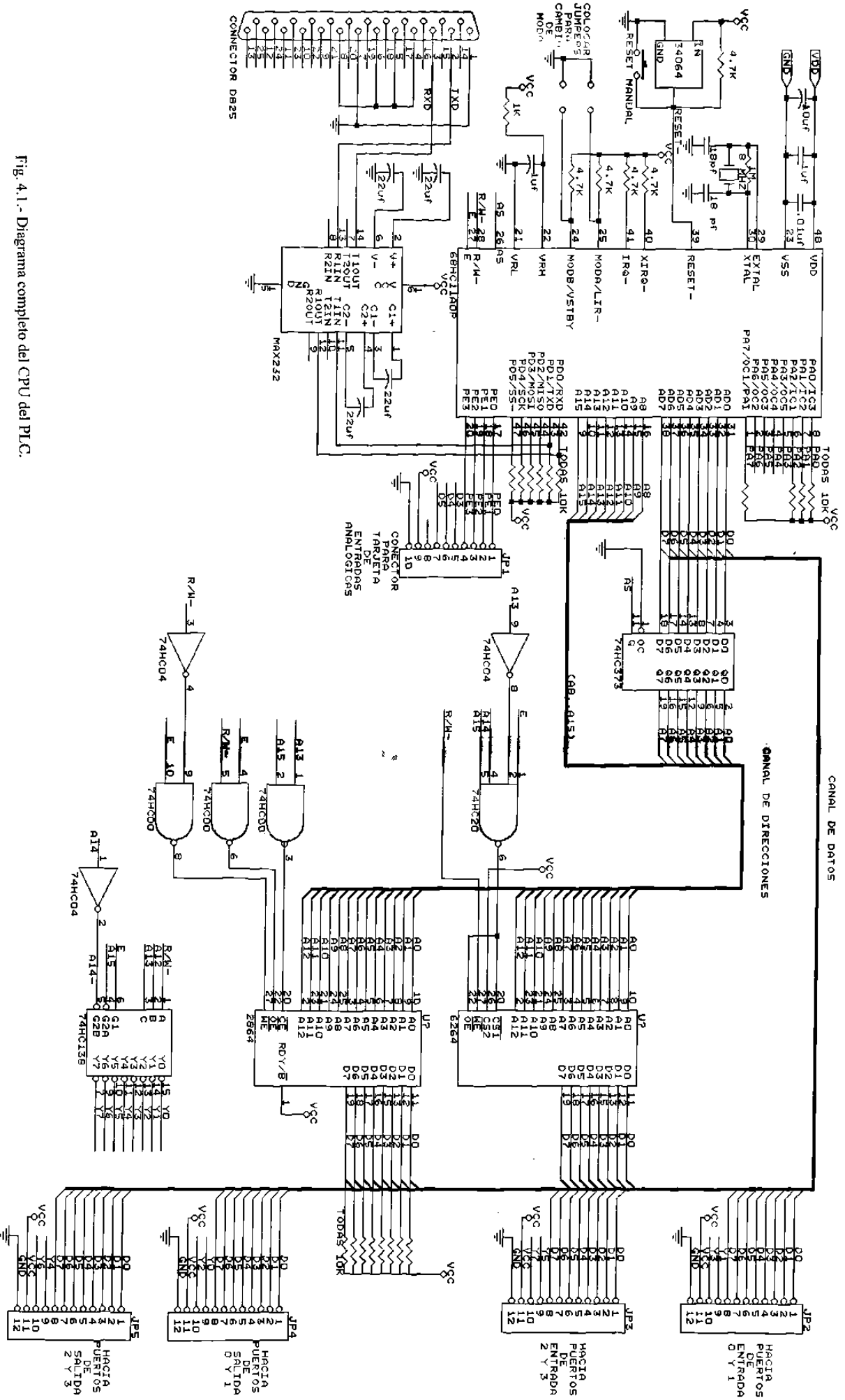


Fig. 4.1.- Diagrama completo del CPU del PLC.





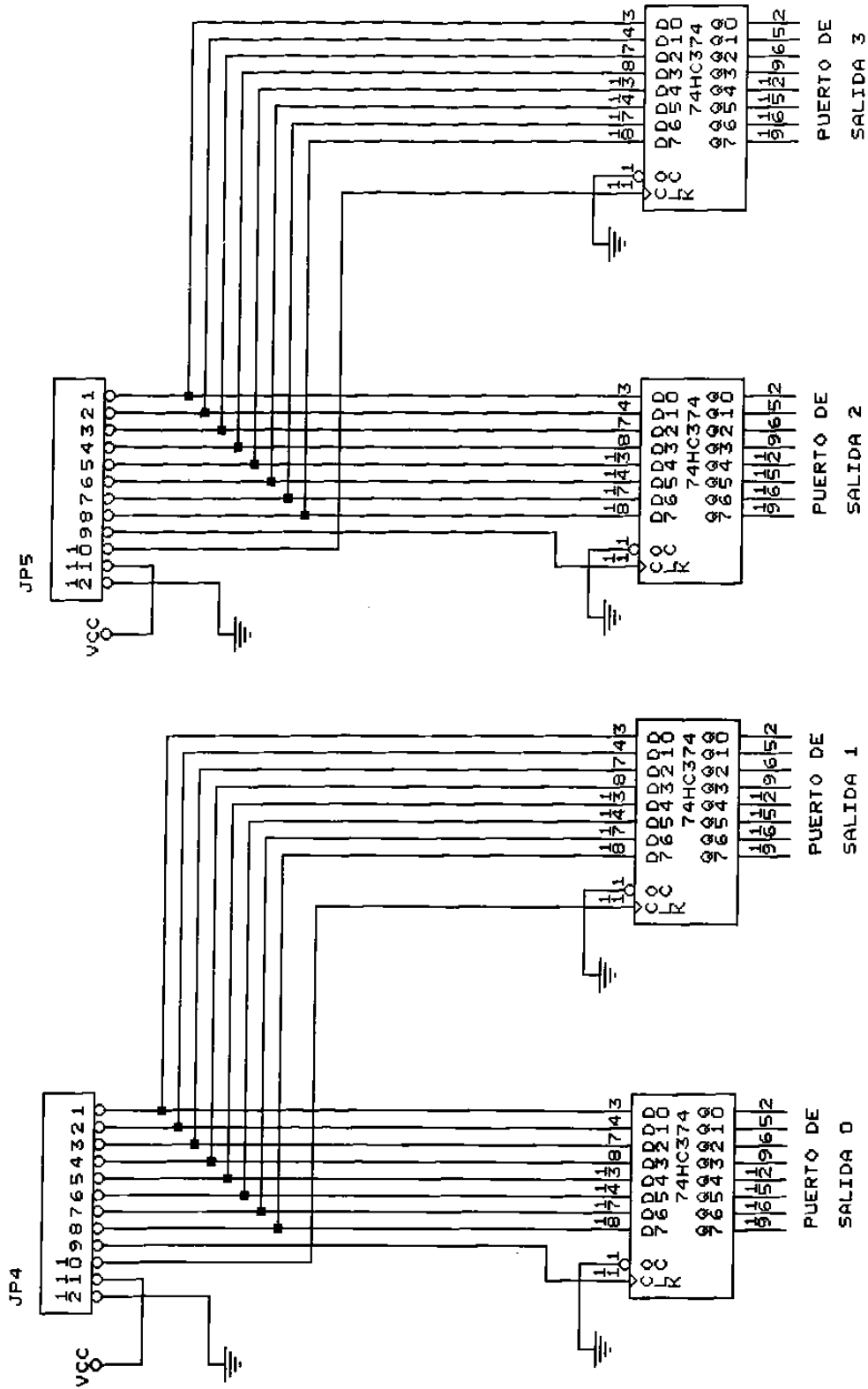


Fig 4.3.- Salidas digitales del PLC.

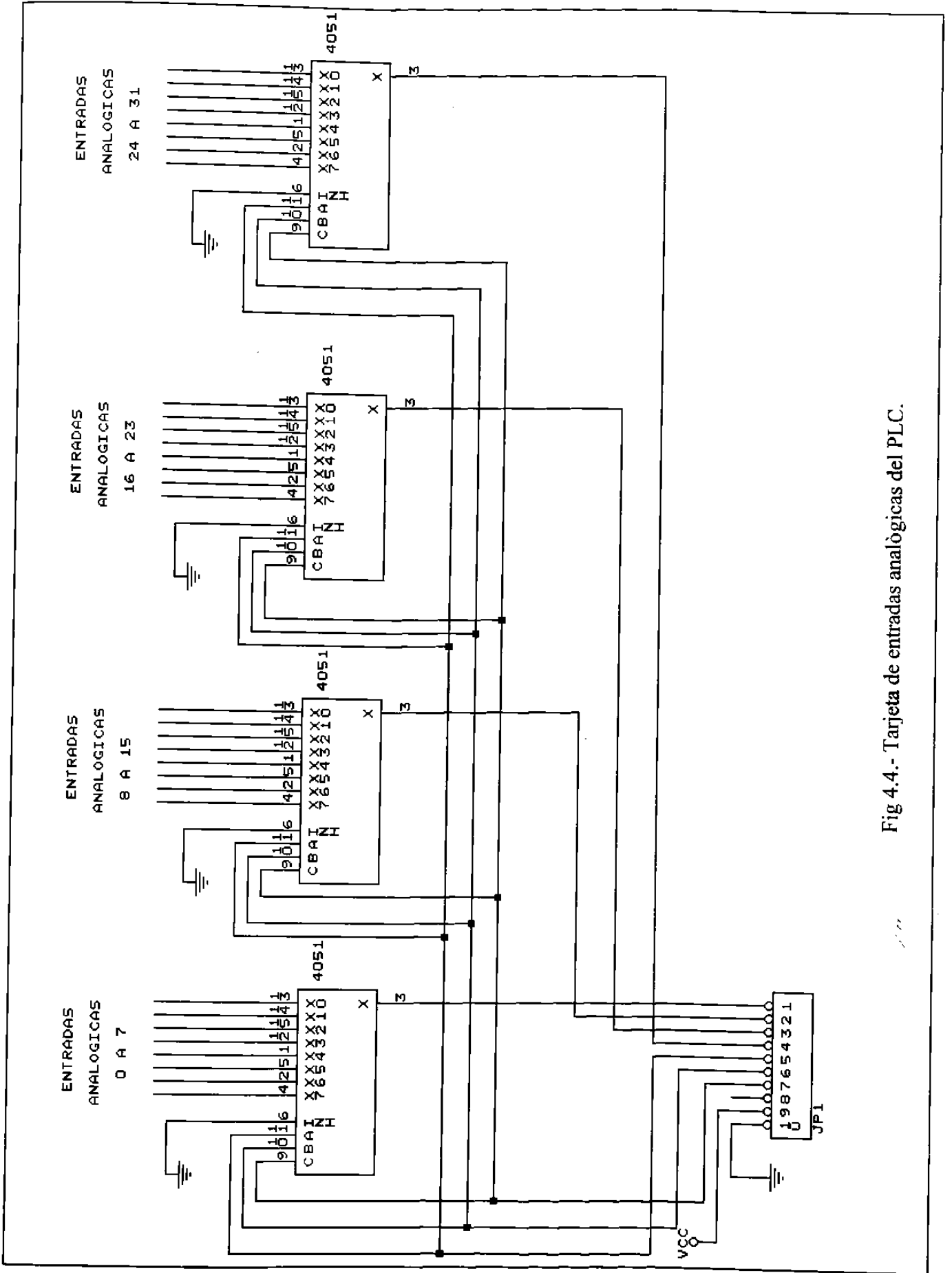


Fig 4.4.- Tarjeta de entradas analógicas del PLC.

**CAPÍTULO 5**  
**PROGRAMACION**

## CAPÍTULO 5

### PROGRAMACION

#### 5.1.- Sintaxis del ensamblador.

##### 5.1.1.- Campos del ensamblador,

Cada línea de código tiene cuatro campos separados por espacios en blanco:

Etiqueta      Mnemónico    Operando(s)    Comentario

a)Etiqueta.- Una etiqueta es un símbolo que empieza en la primera columna y se utiliza para referenciar un punto del programa.

b)Mnemónico.- Un mnemónico es un símbolo precedido por al menos un espacio en blanco, el cual nos indica la operación que se va a realizar. Las letras utilizadas en este campo pueden ser mayúsculas o minúsculas indistintamente.

c)Operando.- El operando siempre sigue al mnemónico, separado por al menos un espacio en blanco. El contenido del operando es interpretado por cada instrucción. Las instrucciones en el modo implícito no requieren operandos. Algunas instrucciones utilizan varios operandos, los cuales son separados por espacios en blanco. Los operandos de **forzamiento de tamaño** pueden ser usados para forzar el direccionamiento directo ( LDAA <CONT ), o el direccionamiento extendido ( LDAA >CONT ).

d)Comentarios.- El comentario es cualquier texto que se coloca después de los operandos, con la intención de hacer mas claro el programa para que pueda ser interpretado mas fácilmente por otros programadores, o por el mismo programador después de cierto tiempo. El colocar un asterisco solo en toda la línea o bien dejar la línea en blanco puede ser considerado como un comentario para separar el programa en segmentos.

##### 5.1.2.- Expresiones.

Las expresiones pueden consistir de símbolos, constantes o el carácter \* (que significa el valor actual del contador del programa) unidos por medio de alguno de los operadores que se listan a continuación:

OPERADOR	SIGNIFICADO
+	SUMA
-	RESTA
*	MULTIPLICACION
/	DIVISION

%	RESIDUO DESPUES DE LA DIVISION
	OR LOGICA
&	AND LOGICA
=	IGUALDAD
^	OR EXCLUSIVO

Todos los operadores son iguales que en lenguaje C, excepto el operador de igualdad, que en programación C se representa por == .

Una expresión de igualdad ( = ) evalúa ambos lados y regresa un 0 si ambos lados son diferentes, y un 1 si ambos lados son iguales. Las expresiones se evalúan de izquierda a derecha, y no está permitido el uso de paréntesis. La aritmética se realiza en precisión entera en complementos a dos.

Las constantes se preceden de alguno de los siguientes símbolos para indicar su tipo:

SIMBOLO	SIGNIFICADO
\$	REPRESENTA UNA CONSTANTE HEXADECIMAL
%	REPRESENTA CONSTANTE BINARIA
NINGUNO	PARA REPRESENTAR CONSTANTE DECIMAL
@	INDICA CONSTANTE OCTAL
'	SEGUIDO DE UN CARÁCTER ASCII

### 5.1.3.- Directivos del ensamblador.

Los directivos se usan para darle instrucciones al ensamblador, algunos de ellos se incluyen pero no realizan ninguna función como END (fin de programa), NAM o NAME (nombre del programa).

A continuación se da una lista de otros directivos:

a) EQU asigna a la etiqueta el valor que aparece a la derecha.

Por ejemplo:

L EQU \$1000 Ocasiona que L adquiera el valor \$1000.

b) ORG se usa para establecer el origen del contador de localidades, le indica al ensamblador donde colocar la siguiente palabra que se genere después de ORG. Por ejemplo:

ORG \$100

LDAA #\$80 Colocará el código de instrucción de LDAA en la localidad \$100 y

el

valor del operando ( \$80 ) en la localidad \$101.

c) RMB reserva la cantidad de bytes especificada al lado derecho, y designa a la etiqueta que aparece a su izquierda como la primera palabra del área que ha sido asignada.

Por ejemplo:

C RMB 2 Reserva dos localidades de memoria, siendo C la etiqueta de la primera localidad.

d) FCB Genera un byte por cada argumento que aparezca a la derecha separados por comas, y asigna la etiqueta que aparece a la izquierda al primer byte. Por ejemplo:

C FCB M1, M2 Forma un byte para cada uno de los dos argumentos, asignando la etiqueta C al primer argumento.

e) FDB Genera dos bytes por cada argumento, asignando la etiqueta que aparece a la izquierda al primer byte del primer argumento. Por ejemplo:

C FDB M1, M2 Forma cuatro bytes, dos para cada argumento; y asigna la etiqueta C al primer byte de M1.

f) FCC Genera el código ASCII para cada carácter que aparece a la derecha entre comillas simples. asignando la etiqueta que aparece a la izquierda al primer carácter.

Por ejemplo:

C FCC 'JOE' Genera el código ASCII para cada uno de los tres caracteres que aparecen entre comillas, y asigna la etiqueta C al primer carácter.

g) OPT Establece las opciones del ensamblador, de acuerdo con el operando que aparece a la derecha. Las opciones pueden ser:

OPT C Causa que el listado del ensamblador muestre el conteo de los ciclos

OPT COND Causa la impresión del código ensamblado condicionalmente.

OPT CRE Imprime una referencia cruzada al final de la lista ensamblada.

OPT GEN Muestra todo el código generado aunque no quepa en una línea.

OPT FORCE Forza la ejecución del segundo paso aunque existan errores en el primero.

OPT NOL Desactiva temporalmente el listado.

OPT I Reactiva el listado.

OPT S Imprime una tabla de símbolos al final.

OPT WARN Habilita el listado de los mensajes de advertencia.

OPT SRECORD muestra el listado del código objeto de los archivos tipo S.

Si se colocan las letras NO antes de la opción se remueve el efecto del mandato, de tal forma que otras opciones son: NOWARN, NOCOND, NOGEN, NOC.

h) BSZ            Almacena ceros en un bloque de memoria, el tamaño del bloque se especifica a la derecha del directivo.

i) ZMB            Es equivalente a BSZ.

j) FILL            Puede ser usado para inicializar la memoria a un valor diferente de cero. Por ejemplo: FILL \$FF,4      Llenará cuatro localidades de memoria con unos.

5.1.4.- Errores y advertencias dadas por el ensamblador.

Algunos errores que cometemos muy frecuentemente al programar son:

1) Confundir el cero (0) con la letra O, y confundir el número uno (1) con la letra l.

2) Dejar espacios en el nombre de la etiqueta, en el código de operación, o en el operando. Por ejemplo:

LDA A R1            Hay un error porque lo que queríamos era LDAA R1.

LDAA R 1            Se debió escribir como LDAA R1.

C 1 LDAA            Tiene un error porque no debemos dejar espacios en la etiqueta.

3) Si olvidamos colocar el directivo de origen ORG no sabrá la primer dirección donde empieza el programa.

4) Olvidar colocar la coma en el direccionamiento indexado. Por ejemplo:

DEC X                Significa decrementar la localidad de memoria X. Mientras que:

DEC,X                Significa decrementar el registro índice X.

5) Dejar de colocar el # cuando se trate de direccionamiento inmediato. Por ejemplo:

LDAA \$80            Carga en el acumulador el contenido de la localidad \$80. Mientras que:

LDAA #80            Carga el acumulador con el número 80.

6) Olvidar el signo \$ cuando queremos usar números hexadecimales. Por ejemplo:

LDAA #80            Carga en el acumulador el número hexadecimal 80. Mientras que:

LDAA #80            Coloca en el acumulador el número decimal 80.

Los mensajes de advertencia ayudan al programador a identificar posibles errores. Una recomendación ocurrirá si una expresión es truncada. Por ejemplo:

LDAA #580 marcará advertencia porque 580 no cabe en el acumulador.

FDB \$45654 no cabe en un campo de 16 bits y marcará advertencia.

Las advertencias también señalarán la posibilidad de ahorrar espacio o tiempo. Por ejemplo cuando usamos un brinco (JUMP) pudiendo haber usado una bifurcación relativa (BRANCH), como se muestra a continuación:

LDAA #5

M      DECA

BEQ SALIDA

JMP M            Marcará advertencia ya que debemos usar BRA en lugar de

JMP.

## 5.2.- Conjunto de instrucciones del microcontrolador.

El conjunto de instrucciones lo podemos clasificar en cuatro grupos de acuerdo con la función que realizan:

- 1.- Instrucciones para movimiento de datos.
- 2.- Instrucciones aritméticas y lógicas.
- 3.- Instrucciones de rotación y corrimiento.
- 4.- Instrucciones de control.

### 5.2.1.- Instrucciones para movimiento de datos.

INSTR	SIGNIFICADO	DIRECCIONAMIENTO
CLR	BORRAR UN BYTE DE MEMORIA	EXT, IND
CLRA	BORRAR EL ACUMULADOR A	INH
CLRB	BORRAR EL ACUMULADOR B	INH
LDAA	CARGAR EL ACUMULADOR A	INM, DIR, EXT,
IND		
LDAB	CARGAR EL ACUMULADOR B	INM, DIR, EXT, IND
LDD	CARGAR EL ACUMULADOR D	INM, DIR, EXT, IND
LDX	CARGAR EL REGISTRO X	INM, DIR, EXT, IND
LDY	CARGAR EL REGISTRO Y	INM, DIR, EXT, IND
LDS	CARGAR EL APUNTADOR	INM, DIR, EXT, IND
PULA	EXTRAER A DE LA PILA	INH
PULB	EXTRAER B DE LA PILA	INH
PULX	EXTRAER X DE LA PILA	INH
PULY	EXTRAER Y DE LA PILA	INH
PSHA	GUARDAR A EN LA PILA	INH
PSHB	GUARDAR B EN LA PILA	INH
PSHX	GUARDAR X EN LA PILA	INH
PSHY	GUARDAR Y EN LA PILA	INH
STAA	GUARDAR EL ACUMULADOR A	INM, DIR, EXT, IND
STAB	GUARDAR EL ACUMULADOR B	INM, DIR, EXT, IND
STD	GUARDAR EL ACUMULADOR D	INM, DIR, EXT, IND
STX	GUARDAR EL REGISTRO X	INM, DIR, EXT, IND
STY	GUARDAR EL REGISTRO Y	INM, DIR, EXT, IND
STS	GUARDAR EL APUNTADOR	INM, DIR, EXT, IND
TAB	TRANSFERIR A HACIA B	INH
TAP	TRANSFERIR A HACIA EL CCR	INH



TBA	TRANSFERIR B HACIA A	INH
TPA	TRANSFERIR CCR HACIA A	INH
TSX	TRANSFERIR EL APUNTADOR A X	INII
TXS	TRANSFERIR X AL APUNTADOR	INH
TSY	TRANSFERIR EL APUNTADOR A Y	INH
TYS	TRANSFERIR Y AL APUNTADOR	INH
TST	PROBAR SI ES CERO O NEG	EXT, IND
TSTA	PROBAR SI A ES CERO O NEG	INH
TSTB	PROBAR SI B ES CERO O NEG	INH
XGDX	INTERCAMBIAR D CON X	INH
XGDY	INTERCAMBIAR D CON Y	INH

Las instrucciones de carga y almacenamiento pueden cargar o guardar cualquiera de los registros de ocho bits A o B, o los registros de 16 bits X, Y, S, y D.

Un registro índice puede ser usado en cálculos de direccionamiento, aún cuando ese registro sea cambiado por la instrucción misma, debido a que en el ciclo de búsqueda y ejecución, las direcciones son calculadas antes de que los datos de la memoria sean cargados dentro del registro. Por ejemplo si el registro x tiene el valor \$8000, entonces la instrucción:

LDX 0,X          cargará en el registro X el valor que hay en las direcciones \$8000 y \$8001.

Las instrucciones de transferencia e intercambio permiten el movimiento de datos entre registros de similar tamaño. Las instrucciones de transferencia no alteran el contenido inicial del registro de origen.

Las instrucciones de carga y almacenamiento modifican los bits N y Z del CCR, los cuales pueden ser usados en bifurcación condicional. El bit N se hace alto si la palabra transferida es negativa, y bajo si es positiva. El bit Z es alto si la palabra transferida es cero, de otra manera será bajo.

Si se desea modificar el CCR como en una instrucción de carga, pero no se quiere modificar el acumulador, se pueden usar las instrucciones TSTA y TSTB.

Para inicializar el acumulador A con cero se puede usar la instrucción CLRA en lugar de usar LDAA #0. Pero deberá tenerse en cuenta que CLRA modifica las banderas V y C, mientras que LDAA #0 no lo hace, por lo que deberá usarse esta última si no se desea modificarlas.

## 5.2.2.- Instrucciones aritméticas y lógicas.

INSTRUCCION	SIGNIFICADO	DIRECCIONAMIENTO
ABX	SUMAR ACUM. B CON X	INH
ABY	SUMAR ACUM B CON Y	INH
ADDA	SUMAR MEMORIA CON A	INM, DIR, EXT, INDEX
ADDB	SUMAR MEMORIA CON B	INM, DIR, EXT, INDEX
ADCA	SUMAR CON ACARREO A	INM, DIR, EXT, INDEX
ADCB	SUMAR CON ACARREO B	INM, DIR, EXT, INDEX
ABA	SUMAR ACUMULADORES	INH
ADDD	SUMAR 16 BITS DE MEM. CON D	INM, DIR, EXT, INDEX
ANDA	AND ACUM. A CON LA MEMORIA	INM, DIR, EXT, INDEX
ANDB	AND ACUM. B CON LA MEMORIA	INM, DIR, EXT, INDEX
BITA	PROBAR BITS DE A CON LA MEM.	INM, DIR, EXT, INDEX
BITB	PROBAR BITS DE B CON LA MEM.	INM, DIR, EXT, INDEX
BCLR	BORRAR BITS EN MEMORIA	DIR, INDEX
BSET	PONER LOS BITS EN MEM. ALTOS	DIR, INDEX
CPD	COMPARE 16 BITS DE MEM. CON D	INM, DIR, EXT INDEX
CPX	COMPARE 16 BITS DE MEM. CON X	INM, DIR, EXT, INDEX
CPY	COMPARE 16 BITS DE MEM. CON Y	INM, DIR, EXT, INDEX
CPMA	COMPARE A CON MEMORIA	INM, DIR, EXT, INDEX
CMPB	COMPARE B CON MEMORIA	INM, DIR, EXT, INDEX
CBA	COMPARE A CON B	INH
COM	COMPL. A UNO DEL BYTE DE M	EXT, INDEX
COMA	COMPL. A UNO DEL ACUM. A	INH
COMB	COMPL. A UNO DEL ACUM. B	INH
DAA	AJUSTE DECIMAL DE A (PARA BCD)	INH
DECA	DECREMENTAR ACUMULADOR A	INH
DECB	DECREMENTAR ACUMULADOR B	INH
DEC	DECREMENTAR BYTE DE MEMORIA	EXT, INDEX
DEX	DECREMENTAR REG.INDICE X	INH
DEY	DECREMENTAR REG. ÍNDICE Y	INH
DES	DECREMENTAR APUNTADOR SP	INH
EORA	EXOR DE A CON MEMORIA	INM, DIR, EXT, INDEX
EORB	EXOR DE B CON MEMORIA	INM, DIR, EXT, INDEX
INCA	INCREMENTAR A	INH

INCB	INCREMENTAR B	INH
INC	INCREMENTAR BYTE DE MEM.	EXT, INDEX
INX	INCREMENTAR REG. INDICE X	INH
INY	INCREMENTAR REG. INDICE Y	INH
INS	INCREMENTAR APUNTAOR SP	INH
IDIV	DIVISION ENTERA (D/X-X; r -X)	INH
FDIV	DIVISION FRACCIONARIA	INH
LSRA	CORR. A LA DER. LOGICO DE A	INH
MUL	MULTIPLICAR (A*B -D)	INH
NEGA	COMPL. A DOS DE ACUM. A	INH
NEGB	COMPL. A DOS DE ACUM. B	INH
NEG	COMPL. A DOS DE UN BYTE DE M	EXT, INDEX
ORAA	OR DEL ACUM. A CON MEMORIA	INM, DIR, EXT, INDEX
ORAB	OR DEL ACUM. B CON MEMORIA	INM, DIR, EXT, INDEX
SUBA	RESTAR MEM. AL ACUM. A	INM, DIR, EXT, INDEX
SUBB	RESTAR MEM. AL ACUM. B	INM, DIR, EXT, INDEX
SBCA	RESTAR CON ACARREO M DE A	INM, DIR, EXT, INDEX
SBCB	RESTAR CON ACARREO M DE B	INM, DIR, EXT, INDEX
SBA	RESTAR ACUMULADORES	INH
SUBD	RESTAR 16 BITS DE MEM. A D	INM, DIR, EXT, IND
SEC	PONER EN ALTO EL ACARREO	INH
SEI	PONER EN ALTO EL BIT I DEL CCR	INH
SEV	PONER EN ALTO EL BIT V DEL CCR	INH
CLC	BORRAR EL BIT DE ACARREO	INH
CLI	BORRAR EL BIT DE INTERRUPCION	INH
CLV	BORRAR EL BIT DE SOBREFLUJO	INH

Las instrucciones aritméticas suman, restan, multiplican o dividen el valor en el acumulador con el valor de una palabra extraída de la memoria.

La instrucción ADD es la misma para la aritmética con signo o sin signo, lo único que varia es la prueba que se haga al registro de código de condición después de efectuar la operación.

La instrucción de suma con acarreo ADC se usa para sumar palabras de varios bytes, esta instrucción suma el número extraído de la memoria con el acumulador, agregando además el bit de acarreo anterior en la posición menos significativa.

La instrucción compare CMPA es muy parecida a la instrucción de resta SUBA, pero no cambia el valor del acumulador, los códigos de condición son los únicos que cambian y pueden ser probados por instrucciones de salto condicional.

Para multiplicar dos números de 8 bits sin signo, se colocan en los acumuladores A y B, y se efectúa la operación MUL; el resultado de 16 bits se podrá leer en el acumulador D.

Para efectuar la suma de dos números codificados en decimal (BCD), se realiza la suma usando la instrucción ADDA o ADCA, seguida inmediatamente de la instrucción DAA (ajuste decimal al acumulador A). La instrucción DAA usa las banderas de acarreo y medio acarreo para corregir el número de tal manera que el resultado será la suma BCD de los dos números que han sido sumados. Hay que tener en cuenta que la dirección implícita para DAA es el acumulador A, y que el medio acarreo sólo se modifica por las instrucciones ADDA y ADCA, por lo que DAA sólo funcionará después de estas instrucciones.

La instrucción FDIV divide el acumulador D entre el registro índice X considerado como una fracción, dejando el cociente en X y el residuo en D.

La instrucción IDIV divide el acumulador D entre el registro índice X considerado como un entero, dejando el cociente en X y el residuo en D.

La instrucción ABX suma el contenido del acumulador B, considerado como un número sin signo con el registro índice X.

La instrucción BITA efectúa la operación AND del acumulador A con el operando, pero solo cambia los códigos de condición sin alterar el acumulador. Puede ser usado como la instrucción CMPA para comparar una palabra, con otras palabras de la memoria, para ver si algunos de sus bits son cero.

Las instrucciones orientadas a bits permiten poner altos algunos bits en particular usando BSET, o borrar algunos bits específicos usando BCLR. Por ejemplo:

BCLR OUT 4 borrará el bit 2 de la variable OUT.

BSET IN %00000011 pondrá en nivel alto los dos bits de menor peso de la variable IN.

Los bits del CCR frecuentemente son modificados. El bit I del CCR puede ser colocado en nivel alto por medio de SEI para evitar que ocurran las interrupciones, o puede ser borrado por medio de CLI para permitir que ocurran.

## 5.2.3.- Instrucciones de rotación y corrimiento.

ASLA	CORRIMIENTO A LA IZQ. ARITM. A	INH
ASLB	CORRIMIENTO A LA IZQ. ARITM. B	INH
ASLD	CORRIMIENTO A LA IZQ. ARITM. D	INH
ASL	CORRIMIENTO A LA IZQ. ARITM. M	EXT, INDEX
ASRA	CORRIMIENTO A LA DER. ARITM A	INH
ASRB	CORRIMIENTO A LA DER. ARITM B	INH
ASR	CORRIMIENTO A LA DER. ARITM M	EXT, INDEX
LSRA	CORR. A LA DER. LOGICO DE A	INH
LSRB	CORR. A LA DER. LOGICO DE B	INH
LSR	CORR A LA DER. LOGICO DE M	EXT, INDEX
LSRD	CORR. A LA DER. LOGICO DE D	INH
*LSLA	CORR. A LA IZQ. LOGICO DE A	INH
*LSLB	CORR. A LA IZQ. LOGICO DE B	INH
*LSL	CORR. A LA IZQ. LOGICO DE M	EXT, INDEX
*LSLD	CORR. A LA IZQ. LOGICO DE D	INH
ROLA	ROTAR A LA IZQ. EL ACUM. A	INH
ROLB	ROTAR A LA IZQ. EL ACUM. B	INH
ROL	ROTAR A LA IZQ. UN BYTE DE MEM.	EXT, INDEX
RORA	ROTAR A LA DERECHA ACUM. A	INH
RORB	ROTAR A LA DERECHA ACUM. B	INH
ROR	ROTAR ALA DERECHA M	EXT, INDEX

Las instrucciones de corrimiento lógico a la izquierda están marcadas con \* para indicar que no hay diferencia entre las instrucciones de corrimiento a la izquierda lógico o aritmético, ya que ambos tienen el mismo código de operación y son reconocidas por el ensamblador como equivalentes. Pero se incluyen ambas para hacer los programas que las usen más fáciles de leer.

La instrucción ASL equivale a multiplicar un número por dos, y ASR equivale a dividirlo entre dos. Similarmente LSR se puede usar para dividir un número sin signo entre dos.

La instrucción LSRA recorre los bits en el acumulador una posición hacia la derecha, llenando el espacio que queda a la izquierda con un cero y colocando el bit que anteriormente estaba a la derecha en el bit C del registro de código de operación. Similarmente la instrucción LSLA recorrerá los bits del acumulador una posición a la izquierda, llenando con un cero el bit del extremo derecho, y pasando el bit anterior del extremo izquierdo en el bit C del CCR.

La instrucción RORA recorre los bits hacia la derecha, el bit que anteriormente estaba en el bit C del CCR se introduce al acumulador A por el lado izquierdo, permitiendo que el bit que sale del acumulador por el extremo derecho se cargue en C.

#### 5.2.4.- Instrucciones de control.

Este tipo de instrucciones afectan el contador del programa. Pueden ser clasificadas en seis grupos: salto incondicional, condicional simple, condicional para complementos a dos, condicional de bit, y de interrupción y subrutina.

##### a) Salto incondicional.

BRA	BIFURCA SIEMPRE A LA DIRECC DADA	REL
BRN	NUNCA BIFURCA (NOP DE 3 CICLOS)	REL
JMP	BRINCA A LA DIRECCION DADA	DIR, EXT, INDEX
NOP	NO OPERACION (RETARDO DE 2 CICLOS)	INH
CPX	SALTA LAS DOS INSTRUCC. QUE SIGUEN	INM

##### b) Salto condicional simple.

BEQ	SALTA SI ES IGUAL A CERO (Z=1?)	REL
BNE	SALTA SI NO ES IGUAL A CERO (N=0?)	REL
BMI	SALTA SI ES NEGATIVO (N=1?)	REL
BPL	SALTA SI ES POSITIVO (N=0?)	REL
BCS	SALTA SI HAY ACARREO (C=1?)	REL
BCC	SALTA SI NO HAY ACARREO (C=0?)	REL
BVS	SALTA SI HAY SOBREFLUJO (V=1?)	REL
BVC	SALTA SI NO HAY SOBREFLUJO (V=0?)	REL

##### c) Salto condicional para complementos a dos.

BGT	SALTA SI ES MAYOR QUE	REL
BGE	SALTA SI ES MAYOR O IGUAL QUE	REL
BEQ	BRINCA SI ES IGUAL A CERO	REL
BLE	BRINCA SI ES MENOR O IGUAL QUE	REL
BLT	BRINCA SI ES MENOR QUE	REL

d) Condicional para aritmética sin signo.

BHI	BRINCA SI ES MAS ALTO QUE	REL
BHS	BRINCA SI ES MAS ALTO O IGUAL (BCC)	REL
BEQ	BRINCA SI ES IGUAL A CERO	REL
BLS	BRINCA SI ES INFERIOR O IGUAL	REL
BLO	BRINCA SI ES INFERIOR QUE (BCS)	REL

e) Condicional con prueba de bits.

BRSET	BRINCA SI LOS BITS SON ALTOS EN EL BYTE DE M	REL
BRCLR	BRINCA SI LOS BITS SON BAJOS EN EL BYTE DE M	REL

c) Interrupciones y subrutinas.

BSR	BIFURCA A LA SUBRUTINA	REL
JSR	BRINCA A LA SUBRUTINA	DIR, EXT, INDEX
RTS	RETORNA DE SUBRUTINA	INH
RTI	RETORNA DE INTERUPCION	INH
STOP	DETIENE LOS RELOJES	INH
SWI	INTERRUPCION POR SOFTWARE	INH
WAI	ESPERA POR INTERRUPCION	INH

La instrucción de no-operación no realiza ninguna función, se utiliza para permitir que el procesador tenga un retardo y pueda manejar periféricos lentos. La instrucción NOP retarda dos ciclos de reloj y la instrucción BRN retarda tres ciclos.

La instrucción CPX usando direccionamiento inmediato se utiliza para saltar las siguientes dos localidades, también se le llama SKIP2. Esta instrucción adicionalmente cambia el CCR, lo cual generalmente no causa problema..

### 5.3.- Ejemplos de aplicación.

A continuación se presentan algunos ejemplos que pueden ser usados para practicar con el sistema o como parte de programas más complejos.

5.3.1.- Diseño de un semáforo, para una intersección de dos calles de doble circulación.

Un semáforo es un ejemplo muy sencillo de un secuenciador, en el cual algunas luces encienden por algunos segundos, mientras que otras permanecen apagadas. Se usará la

tarjeta de simulación de salidas ubicada en la dirección \$4000 para observar las luces del semáforo por medio de los LEDS, como se muestra en la figura 5.1.

Cada salida del registro se emplea para encender dos luces simultáneamente cuando la salida sea alta: la norte y la sur o la este y la oeste.

En la tabla se muestran los estados y la duración de cada uno de ellos, los tiempos se pueden ajustar de acuerdo a las condiciones de tráfico, la duración en el ejemplo es muy corta, sirve solamente como simulación.

El primer estado consiste en encender durante 25 seg. el rojo N-S y el verde E-W, el segundo estado dura 5 seg. donde continua encendido el rojo N-S y el prende el amarillo E-W, el tercer estado con duración de 10 seg. consiste en mantener encendido el rojo N-S y prender la flecha izquierda de las direcciones E y W, el cuarto estado es similar al segundo y dura también 5 seg., el quinto estado enciende el verde N-S y el rojo E-W durante 25 seg., el sexto estado con duración de 5 seg. es prender el amarillo N-S y el rojo E-W, el séptimo estado con duración de 10 seg. consiste en encender la flecha izquierda de la dirección N-S y el rojo E-W, el último estado es igual al sexto.

Tabla 5.1.- Secuencia de salidas y duración.

SALIDAS				CODIGO				TIEMPO EN SEG.	CICLOS ACUM. B	
N-S		E-W		HEXADECIMAL						
R	A	V	F	R	A	V	F	DE SALIDAS		
b7	b6	b5	b4	b3	b2	b1	b0			
1	0	0	0	0	0	1	0	82	25	200
1	0	0	0	0	1	0	0	84	5	40
1	0	0	0	0	0	0	1	81	10	80
1	0	0	0	0	1	0	0	84	5	40
0	0	1	0	1	0	0	0	28	25	200
0	1	0	0	1	0	0	0	48	5	40
0	0	0	1	1	0	0	0	18	10	80
0	1	0	0	1	0	0	0	48	5	40

ROJO	DIRECCION NORTE-SUR			ROJO	DIRECCION ESTE-OESTE		
	AMA	VERDE	FLECHA		AMA	VERDE	FLECHA
	RILLO		IZQ.	RILLO		IZQ.	
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0

Fig. 5.1.- Conexión de las luces del semáforo al puerto de salida.



## CODIFICACION EN LENGUAJE ENSAMBLADOR DEL SEMAFORO.

ORG \$C000 SE ESTABLECE ORIGEN EN RAM  
 LDAA #%10000010 ENCIENDE ROJO N-S Y VERDE E-W  
 STAA \$4000 LO MANDA AL PUERTO DE SALIDA  
 LDAB #200 CARGA EN B CANTIDAD DE OCTAVOS DE SEG.  
 BSR RET TRANSFIERE A LA SUBROUTINA RETARDO  
 LDAA #%10000100 ENCIENDE ROJO N-S Y AMARILLO E-W  
 STAA \$4000 LO MANDA AL PUERTO DE SALIDA  
 LDAB #40 CARGA EN B 40 OCTAVOS DE SEGUNDO  
 BSR RET  
 LDAA #%10000001 ENCIENDE ROJO N-S Y FLECHA IZQ. E-W  
 STAA \$4000 LO MANDA AL PUERTO  
 LDAB #80 RETARDO DE 10 SEG.  
 BSR RET  
 LDAA #%10000100 ENCIENDE ROJO N-S Y AMARILLO E-W  
 STAA \$4000  
 LDAB #40 RETARDO DE 5 SEG.  
 BRS RET

LDAA #%00101000 ENCIENDE VERDE N-S Y ROJO E-W  
 STAA \$4000  
 LDAB #200 RETARDO DE 25 SEG.  
 BSR RET  
 LDAA #%01001000 ENCIENDE AMARILLO N-S Y ROJO E-W  
 STAA \$4000  
 LDAA #40 RETARDO DE 5 SEG.  
 BSR RET  
 LDAA #%00011000 ENCIENDE FLECHA N-S Y ROJO E-W  
 STAA \$4000  
 LDAB #80 RETARDO DE 10 SEG.  
 BSR RET

```

LDAA #%01001000 ENCIENDE AMARILLO N-S Y ROJO E-W
STAA $4000
LDAB #40          RETARDO DE 5 SEG.
BSR RET
L2  BRA L2

RET  LDX #41667   RETARDO DE UN OCTAVO DE SEGUNDO.
L1  DEX           ESTA INSTRUCCION Y LA SIGUIENTE TARDAN
    BNE L1        6 CICLOS AL REPETIRSE 41667 VECES DAN 1/8 S
    DECB
    BNE RET
    RTS

```

5.3.2.-Una mejor alternativa para resolver el problema del semáforo consiste en usar una tabla para almacenar la salida y duración de cada uno de los estados, luego por medio de un apuntador vamos sacando cada una de las condiciones y tiempos. El listado se presenta a continuación:

ORG \$C000	INICIO DE LA RAM
SALIDA EQU \$4000	USAREMOS PUERTO DE SALIDA 0
FCB \$82	ROJO N-S VERDE E-W
FCB 200	RETARDO DE 25 SEG
FCB \$84	ROJO N-S AMARILLO E-W
FCB 40	RETARDO DE 5 SEG
FCB \$81	ROJO N-S FLECHA E-W
FCB 80	RETARDO DE 10 SEG
FCB \$84	ROJO N-S AMARILLO E-W
FCB 40	RETARDO DE 5 SEG
FCB \$28	VERDE N-S ROJO E-W
FCB 200	RETARDO DE 25 SEG
FCB \$48	AMARILLO N-S ROJO E-W
FCB 40	RETARDO 5 SEG
FCB \$18	FLECHA N-S ROJO E-W
FCB 80	RETARDO 10 SEG
FCB \$48	AMARILLO N-S ROJO E-W

## FCB 40                   RETARDO DE 5 SEG

Una vez teniendo la tabla almacenada en memoria el programa resulta muy sencillo, ya que lo único que se necesita es ir sacando las condiciones de salida y los retardos usando un apuntador.

	ORG \$C100	
L	LDY #\$C000	CARGA IY CON EL INICIO DE LA TABLA
L0	LDAA 0,Y	CARGA A CON EL PRIMER ESTADO
	STAA SALIDA	LO EXHIBE POR EL PUERTO 0
	INY	AVANZA APUNTADOR AL PRIMER RETARDO
	LDAA 0,Y	OBTIENE EL PRIMER RETARDO
	INY	
L1	LDX #41667	ESTE RETARDO DURA 1/8 DE SEG
L2	DEX	
	BNE L2	
	DECA	
	BNE L1	
	CPY #\$C000+16	VERIFICA SI YA LLEGO AL FINAL DE LA
TABLA		
	BNE L0	
	BRA L	REPITE EL CICLO

## 5.3.3.- SECUENCIADOR PARA UN MOTOR DE PASOS.

A continuación se muestra el diseño de un secuenciador para hacer girar un motor de pasos, el sistema cuenta con una entrada que se lee en el bit 7 del puerto de entrada 0; si esta entrada es baja el motor gira en sentido de las manecillas, si es alta gira en sentido contrario.

El motor tiene cuatro bobinas A, A', B, y B' las cuales se van a representar por los cuatro LEDs de menor peso de la tarjeta simuladora de salidas que se encuentra conectada al puerto de salida 0 ubicada en la dirección \$4000.

La secuencia de ocho pasos que se lista hace girar el motor 1/400 de revolución por cada paso, nótese que A y A' nunca son ambas altas simultáneamente, lo mismo sucede con B y B'. El tiempo de retardo entre pasos determinará la velocidad de rotación, si se escoge una velocidad muy alta el motor no alcanzará a responder.

La cantidad de pasos aplicada determinará la posición final del motor y el número de giros antes de detenerse.

Secuencia ocho estados para el motor de pasos:

ESTADO	SALIDAS				HEXADECIMAL
	A	A'	B	B'	
SA	1	0	1	0	\$0A
SB	1	0	0	0	\$08
SC	1	0	0	1	\$09
SD	0	0	0	1	\$01
SE	0	1	0	1	\$05
SF	0	1	0	0	\$04
SG	0	1	1	0	\$06
SH	0	0	1	0	\$02

#### LISTADO DEL PROGRAMA DEL MOTOR DE PASOS

	ORG \$C000	ORIGEN EN EL INICIO DE LA RAM
IN	EQU \$4000	DIRECCION DEL PUERTO DE ENTRADA 0
OUT	EQU \$4000	DIRECCION DEL PUERTO DE SALIDA 0
ACA	RMB 1	MEMORIA PARA GUARDAR EL ACUMULADOR
A		
DELX	RMB 2	MEMORIA TEMPORAL PARA IX
	ORG \$C100	INICIO DE LA TABLA
SA	FCB \$0A	PRIMER PASO
	FDB SB	ESTADO SIGUIENTE SI IN ES BAJA
	FDB SH	ESTADO SIGUIENTE SI IN ES ALTA
SB	FCB \$08	SEGUNDO PASO
	FDB SC	SIGUIENTE SI IN ES CERO
	FDB SA	SIG. SI IN ES ALTA
SC	FCB \$09	TERCER PASO
	FDB SD	SIG. SI IN ES BAJA
	FDB SB	SIG. SI IN ES ALTA
SD	FCB \$01	CUARTO PASO
	FDB SE	SIG SI IN ES BAJA
	FDB SC	SIG SI IN ES ALTA

SE	FCB \$05	QUINTO PASO
	FDB SF	SIG SI IN ES BAJA
	FDB SD	SIG SI IN ES ALTA
SF	FCB \$04	SEXTO ESTADO
	FDB SG	SIG SI IN ES CERO
	FDB SE	SIG SI IN ES ALTA
SG	FCB \$06	SEPTIMO ESTADO
	FDB SH	SIG SI IN ES BAJA
	FDB SF	SIG SI IN ES ALTA
SH	FCB \$02	OCTAVO ESTADO
	FDB SA	SIG SI IN ES BAJA
	FDB SG	SIG SI IN ES ALTA
	ORG \$C100	
	LDX #400	SE PARA AL COMPLETAR UNA VUELTA
LO	LDAB #08	CUENTA LA CANTIDAD DE PASOS
L1	LDY #C100	APUNTA AL INICIO DE LA TABLA
L2	LDA 0,Y	OBTIENE EL ESTADO
	STAA OUT	LO EXHIBE POR EL PUERTO 0
	INY	
	BSR WAIT	ESPERA A QUE RESPONDA EL MOTOR
	TST IN	VERIFICA LA DIRECCION DESEADA
	BMI L3	SI IN ES BAJA GIRA A LA DERECHA
	INY	SI IN ES BAJA AVANZA POR GIRO A IZQ
	INY	
L3	LDY 0,Y	EL REGISTRO Y APUNTA AL SIG ESTADO
	DECB	
	BNE L2	REGRESA A L2 SI NO HA COMPLETADO 8
PASOS		
	DEX	
	BNE L0	VERIFICA SI DIO LA VUELTA COMPLETA
L4	BRA L4	SI LA HA COMPLETADO SE DETIENE

LA SUBROUTINA DE RETARDO ESTA AJUSTADA A CINCO SEGUNDOS PARA QUE SE ALCANCE A VER EN LA TARJETA DE SIMULACION, PERO SE TENDRA QUE REDUCIR EL RETARDO SI SE EMPLEA UN MOTOR DE PASOS REAL.

WAIT	STAA ACA	GUARDA EL VALOR ACTUAL DE A
	STX DELX	GUARDA TAMBIEN IX
	LDA #40	CARGA ACUMULADOR CON 40*1/8 DE SEG
L5	LDX #41667	ESTE LAZO TARDA 1/8 DE SEGUNDO
L4	DEX	
	BNE L4	
	DECA	
	BNE L5	
	LDX DELX	RECUPERA EL VALOR INICIAL DE IX
	LDA ACA	REGRESA EL VALOR INICIAL DE A
	RTS	

5.3.4.-En el siguiente programa, la salida actual va a continuar exhibiéndose indefinidamente hasta que los sensores que se van a estar leyendo en la tarjeta de entrada coincidan con el valor almacenado en una tabla, es decir cuando se cumpla la condición de entrada.

	SALIDA	CONDICION DE ENTRADA
SA	%00111000	CAMBIA HASTA QUE LA ENTRADA SEA %10000000
SB	%11000010	%00010001
SC	%00010011	%00110011
SD	%11110000	%11001100

#### CODIFICACION EN LENGUAJE ENSAMBLADOR

	ORG \$C000	INICIO DE LA RAM
IN	EQU \$4000	DIRECCION DEL MODULO DE ENTRADA
OUT	EQU \$4000	DIRECCION DEL MODULO DE SALIDA
	ORG \$C100	
SA	FCB \$38	VALOR DE LA PRIMERA SALIDA
	FCB \$80	PRIMERA CONDICION DE SALIDA
SB	FCB \$C2	VALOR DE LA SEGUNDA SALIDA

	FCB \$11	SEGUNDA CONDICION DE SALIDA
SC	FCB \$13	VALOR DE LA TERCERA SALIDA
	FCB \$33	TERCERA CONDICION DE SALIDA
SD	FCB \$F0	VALOR DE LA CUARTA SALIDA
	FCB \$CC	CUARTA CONDICION DE SALIDA
	ORG \$C100	
L0	LDY #\$C100	APUNTA AL INICIO DE LA TABLA
	LDX #04	
L1	LDAA 0,Y	CARGA A CON EL DATO APUNTADO POR Y
	INY	
L2	STAA OUT	EXHIBE EL DATO EN LA SALIDA
	LDAB IN	VERIFICA LAS ENTRADAS
	EORB 0,Y	LA PRUEBA CON LA CONDICION
	BNE L2	SI SON DIFERENTES REPITE ESTE CICLO
	INY	SI SON IGUALES AVANZA A LA SIG SALIDA
	DEX	
	BNE L1	CUANDO TERMINA LOS CUATRO ESTADOS
	BRA L0	REPITE EL CICLO

El programa anterior se podría modificar por ejemplo para una cerradura de seguridad, que al teclear la clave correcta se abra la puerta, agregando instrucciones para que se quede bloqueado después de un determinado número de intentos, las salidas podrían estar mostrando las condiciones del sistema.

5.3.5.- El siguiente programa realiza la lectura de los ocho canales del multicanalizador conectado a la entrada E0 del convertidor A/D, y almacena las lecturas en la RAM a partir de la dirección \$C150. De una manera similar se pueden leer las otras entradas del convertidor. Esta rutina podría emplearse como parte de un programa más extenso.

	ORG \$C000
OPTION	EQU \$1039
ADCTL	EQU \$1030
ADR1	EQU \$1031
DDRD	EQU \$1009
PORTD	EQU \$1008
ACB	RMB 1

	ORG \$C100	ORIGEN DEL PROGRAMA
	LDAA #\$3F	CARGA SEIS UNOS EN A
	STAA DDRD	PARA DEFINIR LAS LINEAS COMO SALIDAS
	LDX #\$C150	INICIO DE DONDE GUARDAR LAS MUESTRAS
	LDAA #\$90	CARGA UNOS EN BIT 7 Y BIT 4
	STAA OPTION	PARA ACTIVAR EL CONVERTIDOR Y RET
	LDAA #00	ACTIVA CANAL 0 DEL MUX
C3	STAB PORTD	SELECCIONA UN CANAL DEL MUX
	LDAA #01	CARGA UNO EN BIT 0
	STAA ADCTL	PARA SELECCIONAR CANAL UNO
C1	LDAA ADCTL	MUESTREA EL ADCTL
	BITA #\$80	VERIFICA SI ESTA COMPLETA LA CONV
	BEQ C1	ESPERA HASTA QUE SE COMPLETE
	LDAA ADR1	TOMA LA MUESTRA
	STAA 0,X	LA ALMACENA EN LA DIRECC APUNTADA
	INX	
	CPX #\$C158	VERIFICA SI SON LAS OCHO MUESTRAS
	BEQ C2	SI TERMINO SALE
	STAB ACB	GUARDA B EN UN REGISTRO TEMPORAL
	LDAB #\$05	PRODUCE RETARDO DE 5*1/10 SEG
C5	LDY #28570	ESTE CICLO DURA 1/10 SEG
C4	DEY	
	BNE C4	
	DECB	
	BNE C5	
	LDAB ACB	RECUPERA EL VALOR DEL ACUM B
	ADDB #\$08	B5 B4 B3 CUENTAN DE 000 A 111
	BRA C3	
C2	BRA C2	

5.3.6.- El siguiente programa sirve para realizar algunas de las operaciones lógicas de un PLC, la primera parte del programa está compuesta de las subrutinas para lectura de entradas y exhibición de salidas, esta parte se tendrá que incluir en todos los ejercicios de esta sección.



El ejemplo emplea solo ocho entradas y ocho salidas, pero podría extenderse fácilmente a las 32 entradas y 32 salidas.

```

        ORG $C000
IN0    EQU $4000
OUT0   EQU $4000
RLO    RMB 1
TEMP1  RMB 1

        ORG $C100
I00    LDAA IN0  EL BIT 0 NO LO MUEVE
        RTS
I01    LDAA IN0  EL BIT 1 LO CAMBIA A LA POSICION 0
        LSRA
        RTS
I02    LDAA IN0  EL BIT 2 LO CAMBIA A LA POSICION 0
        LDAB #02
        BSR CORD
        RTS
I03    LDAA IN0  EL BIT 3 LO CAMBIA A LA POSICION 0
        LDAB #03
        BSR CORD
        RTS
I04    LDAA IN0  EL BIT 4 LO CAMBIA A LA POSICION 0
        LDAB #04
        BSR CORD
        RTS
I05    LDAA IN0  EL BIT 5 LO CAMBIA A LA POSICION 0
        LDAB #05
        BSR CORD
        RTS
I06    LDAA IN0  EL BIT 6 LO CAMBIA A LA POSICION 0
        LDAB #06
        BSR CORD
        RTS
I07    LDAA IN0  EL BIT 7 LO CAMBIA A LA POSICION 0

```

```

LDAB #07
BSR CORD
RTS
CORD LSRA      HACE LOS CORRIMIENTOS DADOS POR B
DECB
BNE CORD
RTS
NI00 LDAA IN0
COMA      SI LA ENTRADA ES BAJO ACTIVA LA INVIERTE
RTS
NI01 LDAA IN0
COMA      INVIERTE LA ENTRADA
LSRA      CAMBIA EL BIT 1 A LA POSICION 0
RTS
NI02 LDAA IN0
COMA      INVIERTE LA ENTRADA
LDAB #02
BSR CORD  CAMBIA EL BIT 2 A LA POSICION 0
RTS
NI03 LDAA IN0
COMA      INVIERTE LA ENTRADA
LDAB #03
BSR CORD  CAMBIA EL BIT 3 A LA POSICION 0
RTS
NI04 LDAA IN0
COMA      INVIERTE LA ENTRADA
LDAB #04
BSR CORD  CAMBIA EL BIT 4 A LA POSICION 0
RTS
NI05 LDAA IN0
COMA      INVIERTE LA ENTRADA
LDAB #05
BSR CORD  CAMBIA EL BIT 5 A LA POSICION 0
RTS
NI06 LDAA IN0
COMA      INVIERTE LA ENTRADA

```

```
LDAB #06
BSR CORD CAMBIA EL BIT 6 A LA POSICION 0
RTS
NI07 LDAA IN0
COMA INVIERTE LA ENTRADA
LDAB #07
BSR CORD CAMBIA EL BIT 7 A LA POSICION 0
RTS
O00 LDAA RLO
ANDA #$01 EXHIBE SOLO EL BIT 0
RTS
O01 LDAA RLO
LSLA REGRESA EL BIT A LA POSICION 1
ANDA #$02 EXHIBE SOLO EL BIT 1
RTS
O02 LDAA RLO
LDAB #02
BSR CORI REGRESA EL BIT A LA POSICION 2
ANDA #$04 MUESTRA SOLO EL BIT 2
RTS
O03 LDAA RLO
LDAB #03
BSR CORI REGRESA EL BIT A LA POSICION 3
ANDA #$08 MUESTRA SOLO EL BIT 3
RTS
O04 LDAA RLO
LDAB #04
BSR CORI REGRESA EL BIT A LA POSICION 4
ANDA #$10 MUESTRA SOLO EL BIT 4
RTS
O05 LDAA RLO
LDAB #05
BSR CORI REGRESA EL BIT A LA POSICION 5
ANDA #$20 MUESTRA SOLO EL BIT 5
RTS
O06 LDAA RLO
```

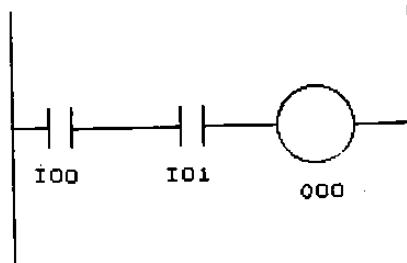
```

LDAB #06
BSR COR1  REGRESA EL BIT A LA POSICION 6
ANDA #$40 MUESTRA SOLO EL BIT 6
RTS
007 LDAA RLO
LDAB #07
BSR COR1  REGRESA EL BIT A LA POSICION 7
ANDA #$80 MUESTRA SOLO EL BIT 7
RTS
COR1 LSLA      LO RECORRE A LAS VECES QUE DIGA B
DECB
BNE COR1
RTS

```

Los siguientes ejercicios hacen uso de las subrutinas anteriores para realizar operaciones lógicas sobre las variables de entrada.

#### 5.3.6.1.- Operación AND entre dos entradas.

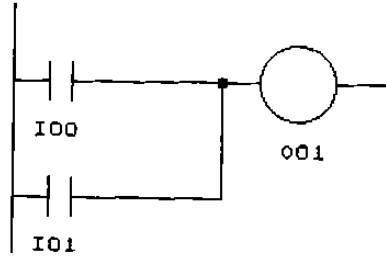


```

ORG $C200
JSR I00
STAA RLO  GUARDA LA ENTRADA EN EL BIT 0 DEL RLO
JSR I01   TOMA LA SIG ENT Y LA REGRESA EN POS 0
ANDA RLO  SE EFECTUA LA OPERACION AND
STAA RLO
JSR O00   REGRESA CON EL BIT EN LA POS CORRECTA
STAA OUT0 LO MUESTRA POR EL PUERTO

```

## 5.3.6.2.- Operación OR entre dos entradas.

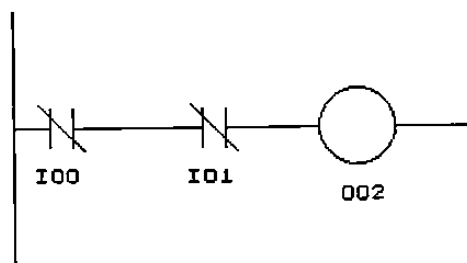


```

ORG $C250
JSR I00
STAA RLO  ALMACENA LA ENTRADA EN EL BIT 0 DEL RLO
JSR I01    REGRESA CON LA ENTRADA EN LA POS 0 DE A
ORA RLO    EFECTUA LA OPERACION OR
STAA RLO
JSR O01    REGRESA CON EL BIT EN LA POSICION 1 DE A
STAA OUT0 LO EXHIBE POR EL PUERTO 0

```

## 5.3.6.3.-Operación NOR.

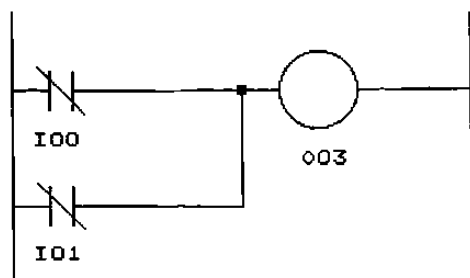


```

ORG $C300
JSR NI00  INVIERTE LA ENTRADA
STAA RLO
JSR NI01  INV LA ENT Y LA REGRESA EN LA POS 0 DE A
ANDA RLO  EFECTUA AND CON ENTR INV (NOR)
STAA RLO
JSR O02   VUELVE CON EL BIT EN LA POSICION 2
STAA OUT0 LO EXHIBE

```

## 5.3.6.4.- Operación NAND

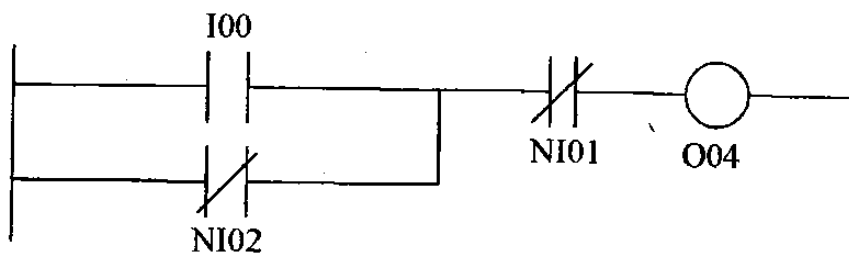


```

ORG $C350
JSR NI00  INVIERTE LA ENTRADA
STAA RLO
JSR NI01  INVIERTE Y LA PASA A LA POSICION 0
ORA RLO   REALIZA OR CON ENT INV (NAND)
STAA RLO
JSR O03   REGRESA CON EL BIT EN LA POS 3
STAA OUT0 LO EXHIIBE

```

## 5.3.6.5.- Circuito con OR y AND.

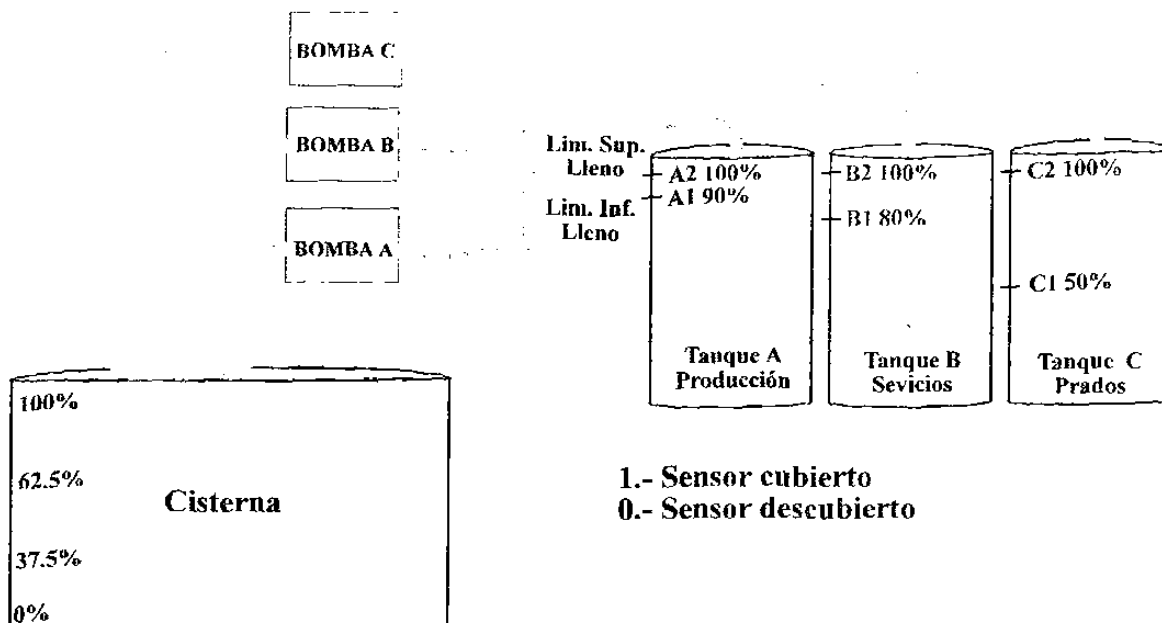


```

ORG $C400
JSR I00   VUELVE CON EL DATO EN EL BIT 0 DE A
STAA RLO
JSR NI02  INVIERTE EL DATO Y LO PASA A LA POS 0
ORA RLO   EFECTUA LA OPERACION OR
STAA RLO
JSR NI01  INVIERTE Y LO PASA A LA POS 0
ANDA RLO  EFECTUA LA AND
STAA RLO
JSR O04   REGRESA CON EL DATO EN LA POS 4
STAA OUT0 LO MUESTRA EN LA SALIDA 0 POS 4

```

## 5.3.7.- Ejemplo de llenado de tanques.



Una empresa tiene una cisterna que suministra agua a 3 tanques:

Tanque A.- Es el más prioritario ya que suministra agua al proceso de producción.

Tanque B.- Es el 2° en prioridad ya que suministra agua a los servicios.

Tanque C.- Es el menos prioritario ya que se usa para regar los prados.

Las condiciones son las siguientes:

El tanque A comienza a llenarse si su nivel está abajo de  $A_1$  (90%) y termina de llenarse si su nivel llega a  $A_2$  (100%).

El tanque B empieza a llenarse si su nivel está abajo de  $B_1$  (80%) y termina de llenarse cuando alcanza el nivel  $B_2$  (100%).

El tanque C empieza a llenarse cuando su nivel baja más de  $C_1$  (50%) y termina de llenarse cuando llega al nivel  $C_2$  (100%).

Si la cisterna tiene un nivel mayor que 62.5% se llena cualquier tinaco que esté vacío.

Si la cisterna tiene un nivel entre el 37.5% y 62.5% se empieza a llenar el tinaco A hasta el máximo, después el tinaco B si está vacío y después el tinaco C si se requiere.

Si el nivel de cisterna es menor de 37.5% pero mayor de cero solo se llena el tanque A.

Si no hay agua en la cisterna se apagan todas las bombas.

		A2	B1	C2	A1	B1	C1
--	--	----	----	----	----	----	----

**Tanque IN**  
\$4000

					A	B	C
--	--	--	--	--	---	---	---

**Bombas OUT**  
\$4000

S7	S6	S5	S4	S3	S2	S1	S0
----	----	----	----	----	----	----	----

**Cisterna IN**  
\$5000

L7	L6	L5	L4	L3	L2	L1	L0
----	----	----	----	----	----	----	----

**Cisterna OUT**  
\$5000

ORG \$C500

NIVEL1 EQU \$5000 Entrada nivel cisterna

NIVEL0 EQU \$5000 Salida nivel cisterna

TANQUE EQU \$4000 Entrada nivel tanques

BOMBAS EQU \$4000 Salida enciende bombas

L0 LDAA \$5000

STAA \$5000

CMPA #1F Si el nivel es mayor de 62.5% llena cualquier tanque

BGE L1

CMPA #07 Si el nivel está entre 37.5% y 62.5% llena los tanques usando la prioridad

BGE L2

CMPA #00 Si el nivel de cisterna es menor de 37.5% pero mayor que cero solo enciende la bomba A

BGT ENCA

JMP APAGA Si no hay agua en la cisterna apaga las bombas

L1 LDAA TANQUE

CMPA #3F Si están llenos los tanques apaga las bombas

BEQ APAGA

COMA

ANDA #07 Enciende la bomba del tinaco que lo solicita

STAA BOMBAS

JMP L0



L2            LDAA TANQUE  
              CMPA #3F Si están llenos los tanques apaga las bombas  
              BEQ APAGA  
              BITA #04 Primero llena el tanque A  
              BEQ ENCA  
              BITA #02 Luego llena el tanque B  
              BEQ ENCB  
              BITA #01 Luego llena el tanque C  
              BEQ ENCC  
              JMP L0

APAGA        LDAA #00 Si los tanques están llenos no enciende ninguna bomba  
              STAA BOMBAS  
              JMP L0

ENCA         LDAA #04 Llena el tanque A  
              STAA BOMBAS  
              JMP L0

ENCB         LDAA #02 Llena el tanque B  
              STAA BOMBAS  
              JMP L0

ENCC         LDAA #04 Llena el tanque C  
              STAA BOMBAS  
              JMP L0

# APENDICE

# LISTADO DE PROGRAMAS

ENSAMBLADOS  
CON  
IASM

C000		1	ORG \$C000
C000		2	RLO RMB 1
C001		3	TEMP1 RMB 1
C002		4	TEMP2 RMB 1
C003		5	TEMP3 RMB 1
C004		6	TEMP4 RMB 1
C005		7	INO EQU \$4000
C005		8	OUT0 EQU \$4000
C005		9	IN1 EQU \$5000
C005		10	OUT1 EQU \$5000
		11	
C005 B64000		12	I00 LDAA INO
C008 39		13	RTS
C009 B64000		14	I01 LDAA INO
C00C 44		15	LSRA
C00D 39		16	RTS
C00E B64000		17	I02 LDAA INO
C011 C602		18	LDAB #02
C013 8D29		19	BSR CORD
C015 39		20	RTS
C016 B64000		21	I03 LDAA INO
C019 C603		22	LDAB #03
C01B 8D21		23	BSR CORD
C01D 39		24	RTS
C01E B64000		25	I04 LDAA INO
C021 C604		26	LDAB #\$04
C023 8D19		27	BSR CORD
C025 39		28	RTS
C026 B64000		29	I05 LDAA INO
C029 C605		30	LDAB #\$05
C02B 8D11		31	BSR CORD
C02D 39		32	RTS
C02E B64000		33	I06 LDAA INO
C031 C606		34	LDAB #\$06
C033 8D09		35	BSR CORD
C035 39		36	RTS
C036 B64000		37	I07 LDAA INO
C039 C607		38	LDAB #\$07
C03B 8D01		39	BSR CORD
C03D 39		40	RTS
C03E 44		41	CORD LSRA
C03F 5A		42	DECB
C040 26FC		43	BNE CORD
C042 39		44	RTS
C043 B64000		45	NI00 LDAA INO
C046 43		46	COMA
C047 39		47	RTS
C048 B64000		48	NI01 LDAA INO
C04B 43		49	COMA
C04C 44		50	LSRA
C04D 39		51	RTS
C04E B64000		52	NI02 LDAA INO
C051 43		53	COMA
C052 C602		54	LDAB #\$02
C054 8DE8		55	BSR CORD
C056 39		56	RTS
C057 B64000		57	NI03 LDAA INO
C05A 43		58	COMA

```

PLCRAM.ASM      Assembled with IASM   02/21/1996  22:20  PAGE 2
C05B C603      59          LDAB # $03
C05D 8DDF      60          BSR CORD
C05F 39        61          RTS
C060 B64000    62   NI04     LDAA IN0
C063 43        63          COMA
C064 C604      64          LDAB # $04
C066 8DD6      65          BSR CORD
C068 39        66          RTS
C069 B64000    67   NI05     LDAA IN0
C06C 43        68          COMA
C06D C605      69          LDAB # $05
C06F 8DCD      70          BSR CORD
C071 39        71          RTS
C072 B64000    72   NI06     LDAA IN0
C075 43        73          COMA
C076 C606      74          LDAB # $06
C078 8DC4      75          BSR CORD
C07A 39        76          RTS
C07B B64000    77   NI07     LDAA IN0
C07E 43        78          COMA
C07F C607      79          LDAB # $07
C081 8DBB      80          BSR CORD
C083 39        81          RTS
C084 B6C000    82   OUT00    LDAA RLO
C087 8401      83          ANDA # $01
C089 39        84          RTS
C08A B6C000    85   OUT01    LDAA RLO
C08D 48        86          LSLA
C08E 8402      87          ANDA # $02
C090 39        88          RTS
C091 B6C000    89   OUT02    LDAA RLO
C094 C602      90          LDAB # $02
C096 8D35      91          BSR CORI
C098 8404      92          ANDA # $04
C09A 39        93          RTS
C09B B6C000    94   OUT03    LDAA RLO
C09E C603      95          LDAB # $03
COA0 8D2B      96          BSR CORI
COA2 8408      97          ANDA # $08
COA4 39        98          RTS
COA5 B6C000    99   OUT04    LDAA RLO
COA8 C604     100          LDAB # $04
COAA 8D21     101          BSR CORI
COAC 8410     102          ANDA # $10
COAE 39      103          RTS
COAF B6C000   104   OUT05    LDAA RLO
COB2 C605     105          LDAB # $05
COB4 8D17     106          BSR CORI
COB6 8420     107          ANDA # $20
COB8 39      108          RTS
COB9 B6C000   109   OUT06    LDAA RLO
COBC C606     110          LDAB # $06
COBE 8D0D     111          BSR CORI
COC0 8440     112          ANDA # $40
COC2 39      113          RTS
COC3 B6C000   114   OUT07    LDAA RLO
COC6 C607     115          LDAB # $07
COC8 8D03     116          BSR CORI

```

C0CA	8480	117		ANDA #80
C0CC	39	118		RTS
C0CD	48	119	CORI	LSLA
C0CE	5A	120		DECB
C0CF	26FC	121		BNE CORI
C0D1	39	122		RTS
		123		
C100		124		ORG \$C100
C100	FE5000	125	OB1	LDX IN1
C103	1E000114	126		BRSET 0,X,\$01,PRO1
C107	1E000213	127		BRSET 0,X,\$02,PRO2
C10B	1E000412	128		BRSET 0,X,\$04,PRO3
C10F	1E000811	129		BRSET 0,X,\$08,PRO4
C113	1E001010	130		BRSET 0,X,\$10,PRO5
C117	1E00800F	131		BRSET 0,X,\$80,PRO8
C11B	7EC200	132	PRO1	JMP PROG1
C11E	7EC250	133	PRO2	JMP PROG2
C121	7EC300	134	PRO3	JMP PROG3
C124	7EC350	135	PRO4	JMP PROG4
C127	7EC400	136	PRO5	JMP PROG5
C12A	7EC900	137	PRO8	JMP SEMAF
C12D	20D1	138		BRA OB1
		139		
C200		140		ORG \$C200
C200	7F4000	141	PROG1	CLR OUT0
C203	B65000	142	C1	LDAA \$5000
C206	B75000	143		STAA \$5000
C209	7FC000	144		CLR RLO
C20C	BDC005	145		JSR I00
C20F	B7C000	146		STAA RLO
C212	BDC009	147		JSR I01
C215	B4C000	148		ANDA RLO
C218	B7C000	149		STAA RLO
C21B	BDC084	150		JSR OUT00
C21E	B74000	151		STAA OUT0
C221	7EC100	152		JMP OB1
		153		
C250		154		ORG \$C250
C250	7F4000	155	PROG2	CLR OUT0
C253	B65000	156	C2	LDAA \$5000
C256	B75000	157		STAA \$5000
C259	7FC000	158		CLR RLO
C25C	BDC005	159		JSR I00
C25F	B7C000	160		STAA RLO
C262	BDC048	161		JSR NI01
C265	B4C000	162		ANDA RLO
C268	B7C000	163		STAA RLO
C26B	BDC08A	164		JSR OUT01
C26E	B74000	165		STAA OUT0
C271	7EC100	166		JMP OB1
		167		
C300		168		ORG \$C300
C300	7F4000	169	PROG3	CLR OUT0
C303	B65000	170	C3	LDAA \$5000
C306	B75000	171		STAA \$5000
C309	7FC000	172		CLR RLO
C30C	BDC005	173		JSR I00
C30F	B7C000	174		STAA RLO

C312	BDC009	175		JSR I01
C315	BAC000	176		ORAA RLO
C318	B7C000	177		STAA RLO
C31B	BDC091	178		JSR OUT02
C31E	B74000	179		STAA OUT0
C321	7EC100	180		JMP OB1
		181		
C350		182		ORG \$C350
C350	7F4000	183	PROG4	CLR OUT0
C353	B65000	184	C4	LDAA \$5000
C356	B75000	185		STAA \$5000
C359	7FC000	186		CLR RLO
C35C	BDC005	187		JSR I00
C35F	B7C000	188		STAA RLO
C362	BDC04E	189		JSR NI02
C365	BAC000	190		ORAA RLO
C368	B7C000	191		STAA RLO
C36B	BDC048	192		JSR NI01
C36E	B4C000	193		ANDA RLO
C371	B7C000	194		STAA RLO
C374	BDC09B	195		JSR OUT03
C377	B74000	196		STAA OUT0
C37A	7EC100	197		JMP OB1
		198		
C400		199		ORG \$C400
C400	7F4000	200	PROG5	CLR OUT0
C403	B65000	201	C5	LDAA \$5000
C406	B75000	202		STAA \$5000
C409	7FC000	203		CLR RLO
C40C	BDC02E	204		JSR I06
C40F	B7C000	205		STAA RLO
C412	BDC07B	206		JSR NI07
C415	B4C000	207		ANDA RLO
C418	B7C001	208		STAA TEMP1
C41B	BDC072	209		JSR NI06
C41E	B7C000	210		STAA RLO
C421	BDC036	211		JSR I07
C424	B4C000	212		ANDA RLO
C427	BAC001	213		ORAA TEMP1
C42A	B7C000	214		STAA RLO
C42D	BDC0C3	215		JSR OUT07
C430	B74000	216		STAA OUT0
C433	7EC100	217		JMP OB1
		218		
C850		219		ORG \$C850
C850	41	220		FCB \$41
C851	FF	221		FCB \$FF
C852	21	222		FCB \$21
C853	20	223		FCB \$20
C854	81	224		FCB \$81
C855	40	225		FCB \$40
C856	21	226		FCB \$21
C857	20	227		FCB \$20
C858	14	228		FCB \$14
C859	FF	229		FCB \$FF
C85A	12	230		FCB \$12
C85B	20	231		FCB \$20
C85C	18	232		FCB \$18

C85D	40	233		FCB \$40
C85E	12	234		FCB \$12
C85F	20	235		FCB \$20
		236		
C900		237		ORG \$C900
C900	B65000	238	SEMAF	LDAA \$5000
C903	B75000	239		STAA \$5000
C906	18CEC850	240		LDY #\$C850
C90A	18A600	241	L0	LDAA 0,Y
C90D	1808	242		INY
C90F	B74000	243		STAA OUT0
C912	18A600	244		LDAA 0,Y
C915	1808	245		INY
C917	CEA2C3	246	L1	LDX #\$A2C3
C91A	09	247	L2	DEX
C91B	26FD	248		BNE L2
C91D	4A	249		DECA
C91E	26F7	250		BNE L1
C920	188CC866	251		CPY #\$C850+16
C924	26E4	252		BNE L0
C926	7EC100	253		JMP OB1
		254		
		255		
		256		
		257		
		258		

Symbol Table

C1	C203
C2	C253
C3	C303
C4	C353
C5	C403
CORD	C03E
CORI	C0CD
I00	C005
I01	C009
I02	C00E
I03	C016
I04	C01E
I05	C026
I06	C02E
I07	C036
IN0	4000
IN1	5000
L0	C90A
L1	C917
L2	C91A
NI00	C043
NI01	C048
NI02	C04E
NI03	C057
NI04	C060
NI05	C069
NI06	C072
NI07	C07B
OB1	C100



