UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DE UN PROGRAMADOR DE MEMORIAS, BASADO EN EL MICROCONTROLADOR MC68HC11

T E S I S

QUE PARA OBTENER EL TITULO DE:
INGENIERA EN COMPUTACION
P R E S E N T A :
LILIANA JOSEFINA MUÑIZ ZAFRA

Y PARA OBTENER EL TITULO DE
I
F
A

DIRECTOR DE TESIS: ING. ROBERTO MACIAS PEREZ

MEXICO, D. F. 1996

TESIS

COMPLETA

Nuestro agradecimiento ...

... a la Universidad Nacional Autónoma de México por el legado de cultura y educación que nos brinda desde nuestro ingreso como estudiantes.

... a la Facultad de Ingeniería por darnos la preparación académica para ingresar al campo profesional.

... a los maestros que nos transmitieron sus conocimientos y nos expusieron sus experiencias en sus cátedras.

... al Ing. Evaristo Sanvicente y al Lic. Javier Sanvicente por su colaboración en la corrección de estilo.

Nuestro más sincero agradecimiento al Ing. Roberto Macías Pérez, por su gran deseo de formar profesionistas y por todo el apoyo desinteresado en la realización de este trabajo.

ÍNDICE

PRÓLOGO

El propósito de este trabajo es realizar el diseño y desarrollo de un prototipo para programación de memorias EPROM, que permita al estudiante de las carreras Ingeniero en Computación e Ingeniero en Electrónica, construir un instrumento de programación de dispositivos de memoria EPROM, que le auxilie en la construcción de otros prototipos que ha de realizar durante su formación universitaria, y posteriormente en su ejercicio profesional. Una de las premisas principales es el bajo costo del equipo y la facilidad para la adquisición de los componentes en el mercado local.

Está dividido en 6 capítulos de acuerdo al siguiente índice.

1.- Introducción.
Este capítulo define términos básicos y plantea en forma global el prototipo de programación.

2.- Memorias.
Hace una referencia histórica de los orígenes de las memorias, la necesidad a que obedecen, clasificación y proceso de manufactura.

3.- Microcontrolador MC68HC11 de Motorola.
Se define el concepto del microcontrolador a partir del microprocesador y se resumen las funciones particulares del MC68HC11.

4.- Diseño del "*hardware*".
Plantea todo el diseño del circuito del programador, a partir de dispositivos TTL.

5.- Programa de aplicación, "*software*".
Explica el manejo, funcionamiento y diseño del programa bajo el cual funciona el instrumento.

6.- Manual de operación.
Explica el manejo del programador y del *software*, para usuarios.

CAPÍTULO I

INTRODUCCIÓN

Un "*chip* de memoria" es un circuito integrado realizado con semiconductores con capacidad para almacenar información o datos en forma binaria, y proporcionarla cuando le sea solicitada.

Las computadoras y otros equipos electrónicos necesitan almacenar datos o tomar códigos de operación cuando están funcionando, es decir, necesitan "recordar" eventos pasados. Haciendo una analogía con una de las principales características del ser humano, se ha llamado "memoria" a los dispositivos semiconductores que hacen posible que un equipo inanimado pueda "recordar".

Una "celda de memoria" almacena un solo dígito binario (bit) y está constituida por un transistor en las RAM dinámicas y alrededor de cinco en las estáticas.

Un programador de memorias es un instrumento capaz de grabar instrucciones y/o datos en dispositivos de memoria no volátiles. Inicialmente los bits de una EPROM son todos 1's lógicos, y el programador se emplea para convertirlos en 0's lógicos, de acuerdo a un patrón de bits particular. Los bits en este tipo de memoria pueden "borrarse" (convertirse otra vez en 1's) exponiendo la ventanilla superior del dispositivo a la luz ultravioleta durante un tiempo que depende de la intensidad de ésta.

El programador propuesto está basado en el microcontrolador Motorola MC68HC11A1 (cuyas características se definen en el capítulo III) y posee la versatilidad de poder conectarse a cualquier computadora personal a la que el estudiante tenga acceso en algún laboratorio.

El dispositivo que se va a programar se coloca sobre una base especial localizada en la parte superior del programador siguiendo la orientación convencional (ranura o marca indicadora).

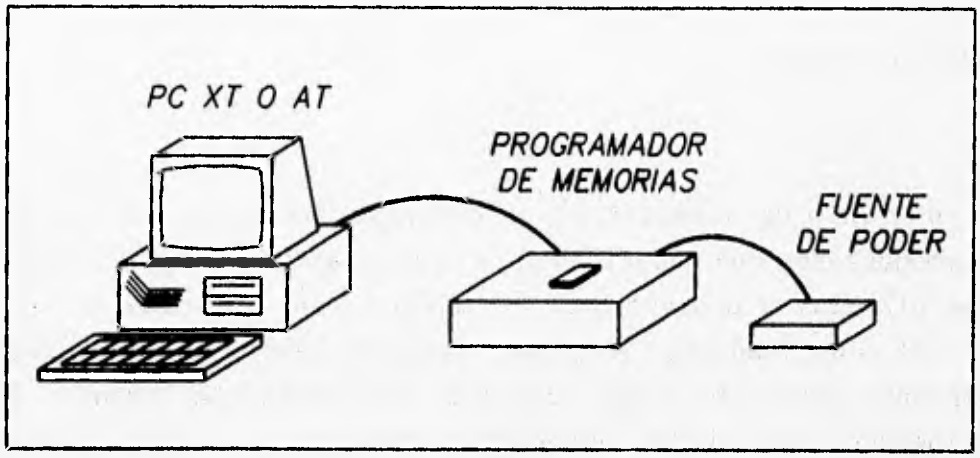


FIG. 1.1 PROGRAMADOR DE EPROMs

Un puerto de comunicación tipo serie realiza la transferencia de los datos, ya sea los que se van a programar en el dispositivo, o de la lectura de los que éste ya contiene, mediante un cable de interconexión, hacia la computadora personal, donde se ejecuta el programa de aplicación (figura 1.1). Este programa permitirá al usuario editar el contenido de las memorias antes de grabarlas y aprovechar las facilidades que brinda una computadora personal en cuanto al almacenamiento de la información y el intercambio de ésta por medios fuera de línea.

Los dispositivos de memoria EPROM que pueden programarse son los modelos 27C16, 27C32, 27C64 y 27C128 de la marca INTEL; los cuales se eligieron por su popularidad en el mercado y su versatilidad para el desarrollo de prototipos.

CAPÍTULO II

MEMORIAS

2.1 ANTECEDENTES DE LAS MEMORIAS Y SU DESARROLLO

Hace 20 años (alrededor de 1975), las memorias del tipo MOS LSI (*Metal Oxide Semiconductor Large Scale of Integration*) eran algo más que curiosidades de laboratorio. Cualquier ingeniero suficientemente hábil para diseñar equipos dotados de memorias de semiconductores tenía pocas opciones para elegir el tipo de memoria que podía usar. Se disponía solamente de dos tipos de dispositivos: la memoria RAM estática 2102 de fácil uso o la RAM dinámica 1103 para las aplicaciones de bajo consumo de energía. Desde entonces, el mercado de las memorias ha recorrido un largo camino, los tipos de dispositivos de memoria han proliferado y ahora hay disponibles más de 3,000 modelos diferentes. Consecuentemente el diseñador tiene mucho de donde escoger, pero la elección es más difícil, y debe basarse en las características particulares de cada aplicación.

Los dispositivos de memoria pueden dividirse en dos categorías principales: volátiles y no volátiles. Las memorias volátiles retienen su información solamente mientras la energía está aplicada y en muchos casos esta limitación no presenta problemas. El término genérico "*Random Access Memory*" (RAM) ha venido a ser sinónimo de memoria volátil, en la cual es constante la reescritura o el almacenamiento de datos.

En otras situaciones sin embargo, es imperativo que se usen dispositivos de memoria no volátil (ROM), porque retienen la información aunque no estén energizados. Un ejemplo de este requerimiento podría ser mantener la información durante una falla de energía eléctrica.

2.2 MEMORIAS VOLÁTILES (RAMs)

La memoria volátil (RAM) permite el acceso a los datos almacenados (lectura), y la habilidad de alterarlos (escritura), por lo cual también se conoce como memoria de lectura/escritura.

Debido a que las RAMs pierden la información cuando se desenergizan, se deben dejar encendidos los sistemas permanentemente, agregar respaldo de batería o almacenar los datos importantes en un medio no volátil antes de desconectar la energía. A pesar de su volatilidad, la memoria RAM ha sido muy popular, naciendo una nueva industria que principalmente alimenta a los sistemas de cómputo que requieren una gran de capacidad de memoria y tiempos de acceso muy rápidos.

Se han desarrollado dos tipos básicos de RAM desde 1970; las RAMs dinámicas están indicadas para alta capacidad, velocidad moderada y bajo consumo de energía. Sus celdas de memoria son básicamente capacitores de almacenamiento de carga con un transistor "*driver*". La presencia o ausencia de carga en el capacitor es interpretada por la RAM como 1 ó 0 lógico. Debido a la tendencia natural de la carga a distribuirse por si misma en una configuración de estado de baja energía, la RAM dinámica requiere períodos de recarga para mantener almacenada la información. Tradicionalmente, este requerimiento ha significado que los diseñadores de sistemas tengan que construir circuitos adicionales para el subsistema de recarga de la RAM dinámica. En algunos tiempos la circuitería de recarga de memoria dificultaba las operaciones de lectura o escritura. Sin embargo, ahora existen alternativas para desplazar esta desventaja. Para memorias relativamente pequeñas en ambiente de microprocesador, la "RAM integrada" o iRAM provee la compleja circuitería de recarga en un *chip*, lo cual simplifica grandemente el diseño. Para requerimientos de almacenamientos mayores, los controladores LSI de memoria dinámica reducen los requerimientos de recarga a un diseño mínimo.

El segundo tipo de RAMs, la RAM estática, está orientada para los usuarios que no les importa el espacio ni el costo, a cambio de una mayor velocidad y simplicidad del circuito.

En contraparte de la memoria dinámica, la RAM estática almacena 1's y 0's usando la tradicional configuración de *flip-flop* de compuerta lógica. Estos son más rápidos y no requieren ser recargados. El usuario simplemente direcciona la RAM estática y después de un breve tiempo obtiene el bit almacenado en esa localidad.

El diseño con RAMs estáticas es más sencillo que con las RAMs dinámicas, pero estas últimas tienen mayor capacidad de celdas de bits por milésima de pulgada cuadrada de silicio. Hay una forma de obtener la simplicidad de diseño de la RAM estática con la alta capacidad de la RAM dinámica y otras ventajas: la RAM integrada o iRAM, combina en un solo substrato, una RAM dinámica, su control y su circuito de recarga, creando un *chip* que tiene las características de densidad de la RAM dinámica, pero que parece como una RAM estática ante el usuario; simplemente se direcciona y se proporciona o colecta el dato sin preocuparse de la recarga.

Antes de la aparición de las iRAMs los usuarios que requerían bloques de memoria menores de 8 KB generalmente usaban RAM estática, porque el alto precio del dispositivo era desplazado por la simplicidad del circuito de soporte. Por otro lado, los usuarios con necesidades de bloques de memoria mayores que 64 KB, usualmente optaban por RAMs dinámicas, debido a la densidad, menor precio y consideraciones de potencia que toman ventaja sobre la complejidad del circuito.

Para las aplicaciones entre estos dos límites, las decisiones dependen de muchos factores, pero las memorias iRAM cumplen las necesidades de esta área media.

2.3 MEMORIAS NO VOLÁTILES (ROMs).

La "*Read Only Memory*" (ROM), es similar a la RAM en cuanto al direccionamiento por la computadora para recuperar el dato almacenado en una localidad. La diferencia es que no incluye mecanismos para alterar su información, de ahí el término de "solo lectura".

Las ROMs son usadas básicamente para almacenar información que no está sujeta a cambio, o al menos no frecuentemente. A diferencia de la RAM, la ROM retiene su contenido cuando la energía del sistema se apaga.

Los dispositivos ROM se hicieron muy populares con el advenimiento de los microprocesadores. La mayoría de las aplicaciones de los primeros microprocesadores fueron sistemas dedicados; el programa del sistema era fijo y almacenado en ROM. Los datos manipulados podían variar y entonces eran almacenados en RAM. Esta división de aplicaciones dio origen a que a la ROM se le haya llamado almacenamiento de programa y a la RAM, almacenamiento de datos.

Las primeras ROM's contenían arreglos de celdas en las cuales la secuencia de 1's y 0's estaba establecida mediante un paso metalizado de interconexión entre máscaras durante su fabricación. Esto es, el usuario tenía que suministrar el mapa de bits para que el fabricante pudiera completar la mascarilla y construir la ROM. Los cambios para ajustes eran muy costosos, de hecho prohibitivos, a menos que el usuario planeara grandes volúmenes de producción de la misma ROM.

Para desplazar estos altos costos por ajustes, los fabricantes desarrollaron una ROM programable por el usuario (PROM), el primer dispositivo usó eslabones-fusibles que podrían ser fundidos o quemados con un sistema programador especial.

Una vez quemada, una PROM era exactamente igual que una ROM. Si el programa quemado era erróneo, el *chip* tenía que ser desechado. Las PROMs proporcionaron una solución de menor costo para memoria de programas o *FIRMWARE* para propósitos de más bajo volumen de producción que las ROM.

Como una alternativa a la programación de eslabón-fusible, la empresa INTEL desarrolló una PROM borrable de tecnología MOS (llamada EPROM), que se usó para cargar o almacenar programación. Se presentó en un Paquete estándar de CERámica con conexiones Duales en (*In*) línea (CERDIP), con una ventana que permitía la exposición del dispositivo a la luz. Cuando el chip era expuesto a

la luz ultravioleta, los protones de alta energía hacían colisión con los electrones y se esparcían en forma aleatoria, borrando de esta forma el dispositivo de memoria.

La EPROM obviamente no estaba dirigida para usarse en aplicaciones de lectura/escritura, pero proporciona gran utilidad en la investigación y desarrollo de prototipos, donde la necesidad de alterar los programas es muy común. Realmente, el mercado de la EPROM está integrado casi exclusivamente por laboratorios de desarrollo.

El proceso de fabricación evolucionó y los volúmenes se incrementaron, los precios de las EPROM bajaron haciéndolas más atractivas aún para las aplicaciones de producción de sistemas de mediano volumen.

Otro avance tecnológico de la ROM ocurrió en 1980 con la introducción de la ROM de 16 KB (2816 de INTEL), que era una ROM programable por el usuario y borrable eléctricamente. Esto es, en vez de removerla del sistema de aplicación y colocarla bajo luz ultravioleta para borrar su programa, la 2816 puede ser reprogramada en su *socket*. Aún más, se pueden borrar bits solos o bytes enteros en una sola operación en vez de borrar el *chip* completo.

Tales EEPROM (*Electrically Erasable Programable* ROM) están abriendo nuevas aplicaciones. En terminales punto-de-venta, por ejemplo, cada terminal está conectada a una computadora central pero también maneja una cantidad de proceso local. Una EEPROM puede almacenar información de descuentos para ser considerados automáticamente durante las transacciones de venta. La computadora central podría actualizar en cada terminal el cargo de descuento, vía línea telefónica mediante la reprogramación de la EEPROM.

En instrumentación digital, un instrumento puede ser autocalibrado usando una EEPROM. Si la calibración del instrumento se impulsara fuera de los límites especificados, el sistema podría emplear un diagnóstico interno para reprogramar un ajuste de parámetros en una EEPROM y regresar la calibración dentro de los límites.

La EEPROM contiene una estructura de celda de túnel de óxido de compuerta flotante (*floating-gate tunnel-oxide Flotox*) basado en un túnel de electrones a través de una capa delgada (de menos de 200 Å, 1 Angström = 1 X 10^{-10}m) de dióxido de silicio, estas celdas permiten escribir y borrar con pulsos de 21 V. y durante la operación de lectura el *chip* usa solo los 5 V. convencionales.

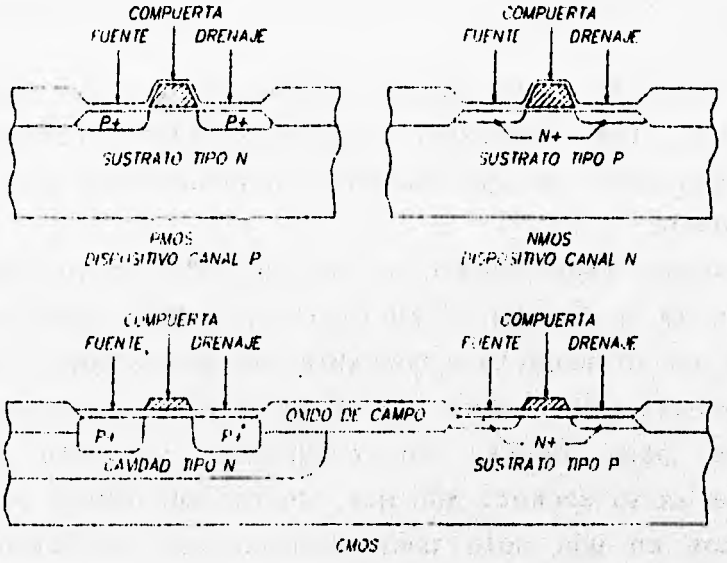


FIG. 2.1 SECCION DE CORTE DEL PROCESO MOS

2.4 TECNOLOGÍAS DE MANUFACTURA DE MEMORIAS

En cuanto a la manufactura, hay 3 familias de tecnología principales: PMOS, NMOS y CMOS (figura 2.1) y se refieren al tipo de canal de los transistores MOS que integran las memorias.

La tecnología PMOS (*Positive MOS*) implementa transistores de canal P mediante difusión de impurezas tipo p (usualmente boro) en un sustrato de silicio tipo n para formar la fuente (*source*) y el drenaje (*drain*). El canal p es así llamado debido a que está compuesto de portadores cargados positivamente.

La tecnología NMOS (*Negative* MOS) es similar, pero utiliza impurezas tipo n (normalmente fósforo o arsénico) para hacer transistores de canal n en el sustrato de silicio tipo p. El canal n es así llamado debido a que está compuesto de portadores cargados negativamente.

La tecnología CMOS (*Complementary* MOS) combina ambos tipos en el mismo silicio: el canal p y el canal n. Se puede usar cualquiera de los sustratos de silicio; sin embargo, se debe definir en el sustrato un área profunda de tipo opuesto para permitir la fabricación del tipo complementario.

La mayoría de los primeros dispositivos de memoria, como la RAM dinámica 1103 INTEL y la EPROM 1702, fueron hechos con tecnología PMOS. Como hubo necesidad de más altas velocidades y densidades más grandes, la mayoría de los nuevos dispositivos fueron fabricados con NMOS. Esto es debido a la mayor velocidad inherente de los portadores de carga del canal n (electrones) en el silicio, aunado con procesos mejorados. Hace poco tiempo, la mayoría de los dispositivos de memoria MOS que se producían eran fabricados con tecnologías NMOS. Hoy, la tecnología CMOS ha empezado a tener un uso comercial difundido, porque permite que los dispositivos de bajo requerimiento de energía puedan ser usados en diseños operados o respaldados por batería. Históricamente, los dispositivos de tecnología CMOS han sido más lentos que cualquier dispositivo NMOS, sin embargo, recientemente la tecnología CMOS ha venido mejorando para producir dispositivos de más alta velocidad. Hasta ahora el costo extra requerido para fabricar ambos tipos de transistores ha limitado a las memorias CMOS a aquellas áreas donde las características especiales de la tecnología justificaran el costo extra.

El proceso de fabricación de un circuito integrado MOS comienza con una oblea de simple cristal de silicio de alto grado de pureza, sometida a un proceso de alisado y abrillantado, con estructura de red geométrica, uniforme y simétrica. La dimensión de cada oblea depende de la cantidad de *chips* (1.27 x 1.27 mm. aprox.) que se desean producir, típicamente de 10 a 12.5 cm. de diámetro,

y aproximadamente 0.5 mm de espesor uniformemente alterada tipo p. Para formar una delgada capa de dióxido de silicio (SiO_2) en la superficie, la oblea se oxida en un horno a aproximadamente 1,000 °C y 10 bares de presión. Entonces se deposita nitruro de silicio en la oblea oxidada dentro de un reactor químico para fase gaseosa. La oblea queda lista para recibir su primer patrón de lo que será un complejo circuito de varias capas. Este patrón es grabado en el nitruro de silicio utilizando un proceso conocido como litografiado, que es similar al utilizado para fabricar los circuitos bipolares convencionales. El primer patrón (figura 2.2) define los límites de las regiones activas de los circuitos integrados, donde se harán los transistores, capacitores, resistores por difusión y las interconexiones de primer nivel.

FIG. 2.2 PRIMERA MASCARILLA

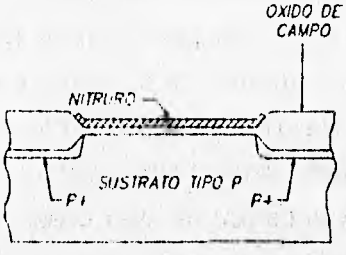


FIG. 2.3 POST-OXIDACION DE CAMPO

A la oblea grabada y modelada se le implantan átomos de boro adicionales, acelerados a alta energía. El boro solamente alcanzará el sustrato de silicio donde el nitruro y el óxido no han sido grabados, proporcionando áreas alteradas fuertemente tipo p (p+), que eléctricamente separarán las áreas activas. Después del implante, la oblea se oxida otra vez, y ésta vez se forma una espesa capa de óxido. El óxido solamente se forma en las áreas grabadas debido a las propiedades antioxidantes del nitruro de silicio. Cuando la capa de óxido ha crecido, algo del sustrato de

silicio se ha consumido y esto le da aislamiento, tanto físico como eléctrico, para los dispositivos adyacentes, como se puede ver en la figura 2.3.

Cumplido este propósito, se remueve la capa remanente de nitruro de silicio. Enseguida se aplica una ligera capa de óxido, levantando o realzando la primera capa de óxido, pero dejando intacto el espesor del óxido de campo.

En este momento las áreas para los transistores activos han sido definidas y aisladas y se puede determinar el tipo de transistores necesarios. La oblea es grabada otra vez, y si se requieren características especiales (como operación en modo agotamiento) se implanta con átomos de impurezas. La energía y dosis de los átomos de impurezas que son implantados, determinan muchas de las características de los transistores. El tipo de las impurezas proporciona el tipo de operación: modo agotamiento (tipo n) o modo enriquecimiento (tipo p).

Los tipos de los transistores ya están definidos, el óxido de la compuerta se incrementa en un horno de alta temperatura. Se debe tener especial cuidado para prevenir contaminación o inclusión de defectos en el óxido y asegurar un espesor uniforme y consistente. Esto es importante para proporcionar características precisas y confiables al dispositivo. La capa de óxido de la compuerta se enmascara y se hacen agujeros para proporcionar un contacto directo desde la compuerta hasta donde se necesite.

La oblea es depositada con una capa de material de compuerta. Generalmente es silicio poli-cristalino, el cual es depositado en un reactor químico para fase gaseosa, similar al utilizado con el nitruro de silicio. El silicio poli-cristalino se altera (usualmente con fósforo) para dar una resistencia laminar abajo de 10 a 20 ohms/milímetro cuadrado. Esta capa también se usa para interconexiones del circuito y si se requiere una resistencia menor, en lugar de este material se puede utilizar un compuesto de poli-silicio/metal refractario o siliciuro de metal refractario. La capa de compuerta se modela para definir las compuertas de los transistores y sus trayectorias de interconexión (figura 2.4).

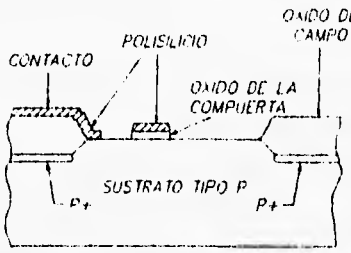


FIG. 2.4 POST-MASCARILLA DE COMPUERTA

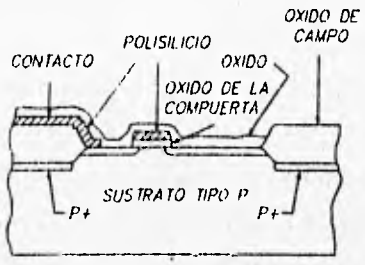


FIG. 2.5 POST-OXIDACION

Enseguida se difunde la oblea con impurezas tipo n (usualmente arsénico o fósforo) para formar las junturas de fuente y drenaje. El material de la compuerta del transistor actúa como barrera para las impurezas, proporcionando un canal no difundido y auto-alineado entre las dos junturas. Se oxida la oblea para sellar las junturas de la contaminación, con una capa de dióxido de silicio SiO_2 (figura 2.5).

Una capa gruesa de cristal se deposita sobre la oblea para proporcionar aislamiento y baja capacitancia entre las capas bajas y las interconexiones de metal. La capa de cristal es modelada con agujeros para los contactos y colocada en un horno de alta temperatura. Este paso de horneado suaviza la superficie y redondea los extremos de los contactos para proporcionar una cobertura uniforme al metal. El metal (generalmente se emplea aluminio o aluminio/silicio) se deposita sobre la oblea con los patrones de interconexión y se definen y graban las almohadillas de conexión externa (figura 2.6). Se hace entonces una aleación de la oblea en un horno a 500 °C para asegurar un buen contacto óhmico entre el aluminio y la difusión o poli-cristalino.

En este punto, el circuito es totalmente funcional, sin embargo, la capa superior de metal es muy suave y se daña fácilmente en el manejo. El dispositivo es también susceptible a la

contaminación o al ataque de la humedad. Para prevenir esto, se sella la oblea con una capa pasiva de nitruro de silicio o un compuesto de óxido de fósforo y silicio. El patrón queda terminado para colocar la ventana superior, sobre las almohadillas de contacto donde se harán las conexiones externas.

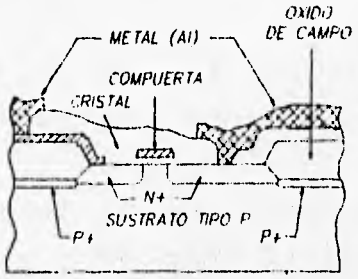


FIG. 2.6 CIRCUITO COMPLETO (SIN CAPA PASIVA).

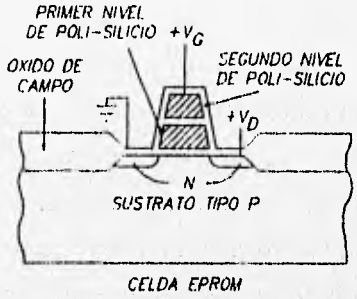


FIG. 2.7 ESTRUCTURA DE DOBLE POLI-SILICIO.

Con esto se finaliza la secuencia de un proceso de simple capa de silicio poli-cristalino. El flujo general del proceso de doble poli-cristalino, que se emplea en las DRAMs de alta densidad, es similar al seguido en las EPROMs y EEPROMs, con la adición en la compuerta de estas últimas, de procesos de depositación de silicio poli-cristalino, impurezas eléctricas y dieléctrico inter-capas, requeridos para la capa adicional de silicio poli-cristalino (figura 2.7). Estos pasos son realizados justo después de que las áreas activas han sido definidas (figura 2.2), proporcionando el capacitor o el nodo de almacenamiento de compuerta flotante, en esos dispositivos.

Después que la fabricación se ha completado, la oblea es enviada a pruebas. Cada circuito es probado individualmente bajo condiciones de diseño, para determinar cuales circuitos funcionarán apropiadamente a bajas temperaturas y en condiciones de operación

reales. Los circuitos que fallan en las pruebas son marcados para distinguirlos. De aquí, las obleas son enviadas a ensamble, donde serán cortadas en circuitos individuales (o dados) con una sierra de diamante del espesor de una hoja de papel. Los dados marcados son separados y los buenos serán enviados para empaque.

Hay dos tipos de empaque, el hermético y el no hermético. El empaque hermético (CERDIP), consiste en dos mitades de cerámica que son selladas con vidrio fundido o cerámica. Los empaques no herméticos son moldeados en plástico.

El empaque cerámico tiene dos partes: la base, en la cual están las terminales (*pins*) de conexión externas y la cavidad para el dado; y la cubierta. La base se coloca sobre un bloque calefactor y sobre ella, una estructura con las terminales; enseguida, se coloca el dado en la cavidad. Todos estos elementos se adhieren a la base mediante el vidrio fundido. Se conectan las terminales con el dado mediante alambres de oro. Finalmente se coloca la cubierta o tapa sobre la base, y el paquete completo se introduce en un horno, en donde se funde el vidrio para pegar y sellar las dos mitades del paquete.

En el empaque de plástico, el dado se adhiere a una plataforma que previamente posee una estructura de terminales, las cuales se conectan mediante alambres de oro. La estructura completa pasa a una máquina de inyección y el empaque de plástico se forma alrededor de la estructura. Al salir, se recortan los excesos de plástico (rebaba).

Después del ensamble final, los paquetes son probados en altas temperaturas para asegurar parámetros de operación críticos, y se separan por grupos de acuerdo a la velocidad y consumo de energía.

Aunque el proceso de fabricación de un dispositivo VLSI suene fácil, actualmente son varios cientos de operaciones que se tienen que realizar correctamente para hacer un circuito integrado. Generalmente, toma alrededor de dos meses completar la manufactura, las pruebas y mediciones que hay alrededor de un circuito integrado.

2.5 PROGRAMACIÓN DE UNA MEMORIA EPROM

Para programar la información en una celda de memoria EPROM, se aplica un alto voltaje entre el drenaje y la compuerta, con lo cual se aumenta la energía de los electrones en el área del canal, y aparecen "electrones libres" capaces de brincar a través de la película de óxido. Empujados por el alto voltaje en la compuerta, los electrones libres son admitidos en la compuerta flotante y la nueva carga cambia el voltaje de disparo en el elemento de memoria, almacenando de esta forma, nueva información.

Cuando se realiza la lectura de la celda, se aplica voltaje en la compuerta y en el drenaje con respecto a la fuente; los 1's y 0's lógicos se identifican mediante la revisión de presencia o ausencia de flujo de corriente. Puesto que el voltaje de drenaje para la lectura es 3 volts, no se da lugar a una escritura errónea durante un ciclo de lectura.

Cuando salen a la venta, todos los bits EPROM permanecen en 1 lógico, con toda la carga liberada (sin datos programados). Los datos se programan cambiando de 1's a 0's mediante la aplicación específica de una forma de onda y un voltaje. Tanto más alto sea el voltaje Vpp y más ancho el pulso de programación tpW, se incrementa la cantidad de electrones libres "atrapados" en la compuerta flotante.

Si Vpp excede el valor del rango permisible, la unión p-n de la celda de memoria puede ceder, provocando un rompimiento permanente. Para prevenir esto, se debe verificar el valor del disparo de Vpp del programador y el ruido inducido de voltaje negativo en las otras terminales, el cual puede crear un efecto de transistor parásito y reducir la cantidad de electrones libres donados.

2.6 BORRADO DE UNA MEMORIA EPROM

Las memorias EPROM generalmente se pueden escribir y borrar más de 100 veces. Los datos almacenados son borrados mediante la exposición del *chip* a la luz ultravioleta, la cual libera la carga de la compuerta flotante. Los electrones en la compuerta flotante

reciben la energía ultravioleta, se convierten en electrones libres otra vez y brincan a la compuerta de control o sustrato. Este proceso borra los datos almacenados.

La longitud de onda y la exposición mínima de luz ultravioleta se especifica como 2,537 Å. y 15 W seg/cm² respectivamente. El borrado ocurre mediante la exposición del dispositivo a una lámpara ultravioleta de 12,000 uW/cm² a una distancia de 1.2 a 3 cm durante aproximadamente 20 minutos. La relación de transmisión de luz ultravioleta de la ventanilla transparente hacia el interior del *chip*, es aproximadamente 70%. La contaminación o materiales extraños en la superficie de vidrio, disminuyen la transmisión de luz y prolongan el tiempo de borrado, por lo que deben ser eliminados mediante el uso de alcohol u otro solvente que no dañe el empaque.

2.7 RETENCIÓN DE DATOS

Como resultado de la escritura, se acumulan en la compuerta flotante, de 0.5 a 2.0 x $(10)^{-13}$ coulombs de electrones. Con el paso del tiempo, la cantidad de electrones decrece, lo cual provoca una conversión de 0's a 1's. Los mecanismos de disipación de electrones se explican como sigue:

1. Disipación por calor
   La acumulación de electrones en la compuerta flotante es un estado no balanceado, así que la disipación de los electrones térmicamente excitados no puede evitarse. El tiempo de permanencia de los datos está cercanamente relacionado con la temperatura (10^4 horas a 100°C, 10^5 horas para 200°C).

2. Disipación mediante luz ultravioleta
   Los rayos ultravioleta con una longitud de onda entre 3,000 y 4,000 Å, liberan la carga almacenada en las compuertas flotantes de la EPROM. La luz fluorescente y la luz de sol contienen algunos rayos ultravioleta; la

exposición prolongada puede causar la corrupción de los datos en un lapso de tiempo de aproximadamente 3 años y una semana respectivamente.

3. Disipación mediante voltaje
   Este tipo de disipación ocurre mientras se escribe la información. El alto voltaje en otras celdas de memoria colocadas en la misma palabra o línea de datos de la celda a ser programada, causa disipación de la carga eléctrica almacenada. Tales defectos son eliminados mediante inspección en la fábrica. El voltaje de programación y el ancho del pulso deben permanecer siempre dentro de los límites especificados.

2.8 PROGRAMADOR DE EPROMs

Un programador de EPROMs almacena los datos en su RAM interna y los transfiere hacia la EPROM, para completar esto, son necesarias tres funciones: Función de revisión de borrado, función de programación y función de verificación. Los programadores también están provistos con una función de verificación de inserción invertida del *chip* y/o una función de verificación de contacto de terminales; anteriores al chequeo de borrado. Estos procedimientos se describen de la siguiente forma:

1. Revisión de inserción invertida
   Esta verificación detecta la inserción invertida del dispositivo, coloca el equipo en modo *reset* y protege al dispositivo y al equipo.

2. Verificación del contacto de las terminales
   Esta verificación se realiza mediante la detección de corriente en sentido directo de cada terminal de la EPROM. La resistencia de entrada polarizada directamente difiere entre los fabricantes.

3. Revisión de borrado

Esta función se realiza antes de la programación para determinar si la EPROM está borrada y prevenir una reprogramación. Da lectura a todos los datos almacenados, que en estado de borrado deben ser 1's y se detiene al detectar cualquier 0 lógico. Normalmente una lámpara o alarma proporciona el aviso.

4. Introducción del programa

Existen varios métodos para introducir los datos a la RAM interna del programador: Introducción por copia, en el cual la entrada es una copia de una ROM maestra; Introducción manual, mediante un teclado en el panel frontal, usado comúnmente para la corrección o revisión de los datos; Entrada por terminal, se puede realizar la entrada, edición y el listado de los datos mediante una terminal. Este último método puede sustituirse mediante el uso de una computadora personal, aprovechando así su propia memoria RAM.

5. Programación

En el flujo normal de programación, se debe leer el byte de la EPROM antes de ser programado y compararse con el dato a programar. Si estos coinciden, la programación no tiene sentido; y si difieren, la programación se debe realizar. Se lee nuevamente el dato y se compara con el que se pretendió programar; si coinciden, se debe incrementar a la siguiente dirección, y continuar el ciclo hasta llegar a la dirección final.

6. Verificación final

Se vigila la correcta programación, comparando la lectura con los datos de la RAM interna del programador. Realiza un paro cuando los datos no coinciden y enciende una lámpara de aviso desplegando la dirección y el dato.

2.9 MANEJO DE UNA EPROM

El contacto con plásticos, ropas secas o el cuerpo humano cargado electrostáticamente, provoca que la ventanilla de vidrio de una EPROM genere electricidad estática, la cual puede causar un daño físico al dispositivo. Los daños típicos son falla en el borrado y ajustes de margen de escritura, los cuales dan la impresión de que la información ha sido correctamente escrita y en realidad puede no ser así. La electricidad estática puede ser eliminada neutralizando las cargas con una exposición corta a los rayos ultravioleta, similar al procedimiento de borrado, el cual reduce las cargas en la compuerta flotante. Para prevenir las cargas en la ventanilla, se utilizan los siguientes métodos:

1. Utilizar sistemas aterrizados para el operador que maneja las EPROMs y evitar el uso de guantes que puedan generar cargas estáticas.

2. Evitar pegar la ventanilla de vidrio con plásticos u otras sustancias que puedan desarrollar cargas estáticas.

3. Evitar el uso de congelantes en aerosol que contengan iones.

4. Usar etiquetas de protección de material conductor que pueden distribuir las cargas.

2.10 ETIQUETAS DE PROTECCIÓN

En ambientes donde puede ocurrir una exposición a la luz ultravioleta, se debe colocar una etiqueta de protección sobre la ventanilla de vidrio de la EPROM, para absorber la radiación.

Existen en el mercado etiquetas especialmente preparadas para este propósito, siendo las de tipo metálico, particularmente efectivas.

Pocas etiquetas de protección cumplen con todos los requerimientos ambientales establecidos para las EPROMs. Se debe

escoger la etiqueta conveniente para cada aplicación considerando las siguientes características:

1. Fuerza del adhesivo
   Evitar el colocamiento repetido y el polvo, los cuales pueden reducir la fuerza del adhesivo. Si las etiquetas deben ser cambiadas, se debe colocar la nueva etiqueta sobre la anterior, puesto que al retirarla se pueden generar cargas estáticas, y si fuera necesario, se recomienda entonces el borrado ultravioleta y la reprogramación.

2. Rangos de temperatura y humedad
   Usar etiquetas de protección cuyo adhesivo soporte los rangos de temperatura y humedad ambientales. Más allá del primer rango, el adhesivo se puede endurecer y permanecer en la ventanilla aún después de que la etiqueta haya sido retirada, y rebasando el segundo rango, el adhesivo puede deformarse y perder adherencia.

CAPÍTULO III

MICROCONTROLADOR MOTOROLA MC68HC11

El presente capítulo comprende un resumen de los elementos y funcionamiento del microcontrolador MC68HC11 de Motorola. De ninguna manera se pretende sustituir todo el manual de datos técnicos en un solo capítulo, pero sí exponer los fundamentos de dicho dispositivo en un modo ordenado, se procura no hacer alusión a conceptos que se definen más adelante.

3.1 DESCRIPCIÓN GENERAL

En los sistemas electrónicos en que sea necesario utilizar un microprocesador, se requiere incorporar dispositivos periféricos tales como memorias RAM y ROM, puertos de entrada/salida y de comunicación, entre los más comunes. Para los diseños sencillos, se puede utilizar un microcontrolador, que tiene todos estos dispositivos en cantidades y combinaciones variables, incorporados en un solo *chip*.

Un microcontrolador contiene adicionalmente al CPU, memoria principal, memoria *caché*, interfases de entrada/salida, controladores de acceso directo a memoria, manejadores de interrupciones, *timers* y otros subsistemas.

La familia MC68HC11 es una serie avanzada de unidades de microcontroladores (MCU) de 8 *bits*, con periféricos altamente sofisticados integrados en un solo *chip*, fabricado con tecnología HCMOS (Semiconductor de Óxido-Metal Complementario de alta densidad). Permite frecuencias de *bus* interno desde cd hasta 2 ó 3 MHz. La tecnología HCMOS empleada, combina el tamaño reducido y altas velocidades con el bajo consumo de energía y la alta inmunidad al ruido. El subsistema de memoria incluye ROM, EPROM, EEPROM y RAM estática en cantidades que varían dependiendo de cada modelo de la familia.

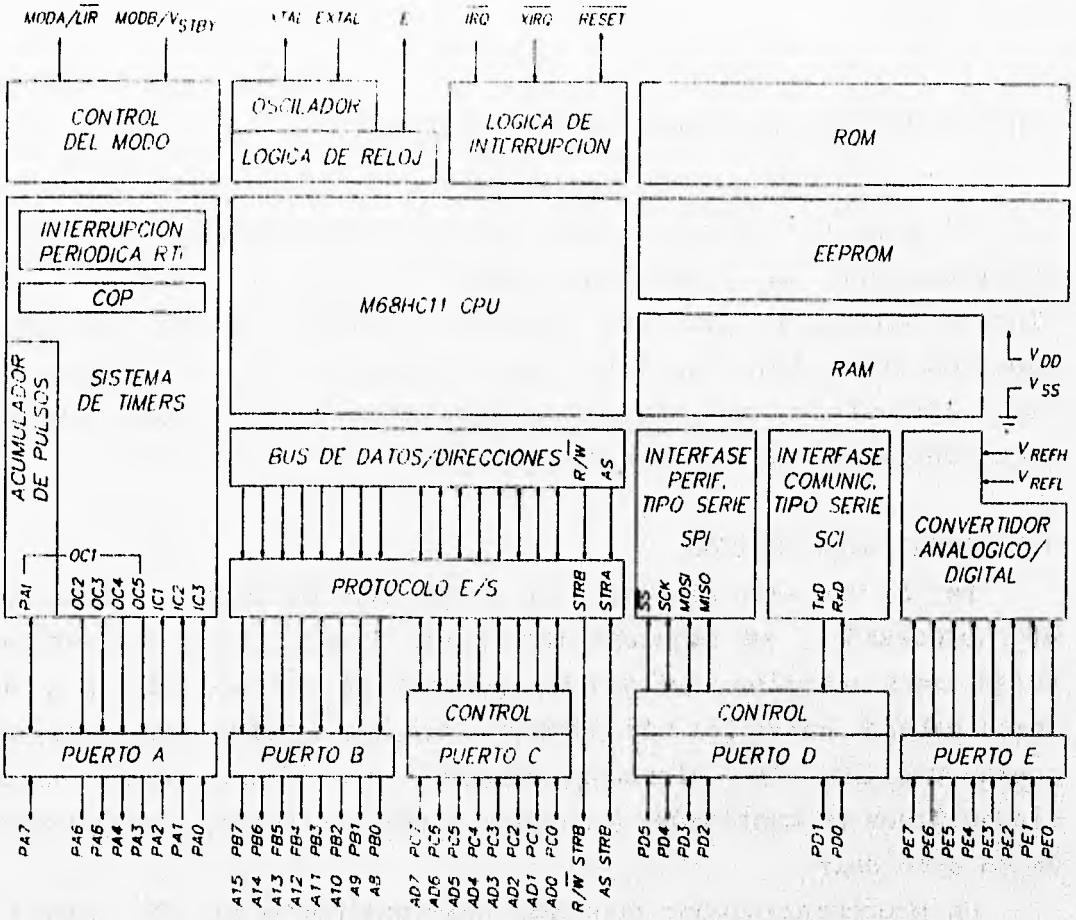


FIG. 3.1 DIAGRAMA A BLOQUES DEL MC68HC11

La figura 3.1 muestra el diagrama de bloques principales de los subsistemas que componen un MCU de esta familia y sus líneas de interacción.

La figura 3.2 muestra la disposición de terminales del MCU MC68HC11A1, con 256 *bytes* de RAM y 512 *bytes* de EEPROM, que es el modelo particular utilizado en este prototipo.

Las funciones periféricas incluidas en el *chip* son las siguientes:

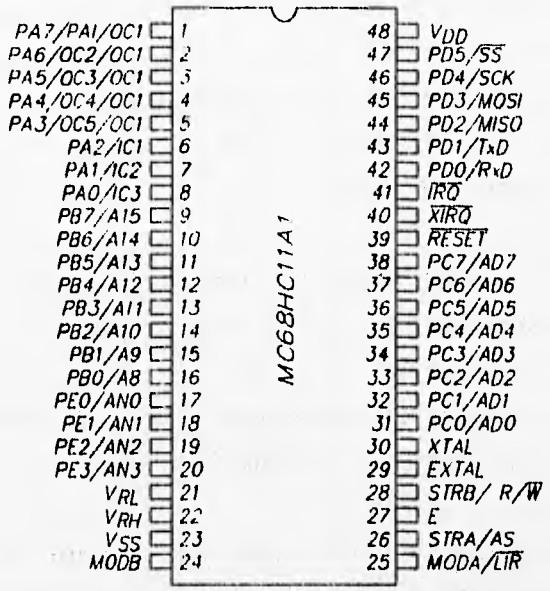


FIG 3.2 ASIGNACION DE TERMINALES DEL MC68HC11A1

- Un subsistema de entradas y salidas tipo paralelo (puertos B y C).

- Un convertidor analógico-digital de 8 canales con 8 *bits* de resolución (Puerto E).

- Una interfase de comunicación tipo serie asíncrona (SCI puerto D).

- Una interfase de comunicación tipo serie síncrona (SPI puerto D).

- Un subsistema de reloj libre de 16 *bits* con 3 líneas de captura de entradas y 5 líneas de comparación de salidas y funciones de interrupción de tiempo real (puerto A).

- Un acumulador de pulsos de 8 *bits* para contar eventos externos o medir períodos.

- Un sistema monitor contra fallas de *software* COP (*Computer*

Operating Properly).

- Un sistema monitor que genera un *reset* del sistema en caso de que el reloj se pierda o su frecuencia sea demasiado baja.
- Un circuito detector de código de operación ilegal que genera una interrupción no mascarable.
- Dos modos de ahorro de energía, *WAIT* y *STOP* que son controlados por *software*.

El MC68HC11 tiene 4 modos de operación, dos de los cuales son para funcionamiento normal y los otros dos son especiales para pruebas:

- Un solo *chip*, es el modo más simple de operar y todas sus funciones y periféricos se encuentran en el mismo *chip*.

- Modo expandido, en el cual se pierde el subsistema de entradas y salidas paralelas, a cambio de ganar un *bus* de datos y un *bus* de direcciones, para tener acceso a memorias y periféricos externos como lo haría un microprocesador. Las entradas y salidas perdidas (puertos A y B), se pueden recuperar instalando la unidad de reemplazo de puertos MC68HC24, con la cual el programa de aplicación funcionará sin alterarse.

- Modo especial de pruebas: Este modo es una variación del modo expandido y es inicialmente utilizado en pruebas internas en la planta de Motorola. Sin embargo es posible accederlo modificando el registro CONFIG ($103F_H$) (ver la sección 3.3.2).

- Modo de arranque especial: Cuando se selecciona este modo, una porción de ROM interna se habilita en las direcciones desde $BF00_H$ hasta $BFFF_H$. El contenido de esta memoria es un programa especial de comunicación, arranque y un conjunto inicial de vectores de interrupción y de *reset*. Al seleccionar este modo se genera un *reset* interno y el programa inicia en la dirección $BF00_H$. Cuando el MCU es nuevo, este modo se utiliza para cargar el programa del usuario y se requiere un transmisor tipo serie con una velocidad de 1,200 *bauds* si el MCU tiene un oscilador externo de 8 MHz (E = 2 MHz), enseguida se debe enviar un carácter con valor FF_H al puerto SCI (ver la sección 3.6.2) e inmediatamente los 512 *bytes* del programa del usuario, los cuales se alojarán desde la dirección 0000_H hasta la dirección $00FF_H$. El programa de inicio especial transfiere entonces la ejecución al nuevo programa del usuario y éste debe acceder a la memoria EEPROM interna para escribir un programa permanente no volátil.

Cualquiera de los modos de operación se selecciona cuando se activa la terminal $\overline{\text{RESET}}$, mediante las terminales MODA y MODB, y al desactivar $\overline{\text{RESET}}$ el MCU ya se encuentra en el modo de operación seleccionado. Hecho esto, se puede disponer de las terminales MODA y MODB y el modo de operación se conservará.

3.2 DESCRIPCIÓN DE TERMINALES

a) V_{DD} y V_{SS} (Terminales de alimentación):

La energía al MCU se aplica a través de estas terminales. V_{DD} son 5 volts nominales y V_{SS} se conecta a tierra. Debido a que en las demás terminales las transiciones de subida o de bajada demandan considerable cantidad de corriente durante corto tiempo, se recomienda conectar un capacitor con características de alta frecuencia, entre estas terminales, lo más cerca posible al MCU. El valor del capacitor varía dependiendo de la carga conectada a las demás terminales, pero un valor de 0.1 uF es recomendable.

b) $\overline{\text{RESET}}$:

Señal de control bidireccional, habilitada a nivel lógico bajo, actúa para iniciar al MCU en cualquiera de sus dos modos de operación normal o sus dos modos de pruebas especiales. También actúa como salida de drenaje abierto (*open drain*) para indicar que se ha detectado una falla interna en el monitor de reloj o en el circuito COP. La CPU distingue una condición de *reset* interna de una externa mediante la detección de la terminal $\overline{\text{RESET}}$, la cual cambia de 0 a 1 lógico en menos de 2 ciclos de reloj E después de que el *reset* ha ocurrido. Por lo tanto no se recomienda conectar circuitos de retraso de tiempo RC en esta terminal, debido a que se altera la constante de carga y puede ocurrir una mala interpretación del tipo de *reset*.

Cuando se habilita esta terminal, el MCU detiene su reloj y no se encuentra en ningún modo de operación, los puertos de salida regresan a 0 lógico y los bidireccionales son reprogramados como entradas. Para salir del estado de *reset* se debe aplicar un nivel lógico alto en esta terminal.

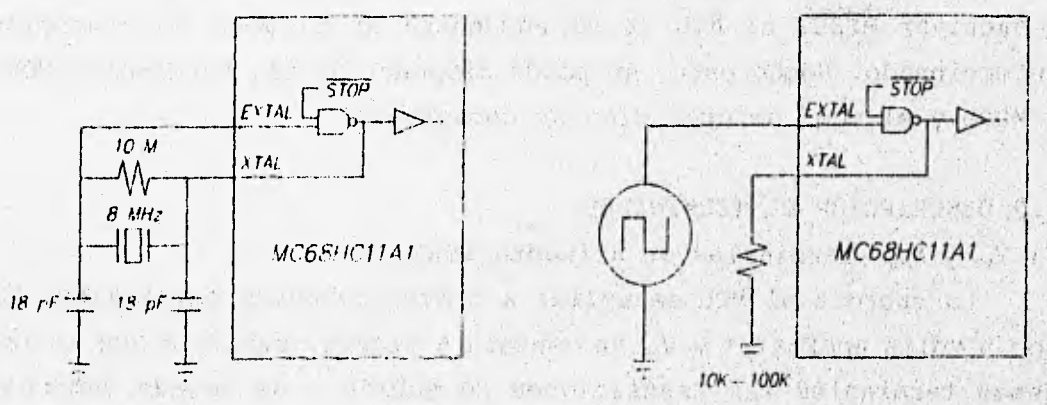


FIG. 3.3 (A) MCU CON OSCILADOR A CRISTAL Y (B) CON GENERADOR EXTERNO

c) XTAL y EXTAL (*Driver* del cristal y entrada de reloj externo):

Estas dos terminales proporcionan la interfase para un cristal de cuarzo (figura 3.3a) para controlar el generador interno de reloj. Se debe evitar el uso de cristales de alta frecuencia que tengan baja impedancia (de corte AT), ya que representan una carga excesiva en la NAND interna. La frecuencia aplicada a estas terminales es cuatro veces mayor que el reloj interno E.

De manera alternativa (figura 3.3b), se puede conectar un oscilador externo, o un circuito de reloj compatible con CMOS en la terminal de entrada EXTAL y la terminal XTAL de salida se puede dejar libre, aunque se recomienda conectar un resistor de carga de 10 kΩ a 100 kΩ a fin de reducir interferencias por ruido (RFI). Se recomienda evitar el uso de osciladores a base de circuitos *Schmitt Trigger* ya que pueden provocar falla en el inicio de la oscilación.

La salida XTAL tiene el propósito de manejar solamente un cristal, aunque se puede conectar a la terminal EXTAL de otro MC68HC11 mediante un *buffer* o resistor opcional (figura 3.4).

d) E (Reloj E):

Terminal de salida del generador interno de reloj E que se utiliza en todas las operaciones internas del MCU. La frecuencia del reloj E es cuatro veces menor que la de las terminales XTAL y EXTAL, pues tiene doble divisor de frecuencia. Esta señal de reloj se detiene cuando el MCU está en modo *STOP* para ahorro de energía.

e) $\overline{\text{IRQ}}$ (Petición de interrupción):

La terminal de entrada $\overline{\text{IRQ}}$ es un medio para petición de interrupción asíncrona al MCU. Se puede seleccionar la activación por nivel o por flanco negativo mediante el registro OPTION (1039_H). $\overline{\text{IRQ}}$ se configura por nivel al salir del estado de *reset*. Se pueden utilizar varios dispositivos conectados en OR alambrada mediante un resistor *pull-up* (ver bandera I en la sección 3.3.2).

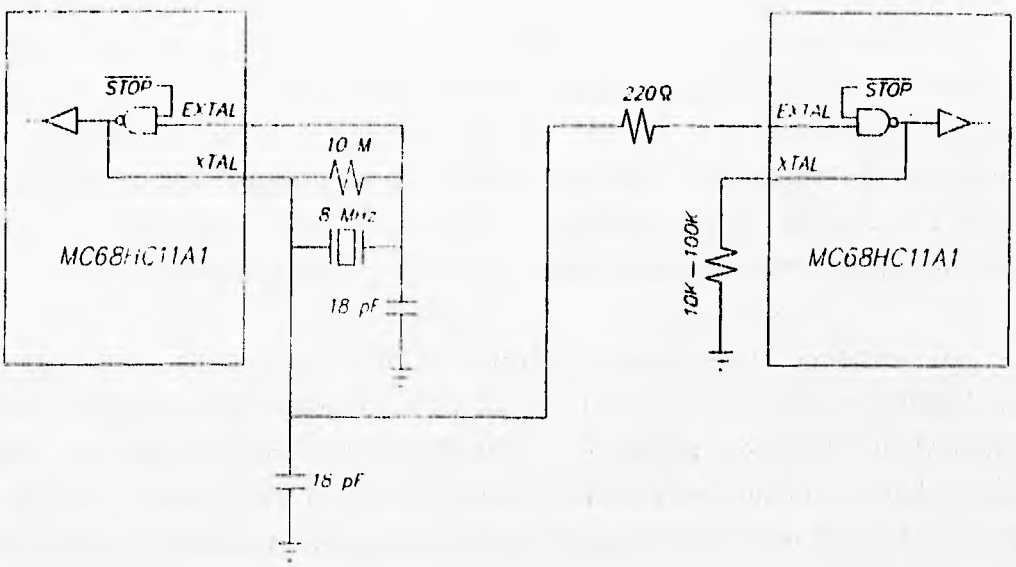


FIG 3.4 DOS MCU CON UN MISMO CRISTAL

f) $\overline{\text{XIRQ}}$ (Interrupción no mascarable):

La terminal de entrada $\overline{\text{XIRQ}}$ proporciona un medio para requerir una interrupción que no se pueda enmascarar una vez que haya sido habilitada. Durante el estado de *reset*, el *bit* X en el registro CCR (Registro de Código de Condición, ver la sección 3.3.1) es puesto en 1 lógico, enmascarando esta interrupción hasta que se habilita por *software* mediante la instrucción TAP (Transferencia del acumulador A hacia CCR, ver la sección 3.3.3). Una vez habilitada esta interrupción no puede enmascararse hasta un nuevo *reset*.

Debido a que esta terminal es activada por nivel, se pueden conectar varios dispositivos en función OR alambrada y una carga resistiva de 4.7 kΩ. Esta terminal se utiliza frecuentemente para detectar falla en el suministro de energía.

g) MODA/$\overline{\text{LIR}}$ y MODB/V_{STBY} (Modo A y modo B):

En estado de *reset*, ambas terminales son entradas, después de éste, estas terminales realizan su función alterna. En la transición, MODA y MODB seleccionan uno de los cuatro modos de operación (ver la sección 3.3)

Los modos de operación del MCU se seleccionan de acuerdo con la tabla 3.1.

MODO DE OPERACIÓN	MODA	MODB
UN *CHIP*	0	1
EXPANDIDO	1	1
ARRANQUE ESPECIAL	0	0
PRUEBA	1	0

TABLA 3.1 SELECCIÓN DEL MODO DE OPERACIÓN

Después de que se ha seleccionado un modo, la terminal $\overline{\text{LIR}}$ proporciona una salida de drenaje abierto para indicar el inicio de la ejecución de una instrucción, y se activa en nivel bajo durante el primer ciclo de reloj E (búsqueda del código de operación). La cantidad de ciclos de reloj E depende de cada instrucción. Esta señal es útil para la verificación por pasos y seguimiento de un programa. Para aplicaciones en el modo de operación en un solo *chip*, esta terminal debe conectarse a V_{SS} (0 V.).

La terminal V_{STBY} es una entrada que se utiliza para energizar la memoria RAM interna cuando se desenergiza V_{DD} y así retener su contenido. La memoria RAM y parte de la lógica de *reset* se energizan a través de esta terminal cuando (V_{STBY} - V_{DD}) es mayor de 0.7 volts. En la mayoría de las aplicaciones, esta terminal puede conectarse a través de un resistor de 4.7 kΩ a V_{DD}.

h) V_{RL} y V_{RH} (Voltajes de referencia bajo y alto):

Estas dos entradas proporcionan los voltajes de referencia para el convertidor analógico/digital. V_{RL} es la referencia baja, típicamente 0 V. V_{RH} es la referencia alta y debe tener al menos 3 volts más que V_{RL}. Ambas deben estar dentro del rango de valores de 0-5 V cd.

i) STRA/AS (Estrobo A):

Esta terminal realiza dos funciones dependiendo del modo de operación. En el modo de un solo *chip*, STRA (estrobo A) es una señal de entrada para protocolo de entrada de datos al puerto C. En el modo expandido multiplexado, AS es una señal de salida para habilitar la parte baja del *bus* de dirección y demultiplexarla del *bus* de datos en el puerto C.

j) STRB/ $R/\overline{W}$ (Estrobo B/Lectura ó Escritura):

Esta terminal realiza dos funciones dependiendo del modo de operación. En el modo de operación de un solo *chip*, STRB actúa como una señal de salida programable para protocolo de salida de datos del puerto B. En el modo expandido multiplexado, $R/\overline{W}$ se usa para indicar la dirección de transferencia en el *bus* de datos externo. Un 0 lógico en $R/\overline{W}$ indica que los datos son escritos hacia el *bus* externo de datos y un 1 lógico indica que un ciclo de lectura está en progreso.

k) Terminales de puertos:

Las terminales de los puertos A, D y E son independientes del modo de operación y sus diversos propósitos se describen posteriormente. Los puertos B y C son afectados por el modo de operación. El puerto B contiene ocho salidas de propósito general en el modo de operación en un solo *chip;* en el modo expandido, estas ocho terminales son las líneas de dirección de alto orden. El puerto C proporciona ocho señales de entrada/salida de propósito general cuando el MCU está en el modo de operación en un solo *chip;* En el modo expandido las terminales del puerto C son el *bus* multiplexado de direcciones de bajo orden/datos.

3.3 UNIDAD CENTRAL DE PROCESO

La CPU está diseñada para tratar a todos los dispositivos periféricos internos de entrada o salida como localidades de memoria, y sus direcciones están contenidas en el mapa de memoria. Es decir, para enviar un *byte* a un periférico se utiliza una instrucción de "almacenar en memoria", y para recibirlo "cargar desde memoria". No hay instrucciones especiales para periféricos internos que los distingan de ser localidades en el mapa de memoria. Sin embargo, la arquitectura del MCU también permite acceder a memoria externa, trabajando en modo expandido.

3.3.1 Registros de la CPU

Los registros de la CPU forman una parte integral de ésta y no tienen dirección en la memoria como los demás registros. La tabla 3.2 muestra los registros de la CPU:

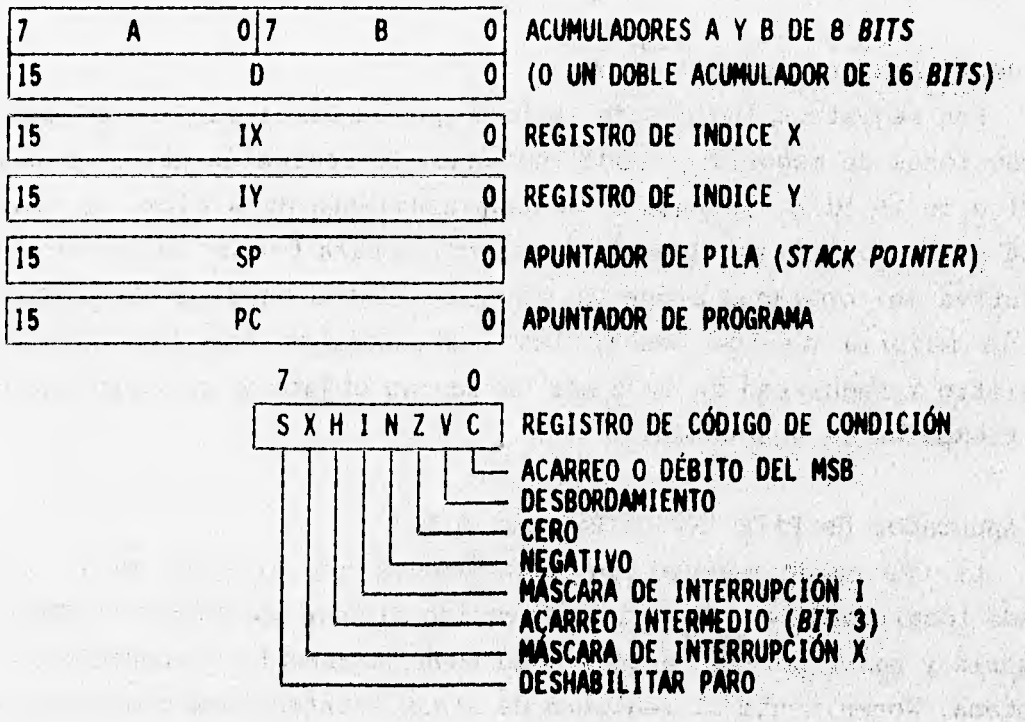


TABLA 3.2 REGISTROS INTERNOS DEL MC68HC11 MCU

a) Acumuladores A y B.

Los acumuladores A y B son, cada uno, registros de 8 *bits* que pueden almacenar operandos y resultados de cálculos aritméticos o manipulación de datos. Algunas instrucciones tratan a la combinación de estos dos como un solo acumulador doble de 16 *bits* (acumulador D). La mayoría de las operaciones pueden usar cualquiera de los acumuladores, sin embargo, las instrucciones ABX y ABY, suman el contenido del acumulador B con el registro de índice X ó Y, y no existe instrucción equivalente para el acumulador A. Las instrucciones TAP y TPA son usadas para transferir datos desde el acumulador A hacia el Registro de Código de Condición y viceversa, y no existe instrucción equivalente para el acumulador B. La instrucción de ajuste a decimal (DAA), solamente existe para el acumulador A y finalmente las instrucciones que utilizan ambos acumuladores como operandos, adición, substracción y comparación, almacenan el resultado solamente en el acumulador A.

b) Registros de índice (X e Y).

Son registros índices de 16 *bits* que se utilizan para acceder direcciones de memoria en modo indexado. El contenido del registro índice de 16 *bits*, se suma a un desplazamiento de 8 *bits*, el cual está incluído como parte de la instrucción para formar la dirección efectiva del operando a que va a ser utilizado en la instrucción. En la mayoría de los casos, las instrucciones que implican al registro Y requieren un *byte* más de código objeto y un ciclo extra de tiempo en la ejecución.

c) Apuntador de Pila (*Stack Pointer* ó SP)

La CPU puede manejar automáticamente una pila de datos que puede localizarse en cualquier dirección disponible en los 64 KB de memoria y puede ser tan grande como toda la memoria disponible del sistema. Normalmente el registro de *Stack Pointer* debe programarse en las primeras instrucciones del programa de aplicación. Cada vez que un dato es "almacenado" en la pila, este registro se decrementa

automáticamente; y a la inversa, cuando el dato se "recupera" de la pila, el registro se incrementa. En un momento dado, el *Stack Pointer* indica la siguiente dirección libre de la pila de datos. En general el *Stack Pointer* y la pila de datos son utilizados por los llamados a subrutinas, interrupciones y para almacenar datos a discreción.

Cuando se llama una subrutina con las instrucciones JSR ó BSR, la dirección de la siguiente instrucción se almacena en la pila, y se recupera para poder regresar al programa principal. De esta forma no se pierde la dirección de retorno y permite varias subrutinas anidadas y aún llamados a sí mismas.

En las interrupciones sucede un proceso análogo, pero además de almacenarse la dirección de regreso, se almacena también una copia del valor de todos los registros del CPU y al regresar, los valores son recuperados en forma inversa al almacenamiento. De este modo los registros aunque se hayan alterado en alguna rutina de servicio de interrupción, siempre recuperan su valor. Si se quiere manipular algún dato obtenido en este tipo de rutinas, se tendrá que almacenar en alguna localidad en RAM.

d) Apuntador de Programa (*Program Counter* ó PC)

Este registro indica la localidad de memoria que se accede en la ejecución del programa y es el que se almacena primero en las llamadas a subrutinas o interrupciones en la pila de datos.

e) Registro de Código de Condición (CCR)

Este registro almacena el código del resultado de la última operación aritmética, lógica o de manipulación de datos que se haya realizado con cualquier acumulador. Contiene cinco banderas de resultado, dos *bits* de interrupción y un *bit* indicador de *STOP* deshabilitado.

El *bit* C indica si existe acarreo en operaciones de adición, o indica débito en la substracción o comparación. Actúa también como indicador de error en las operaciones de multiplicación y división. También es muy útil en las instrucciones de rotación.

El *bit* V (*Overflow*) es utilizado para indicar un desbordamiento en números de complemento a 2 como resultado de una operación.

El *bit* Z (*Zero*) es 1 lógico cuando todos los *bits* del resultado de una operación son 0 lógico.

El *bit* N (*Negative*) es una copia del MSB de un resultado. Para los números con complemento a 2, un número es negativo cuando el MSB es 1 lógico, y es positivo cuando MSB es 0 lógico.

El *bit* H (*Half Carry*) indica un acarreo del *bit* 3 durante la operación de adición. Este indicador permite al CPU ajustar el resultado de una adición de 8 *bits* BCD a su correcto formato BCD mediante la instrucción DAA.

El *bit* S (*Stop Disable*) inhibe o habilita que el reloj E se detenga cuando el programa encuentra la instrucción STOP, empleada para un estado de ahorro de energía.

El *bit* I ($\overline{\text{IRQ}}$) es una bandera global para la habilitación de todas las interrupciones mascarables. Después del *reset*, esta bandera se pone en 1, deshabilitando la atención a subrutinas. Cuando se habilita por *software*, a 0 lógico, se puede atender las subrutinas de interrupción. Al llamado de una interrupción, este *bit* regresa a 1 después de que todos los registros sean guardados en la pila. Este *bit* se debe regresar a 0, mediante *software*, al finalizar la subrutina, para impedir llamadas anidadas.

El *bit* X es una mascarilla utilizada para deshabilitar la interrupción provocada en la terminal $\overline{\text{XIRQ}}$. Después del *reset*, esta bandera es 1 lógico y solamente puede pasar a 0 (habilitar la interrupción) mediante una instrucción. Después que se reconoce la interrupción, el *bit* X (también el I) es automáticamente 1 lógico después de que los registros han sido guardados en pila y antes de buscar el vector de interrupción. Para atender la subrutina este *bit* debe estar habilitado (0 lógico), pero dentro de ésta, el *bit* deshabilita un llamado anidado (1 lógico). Si se requieren varios niveles de subrutinas, se tendría entonces que habilitar nuevamente en el nivel anterior.

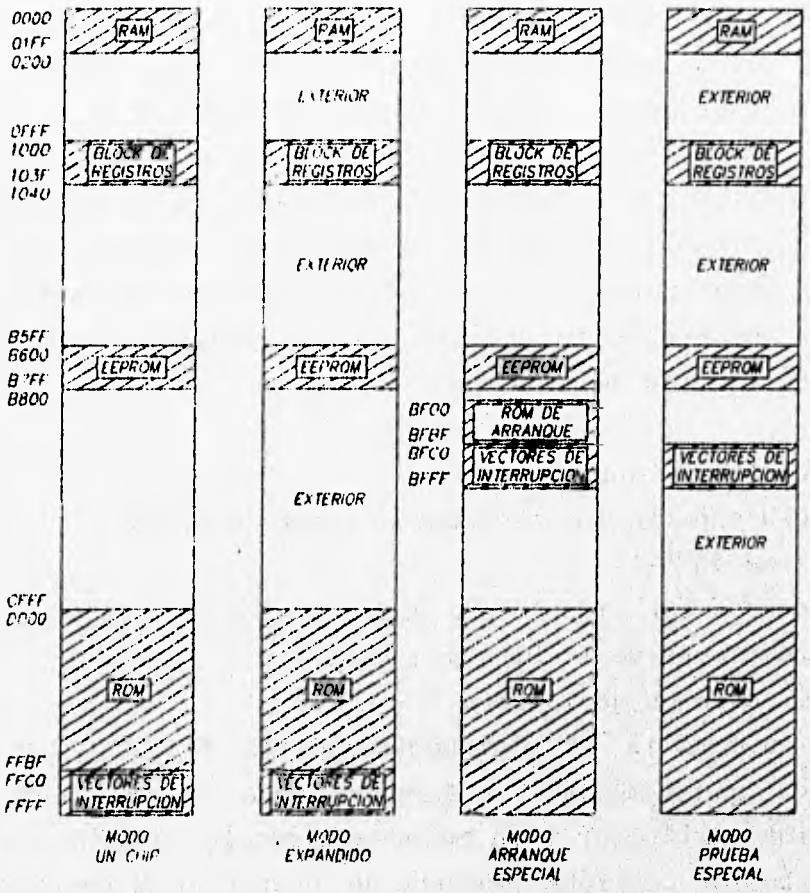


TABLA 3.3 MAPA DE MEMORIA

3.3.2 Registros en el mapa de memoria

El modo de operación determina el mapa de memoria y la dirección interna o externa de ésta. La tabla 3.3 muestra el mapa de memoria para cada uno de los cuatro modos de operación. Las localidades de memoria interna en el *chip* son las mismas para los modos en un solo *chip* y expandido. Los *bits* de control en el

registro CONFIG ($103F_H$) permiten a la EPROM y EEPROM ser deshabilitadas del mapa de memoria. La RAM de 512 *bytes* se localiza en la dirección 0000_H después del *reset*, y puede relocalizarse en cualquier otro lugar múltiplo de 4 KB, escribiendo el valor apropiado en el registro INIT ($103D_H$). Los dispositivos periféricos internos, que son tratados como localidades en el mapa de memoria, forman un grupo de 64 registros localizados en la dirección 1000_H después del *reset*, y puede ser relocalizada en cualquier múltiplo de 4 KB, escribiendo el valor apropiado en el registro INIT. La tabla 3.4 muestra las direcciones de los registros localizables en el mapa de memoria después del *reset*.

3.3.3 Conjunto de instrucciones

La CPU soporta los siguientes tipos de datos:

- Datos de *bits*
- Enteros con signo o sin signo de 8 y 16 *bits*
- Fracciones de 16 *bits* sin signo
- Direcciones de 16 *bits*

En la familia de microcontroladores MC68HC11 se utilizan códigos de operación de 1 ó 2 *bytes*. Cada código identifica una instrucción particular y un modo de dirección asociado a la CPU. Una instrucción completa consiste de un código de operación, y de cero hasta tres operandos. Los operandos contienen la información que la CPU necesita para ejecutar la instrucción. Una instrucción completa puede tener desde uno hasta 5 *bytes* de longitud.

Existen seis modos de direccionamiento: inherente, inmediato, directo, extendido, indexado y relativo, en todos ellos, excepto en el primero, se utiliza una dirección de memoria efectiva en la cual se accede al operador de la instrucción.

- Modo Inherente: El código de operación contiene toda la información necesaria para ejecutar la instrucción. En este modo se incluyen las operaciones que usan solamente los registros de índice o acumuladores, así como las instrucciones de control sin argumentos, pueden ser de uno o dos *bytes*.

OIR. (HEX)	BITS								REGISTRO
	Bit 7	6	5	4	3	2	1	Bit 0	
1000	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	PORTA
1001									Reservado
1002	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	PIOC
1003	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	PORTC
1004	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	PORTB
1005	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	PORTCL
1006									Reservado
1007	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	DDRC
1008			Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	PORTD
1009			DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0	DDRD
100A	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	PORTE
100B	FOC1	FOC2	FOC3	FOC4	FOC5				CFORC
100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3				OC1M
100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3				OC1D
100E	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TCNT
100F	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
1010	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TIC1
1011	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
1012	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TIC2
1013	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
1014	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TIC3
1015	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
1016	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TOC1
1017	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
101B	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TOC2
1019	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
101A	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TOC3
101B	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
101C	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TOC4
101D	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
101E	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	TOC5
101F	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
	Bit 7	6	5	4	3	2	1	Bit 0	

TABLA 3.4A ASIGNACION DE REGISTROS Y BITS DE CONTROL

DIR. (HEX)	BITS								REGISTRO
	Bit 7	6	5	4	3	2	1	Bit 0	
1020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1
1021			EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2
1022	OC1I	OC2I	OC3I	OC4I	OC5I	IC1I	IC2I	IC3I	TMSK1
1023	OC1F	OC2F	OC3F	OC4F	OC5F	IC1F	IC2F	IC3F	TFLG1
1024	**TOI**	RTII	PAOVI	PAII			PR1	PR0	TMSK2
1025	TOF	RTIF	PAOVF	PAIF					TFLG2
1026	DDRA7	PAEN	PAMOD	PEDGE			RTR1	RTR0	PACTL
1027	Bit 7	Bit 6	**Bit 5**	Bit 4	Bit 3	Bit 2	Bit 1	**Bit 0**	PACNT
1028	SPIE	**SPE**	DWOM	MSTR	CPOL	CPHA	**SPR1**	SPR0	SPCR
1029	SPIF	WCOL		MODF					SPSR
102A	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	**Bit 1**	Bit 0	SPDR
102B	TCLR		SCP1	SCP0	RCKB	SCR2	SCR1	SCR0	BAUD
102C	R8	T8		M	WAKE				SCCR1
102D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2
102E	**TDRE**	TC	RDRF	IDLE	OR	NF	FE		SCSR
102F	Bit 7	Bit 6	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	Bit 1	**Bit 0**	SCDR
1030	CCF		SCAN	MULT	CD	CC	CB	CA	ADCTL
1031	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	ADR1
1032	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	ADR2
1033	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	ADR3
1034	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	ADR4
1035	No existe este registro en MC68HC11A1/A8								BPROT
1036									Reservado
1037									Reservado
1038									Reservado
1039	ADPU	CSEL	IRQE	DLY	CME		CR1	CR0	OPTION
103A	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	COPRST
103B	ODD	EVEN		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		OCCR	CBYP	DISR	FCM	FCOP	TCON	TEST1
103F					NOSEC	NOCOP	ROMON	EEON	CONFIG
	Bit 7	6	5	4	3	2	1	Bit 0	

TABLA 3.4B ASIGNACION DE REGISTROS Y BITS DE CONTROL

- Modo Inmediato: El argumento está contenido en uno o dos *bytes* que inmediatamente siguen del código de operación, dependiendo del tipo de instrucción. La instrucción completa puede ser de dos, tres o cuatro *bytes*.
- Modo Directo: El operador se debe encontrar en la memoria entre las direcciones 0000_H y $00FF_H$. El *byte* siguiente al código de operación se utiliza como *byte* de dirección de bajo orden y el de alto orden se asume que será 00_H. Este modo reduce tiempo de ejecución mediante la eliminación del acceso adicional del *byte* de alto orden de la dirección de memoria.
- Modo Extendido: La dirección efectiva del argumento está contenida en los dos *bytes* siguientes al código de operación. Toda la instrucción puede ser de tres o cuatro *bytes*.
- Modo Indexado: La dirección efectiva del operador se indica con la suma de los valores del registro índice X ó Y más un desplazamiento representado por el *byte* que sigue al código de operación. El desplazamiento siempre será positivo. Las instrucciones son de dos a cinco *bytes*.
- Modo Relativo: Este modo se usa solo en las instrucciones de salto. Si la condición para esto es verdadera, se agrega un desplazamiento de ocho *bits* con signo al apuntador de programa (PC) para formar el salto a la dirección efectiva. El desplazamiento es un *byte* inmediatamente después del código de operación. Estas instrucciones son generalmente de dos *bytes*, y en ocasiones de cuatro o cinco *bytes*.

El conjunto de instrucciones con todos los modos de direccionamiento posibles se muestra en la Tabla 3.5, en la cual se indican los códigos de operación, la construcción de los operandos, y el tiempo de ejecución en ciclos de reloj E. Las instrucciones se han agrupado atendiendo al propósito esencial de éstas, formando 14 grupos:

- Carga, almacenamiento, transferencia e intercambio: se refiere a toda instrucción cuya finalidad es cambiar el valor de un registro o de una localidad de memoria.

- Operaciones lógicas: AND, OR y XOR que pueden realizarse a nivel de 8 *bits*.
- Incrementos y decrementos: comprende las instrucciones que alteran el valor de los registros o *bytes* de la memoria en una unidad.
- Suma y resta: Se refiere a la adición y sustracción aritmética de registros y/o *bytes* de memoria, ya sea con acarreo o sin él.
- Multiplicación y división: Se maneja multiplicación de 8 por 8 *bits*, con resultado en 16 *bits* y división entera y/o fraccionaria de 16 por 16 *bits*.
- Rotaciones y corrimientos: Son desplazamientos aritméticos ó lógicos de *bits*, en ambos sentidos.
- Manejo y verificación de *bits*: El primero se refiere al cambio a nivel binario de uno o varios *bits* de una localidad de memoria especificados por una mascarilla. El segundo, a la prueba de *bits* con la finalidad de actualizar las banderas del registro CCR.
- Comparación y prueba de *bytes*: Estas instrucciones se utilizan para comparar valores de registros y localidades de memoria y actualizar las banderas de registro CCR.
- Saltos de dirección: Agrupa las instrucciones usadas para alterar el valor de Apuntador de Programa (PC) dependiendo de una condición.
- Complementos a 1, a 2 y ajuste decimal: Son instrucciones que realizan complementos y negativos de los acumuladores y de localidades de memoria.
- Interrupciones: Instrucciones orientadas a la atención a las interrupciones, así como habilitación y retorno de las mismas.
- Subrutina: Son las instrucciones que transfieren el flujo del programa a una dirección especificada y tienen asociado un retorno.
- Otras instrucciones: Grupo de instrucciones diversas orientadas al desarrollo y pruebas del MCU.

CARGA, ALMACENAMIENTO, TRANSFERENCIA E INTERCAMBIO						REG. COD. CONDIC.						
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
CLR (oper)	Carga 00 en byte de Memoria	0→M	EXTEND.	7F hh ll	6	–	–	–	–	0	1	0	0
			IND.,X	6F ff	6								
			IND.,Y	18 6F ff	7								
CLRA	Carga 00 en acumulador A	0→A	INH.	4F	2	–	–	–	–	0	1	0	0
CLRB	Carga 00 en acumulador B	0→B	INH.	5F	2	–	–	–	–	0	1	0	0
LDAA (oper)	Carga el acumulador A	M→A	INMED.	86 ii	2	–	–	–	–	^	^	0	–
			DIR.	96 dd	3								
			EXTEND.	B6 hh ll	4								
			IND.,X	A6 ff	4								
			IND.,Y	18 A6 ff	5								
LDAB (oper)	Carga el acumulador B	M→B	INMED.	C6 ii	2	–	–	–	–	^	^	0	–
			DIR	D6 dd	3								
			EXTEND.	F6 hh ll	4								
			IND.,X	E6 ff	4								
			IND.,Y	18 E6 ff	5								
LDD (oper)	Carga el doble acumulador D	M→A,M+1→B	INMED.	CC jj kk	3	–	–	–	–	^	^	0	–
			DIR.	DC dd	4								
			EXTEND.	FC hh ll	5								
			IND.,X	EC ff	5								
			IND.,Y	18 EC ff	6								
LDS (oper)	Carga el Apunt. Pila	M:M+1→SP	INMED.	8E jj kk	3	–	–	–	–	^	^	0	–
			DIR	9E dd	4								
			EXTEND.	BE hh ll	5								
			IND.,X	AE ff	5								
			IND.,Y	18 AE ff	6								
LDX (oper)	Carga el registro X	M:M+1→IX	INMED.	CE jj kk	3	–	–	–	–	^	^	0	–
			DIR	DE dd	4								
			EXTEND.	FE hh ll	5								
			IND.,X	EE ff	5								
			IND.,Y	CD EE ff	6								
LDY (oper)	Carga el registro Y	M:M+1→IY	INMED.	18 CE jj kk	4	–	–	–	–	^	^	0	–
			DIR.	18 DE dd	5								
			EXTEND.	18 FE hh ll	6								
			IND.,X	1A EE ff	6								
			IND.,Y	18 EE ff	6								
STAA (oper)	Almacena el acumulador A	A→M	DIR	97 dd	3	–	–	–	–	^	^	0	–
			EXTEND.	B7 hh ll	4								
			IND.,X	A7 ff	4								
			IND.,Y	18 A7 ff	5								
STAB (oper)	Almacena el acumulador B	B→M	DIR.	D7 dd	3	–	–	–	–	^	^	0	–
			EXTEND.	F7 hh ll	4								
			IND.,X	E7 ff	4								
			IND.,Y	18 E7 ff	5								
STD (oper)	Almacena el acumulador D	A→M,B→M+1	DIR	DD dd	4	–	–	–	–	^	^	0	–
			EXTEND.	FD hh ll	5								
			IND.,X	ED ff	5								
			IND.,Y	18 ED ff	6								
STS (oper)	Almacena el Apunt. de Pila	SP→M:M+1	DIR.	9F dd	4	–	–	–	–	^	^	0	–
			EXTEND.	BF hh ll	5								
			IND.,X	AF ff	5								
			IND.,Y	18 AF ff	6								
STX (oper)	Almacena el registro X	IX→M:M+1	DIR.	DF dd	4	–	–	–	–	^	^	0	–
			EXTEND.	FF hh ll	5								
			IND.,X	EF ff	5								
			IND.,Y	CD EF ff	6								
STY (oper)	Almacena el registro Y	IY→M:M+1	DIR.	18 DF dd	5	–	–	–	–	^	^	0	–
			EXTEND.	18 FF hh ll	6								
			IND.,X	1A EF ff	6								
			IND.,Y	18 EF ff	6								
TAB	Transferencia de A a B	A→B	INH.	16	2	–	–	–	–	^	^	0	–
TBA	Transferencia de B a A	B→A	INH.	17	2	–	–	–	–	^	^	0	–
TAP	Transferencia de A al CCR	A→CCR	INH.	06	2	·	1	^	^	^	^	^	^
TPA	Transferencia del CCR a A	CCR→A	INH.	07	2	–	–	–	–	–	–	–	–
TSX	Transferencia de SP a reg. X	SP+1→IX	INH.	30	3	–	–	–	–	–	–	–	–
TSY	Transferencia de SP a reg. Y	SP+1→IY	INH.	18 30	4	–	–	–	–	–	–	–	–
TXS	Transferencia del reg. X a SP	IX–1→SP	INH.	35	3	–	–	–	–	–	–	–	–
TYS	Transferencia del reg. Y a SP	IY–1→SP	INH.	18 35	4	–	–	–	–	–	–	–	–
XGDX	Intercambio de D con X	IX→D	INH.	8F	3	–	–	–	–	–	–	–	–
XGDY	Intercambio de D con Y	IY→D	INH.	18 8F	4	–	–	–	–	–	–	–	–

TABLA 3.5 CONJUNTO DE INSTRUCCIONES (HOJA 1 DE 7)

OPERACIONES LOGICAS						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
ANDA (oper)	AND acum. A con Memoria	A•M→A	INMED.	84 ii	2	-	-	-	-	^	^	0	-
			DIR.	94 dd	3								
			EXTEND.	B4 hh ll	4								
			IND.,X	A4 ff	4								
			IND.,Y	18 A4 ff	5								
ANDB (oper)	AND acum. B con Memoria	B•M→B	INMED.	C4 ii	2	-	-	-	-	^	^	0	-
			DIR.	D4 dd	3								
			EXTEND.	F4 hh ll	4								
			IND.,X	E4 ff	4								
			IND.,Y	18 E4 ff	5								
EORA (oper)	OR exclusiva A con Memoria	A ⊕ M→A	INMED.	88 ii	2	-	-	-	-	^	^	0	-
			DIR.	98 dd	3								
			EXTEND.	B8 hh ll	4								
			IND.,X	A8 ff	4								
			IND.,Y	18 A8 ff	5								
EORB (oper)	OR exclusiva B con Memoria	A ⊕ M→A	INMED.	C8 ii	2	-	-	-	-	^	^	0	-
			DIR.	D8 dd	3								
			EXTEND.	F8 hh ll	4								
			IND.,X	E8 ff	4								
			IND.,Y	18 E8 ff	5								
ORAA (oper)	OR acum. A con Memoria	A + M→A	INMED.	8A ii	2	-	-	-	-	^	^	0	-
			DIR.	9A dd	3								
			EXTEND.	BA hh ll	4								
			IND.,X	AA ff	4								
			IND.,Y	18 AA ff	5								
ORAB (oper)	OR acum. B con Memoria	B + M→B	INMED.	CA ii	2	-	-	-	-	^	^	0	-
			DIR.	DA dd	3								
			EXTEND.	FA hh ll	4								
			IND.,X	EA ff	4								
			IND.,Y	18 EA ff	5								

INCREMENTOS Y DECREMENTOS						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
DEC (oper)	Decrementa byte de Memoria	M-1→M	EXTEND.	7A hh ll	6	-	-	-	-	^	^	^	-
			IND.,X	6A ff	6								
			IND.,Y	18 6A ff	7								
DECA	Decrementa acum. A	A-1→A	INH.	4A	2	-	-	-	-	^	^	^	-
DECB	Decrementa acum. B	B-1→B	INH.	5A	2	-	-	-	-	^	^	^	-
DES	Decrementa SP	SP-1→SP	INH.	34	3	-	-	-	-	-	-	-	-
DEX	Decremento registro X	IX-1→IX	INH.	09	3	-	-	-	-	-	^	-	-
DEY	Decremento registro Y	IY-1→IY	INH.	18 09	4	-	-	-	-	-	^	-	-
INC (oper)	Incrementa byte de Memoria	M+1→M	EXTEND.	7C hh ll	6	-	-	-	-	^	^	^	-
			IND.,X	6C ff	6								
			IND.,Y	18 6C ff	7								
INCA	Incrementa acum. A	A+1→A	INH.	4C	2	-	-	-	-	^	^	^	-
INCB	Incrementa acum. B	B+1→B	INH.	5C	2	-	-	-	-	^	^	^	-
INS	Incrementa SP	SP+1→SP	INH.	31	3	-	-	-	-	-	-	-	-
INX	Incrementa registro X	IX+1→IX	INH.	08	3	-	-	-	-	-	^	-	-
INY	Incrementa registro Y	IY+1→IY	INH.	18 08	4	-	-	-	-	-	^	-	-

TABLA 3.5 CONJUNTO DE INSTRUCCIONES (HOJA 2 DE 7)

SUMA Y RESTA													
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD DE OPER.	CICLOS	REG. COD. CONDIC S	X	H	I	N	Z	V	C
ABA	Suma los dos acumuladores	A+B→A	INH.	1B	2	–	–	^	–	^	^	^	^
ABX	Suma B a X	IX+(00:B)→IX	INH.	3A	3	–	–	–	–	–	–	–	–
ABY	Suma B a Y	IY+(00:B)→IY	INH.	18 3A	4	–	–	–	–	–	–	–	–
ADCA (oper)	Suma con acarreo a A	A+M+C→A	INMED.	89 ii	2	–	–	^	–	^	^	^	^
			DIR.	99 dd	3								
			EXTEND.	B9 hh ll	4								
			IND.,X	A9 ff	4								
			IND.,Y	18 A9 ff	5								
ADCB (oper)	Suma con acarreo a B	B+M+C→B	INMED.	C9 ii	2	–	–	^	–	^	^	^	^
			DIR.	D9 dd	3								
			EXTEND.	F9 hh ll	4								
			IND.,X	E9 ff	4								
			IND.,Y	18 E9 ff	5								
ADDA (oper)	Suma byte de Memoria a A	A+M→A	INMED.	8B ii	2	–	–	^	–	^	^	^	^
			DIR.	9B dd	3								
			EXTEND.	BB hh ll	4								
			IND.,X	AB ff	4								
			IND.,Y	18 AB ff	5								
ADDB (oper)	Suma byte de Memoria a B	B+M→B	INMED.	CB ii	2	–	–	^	–	^	^	^	^
			DIR.	DB dd	3								
			EXTEND.	FB hh ll	4								
			IND.,X	EB ff	4								
			IND.,Y	18 EB ff	5								
ADDD (oper)	Suma de 16 bits a D	D+(M:M+1)→D	INMED.	C3 jj kk	4	–	–	–	–	^	^	^	^
			DIR.	D3 dd	5								
			EXTEND.	F3 hh ll	6								
			IND.,X	E3 ff	6								
			IND.,Y	18 E3 ff	7								
SBA	Resta B de A	A–B→A	INH.	10	2	–	–	–	–	^	^	^	^
SBCA (oper)	Resta con acarreo de A	A–M–C→A	INMED.	82 ii	2	–	–	–	–	^	^	^	^
			DIR.	92 dd	3								
			EXTEND.	B2 hh ll	4								
			IND.,X	A2 ff	4								
			IND.,Y	18 A2 ff	5								
SBCB (oper)	Resta con acarreo de B	B–M–C→B	INMED.	C2 ii	2	–	–	–	–	^	^	^	^
			DIR.	D2 dd	3								
			EXTEND.	F2 hh ll	4								
			IND.,X	E2 ff	4								
			IND.,Y	18 E2 ff	5								
SUBA (oper)	Resta un byte de Memoria del acumulador A	A–M→A	INMED.	80 ii	2	–	–	–	–	^	^	^	^
			DIR.	90 dd	3								
			EXTEND.	B0 hh ll	4								
			IND.,X	A0 ff	4								
			IND,Y	18 A0 ff	5								
SUBB (oper)	Resta un byte de Memoria del acumulador B	B–M→B	INMED.	C0 ii	2	–	–	–	–	^	^	^	^
			DIR.	D0 dd	3								
			EXTEND.	F0 hh ll	4								
			IND.,X	E0 ff	4								
			IND.,Y	18 E0 ff	5								
SUBD (oper)	Resta de 16 bits de memoria del acumulador D	D–(M:M+1)→D	INMED.	83 jj kk	4	–	–	–	–	^	^	^	^
			DIR.	93 dd	5								
			EXTEND.	B3 hh ll	6								
			IND.,X	A3 ff	6								
			IND.,Y	18 A3 ff	7								

TABLA 3.6 CONJUNTO DE INSTRUCCIONES (HOJA 3 DE 7)

MULTIPLICACION Y DIVISION						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
FDIV	División fraccionaria de 16 por 16 bits	D/IX→IX; r→D	INH.	03	41	–	–	–	–	–	^	^	^
IDIV	División entera de 16 por 16 bits	D/IX→IX; r→D	INH.	02	41	–	–	–	–	–	^	0	^
MUL	Multiplicación de 8 bits	A*B→D	INH.	3D	10	–	–	–	–	–	–	–	^

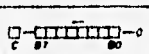
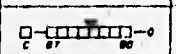
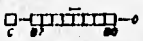
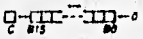
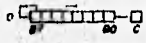
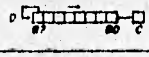
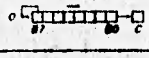
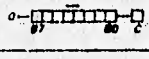
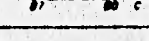
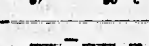
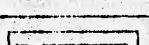
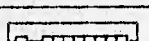
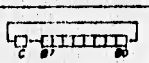
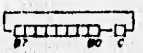
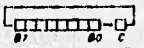
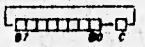
ROTACIONES Y CORRIMIENTOS						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
ASL (oper) ó LSL (oper)	Corrimiento aritmetico o lógico a la izquierda		EXTEND. IND.,X IND.,Y	78 hh ll 68 ll 18 68 ll	6 6 7	–	–	–	–	^	^	^	^
ASLA ó LSLA	Corrimiento aritmetico o lógico de A a la izquierda		INH.	48	2	–	–	–	–	^	^	^	^
ASLB ó LSLB	Corrimiento aritmetico o lógico de B a la izquierda		INH.	58	2	–	–	–	–	^	^	^	^
ASLD ó LSLD	Corrimiento aritmetico o lógico de D a la izquierdo		INH.	05	3	–	–	–	–	^	^	^	^
ASR	Corrimiento aritmetico a la derecha		EXTEND. IND.,X IND.,Y	77 hh ll 67 ll 18 67 ll	6 6 7	–	–	–	–	^	^	^	^
ASRA	Corrimiento aritmetico de A a la derecha		INH.	47	2	–	–	–	–	^	^	^	^
ASRB	Corrimiento aritmetico de B a la derecha		INH.	57	2	–	–	–	–	^	^	^	^
LSR (oper)	Corrimiento lógico a la derecha		EXTEND. IND.,X IND.,Y	74 hh ll 64 ll 18 64 ll	6 6 7	–	–	–	–	0	^	^	^
LSRA	Corrimiento lógico de A a la derecha		INH.	44	2	–	–	–	–	0	^	^	^
LSRB	Corrimiento lógico de B a la derecha		INH.	54	2	–	–	–	–	0	^	^	^
LSRD	Corrimiento lógico de D a la derecha		INH.	04	3	–	–	–	–	0	^	^	^
ROL (oper)	Rotación a la izquierda		EXTEND. IND.,X IND.,Y	79 hh ll 69 ll 18 69 ll	6 6 7	–	–	–	–	^	^	^	^
ROLA	Rotación de A a la izquierda		INH.	49	2	–	–	–	–	^	^	^	^
ROLB	Rotación de B a la izquierda		INH.	59	2	–	–	–	–	^	^	^	^
ROR (oper)	Rotación a la derecha		EXTEND. IND.,X IND.,Y	76 hh ll 66 ll 18 66 ll	6 6 7	–	–	–	–	^	^	^	^
RORA	Rotación de A a la derecha		INH.	46	2	–	–	–	–	^	^	^	^
RORB	Rotación de B a la derecha		INH.	56	2	–	–	–	–	^	^	^	^

TABLA 3.5 CONJUNTO DE INSTRUCCIONES (HOJA 4 DE 7)

MANEJO Y VERIFICACION DE BITS						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
BCLR (oper) (masc)	Fijar bit(s) a 0	$M \cdot (\overline{mm}) - M$	DIR.	15 dd mm	6	-	-	-	-	^	^	0	-
			IND.,X	1D ff mm	7								
			IND.,Y	18 1D ff mm	8								
BSET (oper) (masc)	Fijar bit(s) a 1	$M + (mm) - M$	DIR.	14 dd mm	6	-	-	-	-	^	^	0	-
			IND.,X	1C ff mm	7								
			IND.,Y	18 1C ff mm	8								
BITA (oper)	Verificar bit(s) de A con Memoria	A•M	INMED.	85 ii	2	-	-	-	-	^	^	0	-
			DIR.	95 dd	3								
			EXTEND.	B5 hh ll	4								
			IND.,X	A5 ff	4								
			IND.,Y	18 A5 ff	5								
BITB (oper)	Verificar bit(s) de B con Memoria	B•M	INMED.	C5 ii	2	-	-	-	-	^	^	0	-
			DIR.	D5 dd	3								
			EXTEND.	F5 hh ll	4								
			IND.,X	E5 ff	4								
			IND.,Y	18 E5 ff	5								
CLC	Fijar en 0 la bandera C	0 — C	INH.	0C	2	-	-	-	-	-	-	-	0
SEC	Fijar en 1 la bandera C	1 — C	INH.	0D	2	-	-	-	-	-	-	-	1
CLV	Fijar en 0 la bandera V	0 — V	INH.	0A	2	-	-	-	-	-	-	0	-
SEV	Fijar en 1 la bandera V	1 — V	INH.	0B	2	-	-	-	-	-	-	1	-

COMPARACION Y PRUEBA DE BYTES						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
CBA	Compara A con B	A - B	INH.	11	2	-	-	-	-	^	^	^	^
CMPA (oper)	Compara A con Memoria	A - M	INMED.	81 ii	2	-	-	-	-	^	^	^	^
			DIR.	91 dd	3								
			EXTEND.	B1 hh ll	4								
			IND.,X	A1 ff	4								
			IND.,Y	18 A1 ff	5								
CMPB (oper)	Compara B con Memoria	B - M	INMED.	C1 ii	2	-	-	-	-	^	^	^	^
			DIR.	D1 dd	3								
			EXTEND.	F1 hh ll	4								
			IND.,X	E1 ff	4								
			IND.,Y	18 E1 ff	5								
CPD (oper)	Compara D con 16 bits de Memoria	D - (M:M+1)	INMED.	1A 83 jj kk	5	-	-	-	-	^	^	^	^
			DIR.	1A 93 dd	6								
			EXTEND.	1A B3 hh ll	7								
			IND.,X	1A A3 ff	7								
			IND.,Y	CD A3 ff	7								
CPX (oper)	Compara X con 16 bits de Memoria	IX - (M:M+1)	INMED.	8C jj kk	4	-	-	-	-	^	^	^	^
			DIR.	9C dd	5								
			EXTEND.	BC hh ll	6								
			IND.,X	AC ff	6								
			IND.,Y	CD AC ff	7								
CPY (oper)	Compara Y con 16 bits de Memoria	IY - (M:M+1)	INMED.	18 8C jj kk	5	-	-	-	-	^	^	^	^
			DIR.	18 9C dd	6								
			EXTEND.	18 BC hh ll	7								
			IND.,X	1A AC ff	7								
			IND.,Y	18 AC ff	7								
TST (oper)	Prueba si es Cero o Negativo	M - 0	EXTEND.	7D hh ll	6	-	-	-	-	^	^	0	0
			IND.,X	6D ff	6								
			IND.,Y	18 6D ff	7								
TSTA	Prueba si A es Cero o Negativo	A - 0	INH.	4D	2	-	-	-	-	^	^	0	0
TSTB	Prueba si B es Cero o Negativo	B - 0	INH.	5D	2	-	-	-	-	^	^	0	0

TABLA 3.5 CONJUNTO DE INSTRUCCIONES (HOJA 5 DE 7)

SALTO DE DIRECCION						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
BCC (rel) BHS (rel)	Salto si no hay acarreo ó si es mayor o igual	?C=0	REL.	24 rr	3	-	-	-	-	-	-	-	-
BCS (rel) BLO (rel)	Salto si hay acarreo ó si es menor	?C=1	REL.	25 rr	3	-	-	-	-	-	-	-	-
BNE (rel)	Salto si no es = cero	?Z=0	REL.	26 rr	3	-	-	-	-	-	-	-	-
BEQ (rel)	Salto si = cero	?Z=1	REL.	27 rr	3	-	-	-	-	-	-	-	-
BGT (rel)	Salto si > cero	?Z+(N ⊕ V)=0	REL.	2E rr	3	-	-	-	-	-	-	-	-
BLE (rel)	Salto si < = cero	?Z+(N ⊕ V)=1	REL.	2F rr	3	-	-	-	-	-	-	-	-
BGE (rel)	Salto si >= cero	?N ⊕ V=0	REL.	2C rr	3	-	-	-	-	-	-	-	-
BLT (rel)	Salto si < cero	?N ⊕ V=1	REL.	2D rr	3	-	-	-	-	-	-	-	-
BPL (rel)	Salto si es positivo	?N=0	REL.	2A rr	3	-	-	-	-	-	-	-	-
BMI (rel)	Salto si es negativo	?N=1	REL.	2B rr	3	-	-	-	-	-	-	-	-
BHI (rel)	Salto si es mayor	?C+Z=0	REL.	22 rr	3	-	-	-	-	-	-	-	-
BLS (rel)	Salto si es menor o igual	?C+Z=1	REL.	23 rr	3	-	-	-	-	-	-	-	-
BVC (rel)	Salto si no hay desbordamiento	?V=0	REL.	28 rr	3	-	-	-	-	-	-	-	-
BVS (rel)	Salto si hay desbordamiento	?V=1	REL.	29 rr	3	-	-	-	-	-	-	-	-
BRA (rel)	Salto siempre (incondicionado)	?1=1	REL.	20 rr	3	-	-	-	-	-	-	-	-
BRN (rel)	Salto nulo	?1=0	REL.	21 rr	3	-	-	-	-	-	-	-	-
BRCLR (oper) (masc) (rel)	Salto si el bit(s) es cero	?M•mm=0	DIR. IND.,X IND.,Y	13 dd mm rr 1F ii mm rr 18 1F ii mm rr	6 7 8	-	-	-	-	-	-	-	-
BRSET (oper) (masc) (rel)	Salto si el bit(s) es 1	?M•mm=0	DIR. IND.,X IND.,Y	12 dd mm rr 1E ii mm rr 18 1E ii mm rr	6 7 8	-	-	-	-	-	-	-	-
JMP (oper)	Salto incondicional	PC+(oper)→PC	EXTEND. IND.,X IND.,Y	7E hh ll 6E ii 18 6E ii	3 3 4	-	-	-	-	-	-	-	-

COMPLEMENTOS A 1, A 2 Y AJUSTE DECIMAL						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
COM (oper)	Complemento a 1 de un byte de Memoria	FF_H − M → M	EXTEND. IND.,X IND.,Y	73 hh ll 63 ii 18 63 ii	6 6 7	-	-	-	-	^	^	0	1
COMA	Complemento a 1 de A	FF_H − A → A	INH.	43	2	-	-	-	-	^	^	0	1
COMB	Complemento a 1 de B	FF_H − B → B	INH.	53	2	-	-	-	-	^	^	0	1
NEG (oper)	Complemento a 2 de un byte de Memoria	0 − M → M	EXTEND. IND.,X IND.,Y	70 hh ll 60 ii 18 60 ii	6 6 7	-	-	-	-	^	^	^	^
NEGA	Complemento a 2 de A	0 − A → A	INH.	40	2	-	-	-	-	^	^	^	^
NEGB	Complemento a 2 de B	0 − B → B	INH.	50	2	-	-	-	-	^	^	^	^
DAA	Ajuste decimal de A	Ajusta suma a BCD	INH.	19	2	-	-	-	-	^	^	^	^

MANEJO DE PILA DE DATOS						REG. COD. CONDIC.							
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	S	X	H	I	N	Z	V	C
PSHA	Guardar A en la Pila	A → Stk; SP=SP−1	INH.	36	3	-	-	-	-	-	-	-	-
PSHB	Guardar B en la Pila	B → Stk; SP=SP−1	INH.	37	3	-	-	-	-	-	-	-	-
PSHX	Guardar X en la Pila	IX → Stk; SP=SP−2	INH.	3C	4	-	-	-	-	-	-	-	-
PSHY	Guardar Y en la Pila	IY → Stk; SP=SP−2	INH.	18 3C	5	-	-	-	-	-	-	-	-
PULA	Extraer A de la Pila	SP=SP+1; Stk → A	INH.	32	4	-	-	-	-	-	-	-	-
PULB	Extraer B de la Pila	SP=SP+1; Stk → B	INH.	33	4	-	-	-	-	-	-	-	-
PULX	Extraer X de la Pila	SP=SP+2; Stk → IX	INH.	38	5	-	-	-	-	-	-	-	-
PULY	Extraer Y de la Pila	SP=SP+2; Stk → IY	INH.	18 38	6	-	-	-	-	-	-	-	-

TABLA 3.5 CONJUNTO DE INSTRUCCIONES (HOJA 6 DE 7)

INTERRUPCIONES															
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	REG. COD. CONDIC. S	X	H	I	N	Z	V	C
CLI	Habilitar interrupción	0→I	INH.	0E	2	–	–	–	0	–	–	–	–
SEI	Deshabilitar interrupción	1→I	INH.	0F	2	–	–	–	1	–	–	–	–
SWI	Interrupción por software	PC→Stk IY→Stk IX→Stk A→Stk B→Stk CCR→Stk SP–9→SP	INH.	3F	14	–	–	–	1	–	–	–	–
RTI	Retorno de interrupción	Stk→CCR Stk→B Stk→A Stk→IX Stk→IY Stk→PC SP+9→SP	INH.	3B	12	^	↕	^	^	^	^	^	^
WAI	En espera de interrupción	PC→Stk IY→Stk IX→Stk A→Stk B→Stk CCR→Stk SP–9→SP	INH.	3E	14+	–	–	–	–	–	–	–	–

SUBRUTINAS															
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	REG. COD. CONDIC. S	X	H	I	N	Z	V	C
BSR (rel)	Salto a subrutina	PC→Stk SP–2→SP	REL.	8D rr	6	–	–	–	–	–	–	–	–
JSR (rel)	Salto a subrutina	PC→Stk SP–2→SP	DIR. EXTEND. IND., X IND., Y	9D dd BD hh ll AD ff 18 AD ff	5 6 6 7	–	–	–	–	–	–	–	–
RTS	Retorno de subrutina	Stk→PC SP+2→SP	INH.	39	5	–	–	–	–	–	–	–	–

OTRAS INSTRUCCIONES															
MNEMONICO	DESCRIPCION	OPERACION	MODO DE DIREC.	COD. DE OPER.	CICLOS	REG. COD. CONDIC. S	X	H	I	N	Z	V	C
NOP	No operación	No opera	INH.	01	2	–	–	–	–	–	–	–	–
STOP	Detener el reloj E	–	INH.	CF	2	–	–	–	–	–	–	–	–
TEST	Prueba (sólo en modo PRUEBA)	PC+1→PC	INH.	00	*	–	–	–	–	–	–	–	–

SIMBOLOGIA

OPERANDOS

- ii: Un byte en modo inmediato
- dd: Un byte (parte baja) en modo directo
- hh ll: Dirección de 16 bits en modo extendido
- jj kk: Dirección de 16 bits en modo inmediato
- ff: Un byte de desplazamiento en modo indexado
- mm: Mascarilla de 8 bits (indica los bits a ser afectados)
- rr: Desplazamiento de salto relativo

CODIGOS DE CONDICION

- – La bandera no cambia
- 0 La bandera es 0
- 1 La bandera es 1
- ^ La bandera es 0 ó 1, dependiendo de la operación.
- ↕ La bandera nunca puede cambiar de 0 a 1

OPERADORES

- ( ) El valor del registro se muestra dentro
- → Se transfiere hacia
- • Operación Booleana AND
- + Símbolo de suma aritmética u OR booleano
- ⊕ OR exclusiva, XOR
- * Multiplicación
- : Concatenación (une parte alta y baja de una dirección).
- – Símbolo de resta aritmética
- ¯ Símbolo de negación y complemento a 2

TABLA 3.5 CONJUNTO DE INSTRUCCIONES (HOJA 7 DE 7)

3.4 PUERTO A

El puerto A tiene tres terminales de entrada de dirección fija (PA0, PA1 y PA2), cuatro salidas de dirección fija (PA3, PA4, PA5 y PA6) y una terminal bidireccional (PA7). La dirección de la terminal PA7 está controlada por el *bit* DDRA7 (*Data Direction Register port* A *bit* 7) en el registro de control del acumulador de pulsos PACTL (1026_H) (*Pulse Accumulator Control*).

Las ocho terminales del puerto A pueden ser configuradas independientemente como entradas o salidas de propósito general, como *timer* o como funciones de acumulador de pulsos. Los datos en el puerto A son leídos o escritos a través del registro PORTA (1000_H). Los datos significativos pueden ser leídos del puerto A aún cuando las terminales estén configuradas para funciones de *timer* alterno o acumulador de pulsos. Los datos escritos al puerto A no afectan directamente las terminales del puerto A configuradas para funciones de salida de un *timer* alterno, pero el dato es almacenado en un *latch* interno de manera que, si la función alterna es deshabilitada, el último dato escrito al puerto A va a ser conducido hacia la terminal correspondiente.

La lectura de los *bits* 0, 1 y 2 es independiente de las funciones de captura de entradas temporizadas y se hace a través de un *flip-flop* R-S que actúa como *buffer*.

Las funciones compartidas del puerto A incluyen:

- Sistema *timer* principal (contador libre).
- Entradas y salidas de propósito general.
- Acumulador de pulsos

3.4.1 Sistema de *timer* principal (contador libre)

El elemento central del *timer* principal es un contador libre de 16 *bits* y un pre-escalador de 4 etapas. El contador inicia desde 0000_H tan pronto como el MCU sale del estado de *reset* y se incrementa en forma continua. Cuando alcanza su valor máximo $FFFF_H$, el contador regresa a 0000_H y acciona la bandera de desbordamiento (overflow) y continúa su conteo incremental.

Mientras el MCU está funcionando normalmente no hay forma posible para restablecer, modificar o interrumpir este contador. Todas las actividades del sistema del *timer* principal están referidas a este contador libre.

El registro doble de conteo del *timer* (TCNT: $100E_H$ y $100F_H$), debe ser leído utilizando una instrucción de lectura de doble *byte*, como LOAD D (LDD) ó LOAD X (LDX).

En este contador de 16 *bits*, el *byte* de menor orden pasa a la parte baja del registro TCNT ($100F_H$) a través de un *buffer*. Cuando este *byte* se lee mediante una instrucción de lectura de 1 *byte*, el valor obtenido es simplemente el de los 8 *bits* de menor orden del contador. Sin embargo, cuando se lee el *byte* de alto orden, se inhibe el *buffer* del registro de bajo orden de TCNT durante el siguiente ciclo de *bus*. En caso de una lectura de doble *byte* a TCNT, primero se da acceso al *byte* de alto orden y se "congela" el valor del *byte* de bajo orden en el *buffer*, el cual se lee en el siguiente ciclo de *bus*. Este procedimiento asegura que los dos *bytes* leídos correspondan uno al otro.

El pre-escalador programable permite al contador libre seleccionar una velocidad de entre cuatro posibles. Esto afecta la resolución y el rango del conteo. Para un reloj E de 2 Mhz, la resolución del contador es 500 nS y su rango es 32.77 mS. El factor del pre-escalador se selecciona con los *bits* PR1 y PR0 del registro TMSK2 (1024_H) de acuerdo tabla 3.6. El valor por omisión es 0 y si se modifica debe hacerse en los primeros 64 ciclos de reloj después del *reset*.

FACTOR	RESOLUCIÓN	RANGO	PR1	PR0
1	500 nS	32.77 mS	0	0
4	2 uS	131.10 mS	0	1
8	4 uS	262.20 mS	1	0
16	8 uS	524.30 mS	1	1

TABLA 3.6 SELECCIÓN DEL FACTOR DEL PRE-ESCALADOR

En los casos donde se requiera de períodos más largos que el rango del contador, se debe utilizar el *bit* de desbordamiento (*overflow*) TOF del registro TFLG2 (1025_H), el cual se fija en 1 lógico en cada transición de $FFFF_H$ a 0000_H del contador libre. El *bit* se puede restablecer escribiendo un 1 lógico en el mismo. Se puede también generar una interrupción cada vez que esto ocurra fijando el *bit* TOI del registro TMSK2 (1024_H), en caso de no hacerlo, se debe revisar periódicamente la bandera TOF.

3.4.2 Concepto de captura de entradas

La función de captura de entradas es un elemento fundamental de la arquitectura de *timers* del MC68HC11A1. El contador libre de 16 *bits* se utiliza para grabar el tiempo en el cual algún evento exterior ha ocurrido, se acompaña mediante un registro *latch* que almacena el contenido de este contador cuando el flanco seleccionado es detectado en la terminal de entrada elegida. El tiempo (valor del contador) en el cual el evento ha ocurrido se almacena en el registro de captura de entrada (*latch* de 16 *bits*); de esta forma, aunque el evento tome un tiempo indeterminado, el *software* puede identificar el momento en que ha ocurrido el evento.

Mediante el grabado de los tiempos de flancos sucesivos de la misma polaridad de una señal, el *software* puede determinar el período de ésta y con flancos alternos se puede medir el ancho del pulso.

Cada una de las tres funciones de captura de entradas pueden ser programadas y operadas independientemente para detectar solo flancos de subida, solo flancos de bajada, o cualquier flanco (subida o bajada). La lógica de generación de interrupción incluye una bandera de estado, la cual indica que el flanco ha sido detectado, y un *bit* de habilitación local de interrupción, el cual determina si se va a generar la interrupción.

Los registros de captura de entradas, TIC1, TIC2 y TIC3, se encuentran desde la dirección 1010_H hasta la dirección 1015_H, según la tabla 3.4. Sus contenidos no pueden alterarse mediante escrituras por *software* o *reset*, únicamente pueden realizarse lecturas.

Los *bits* IC1F, IC2F e IC3F del registro TFLG1 (1023_H) son las banderas de estado, las cuales se fijan en 1 lógico cada vez que el flanco seleccionado se detecta en la terminal correspondiente de captura de entradas. Este *bit* se regresa a 0 lógico escribiendo en este registro un 1 lógico en el *bit* correspondiente a la posición de la bandera.

Los *bits* IC1I, IC2I e IC3I del registro TMSK1 (1022_H) determinan la configuración para la atención de las funciones: por revisión periódica de bandera cuando es 0 lógico o por interrupción por *hardware* cuando es 1 lógico. En este último caso antes de abandonar la rutina de atención a interrupción, se debe desactivar la bandera de estado con el procedimiento ya descrito.

La polaridad de cada función de captura de entradas se puede programar individualmente en cada terminal mediante los *bits* de control EDGxB y EDGxA (x puede ser 1, 2 ó 3) en el registro 2 de control de *timers*, TCTL2 (1021_H). Este par de *bits* están codificados como lo muestra la siguiente tabla:

CONFIGURACIÓN	EDGxB	EDGxA
Deshabilitado	0	0
Captura sólo en flanco de subida	0	1
Captura sólo en flanco de bajada	1	0
Captura en cualquier flanco	1	1

TABLA 3.7 SELECCIÓN DEL FLANCO EN CAPTURA DE ENTRADAS

3.4.3 Concepto de comparación de salidas

La función de comparación de salidas es usada para programar una acción que debe ocurrir en un tiempo específico cuando el contador libre alcance el valor especificado. Por cada una de las

cinco funciones de comparación de salidas existe un registro de comparación de 16 *bits* y un circuito comparador dedicado de 16 *bits*. El valor en el registro de comparación se compara contra el valor del contador libre en cada ciclo de *bus*. Cuando el registro de comparación concuerda con el valor del contador se genera una salida, la cual levanta una bandera de estado y se inicia la acción automática de salida en la terminal.

Uno de los usos más prácticos de la función de comparación de salidas es el de producir un pulso de duración específica. Primero se escribe en el registro de comparación el valor correspondiente para el primer flanco del pulso. La comparación de salida se configura automáticamente a la correspondencia de salida alta o baja, dependiendo de la polaridad del pulso que se desea producir. Después que esta comparación ha ocurrido, la función se reprograma para que la terminal de salida regrese a su estado inactivo en la siguiente comparación. Un valor correspondiente al ancho del pulso se agrega al valor del registro de comparación.

Una combinación de las dos funciones es cuando se desea activar una señal de salida después de cierto número de ciclos de reloj de detectado un evento. El tiempo en la captura de entrada es usado como referencia para la función de comparación de salida.

Cada una de las comparaciones de salida está asociada con una terminal del puerto A y la acción a tomar por cada terminal es controlada independientemente.

Las salidas de comparación OC2...OC5 actúan de forma independiente. La función OC1 puede controlar con prioridad cualquier acción de las cinco terminales de salida de estas comparaciones, aún cuando éstas ya estén controladas por su respectivo *timer*.

Para las salidas OC2...OC5, dos *bits* de control en el registro TCTL1 (1020_H) controlan la acción automática que debe ocurrir en la terminal de salida cuando el contador libre alcance el valor del registro de la salida OCx.

Los *bits* de control OMx y OLx, están codificados para permitir las siguientes cuatro posibilidades:

CONFIGURACIÓN	OMx	OLx
OCx no se afecta (OC1 puede afectar)	0	0
Conmuta el estado de la terminal OCx	0	1
Fija a 1 el estado de la terminal OCx	1	0
Fija a 0 el estado de la terminal OCx	1	1

TABLA 3.8 ACCIÓN DE COMPARACIÓN DE SALIDAS

Para la función OC1, la acción automática en la terminal está controlada por los registros: mascarilla OC1M ($100C_H$) y el de datos OC1D ($100D_H$). La mascarilla OC1M determina que terminal(es) del puerto A va(n) a ser afectada(s) por OC1. El registro OC1D indica el dato a ser enviado a la(s) terminal(es) del puerto A que va(n) a ser afectada(s) cuando sea verdadera la comparación de OC1.

Cuando estas terminales se utilizan con el sistema de *timers*, no podrán utilizarse como salidas de propósito general del puerto A. Cada uno de estos 5 *bits* más significativos corresponde, *bit* a *bit*, con las terminales del puerto A, los tres restantes no se utilizan. La función OC1 puede afectar a la terminal PA7 del puerto solamente cuando esté configurada como salida, esto se hace fijando a 1 el *bit* DDRA7 del registro PACTL (1026_H).

Para cambiar el estado de una terminal, se debe alterar el registro TOCx correspondiente para verificar la comparación o utilizar el sistema de "forzamiento de salidas", por ejemplo, para dar valores iniciales.

Los registros de cada función de comparación de salidas son los registros TOC1...TOC5, ubicados en las direcciones 1016_H hasta la $101F_H$ (ver la tabla 3.4), y pueden ser leídos por instrucciones de 8 ó 16 *bits*. Es necesario realizar la escritura mediante instrucciones de 16 *bits* para inhibir la función hasta que ambos *bytes* sean modificados.

El registro TMSK1 (1022_H) contiene los *bits* OCxI para la habilitación de las interrupciones con estas funciones. Cuando alguno de estos *bits* es 1 lógico se genera una interrupción, en caso de que el *bit* sea 0 lógico, la bandera OCxF se puede revisar periódicamente para saber si se ha realizado la acción. En ambos casos, la acción en la terminal se realiza automáticamente.

El registro TFLG1 (1023_H) contiene las banderas o *bits* de estado de cada una de las funciones, OCxF, que indican cada vez que el contador libre ha alcanzado el valor de alguna comparación. Estas banderas deben regresarse a 0 lógico cuando se atienden por interrupción, antes de salir de ésta, mediante la escritura de un 1 lógico en el *bit* que le corresponda.

El forzamiento de comparación de salidas proporciona un mecanismo para cambiar el estado de las terminales sin esperar a que se verifique la función de comparación. Para utilizar este mecanismo se debe escribir un 1 lógico en el/los *bit*(s) FOCx del registro CFORC ($100B_H$) correspondiente(s) a la(s) terminal(es) a ser afectada(s). En el siguiente incremento del contador libre la terminal tomará el estado que tiene programado en la comparación. No se genera interrupción ni se indica la bandera de estado. Este procedimiento no se recomienda para la opción de "conmutación de estado", ya que ocurre un doble cambio si se fuerza y luego se verifica la comparación. El estado en 1 de los *bits* FOCx es transitorio y su lectura siempre dará un 0 lógico.

3.4.4 Entradas y salidas de propósito general

Se podrán utilizar las terminales PA3...PA6, como salidas para propósito general cuando no se habiliten como funciones de comparación de salidas. Adicionalmente, si se desea utilizar como salida a PA7 se deberá configurar como tal fijando a 1 lógico el *bit* DDRA7 del registro PACTL (1026_H).

Las terminales de entrada PA0...PA2 se podrán utilizar para propósito general aunque estén habilitadas con la función de captura de entradas.

Si se desea utilizar la terminal PA7 como entrada, se deberá configurar como tal fijando en 0 lógico (por omisión) el *bit* DDRA7 del registro PACTL (1026_H). En ambos casos, las terminales serán manejadas mediante el registro PORTA (1000_H).

3.4.5 Sistema acumulador de pulsos.

Este sistema es mucho más simple que el de *timer* principal. Está basado en un contador de 8 *bits* y puede ser configurado en los modos de "conteo de eventos externos" o de "acumulación de tiempo". A diferencia del *timer* principal, este contador de 8 *bits* sí puede leerse o escribirse en cualquier momento. Con el sistema están asociadas dos interrupciones con sus propios *bits* de control y vectores de interrupción. El *bit* 7 del puerto A (PA7/PAI/OC1), está asociado con el acumulador de pulsos.

Cuando el sistema se configura en el modo de conteo de eventos externos, el contador de 8 *bits* se incrementa cada vez que se presenta el flanco de activación en la terminal PA7/PAI.

En el modo de acumulación de tiempo, el *bit* PA7/PAI actúa como señal de habilitación para alimentar al contador de 8 *bits* con una señal de reloj de frecuencia E/64 (32 KHz si E es 2 MHz).

El contador del acumulador de pulsos es el registro PACNT (1027_H), el *bit* DDRA7 del registro PACTL (1026_H) define la dirección de la terminal. Para el caso del sistema acumulador de pulsos debe configurarse como entrada, fijándolo en 0 lógico.

PAEN (Habilitación del acumulador de pulsos):
0 = Acumulador de pulsos deshabilitado.
1 = Acumulador de pulsos habilitado.

Cuando se deshabilita el sistema acumulador de pulsos, el contador de 8 *bits* se detiene y se inhiben sus dos interrupciones, entonces las banderas permanecen en el mismo estado que tenían antes de la deshabilitación y no podrán entonces activarse por *software* si estuvieron desactivadas.

PAMOD (Selección del modo del acumulador de pulsos):

0 = Modo de conteo de eventos externos.

1 = Modo de acumulación de tiempo (conteo de E/64).

PEDGE (Selección del flanco/nivel de activación):

0 = Se seleccionan los flancos de bajada en el modo de conteo de eventos. Se selecciona el nivel alto para el modo de acumulación de tiempo.

1 = Se seleccionan los flancos de subida en el modo de conteo de eventos. Se selecciona el nivel bajo para el modo de acumulación de tiempo.

El *bit* PAOVI del registro TMSK2 (1024_H) es la bandera para habilitar la interrupción de desbordamiento del contador de 8 *bits*. Cuando está puesto en 1 lógico se genera una interrupción cada vez que el contador de 8 *bits* cambia de FF_H a 00_H. Si es 0 lógico, la acción a tomar en el caso mencionado se debe manejar mediante la inspección de la bandera de estado de desbordamiento de conteo PAOVF.

El *bit* PAOVF del registro TFLG2 (1025_H) se fija en 1 lógico automáticamente cada vez que el conteo del acumulador de pulsos cambie de FF_H a 00_H, este *bit* de estado se puede regresar a 0 lógico escribiendo en él mismo un 1 lógico cuando se ha tomado una acción.

El *bit* de estado PAIF del registro TFLG2 (1025_H) se fija automáticamente en 1 lógico cada vez que el flanco seleccionado se detecta en la terminal PA7/PAI, este *bit* de estado se puede regresar a 0 lógico mediante la escritura de un 1 lógico en él mismo. El *bit* PAII del registro TMSK2 (1024_H) permite al usuario configurar la detección de flancos del acumulador de pulsos en el modo de operación de revisión periódica o por manejo de interrupción, en ambos casos no afecta el funcionamiento de la bandera PAIF. Cuando PAII es 0, la interrupción de entrada del acumulador de pulsos se inhibe, en este caso el *software* debe tomar muestras para determinar cuando ha ocurrido el flanco. Cuando el *bit* de control PAII es 1, se genera una interrupción cada vez que

la bandera PAIF es 1. Antes de abandonar la rutina de servicio de interrupción, el *software* debe regresar a 0 lógico la bandera PAIF, escribiendo en ella misma un 1 lógico.

3.4.6 Interrupción de Tiempo Real (RTI)

La función RTI sirve para generar interrupciones mascarables requeridas por *hardware* en forma periódica o señales de referencia de tiempo. Se puede seleccionar uno de entre cuatro períodos de tiempo. La duración del período es una función del oscilador del MCU y del estado de los *bits* de control RTR1 y RTR2, los cuales se pueden modificar por *software* para realizar la selección del período.

La función RTI no tiene ninguna terminal de salida en puerto alguno, pero se ha clasificado dentro de este subcapítulo debido a que comparte el contador libre utilizado en las funciones de captura de eventos y comparación de salidas. Esta señal de reloj hace que el tiempo de expiración del período sea constante e independiente de la duración de la rutina de atención a la interrupción. Esto es, el nuevo período empieza al terminar el tiempo del anterior y no cuando se regresa a 0 lógico la bandera de la interrupción.

La función RTI contiene una bandera de estado, una de habilitación de interrupción y dos *bits* de selección del período:

La bandera RTIF del registro TFLG2 (1025_H) es un *bit* de estado que se pone en 1 lógico al final de cada período de RTI. Este *bit* se regresa a 0 lógico escribiendo sobre sí mismo un 1 lógico, lo cual debe hacerse al terminar el servicio de la función por interrupción o por revisión periódica.

El *bit* de control RTII del registro TMSK2 (1024_H) permite al usuario habilitar una interrupción al final de cada período o manejarlo por revisión periódica. Este *bit* no afecta la bandera de estado RTIF. Cuando RTII es 0 lógico, se inhibe la interrupción y el sistema debe operar por muestreo periódico por *software* de la bandera RTIF para determinar el momento en que el período ha finalizado. Cuando RTII es 1 lógico, se genera una requisición de

interrupción por *hardware* cada vez que RTIF es puesto en 1 lógico y antes de abandonar la rutina de servicio de interrupción se debe regresar RTIF a 0 lógico.

RTR0 y **RTR1** (Selección del período de tiempo):

Estos dos *bits* del registro PACTL (1026_H), determinan el período de tiempo mediante un factor de división aplicado a la señal de reloj interno E. La tabla 3.9 muestra estos períodos para el caso particular de una señal E de 2 MHz.

RTR0	RTR1	FACTOR DIVISOR	PERÍODO
0	0	2^{13}	4.1 ms
0	1	2^{14}	8.19 ms
1	0	2^{15}	16.38 ms
1	1	2^{16}	32.77 ms

TABLA 3.9 SELECCIÓN DEL PERÍODO DE RTI

3.5 ENTRADA Y SALIDA TIPO PARALELO

Todos los puertos del MC68HC11 pueden ser utilizados como entradas o salidas de propósito general, pero los puertos A, D y E presentan funciones alternas específicas en sus terminales, quedando en segundo plano la función de simples entradas o salidas. Los puertos B y C no tienen funciones alternas y su acción fundamental es el intercambio de *bytes* en forma paralela.

3.5.1 Puerto B

Es un puerto de ocho *bits* de salidas de dirección fija de propósito general. Cuando el MCU está operando en modo expandido el puerto B se utiliza como salida de direcciones de alto orden (desde ADDR8 hasta ADDR15). En modo de un solo *chip*, el puerto B es usado para salidas de propósito general o salidas con estrobo.

La escritura se realiza a través del registro PORTB, haciendo que los datos sean almacenados en un *latch* y conducidos a las terminales del puerto B. La lectura del registro PORTB transfiere el último dato escrito hacia el puerto B. Cuando el subsistema de protocolo de entrada/salida está operando en modo de estrobo sencillo, la escritura al registro PORTB automáticamente causa un pulso en la terminal de salida STRB. El modo de estrobo sencillo se selecciona haciendo el *bit* de control de protocolo HDNDS (*handshake*) igual a 0 lógico en el registro de control E/S paralelo (PIOC).

Cuando el MCU opera en modo expandido, las lecturas y escrituras a la dirección del puerto B, son tratadas como accesos externos para permitir a las funciones del puerto B ser emuladas con lógica externa; la unidad de reemplazo de puertos MC68HC24 duplica las funciones de entrada y salida de propósito general de los puertos B y C y las terminales STRA y STRB. La MC68HC24 se conecta al *bus* multiplexado de datos y direcciones del MC68HC11.

3.5.2 Puerto C

Es un puerto de 8 *bits* bidireccionales de entrada/salida de propósito general. La dirección del flujo de datos en el puerto C se controla individualmente por su correspondiente *bit* en el registro de control de dirección para el puerto C (DDCR).

Adicionalmente a las funciones normales de entrada/salida al puerto C, existe un registro *latch* paralelo de ocho *bits* que captura los datos cuando se detecta el flanco de activación en la terminal de entrada STRA, la lectura del registro PORTCL tranfiere el contenido de este *latch* del puerto C, mientras que la lectura del registro PORTC transfiere el dato actual de las terminales del puerto C.

La escritura a los registros PORTC o PORTCL provoca que los datos escritos sean conducidos a las terminales de salida del puerto C, sin embargo, la escritura a PORTCL acciona la secuencia de protocolo mientras que la escritura a PORTC no lo hace. La escritura hacia las terminales del puerto C que no están

configuradas como salidas, no causan que los datos sean conducidos a esas terminales pero los datos son almacenados en el *latch* interno. De este modo si estas terminales después se convierten en salidas, el último dato escrito va a ser conducido a las terminales del puerto C. El usuario puede programar cualquier combinación de entradas o salidas en este puerto y modificarla cuando así lo requiera.

En el caso de lectura los datos se pueden almacenar en el *latch* cuando el periférico externo habilite la entrada STRA, así aunque el MCU esté realizando otras operaciones, los datos se encuentran ya dentro del puerto C y podrán ser procesados cuando realice una lectura al registro PORTCL.

El puerto C puede ser configurado para la operación OR alambrada fijando en 1 lógico el *bit* de control de modo (CWOM) del registro PIOC. Este procedimiento deshabilita los *drivers* de *pull-up* de canal p de las terminales de salida del puerto C permitiendo conectarse directamente entre ellas o a otras terminales de tipo drenaje abierto. En esta configuración no hay peligro de conflictos destructivos si dos *drivers* tratan de habilitar el nodo al mismo tiempo. Como cualquier línea de drenaje abierto se requiere un resistor de *pull-up*.

Cuando el sistema de protocolos está configurado como protocolo completo (*full handshake*), el puerto C se utiliza para la entrada o salida de datos. La terminal STRA causa que los datos en el puerto C sean capturados cuando se detecta el flanco de activación.

3.5.3 Sistema de protocolos de entrada/salida

El sistema de protocolos de E/S involucra a los puertos B y C, la terminal de entrada STRA, la terminal de salida STRB y al registro PIOC (1002_H).

El sistema de protocolos E/S tiene tres modos principales de operación. El primero de ellos (por omisión) es el de "estrobo sencillo", el cual utiliza al puerto B como salida con estrobo y al puerto C como *latch* de entradas. El segundo modo de operación es el

de "estrobo completo de entradas" (*full input handshake*) y el tercer modo es el de "estrobo completo de salidas" (*full output handshake*). En los modos de operación de estrobo completo (*full handshake*) no se involucra al puerto B, de forma que continúa siendo un puerto de salida de propósito general.

Modo estrobo sencillo: Este modo se selecciona fijando el *bit* HNDS del registro PIOC (1002_H) a un valor 0 lógico. Después del *reset*, HNDS es forzado a 0 lógico, que es el modo de operación por omisión del subsistema de protocolos. En este modo los *bits* de control OIN y PLS del registro PIOC no tienen efecto.

En el modo de estrobo sencillo, el puerto B se utiliza como puerto de salida con estrobo en conjunto con la terminal de salida STRB. El puerto C se usa simultáneamente como *latch* de entradas en conjunto con la terminal de entrada STRA. La función de salida con estrobo en el puerto B es independiente de la función del *latch* de entradas en el puerto C.

Como respuesta a la escritura en el registro PORTB, el dato es actualizado en las terminales del puerto B y se genera un pulso en la terminal STRB de dos ciclos E de duración. El *bit* de control INVB del registro PIOC permite escoger la polaridad de los pulsos de la terminal STRB; con INVB = 1 lógico se seleccionan los pulsos activados en nivel alto.

Los datos de entrada en el puerto C, requieren un cierto tiempo antes y después del flanco de activación en la entrada STRA para poder ser capturados. El *bit* de control STAF del registro PIOC indica la entrada de un nuevo dato al *latch* del puerto C. El flanco de activación se selecciona con el *bit* EGA del registro PIOC.

Modo Protocolo completo de entradas: Este modo se selecciona con el *bit* HNDS = 1 lógico y OIN = 0 lógico. En este modo el dispositivo externo presenta los datos en las terminales del puerto C y sólo serán transferidos cuando la terminal STRB está verificada, indicando una condición de "listo para recibir".

El dispositivo externo presenta entonces un flanco de activación en la terminal STRA con lo cual los datos se almacenan en los *latches* de entrada del puerto C y pone en 1 lógico la bandera STAF (opcionalmente puede generar una interrupción) y desactiva la terminal STRB para evitar que el dispositivo externo pretenda transferir nuevos datos por el momento. Leyendo el dato del registro PORTCL (1005_H) la bandera STAF se regresa a 0 lógico y nuevamente se verifica la terminal STRB. El puerto está listo para un nuevo dato.

El *bit* de control INVB selecciona la polaridad de activación de la terminal STRB y EGA determina el tipo de flanco de acticación (ascendente o descendente) de la terminal STRA.

Este modo presenta dos variantes: el modo "enclavado" (*interlocked*) que es el que se ha descrito anteriormente y el modo "por pulso" en el cual la verificación de la terminal STRB sólo dura dos ciclo de reloj E. Esta variante se selecciona con el *bit* de control PLS.

Modo Protocolo completo de salidas: Este modo se selecciona con el *bit* HNDS y OIN = 1 lógico. En este modo la terminal STRB indica que el puerto C está listo para un sistema externo y la terminal de entrada STRA, que es activada por el sistema externo, indica la aceptación del dato por éste.

En una operación de salida con protocolo completo, el dato se lleva a las terminales del puerto C mediante la escritura al registro PORTCL (1005_H), que automáticamente provoca que la terminal STRB se verifique. El sistema externo reconoce esta señal como una indicación de disponibilidad y después de aceptar el dato del puerto C, el sistema externo provoca un pulso en la terminal de entrada STRA como acuse de recepción del dato. El flanco de activación de la terminal STRA provoca que la terminal STRB se deshabilite y la bandera STAF se ponga en 1 lógico, que indica que el puerto está listo para la siguiente transferencia mediante la escritura al registro PORTCL.

El protocolo completo de salidas puede manejar las terminales del puerto C como salidas de tres estados; el estado de alta impedancia se selecciona por *bits*, con un 0 lógico en el *bit* correspondiente a cada terminal del registro DDRC (1007_H), que en este modo, no selecciona la dirección de la terminal, ya que se asume que son salidas. La habilitación de salidas en estado de alta impedancia se realiza con la terminal STRA, en este caso el *bit* EGA selecciona el nivel de activación. Cuando EGA es 0 lógico el nivel de activación de salidas será alto.

El registro PIOC (1002_H) es utilizado para configurar y controlar el sistema de protocolo y sus *bits* se describen a continuación con mayor detalle:

STAF (bandera de estrobo A):

Esta bandera siempre es 1 lógico al presentarse el flanco seleccionado en la terminal STRA, evento que puede ser asíncrono al reloj interno E. Para regresar a 0 lógico este *bit*, se debe leer o escribir el registro PORTCL dependiendo del protocolo utilizado.

STAI (Habilitar interrupción con STRA):

Cuando este *bit* de control es 1 lógico, habilita una interrupción cada vez que el *bit* STAF cambia a 1 lógico.

CWON (Puerto C en modo OR alambrada):

Este *bit* se utiliza para configurar las ocho terminales del puerto C en modo OR alambrada cuando actúan como salidas. Cuando CWON es 1 lógico, todas las terminales del puerto C actúan como salidas de drenaje abierto y requieren un resistor externo *pull-up*.

HNDS (Protocolo sencillo):

0 = Se selecciona el modo de estrobo sencillo.

1 = Se selecciona el modo de protocolo completo de entradas o salidas.

OIN (selección de protocolo completo de entradas o salidas):

Este *bit* no tiene efecto a menos que HNDS sea 1 lógico.

0 = Selecciona el modo de protocolo completo de entradas.

1 = Selecciona el modo de protocolo completo de salidas.

PLS (Selección del modo de la terminal STRB):

Este *bit* funciona cuando se selecciona el modo de protocolo completo de entradas para elegir una de las dos variantes: enclavamiento o por pulso.

0 = Selecciona la variante de enclavamiento, donde la terminal STRB permanece verificada hasta recibir el flanco de activación en la terminal STRA.

1 = Selecciona la variante por pulso, donde la terminal STRB permanece verificada por un lapso de 2 ciclos de reloj E.

EGA (Selección del flanco del estrobo A):

Este *bit* de control selecciona la polaridad del flanco a ser reconocido como "activo" en la terminal de entrada STRA. Si esta terminal es 0 lógico solamente se detectan los flancos descendentes y si es 1 lógico, los ascendentes. Para el modo de alta impedancia, si EGA es 0 lógico el nivel de activación de salidas será alto, y si EGA es 1 lógico, será bajo.

INVB (Invertir estrobo B):

Este *bit* selecciona la salida Q o $\overline{Q}$ del *flip-flop* acoplado a la terminal de salida STRB. Si INVB es 0 lógico, esta terminal es verificada a nivel bajo, si INVB es 1 lógico se activa en nivel lógico alto.

3.6 ENTRADA Y SALIDA SERIE (PUERTO D)

El puerto D sólo tiene seis terminales y comparte dos subsistemas de comunicación con entradas/salidas bidireccionales de propósito general.

Los dos subsistemas de comunicación tipo serie del MC68HC11 son independientes entre sí: Una interfase asíncrona SCI, que se utiliza generalmente para la comunicación con monitores, teclados o microcomputadoras; emplea las terminales PD0 y PD1 que se tornan unidireccionales cuando se habilita esta función. El otro, SPI, es un subsistema síncrono de alta velocidad de comunicación, usado con frecuencia con otros microcontroladores o periféricos que se encuentren en la misma tarjeta. Utiliza las terminales PD2 a PD5 y sólo esta última se torna unidireccional.

3.6.1 Comunicación asíncrona

La interfase de comunicación tipo serie (SCI) es un receptor transmisor universal asíncrono (UART) *Full-Duplex*. Tiene un formato estándar NRZ (no regreso a cero) que requiere de las siguientes condiciones:

1. Una línea no activada, en estado alto antes de la transmisión o recepción de un mensaje.
2. Un *bit* de arranque, 0 lógico, transmitido o recibido, que indica el inicio de cada carácter.
3. 8 ó 9 *bits* de datos que son transmitidos y recibidos iniciando con el *bit* menos significativo (LSB).
4. Un *bit* de paro, 1 lógico, usado para indicar el final de la trama.
5. Una señal de corte o fin, definida como 0 lógico en transmisión y recepción para múltiples tramas.

La relación de *bauds* (velocidad de transmisión) derivada de la frecuencia del oscilador interno del MCU se puede selecionar de entre varias relaciones de *bauds*. El transmisor y el receptor SCI son independientes, pero utilizan el mismo formato de datos y la misma relación de *bauds*. Debido a que la SCI utiliza los niveles lógicos del MCU, el usuario debe proporcionar los circuitos convenientes para convertir estos niveles a los de las interfases RS-232 o RS-422.

El receptor SCI incluye un número de características avanzadas para asegurar la alta confiabilidad en la recepción de datos y para asistir al desarrollo de redes de comunicación eficientes. El MC68HC11 resincroniza el reloj del receptor en cada transición de 1 a 0 lógico en el flujo de *bits*, en vez de hacerlo solamente con el *bit* de arranque; un circuito de recuperación de datos distingue datos válidos del ruido, para tal fin, se toman selectivamente cerca de la mitad de cada *bit*, tres muestras en la terminal de entrada para detectar el dato recibido, un circuito de comparación determina el valor y la integridad de cada *bit*. Aún si el ruido obliga a una toma errónea, el *bit* será recibido correctamente.

En los sistemas con múltiples receptores, se puede reducir el trabajo de identificación del receptor mediante un modo temporal de espera proporcionado por la función "*Wake up*". Con esta lógica, el *software* de cada receptor evalúa el primer carácter del mensaje al recibirlo, y automáticamente se "despierta". El receptor se pone en modo *Wake up* mediante la escritura de un 1 lógico en el *bit* RWU del registro SCCR2 y mientras todas las banderas relacionadas con la recepción (RDRF, IDLE, OR, NF y FE) se inhiben y no pueden ponerse en 1 lógico. Cuando RWU se pone en 1 lógico, es regresado automáticamente a 0 lógico cuando llega un nuevo mensaje. Aunque también puede hacerse por *software*.

El transmisor del SCI incluye un registro de datos en paralelo (SCDR) y un registro de corrimiento tipo serie, el cual sólo puede ser escrito a través del registro SCDR. Esta operación de doble registro permite que un carácter pueda ser desplazado hacia afuera en forma serie mientras otro carácter espera en el registro SCDR para ser transferido en el registro de corrimiento serie. La salida del registro de corrimiento serie es aplicada a la terminal TxD mientras la transmisión está en progreso o el *bit* de habilitación de transmisión TE del registro 2 de control de comunicación SCCR2 ($102D_H$) está en 1 lógico.

Durante el proceso de recepción otro registro de corrimiento serie recibe los datos y los transfiere al registro paralelo de recepción SCDR como una palabra completa. Esta operación de doble *buffer* permite a un carácter ser desplazado en forma serie mientras otro carácter ya se encuentra en el registro SCDR.

El registro SCDR está compuesto por dos *buffers*: TDR es un *buffer* de transmisión de datos de sólo escritura y RDR es un *buffer* de recepción de sólo lectura. Cuando el *software* lee del registro SCDR, se accesa el *buffer* RDR y cuando el *software* escribe, es el *buffer* TDR el que se acciona.

El transmisor SCI puede producir colas de caracteres de 1 ó 0 lógicos. Cuando se habilita el receptor o el transmisor del SCI, la lógica de éste toma control de los *buffers* de las terminales. Las terminales RxD y TxD son sobrepuestas a PD0 y PD1 y actúan como

entrada y salida respectivamente, no importando la dirección que tengan como puerto D de entrada/salida general. En este caso el registro DDRD determinará la dirección de las terminales cuando se deshabilite el SCI.

Inicialmente el subsistema SCI está configurado y controlado por cinco registros (BAUD, SCCR1, SCCR2, SCSR y SCDR); además el Registro PORTD, el registro de dirección de datos del puerto D (DDRD) y el *bit* de modo de OR alambrada del puerto D del registro de control SPCR que están ordinariamente relacionados con el subsistema SCI. Este último tiene efecto en las terminales para todas las funciones SCI, SPI o entradas/salidas de propósito general.

REGISTRO BAUD ($102B_H$):

El registro de control BAUD, se usa para selecionar la relación de *bauds* para las operaciones del SCI y contiene dos *bits* de control para prueba de manufactura. El SCI divide la frecuencia del reloj E por 16 para alimentar al UART mediante un factor de división. La relación de *bauds*, se obtiene entonces, seleccionando este factor mediante los *bits* SCP0, SCP1, SCR0, SCR1 y SCR2. Los primeros no ofrecen un factor de división binario, incluso dividen con factores primos como se observa en la tabla 3.10, en la que se resaltan las relaciones de *bauds* más comunes.

TCLR (Limpiar la cadena de *timers* de la relación de *bauds*):

Este *bit* sólo tiene efecto en los modos de operación de prueba y arranque especial y su finalidad son las pruebas de manufactura, la función es restablecer la cadena de contadores/divisores que afectan el valor del factor de la relación de *bauds*.

RCKB (Prueba del reloj de la relación de *bauds*)

Este *bit* sólo tiene efecto en los modos de operación de prueba y arranque especial y su finalidad son las pruebas de manufactura, la función es conducir en la terminal TxD una señal de (16 x relación de *bauds*) XOR (relación de *bauds*).

SCP1	SCP0	SCR2	SCR1	SCR0	CRISTAL 8MHz *BAUDS*
0	0	0	0	0	125 000
0	0	0	0	1	62 500
0	0	0	1	0	31 250
0	0	0	1	1	15 625
0	0	1	0	0	7 812.5
0	0	1	0	1	3 906
0	0	1	1	0	1 953
0	0	1	1	1	977
0	1	0	0	0	41 666
0	1	0	0	1	20 833
0	1	0	1	0	10 417
0	1	0	1	1	5 208
0	1	1	0	0	2 604
0	1	1	0	1	1 302
0	1	1	1	0	651
0	1	1	1	1	326
1	0	0	0	0	31 250
1	0	0	0	1	15 625
1	0	0	1	0	7 812.5
1	0	0	1	1	3 906
1	0	1	0	0	1 953
1	0	1	0	1	977
1	0	1	1	0	488
1	0	1	1	1	244
1	1	0	0	0	*9 600(+0.16%)*
1	1	0	0	1	*4 800*
1	1	0	1	0	*2 400*
1	1	0	1	1	*1 200*
1	1	1	0	0	*600*
1	1	1	0	1	*300*
1	1	1	1	0	*150*
1	1	1	1	1	*75*

TABLA 3.10 SELECCIÓN DE LA RELACIÓN DE *BAUDS*

REGISTRO SCCR1 ($102C_H$):

El registro de control SCCR1 incluye a tres *bits* asociados con el formato especial de 9 *bits* de datos y un *bit* para seleccionar uno de los dos métodos *Wake up* del receptor.

R8 (*Bit* 8 de recepción de datos):
Cuando el SCI está configurado para 9 *bits* de datos, este *bit* actúa como *bit* 8 ó MSB, los restantes *bits* (del 0 al 7) se podrán leer del registro SCDR (*buffer* RDR de lectura).
T8 (*Bit* 8 en transmisión de datos):
Cuando el SCI está configurado para 9 *bits* de datos, este *bit* actúa como *bit* 8 ó MSB, cuando los 8 *bits* de bajo orden han sido transmitidos desde el registro SCDR, este *bit* se transfiere a la novena posición del corrimiento.
M (Formato del carácter):
0 = Un *bit* de arranque, ocho *bits* de datos, un *bit* de paro.
1 = Un *bit* de arranque, nueve *bits* de datos, un *bit* de paro. El *bit* M controla la longitud del carácter para el transmisor y el receptor. El noveno *bit* es frecuentemente usado como segundo *bit* de paro o como *bit* de paridad en cualquiera de sus formas (espacio, marca, par o impar).
WAKE (Selección del método *Wake up*):
0 = Línea no activa; La detección de al menos un carácter hace que el receptor "despierte".
1 = Marca de dirección; Un 1 lógico en el *bit* MSB (octavo o noveno dependiendo del formato) provoca que el receptor "despierte".

REGISTRO SCCR2 ($102D_H$):
El registro SCCR2 contiene los principales controles del SCI. Los cuatro *bits* de alto orden (TIE, TCIE, RIE y ILIE) son controles de habilitación de interrupciones locales, que determinan si las banderas de estado del SCI van a ser revisadas periódicamente o se generará una interrupción. Los *bits* TE y RE son respectivamente controles para la habilitación de transmisor y receptor. El *bit* RWU permite al *software* poner al receptor en modo *Wake Up*, y se pone en 0 lógico cuando llega un mensaje. El *bit* SBK permite al *software* generar un mensaje *break* (todos a 0 lógico) en la terminal TxD.

TIE (Habilitar Interrupción del Transmisor):

0 = Interrupción de la bandera TDRE (Registro de datos del transmisor está vacío) deshabilitada.

1 = Se provoca una interrupción cada vez que TDRE es 1 lógico.

TCIE (Habilitar interrupción de transmisión completa):

0 = Interrupción de la bandera TC (Transmisión completa) deshabilitada.

1 = Se genera una interrupción con la bandera TC.

RIE (Habilitar interrupción de recepción):

0 = Deshabilita la interrupción de las banderas RDRF (Registro de datos del receptor lleno) y OR (Error por no lectura).

1 = La interrupción SCI se genera cuando las banderas RDRF u OR se ponen en 1 lógico.

ILIE (Habilitar interrupción de detección de línea no activa):

0 = Deshabilita la interrupción señalada por la bandera IDLE (Detector de línea no activa).

1 = Se genera la interrupción SCI cada vez que la bandera IDLE es puesta en 1 lógico.

TE (Habilitar transmisor):

0 = Deshabilita el transmisor SCI.

1 = Habilita el transmisor SCI.

Cuando este *bit* es puesto en 0 lógico, el transmisor conserva el control de la terminal TxD hasta que cualquier carácter en progreso haya terminado de enviarse. Cuando se escribe con 1 lógico, el transmisor envía un carácter de preámbulo que consiste de 10 u 11 *bits* en 1 lógico. Este mecanismo es usado para agregar un carácter de línea no activa como separación de mensajes.

RE (Habilitar receptor):

0 = Receptor SCI deshabilitado.

1 = Receptor SCI habilitado.

Mientras el receptor SCI está deshabilitado, las banderas RDRF, IDLE, OR, NF y FE no podrán ser 1 lógico.

RWU (Habilitar función *Wake up* del receptor):

0 = Operación de recepción normal.

1 = Habilitar la función de *Wake up*, que pone al receptor SCI en estado de espera. Las interrupciones relativas a la recepción se inhiben hasta que se encuentra la condición de *hardware* que haga "despertar" al receptor.

SBK (Enviar caracteres todos 0 lógico o *break*):

0 = Operación normal del transmisor.

1 = Envía continuamente caracteres a 0 lógico (*break*).

Un carácter de corte (*break*) es una señal en 0 lógico durante el tiempo que éste dura; no tiene *bits* de arranque, paro o paridad. Al menos un carácter de señal de corte se envía, mientras este *bit* es 1 lógico. Un receptor que recibe esta señal, indica un error de trama (bandera FE) debido a que se espera un *bit* de paro en nivel alto.

REGISTRO SCSR ($102E_H$):

En este registro se localizan los siete *bits* de estado relacionados con el sistema: dos banderas del transmisor y cinco relacionadas al receptor. Algunas opcionalmente generan una requisición de interrupción y otros simplemente indican el error en la recepción de caracteres. Estos *bits* de estado son automáticamente 1 lógico cuando encuentran la condición que indican y así permanecen hasta completar su secuencia de puesta en 0 lógico. El transmisor genera banderas para TDRE (Registro de datos de transmisor vacío) y TC (Transmisión Completa). El receptor genera banderas para RDRF (Registro de datos del receptor lleno), OR (Error de caracter no leído), IDLE (Detector de línea no activa), NF (Bandera de ruido en la línea) y FE (Error de formato). En adición a la bandera de estado de Registro de Transmisor Vacío, TDRE, el SCI también proporciona una bandera de Transmisión Completa, TC, que puede ser utilizada en aplicaciones, por ejemplo con algún modem.

TDRE (Registro de datos del transmisor vacío):

0 = No está vacío; un carácter que fue previamente enviado al registro SCDR aún no ha sido transferido al registro de corrimiento en serie.

1 = Un nuevo carácter puede ser escrito en el registro SCDR.

En condiciones normales de operación, debe ser revisado antes de enviar un nuevo carácter para verificar si el registro SCDR puede aceptar un nuevo dato. Cuando se lee de este registro (SCSR) y seguida de la escritura en el registro SCDR esta bandera regresa a 0 lógico.

TC (Transmisión completa):

0 = Indica que el transmisor está ocupado enviando un carácter, un preámbulo o una señal de corte.

1 = El transmisor ha completado el envío de algún carácter, preámbulo o una señal de corte y está en estado inactivo.

Este *bit* es útil en sistemas donde SCI está manejando un modem. El *bit* TDRE solamente indica que el último carácter ha sido transferido al registro tipo serie, pero no cuando se ha completado la transmisión; el *bit* TC sí lo hace.

RDRF (Registro de recepción de datos lleno):

0 = Vacío; no hay nuevo carácter en el registro SCDR.

1 = Un carácter ha sido recibido y transferido desde el registro de corrimiento serie al registro SCDR.

El *software* debe revisar periódicamente esta bandera para la recepción de datos, y también las banderas OR, NF y FE para asegurar una recepción libre de errores. Para regresar esta bandera a 0 lógico, se debe leer el registro SCDR. De la misma forma las demás banderas mencionadas regresarán a 0 lógico, ya que se levantarían al mismo tiempo para el mismo carácter.

IDLE (Detector de línea no ocupada):

0 = La línea de recepción RxD está activa.

1 = La línea de recepción está inactiva.

La condición no activa se define cuando la línea RxD está en 1 lógico por lo menos el tiempo de transmisión de un carácter completo.

OR (Detector de no lectura -dato perdido-):

0 = No hay error de dato perdido.

1 = Se indica que otro carácter ha sido recibido en el registro serie y está listo para ser transferido al registro SCDR, pero el carácter previamente recibido aún no ha sido leído por la CPU.

El/los carácter(es) que llegue(n) cuando el anterior no ha sido leído, se perderá(n). Esta bandera regresa a 0 lógico con la lectura al registro SCDR.

NF (Bandera de ruido en recepción):

0 = No se detectó ruido en la recepción del último carácter.

1 = La lógica de recuperación de datos ha detectado ruido en la recepción del carácter. Esta bandera es puesta en 1 lógico (si fuera el caso) al mismo tiempo que RDRF (Registro de datos lleno), en caso de que se perciba ruido en cualquier *bit* de la trama, ya sea en el *bit* de arranque, de datos o paro.

FE (Error de trama):

0 = No se detecta error en la trama.

1 = Un error de trama ha sido detectado en el carácter que se recibió.

La recepción tipo serie asíncrona requiere que el receptor "alinee" apropiadamente la trama del carácter. Este alineamiento se logra mediante la búsqueda del flanco descendente del *bit* de arranque; y la alineación se verifica buscando el *bit* de paro esperado (nivel alto) del carácter. El error de trama ocurre si se detecta un nivel bajo en el lugar donde se espera el *bit* de paro.

3.6.2 Comunicación síncrona

La interfase periférica de comunicación serie, SPI, es esencialmente utilizada para comunicar al MCU con dispositivos periféricos, éstos pueden ser tan simples como un registro de corrimiento TTL o tan complejos como un *driver* para pantalla de cristal líquido o un convertidor analógico a digital.

El sistema de comunicación síncrona puede ser configurado como dispositivo maestro o esclavo, y las velocidades de transferencia de datos recomendables son hasta 1 Mbit/seg. como maestro y hasta 2 Mbit/seg. como esclavo. La SPI es capaz de comunicarse en sistemas con múltiples maestros.

La lógica de control de reloj permite la selección de la polaridad y elegir entre dos protocolos fundamentales de reloj para ser compatible con la mayoría de los dispositivos de este tipo de comunicación. Cuando la SPI está configurada como maestro, se puede elegir de entre cuatro relaciones de transferencia de datos para la señal de reloj. Cuando es esclavo no tiene sentido esta selección, debido a que la SPI recibe esta señal de su maestro y la utiliza para recuperar datos en la recepción o definir su velocidad de transmisión.

Para soportar la comunicación entre procesadores se incluye una lógica de detección de errores. Un detector de colisión de escritura indica cuando se hace un intento de escritura de datos al registro serial mientras una transferencia está en progreso. Un detector de falla para el modo de múltiples maestros deshabilita automáticamente los *drivers* de salida si más de un MCU intenta simultáneamente tener acceso al *bus* de transmisión.

Durante una transferencia SPI, los datos son transmitidos y recibidos en forma serial mediante registros de corrimiento. Una línea de reloj sincroniza los desplazamientos y las muestras de la información en las dos líneas de datos tipo serie. Una línea de selección de esclavo permite la selección individual del dispositivo esclavo SPI; los dispositivos que no sean seleccionados no interfieren en las actividades de *bus* del SPI.

El *software* puede seleccionar de entre cuatro combinaciones de señal de reloj serial (SCK) de fase y polaridad utilizando dos *bits* del registro de control SPI, SPCR (1028_H). La polaridad del reloj se especifica mediante el *bit* de control CPOL, el cual selecciona la señal de reloj como activa en estado alto o activa en estado bajo y no tiene efecto significativo en el formato de transferencia. El *bit* de control SPHA selecciona uno de los dos

diferentes formatos de transferencia y la fase de la señal de reloj. La fase de reloj y polaridad deben ser idénticas para el dispositivo SPI maestro y el dispositivo esclavo.

Para ambos formatos de transferencia, la transmisión inicia con el *bit* más significativo, y es un *bit* en cada ciclo de reloj de la señal SCK. Las terminales MOSI (salida del maestro/entrada del esclavo) están interconectadas entre los dispositivos maestro y esclavo. De igual forma se conectan las terminales MISO (entrada del maestro/salida del esclavo). La línea $\overline{SS}$ es la entrada de selección de esclavo en este tipo de dispositivos; en el maestro, esta terminal se asume que debe estar inactiva (en estado alto) o debe reconfigurarse como salida de propósito general no afectando la SPI.

Cuando CPHA es 0 lógico, la línea $\overline{SS}$ debe deshabilitarse y verificarse nuevamente entre cada *byte* sucesivo en serie. También, si el esclavo escribe en el registro de datos de la SPI mientras $\overline{SS}$ está activada en nivel bajo, resulta un error de colisión de escritura.

Cuando CPHA es 1 lógico, la línea $\overline{SS}$ puede permanecer activada en nivel bajo entre transferencias sucesivas (puede estar fija a nivel bajo todo el tiempo). Este formato es algunas veces preferido en sistemas que tienen un solo maestro fijo y un solo esclavo interconectados mediante las terminales MOSI.

Cuando ocurre una transferencia de la SPI, un carácter de 8 *bits* se desplaza hacia la terminal de salida mientras otro carácter de 8 *bits* es simultáneamente desplazado hacia adentro en una segunda terminal de datos. El registro de datos SPDR ($102A_H$) tiene simple *buffer* en la dirección de transmisión y doble *buffer* en la dirección de recepción.

Hay cuatro terminales de entrada/salida asociadas con la transferencia de la SPI: la terminal SCK, la activación de esclavo $\overline{SS}$ las líneas de datos MOSI y MISO. Cuando el sistema SPI está deshabilitado estas cuatro terminales están configuradas como entradas/salidas de propósito general, y su dirección original está

controlada por el *bit* de control de dirección correspondiente para cada terminal.

Cuando el sistema SPI está habilitado, los *bits* de control de dirección aún tienen influencia en la dirección de datos en la terminal. Si se espera que una terminal sea entrada, esta terminal lo será a pesar del estado del *bit* de control de dirección. Si se espera que una terminal sea salida, lo será solamente si el *bit* de control de dirección está en 1 lógico. Cuando la SPI está configurada como maestro, la terminal PD5/$\overline{SS}$ es un caso especial.

La terminal SCK es una salida cuando la SPI se configura como maestro y entrada cuando la SPI es esclavo.

Cuando el maestro inicia una transferencia, se generan automáticamente ocho ciclos de reloj en la terminal SCK, derivados del *bus* interno E.

Cuando la SPI es configurada como esclavo, la terminal SCK es una entrada y la señal del reloj del maestro sincroniza la transferencia de datos hacia los dispositivos esclavos, los cuales ignoran la señal SCK a menos que sean seleccionados con la terminal $\overline{SS}$ en nivel bajo.

En ambos tipos de dispositivos los datos son desplazados en un flanco ascendente o descendente de la señal SCK y son muestreados en el flanco opuesto cuando el dato ya está estable.

La terminal $\overline{SS}$ funciona de forma diferente para los dispositivos maestros, en los cuales puede ser una entrada de detección de errores para la SPI o una salida de propósito general. Cuando el *bit* de control DDRD5 es 1 lógico, la terminal PD5/$\overline{SS}$ actúa como salida de propósito general, cuando DDRD5 es 0 lógico, la terminal $\overline{SS}$ actúa como entrada de detección de errores, la cual debe permanecer en nivel alto. Si esta terminal se va a nivel bajo indica que algún otro dispositivo en el *bus* SPI está intentando ser el maestro.

Las terminales del puerto D, incluyendo las cuatro de la SPI, pueden ser configuradas para funcionar como *drivers* de drenaje abierto. El *bit* de control de modo OR alambrada (DWOM) del puerto D es usado para habilitar esta opción y se requiere un resistor

pull-up para cada terminal del puerto D. En sistemas de múltipl maestros, esta opción proporciona una protección extra cont colisiones si más de un dispositivo SPI trata simultáneamente tomar la misma línea de *bus*.

Los registros de acceso por *software* utilizados pa configurar y operar el sistema SPI son: el registro de control SP (1028_H), el registro de estado SPSR (1029_H) y el registro de dat SPDR (1024_H). Debido a que el registro de control de dirección d puerto D (DDRD) tiene influencia en las actividades de la SPI, describirá brevemente.

DDRD: Registro de control de dirección de datos del puerto (1029_H).

Este registro es usado para controlar la dirección original las terminales del puerto D, los *bits* 5, 4, 3 y 2 del puer D son usados por el sistema SPI cuando éste se habilita.

DDRD5 (Control de dirección para el *bit* 5/$\overline{SS}$):

Cuando el sistema SPI se habilita como esclavo (SPE=1 MSTR=0) la terminal PD5/$\overline{SS}$ es la entrada de selección esclavo, sin importar el valor de DDRD5. Cuando el sistema S se habilita como maestro (SPE=1 y MSTR=0), la función de terminal PD5/$\overline{SS}$ depende del valor de DDRD5.

0 = La terminal $\overline{SS}$ es usada como entrada para detect errores. Un nivel bajo en esta terminal indica que alg otro dispositivo, en un sistema de múltiples maestros, llegado a ser el maestro y está tratando de seleccion este MCU como un esclavo.

1 = La terminal PD5/$\overline{SS}$ actúa como salida de propósito gener sin afectar al sistema SPI.

DDRD4 (Control de dirección para el *bit* 4/SCK):

Cuando el sistema SPI está habilitado como esclavo, terminal PD4/SCK actúa como entrada de reloj serial SPI, s importar el estado de DDRD4. Cuando el sistema SPI es habilitado como maestro, el *bit* DDRD4 debe estar en 1 lógi para habilitar la salida de reloj SCK.

DDRD3 (Control de dirección para el *bit* 3/MOSI):
Cuando el sistema SPI está habilitado como esclavo, la terminal PD3/MOSI actúa como entrada serial de datos, sin importar el estado de DDRD3. Cuando el sistema SPI está configurado como maestro, el *bit* DDRD3 debe ser puesto en 1 lógico para habilitar la salida serial de datos del maestro.
DDRD2 (Control de dirección para el *bit* 2/MISO):
Cuando el sistema SPI está configurado como esclavo, el *bit* DDRD2 debe ser puesto en 1 lógico para habilitar la salida serial de datos del esclavo. Cuando el sistema SPI está habilitado como maestro, la terminal PD2/MISO actúa como entrada serial de datos, sin importar el estado de DDRD2.

SPCR Registro de control de SPI (1028_H):
Este registro es utilizado para configurar el sistema SPI y puede ser leído o escrito en cualquier tiempo.
SPIE (Habilitación de interrupción SPI):
0 = La interrupción SPI se deshabilita y se deben realizar periódicamente las banderas SPIF y MODF.
1 = La interrupción SPI se habilita y es requerida si SPIF o MODF son 1 lógico (y el *bit* I del registro CCR es 0 lógico).
SPE (Habilitación del sistema SPI):
0 = El sistema SPI está deshabilitado.
1 = El sistema SPI está habilitado.
DWOM (Selección del modo OR alambrada del puerto D):
0 = Las salidas del puerto D pueden tomar niveles lógicos altos o bajos.
1 = Las salidas del puerto D actúan como *drivers* de drenaje abierto.
MSTR (Selección del modo maestro/esclavo):
0 = La SPI está configurada como esclavo.
1 = La SPI está configurada como maestro.

CPOL (Selección de polaridad del reloj):

0 = Selecciona como activo al nivel alto de la señal.

1 = Selecciona como activo al nivel bajo de la señal.

CPHA (Selección de formato de transferencia de datos y fase de reloj):

Este *bit* de control selecciona alguno de los dos formatos fundamentales de transferencia que se han descrito anteriormente.

SPR0, SPR1 Selección de la relación de *bits* de la SPI:

Estos dos *bits* de control seleccionan la relación de *bits* para la transferencia de datos cuando el SPI está operando como maestro de acuerdo con la tabla 3.11. Estos *bits* no tienen significado cuando el dispositivo está configurado como esclavo.

SPR1	SPR0	RELOJ E DIVIDIDO POR
0	0	2
0	1	4
1	0	16
1	1	32

TABLA 3.11 SELECCIÓN DE LA RELACIÓN DE *BITS* DE LA SPI

SPSR: Registro de estado de la SPI (1029_H)

Este registro de sólo lectura contiene las banderas que indican la terminación de una transferencia de datos de la SPI y la ocurrencia de algunos errores del sistema. Las banderas indican automáticamente la ocurrencia del correspondiente evento de la SPI y regresan también automáticamente mediante una secuencia de *software*.

SPIF (Bandera de transferencia completa):
Esta bandera es puesta en 1 lógico automáticamente al final de una transferencia de la SPI, y se puede regresar a 0 lógico mediante la lectura del registro SPSR seguido de un acceso al registro de datos SPDR. La definición del final de una transmisión varía entre el maestro y el esclavo y el formato de transferencia especificado por el *bit* SPHA.
WCOL (Bandera de error de colisión de escritura):
Esta bandera es puesta en 1 lógico automáticamente si el registro SPDR es escrito mientras una transferencia está en progreso, y se puede regresar a 0 lógico mediante la lectura del registro SPSR seguido de un acceso al registro SPDR.
MODF (Bandera de error de modo inválido):
Esta bandera es 1 lógico si la terminal $\overline{SS}$ se activa en nivel bajo mientras la SPI está configurada como maestro (MSTR=1) y se puede regresar a 0 lógico mediante la lectura del registro SPSR seguido de una escritura al registro SPCR.

3.7 CONVERSIÓN ANALÓGICA/DIGITAL (PUERTO E)

Es un puerto de ocho *bits* de entrada de dirección fija. Las terminales del puerto E funcionan alternativamente como canales analógicos de entrada al convertidor analógico/digital (A/D). Los *buffers* de entrada del puerto E están especialmente diseñados para no drenar corriente en exceso cuando sus entradas están en niveles de voltaje intermedios de los niveles lógicos.

3.7.1 Descripción del método de conversión A/D.

El puerto E es el puerto de entradas analógicas de propósito general. El sistema convertidor A/D consiste de ocho canales de entrada de niveles analógicos de voltaje (las ocho terminales del puerto E), un multiplexor analógico de 16 a 1 para seleccionar un canal a la vez para su muestreo y conversión, un convertidor A/D de una entrada con una resolución de conversión de 8 *bits*, un oscilador RC de 2 MHz y circuitos de control para configurar el sistema, por lo que no requiere circuitería adicional para la

conversión. La razón de que el multiplexor sea de 16 entradas es que existen 8 canales internos para verificación y pruebas de manufactura.

El rango del voltaje a convertir debe estar dentro de los límites de referencia V_{RH} (referencia nivel alto) y V_{RL} (referencia nivel bajo) del convertidor. La mayoría de las aplicaciones sugiere conectar V_{RL} a 0 volts (V_{SS}) y V_{RH} a 5 volts (V_{DD}).

El método de conversión empleado es conocido como "Redistribución de cargas", el cual consiste en cargar un arreglo en paralelo de capacitores, cuyo valor individual es del doble que el anterior. Para ejemplificar brevemente el método se hará con 4 *bits* de resolución. Se harán consideraciones importantes para este ejemplo: la corriente de entrada al comparador y la fuga interna de los capacitores es cero, es decir, la carga se conserva. Los capacitores serán conmutados por medio de interruptores analógicos de impedancia cero cuando conectan, e infinita cuando abren. Las unidades de los capacitores no importan, pero sí la proporción entre ellos, por lo que pueden especificarse en cualquier submúltiplo del Faradio. El voltaje a convertir debe estar entre V_{RL} y V_{RH} incluyendo varios extremos.

Paso 1: Muestreo.

Los capacitores C_0 a C_3 de la figura 3.5(a) se conectan con una terminal referida a V_{RL} y la otra al voltaje V_X a convertir. El comparador tiene la terminal positiva de entrada conectada a V_{RL}. La tensión en los capacitores resulta:

$$Vcapac. = V_{RL} - V_X \qquad \ldots \quad (1)$$

De una ecuación básica que relaciona la carga eléctrica (Q), capacitancia (C) y voltaje (V):

$$Q = C\,V \qquad \ldots \quad (2)$$

Se calcula la carga total en los capacitores.

$$Q_{TOT} = C\,(V_{RL} - V_X) \qquad \ldots \quad (2\text{-a})$$

si $V_{RL} = 0$ y $C = 16$ F se obtiene:

$$Q_{TOT} = -16\,V_X \text{ [Coulomb]} \qquad \ldots \quad (2\text{-b})$$

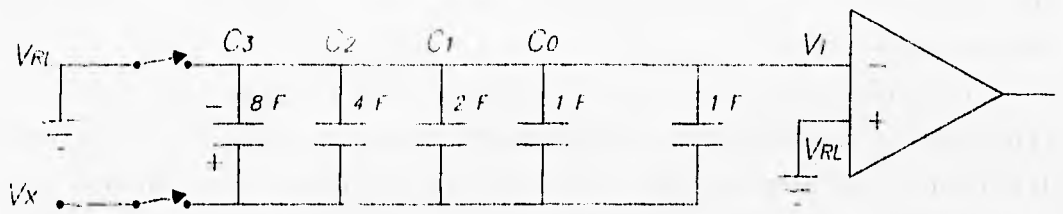


FIG 3.5(a) TOMA DE MUESTRA DEL CONVERTIDOR A/D

Paso 2: Retención de carga

Durante el paso de retención de carga los interruptores se conmutan para conectar el arreglo en paralelo de los capacitores entre la terminal negativa de entrada del comparador (antes V_{RL}) y V_{RL} (antes V_x); ver figura 3.5(b). El voltaje en la terminal negativa del comparador es entonces $-V_x$.

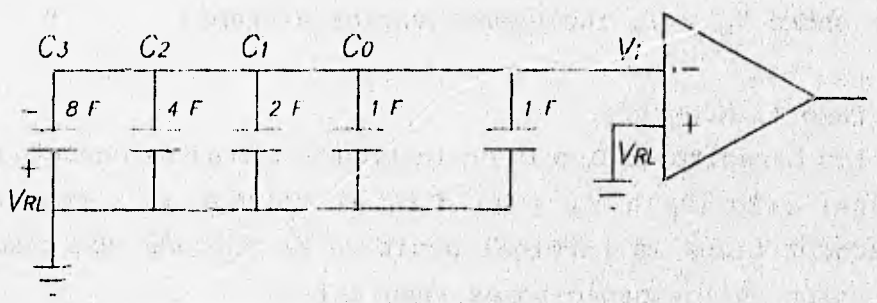


FIG 3.5(b) RETENCION DE CARGA

Paso 3: Aproximación

Para la aproximación del valor digital binario, se supondrá:

V_{RH} = 5 Volts

V_{RL} = 0 Volts

y el voltaje a convertir V_x = 3 Volts.

En la conversión cada capacitor es conmutado desde V_{RL} a V_{RH}, principiando con el capacitor de mayor valor que corresponderá al dígito más significativo (MSB). De esta manera se altera la tensión Vi en la entrada negativa del comparador, si al hacer esto Vi < V_{RL}, el comparador tiene salida en nivel lógico alto y significa que este paso de la aproximación está correcto. De lo contrario se deberá regresar el capacitor conmutado a V_{RL}.

Este proceso se realiza consecutivamente para todos los capacitores de la aproximación. El último capacitor, que tiene valor de 1 unidad, no entra en la secuencia de conmutación y su existencia es necesaria para dar un factor de exactitud de 1/16 en cada paso de la conversión (8+4+2+1+1 = 16).

Para encontrar, entonces el valor lógico de la salida del comparador, se calcula la carga total de los capacitores en el paso de muestreo, con ayuda de la ecuación (2-b), y recordando que el arreglo se conmutó:

$$Q_{TOT} = -16\ V_x = -3\ (16) = -48 \text{ Coulombs} \quad \ldots \quad (3)$$

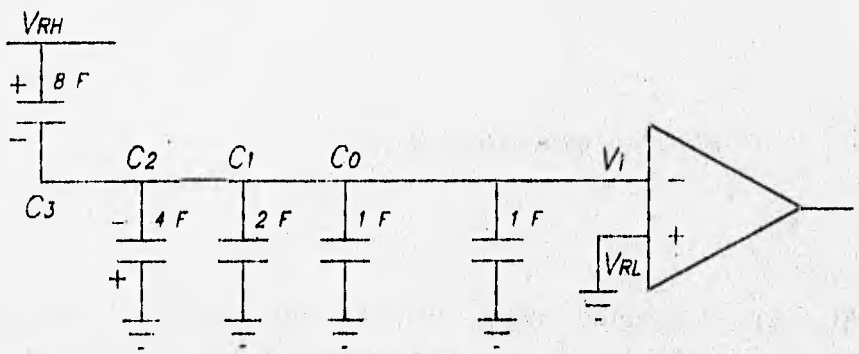


FIG. 3.5(c) APROXIMACION DEL BIT 3

Si se conmuta C_3 a V_{RH}, como se indica en la figura 3.5 (c), y se calcula Vi por medio de cargas:

$$Q_{TOT} = 8\ (Vi-V_{RH}) + (4+2+1+1)\ (Vi-V_{RL}) = -48 \text{ Coulombs}$$

$$Q_{TOT} = 8\ Vi - 8\ V_{RH} + 8\ Vi - 8\ V_{RL} = -48 \text{ Coulombs}$$

$$Q_{TOT} = 16\ Vi - 8\ (5) + 8\ (0) = -48 \text{ Coulombs}$$

Por lo tanto

$$Vi = (-48+40)/16 = -\ 0.5 \text{ V.}$$

Como $Vi < V_{RL}$, el comparador resulta en nivel alto indicando que MSB será 1 lógico. El capacitor C_3 ahora permanecerá conectado a V_{RH}.

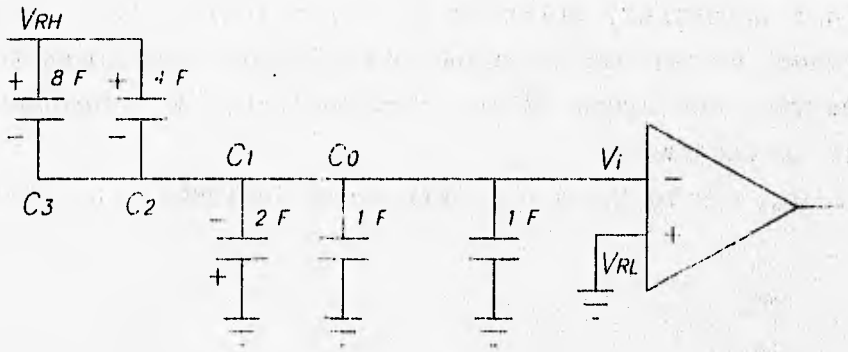


FIG. 3.5(d) APROXIMACION DEL BIT 2

Para el siguiente paso, con el capacitor C_2, se tiene el circuito de la figura 3.5(d). Calculando la carga:

$$Q_{TOT} = (8+4)\ (Vi-V_{RH}) + (2+1+1)\ (Vi-V_{RL}) = -48 \text{ Coulombs}$$

$$Q_{TOT} = 12\ Vi - 12\ V_{RH} + 4\ Vi - 4\ V_{RL} = -48 \text{ Coulombs}$$

$$Q_{TOT} = 16\ Vi - 12\ (5) - 4\ (0) = -48 \text{ Coulombs}$$

$$Vi = (-48+60)/16 = 0.75 \text{ V.}$$

Como $Vi > V_{RL}$, el comparador resulta en nivel bajo indicando que el dígito será 0 lógico, y el capacitor C_2 ahora regresará a conectarse a V_{RL}.

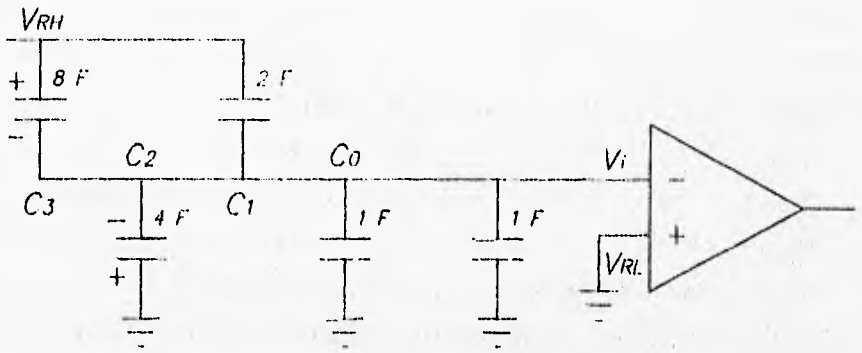


FIG. 3.5(e) APROXIMACION DEL BIT 1

Para el paso con el capacitor C_1, se tiene el circuito de la figura 3.5(e) Calculando la carga:

$$Q_{TOT} = (8+2)\ (Vi-V_{RH}) + (4+1+1)\ (Vi-V_{RL}) = -48 \text{ Coulombs}$$

$$Q_{TOT} = 10\ Vi - 10\ V_{RH} + 6\ Vi - 6\ V_{RL} = -48 \text{ Coulombs}$$

$$Q_{TOT} = 16\ Vi - 10\ (5) - 6\ (0) = -48 \text{ Coulombs}$$

$$Vi = (-48+ 50)/16 = 0.125 \text{ V.}$$

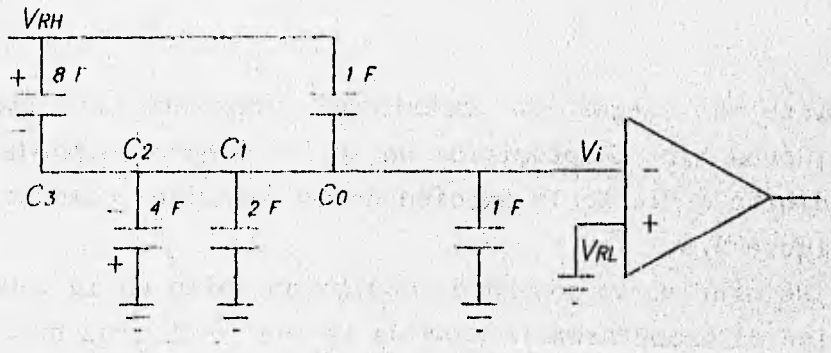


FIG. 3.5(f) APROXIMACION DEL BIT 0

El comparador resulta en nivel bajo indicando que el dígito será 0 lógico, y el capacitor C_1 ahora regresará a conectarse a V_{RL}.

Para el último paso, con el capacitor C_0, se tiene el circuito de la figura 3.5(f). Calculando la carga:

$$Q_{TOT} = (8+1)\ (Vi-V_{RH}) + (4+2+1)\ (Vi-V_{RL}) = -48 \text{ Coulombs}$$

$$Q_{TOT} = 9\ Vi - 9\ V_{RH} + 7\ Vi - 7\ V_{RL} = -48 \text{ Coulombs}$$

$$Q_{TOT} = 16\ Vi - 9\ (5) - 7\ (0) = -48 \text{ Coulombs}$$

$$Vi = (-48+ 45)/16 = - 3/16 = - 0.1875 \text{ V.}$$

Como $Vi < V_{RL}$, el comparador resulta en nivel alto indicando que LSB será 1 lógico. La conversión ha terminado y el resultado es 1001_2.

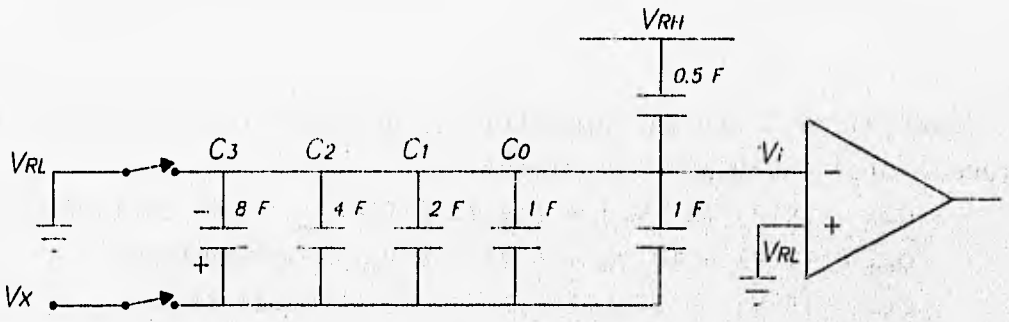


FIG. 3.6 CONVERTIDOR QUE DESPLAZA EL VALOR A CONVERTIR

Existe un error de precisión inherente en todos los convertidores A/D. La precisión de -0, +1 dígito se puede centrar a ±1/2 dígito mediante la adición de un capacitor como se muestra en la figura 3.6.

Si se grafica V_x contra el código obtenido en la conversión, la función de transferencia pasa de la que se observa en la figura 3.7 a la figura 3.8.

Con lo que el error en la conversión se distribuye de igual manera en ambos lados de cada múltiplo de la resolución del convertidor A/D.

El convertidor del microcontrolador MC68HC11 es algo más complejo que el ejemplo analizado, como lo muestra la figura 3.9. El capacitor C_s "divide" el valor de los capacitores de la parte izquierda del diagrama por 16, que son los de menor orden. Esto es con fines prácticos en la fabricación del circuito integrado.

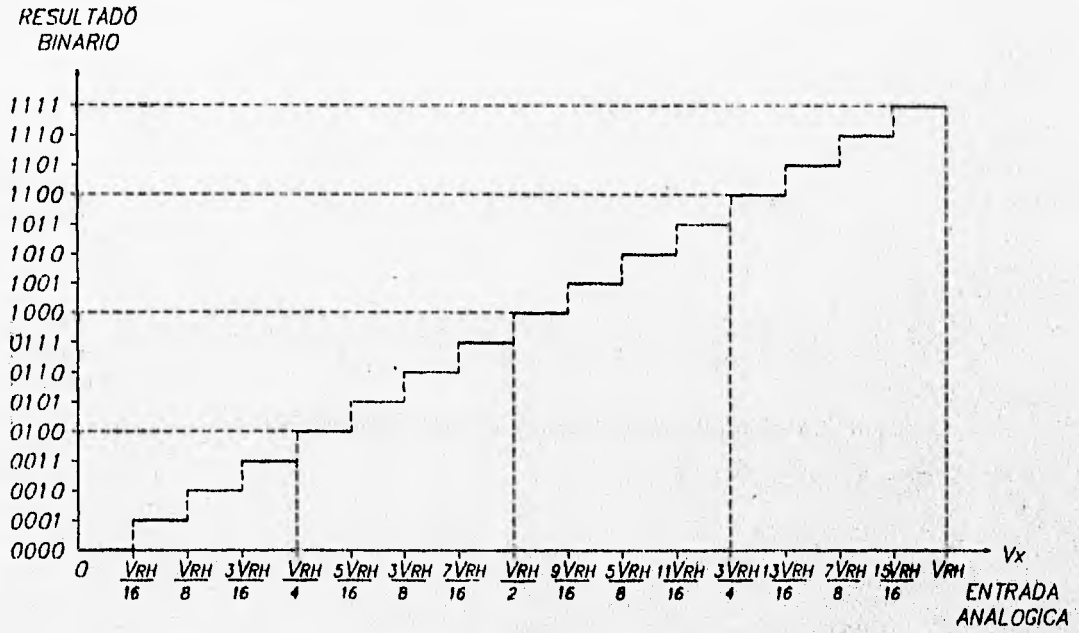


FIG. 3.7 FUNCION DE TRANSFERENCIA DE UN CONVERTIDOR A/D DE 4 BITS

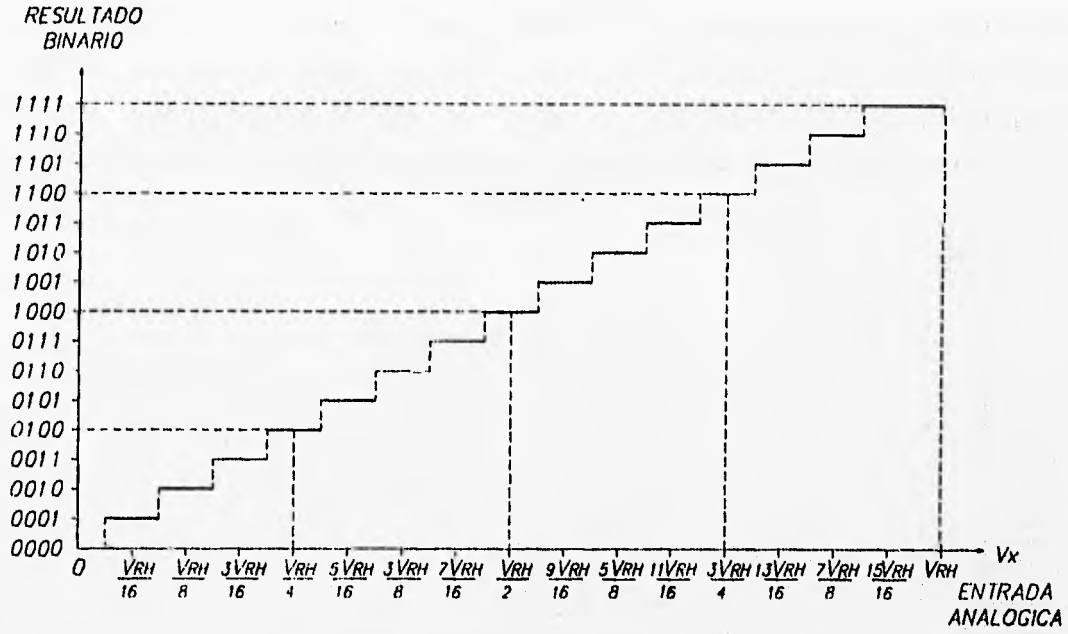


FIG. 3.8 FUNCION DE TRANSFERENCIA CON CORRIMIENTO

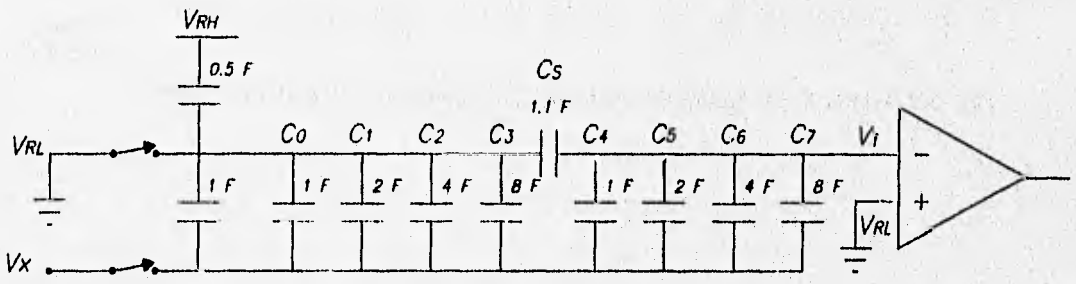


FIG. 3.9 CONVERTIDOR A/D DE 8 BITS

3.7.2 Puesta en marcha del convertidor A/D.

Dentro del circuito integrado se incluye un generador de 8 volts para manejar las compuertas de los interruptores analógicos, tanto del arreglo de capacitores como del multiplexor. Para iniciar la conversión se debe encender este generador poniendo un 1 lógico en el *bit* ADPU (*bit* 7) en el registro OPTION (1039_H),

y se requiere un lapso de 10 mS para permitir al generador lograr el nivel de voltaje.

Debido a que el proceso de conversión es dinámico, éste debe finalizar antes de que la fuga de carga de los capacitores sea considerable. Se requiere que el reloj interno (E) del microcontrolador tenga una frecuencia mayor a 750 KHz., de lo contrario, se corre el riesgo de fuga de carga a temperaturas extremas. Para prevenir este problema, se ha incorporado un oscilador tipo RC, el cual proporciona una fuente de reloj alterna de 2 MHz. El reloj alterno se selecciona escribiendo un 1 lógico en el *bit* CSEL (*bit* 6) en el mismo registro OPTION.

La desventaja de usar este reloj alterno es que no necesariamente está sincronizado al reloj interno del microcontrolador y requiere de un retraso de tiempo al término de la conversión y habilitar la bandera de fin de conversión para su lectura.

Se debe tener cuidado cuando se habilita este reloj ya que el *bit* realiza una segunda función: habilita al generador de alto voltaje para la programación de la EEPROM interna; también se debe considerar un retraso de 10 mS cuando se modifique el contenido de la EEPROM al tiempo que se realiza una conversión A/D.

Se dispone de cuatro registros de memoria (ADR1 hasta ADR4) para almacenar cuatro resultados de las conversiones A/D.

La conversión A/D puede ser sobre un grupo seleccionado de 4 canales (uno a la vez, por el multiplexor) o sobre uno solo cuatro veces, por cada secuencia de conversión. Las secuencias de conversión pueden configurarse para repetirse continuamente o para detenerse al terminar el grupo de cuatro conversiones. De esta forma, para hacer una conversión de los ocho canales externos de que dispone el sistema, será necesario seleccionar un grupo de cuatro canales desde PE0 hasta PE3 y realizar una conversión en modo no repetido, almacenar los resultados en alguna localidad de RAM, seleccionar los canales restantes y realizar las mismas operaciones. La conversión en un canal toma 32 ciclos de reloj E y la secuencia completa, 128.

El registro de control y estado del convertidor A/D es ADCTL, está en la localidad 1030_H y las banderas que contiene se describen a continuación:

CCF (Bandera de conversión terminada):
Este indicador de sólo lectura está en alto cuando los cuatro registros del resultado de la conversión A/D contienen un dato válido. Cada vez que se escribe en el registro ADCTL, este *bit* es automáticamente puesto en 0 lógico y una nueva secuencia de conversión empieza inmediatamente. En el modo de conversión contínua, los registros de resultados son actualizados aún cuando CCF permanezca en nivel alto.

SCAN (Control de rastreo contínuo):
Cuando este *bit* está en 0 lógico, se realiza la conversión en cada uno de los cuatro canales preseleccionados, uno a la vez, para llenar los cuatro registros de resultados. Cuando este *bit* está en 1 lógico, la conversión se realiza en forma contínua y los registros de resultados son actualizados tan pronto como un nuevo dato esté disponible.

MULT (Canal múltiple o canal simple):
Cuando este *bit* es 0 lógico el sistema está configurado para realizar cuatro conversiones consecutivas de un solo canal, especificado mediante los *bits* de selección de canal (de CD a CA de este registro). Cuando este *bit* es 1 lógico el sistema se configura para realizar las conversiones de cada canal en el grupo especificado por los *bits* de selección de canal CD y CC . En este modo cada canal está asociado con un registro específico de resultados (consultar la tabla 3.12).

CD	CC	CB	CA	SEÑAL DEL CANAL	Resultado en: si MULT=1
0	0	0	0	PE0	ADR1
0	0	0	1	PE1	ADR2
0	0	1	0	PE2	ADR3
0	0	1	1	PE3	ADR4
0	1	0	0	PE4 *	ADR1
0	1	0	1	PE5 *	ADR2
0	1	1	0	PE6 *	ADR3
0	1	1	1	PE7 *	ADR4
1	0	0	0	Reservado	ADR1
1	0	0	1	Reservado	ADR2
1	0	1	0	Reservado	ADR3
1	0	1	1	Reservado	ADR4
1	1	0	0	Pruebas en planta	ADR1
1	1	0	1	Pruebas en planta	ADR2
1	1	1	0	Pruebas en planta	ADR3
1	1	1	1	Pruebas en planta	ADR4

* Este canal no está disponible en los paquetes de 48 terminales

TABLA 3.12 ASIGNACIÓN DE CANALES DEL CONVERTIDOR A/D

CD, CC, CB y CA (Selectores de canal):
Estos cuatro *bits* son usados para especificar el canal o canales con que se va a operar en la conversión A/D, la tabla 3.12 muestra la relación del canal seleccionado en modo simple. Cuando se selecciona el modo de canal multiple, los *bits* CB y CA no tienen efecto y el grupo de cuatro canales se selecciona mediante los *bits* CD y CC.

Los registros de resultados de conversión A/D, ADR1..ADR4 son registros de sólo lectura usados para retener el resultado de la conversión de ocho *bits*. Después de que los registros han sido llenados con datos válidos de una secuencia de conversión, el *bit* de estado CCF se fija por sí mismo en 1 lógico para indicar que los resultados son válidos. Los resultados de una nueva conversión son calculados dentro del sistema convertidor, y son transferidos a los

registros de resultados en una parte de ciclo de reloj donde las lecturas no se pueden realizar, así pues no puede ocurrir interferencia entre lecturas por *software* y actualización de resultados.

3.8 SISTEMA DE VIGILANCIA (COP)

El sistema de *timers* COP está dirigido para detectar errores en el procesamiento del *software*. Esta función equivale en otros procesadores a la función *WATCHDOG TIMER* (*timer* de perro guardián). Cuando la función COP es utilizada, el *software* es responsable de mantener un *timer* sin que alcance su tiempo de expiración. Si el *timer* expira, es una indicación de que el *software* no se ha ejecutado con su secuencia intencional, y entonces se inicia un *reset* del sistema.

El sistema COP se habilita o deshabilita dependiendo del estado del *bit* NOCOP en el registro CONFIG ($103F_H$) basado en la memoria EEPROM. Después que el *bit* ha sido cambiado, el MCU debe restablecerse para hacer efectivo el nuevo estado.

En los modos de operación de prueba especial o arranque especial, el sistema COP es inicialmente deshabilitado mediante el *bit* de control DISR del registro TEST1. Este *bit* puede ser fijado en 0 lógico para habilitar un *reset* del sistema COP en estos 2 modos de operación.

El período del sistema COP se selecciona mediante los *bits* de control CR1 y CR0 del registro OPTION (1039_H). Después del *reset*, estos *bits* son ambos 0 lógico, lo cual selecciona el período más corto y el reloj interno E se divide por 2^{15} antes de entrar al sistema COP. Los *bits* CR1 y CR0 seleccionan el período para el *timer*, como puede verse en la tabla 3.13. En los modos de operación normales (un solo *chip* y expandido), estos *bits* solamente pueden ser escritos una vez, y esto debe hacerse en los primeros 64 ciclos de reloj después del *reset*.

CR1	CR0	FACTOR DIVISOR DE E	TIEMPO DE EXPIRACIÓN
0	0	2^{15}	16.384 ms *
0	1	2^{17}	65.536 ms *
1	0	2^{19}	262.14 ms *
1	1	2^{21}	1.049 s *

* Frecuencia del oscilador 8 MHz

TABLA 3.13 SELECCIÓN TIEMPO DEL SISTEMA COP

El *timer* del sistema COP debe restablecerse mediante una secuencia de *software* antes de su expiración para evitar un *reset* provocado por el sistema COP. Este procedimiento es una secuencia de dos pasos, el primero es escribir 55_H en el registro COPRST ($103A_H$) para armar el mecanismo de limpieza del *timer*. El segundo paso es escribir AA_H en el mismo registro, con lo cual se restablece el *timer* COP. Cualquier cantidad de instrucciones se puede realizar entre estos pasos siempre que se realicen en la secuencia correcta antes de que el *timer* expire.

3.9 INTERRUPCIONES

La CPU de un microcontrolador ejecuta secuencialmente las instrucciones, pero en muchas aplicaciones es necesario ejecutar un grupo de instrucciones en respuesta al requerimiento de varios dispositivos periféricos. Estos requerimientos usualmente son asincronos a la ejecución del programa principal. Las interrupciones proporcionan un medio para suspender temporalmente la ejecución normal del programa, de este modo la CPU se libera para atender estos requerimientos. Después que se ha atendido la interrupción, el programa principal se reanuda como si no se hubiera interrumpido.

Al grupo de instrucciones ejecutadas en respuesta a la interrupción se le llama "Rutina de Servicio de Interrupción". Estas rutinas semejan a las subrutinas excepto en que son llamadas automáticamente mediante un mecanismo de interrupción por *hardware* en vez de una instrucción de llamada, y todos los registros de la CPU son almacenados en la Pila de datos, en vez de que solamente sea el Apuntador de Programa.

Una interrupción provoca que el flujo normal del programa se suspenda tan pronto como termina la instrucción que se ejecuta. La lógica de interrupción entonces almacena el contenido de todos los registros de la CPU en la Pila para que se puedan recuperar después del servicio a la interrupción. Después de guardar los registros, el vector de máxima prioridad de las fuentes de interrupción solicitantes se carga en el Apuntador de Programa y la ejecución continúa con la instrucción que se encuentra en esa dirección. Un servicio de interrupción se concluye con la instrucción RTI (*Return from Interrupt*) que provoca que los registros de la CPU, así como la dirección del retorno, sean recuperados de la Pila, para continuar con la ejecución normal del programa.

Las interrupciones pueden ser habilitadas o no globalmente mediante los *bits*: mascarilla I y X del registro CCR o por las mascarillas locales del propio registro de control de los dispositivos. Existen seis fuentes de interrupción que están siempre habilitadas a las cuales se les llama "no mascarables". La terminal $\overline{\text{XIRQ}}$ es precisamente una fuente de interrupción no mascarable excepto que se encuentra deshabilitada después del *reset*, pero quedará permanentemente habilitada al escribirse un 1 lógico en la mascarilla X. Las demás fuentes de interrupción son mascarables con el *bit* I del registro CCR.

Las interrupciones obedecen a un circuito de prioridades para resolver los requerimientos de interrupciones simultáneas, en el orden siguiente:

1. POR ó terminal $\overline{\text{RESET}}$
2. *Reset* por el monitor de reloj
3. *Reset* por el sistema COP
4. Terminal $\overline{\text{XIRQ}}$
5. Interrupción por código de operación ilegal
6. Interrupción por *software*
7. Desbordamiento del *timer*
8. Desbordamiento del acumulador de pulsos
9. Flanco de entrada del acumulador de pulsos
10. Transferencia completa en SPI
11. Sistema SCI
12. Terminal $\overline{\text{IRQ}}$
13. Interrupción de tiempo real RTI
14. *Timer* de captura de entrada 1
15. *Timer* de captura de entrada 2
16. *Timer* de captura de entrada 3
17. *Timer* de comparación de salidas 1
18. *Timer* de comparación de salidas 2
19. *Timer* de comparación de salidas 3
20. *Timer* de comparación de salidas 4
21. *Timer* de comparación de salidas 5

Sin embargo, cualquier fuente de interrupción mascarable puede ser elevada a la prioridad más alta de esta categoría. La prioridad más alta de éstas se asigna mediante los *bits* PSEL3,...,PSEL0 del registro HPRIO ($103C_H$) de acuerdo a la tabla 3.14.

PSEL3	PSEL2	PSEL1	PSEL0	FUENTE DE INTERRUPCIÓN
0	0	0	0	Desbordamiento del *timer*
0	0	0	1	Desbordamiento del acumulador de pulsos
0	0	1	0	Flanco de entrada del acumulador de pulsos
0	0	1	1	Transferencia completa en SPI
0	1	0	0	Sistema SCI
0	1	0	1	Reservado (omisión a $\overline{IRQ}$)
0	1	1	0	$\overline{IRQ}$ (Terminal externa o ent./sal. paralelo)
0	1	1	1	Interrupción de tiempo real RTI
1	0	0	0	*Timer* de captura de entrada 1
1	0	0	1	*Timer* de captura de entrada 2
1	0	1	0	*Timer* de captura de entrada 3
1	0	1	1	*Timer* de comparación de salidas 1
1	1	0	0	*Timer* de comparación de salidas 2
1	1	0	1	*Timer* de comparación de salidas 3
1	1	1	0	*Timer* de comparación de salidas 4
1	1	1	1	*Timer* de comparación de salidas 5

TABLA 3.14 SELECCIÓN DE FUENTE DE INTERRUPCIÓN DE MAYOR PRIORIDAD

El MCU tiene 21 vectores de interrupción que soportan a 26 fuentes de interrupción. Las 15 interrupciones mascarables son generadas por sistemas periféricos en el *chip*. La tabla 3.15 muestra las fuentes de interrupción con su correspondiente vector y su mascarilla local.

DIRECCIÓN DEL VECTOR	FUENTE DE INTERRUPCIÓN	MASCARILLA CCR	MASCARILLA LOCAL
FFC0, C1-FFD4, D5	RESERVADOS		-
FFD6, D7	SISTEMA SCI	I	
	* SCI TRANSMISIÓN COMPLETA		TCIE
	* SCI REGISTRO DE TRANS. VACÍO		TIE
	* SCI DETEC. DE LÍNEA NO ACTIVA		ILIE
	* SCI DATO PERDIDO EN RECEPCIÓN		RIE
	* SCI REGISTRO RECEPTOR LLENO		RIE
FFD8, D9	TRANSFERENCIA COMPLETA EN SPI	I	SPIE
FFDA, DB	FLANCO DE ENT. DEL ACUM. DE PULSOS	I	PAII
FFDC, DD	DESBORDAMIENTO DEL ACUM. DE PULSOS	I	PAOVI
FFDE, DF	DESBORDAMIENTO DEL *TIMER*	I	TOI
FFE0, E1	*TIMER* DE COMPARACIÓN DE SALIDAS 5	I	OC5I
FFE2, E3	*TIMER* DE COMPARACIÓN DE SALIDAS 4	I	OC4I
FFE4, E5	*TIMER* DE COMPARACIÓN DE SALIDAS 3	I	OC3I
FFE6, E7	*TIMER* DE COMPARACIÓN DE SALIDAS 2	I	OC2I
FFE8, E9	*TIMER* DE COMPARACIÓN DE SALIDAS 1	I	OC1I
FFEA, EB	*TIMER* DE CAPTURA DE ENTRADA 3	I	IC3I
FFEC, ED	*TIMER* DE CAPTURA DE ENTRADA 2	I	IC2I
FFEE, EF	*TIMER* DE CAPTURA DE ENTRADA 1	I	IC1I
FFF0, F1	INTERRUPCIÓN DE TIEMPO REAL RTI	I	RTII
FFF2, F3	PROTOCOLO DE ENT./SAL. EN PARALELO	I	STAI
	$\overline{\text{IRQ}}$ (Terminal externa)		-
FFF4, F5	$\overline{\text{XIRQ}}$ (Terminal externa)	X	-
FFF6, F7	INTERRUPCIÓN POR *SOFTWARE*	-	-
FFF8, F9	CÓDIGO DE OPERACIÓN ILEGAL	-	-
FFFA, FB	FALLA POR SISTEMA COP	-	NOCOP
FFFC, FD	FALLA POR MONITOR DE RELOJ (COP)	-	CME
FFFE, FF	RESET	-	-

TABLA 3.15 RELACIÓN DE FUENTES Y VECTORES DE INTERRUPCIÓN

CAPÍTULO IV

DISEÑO Y CONSTRUCCIÓN DEL PROTOTIPO, *HARDWARE*

El diseño electrónico del Programador de Memorias EPROM 27C16, 27C32, 27C64 y 27C128 está basado en el microcontrolador Motorola MC68HC11A1, el cual se definió ampliamente en el capítulo anterior.

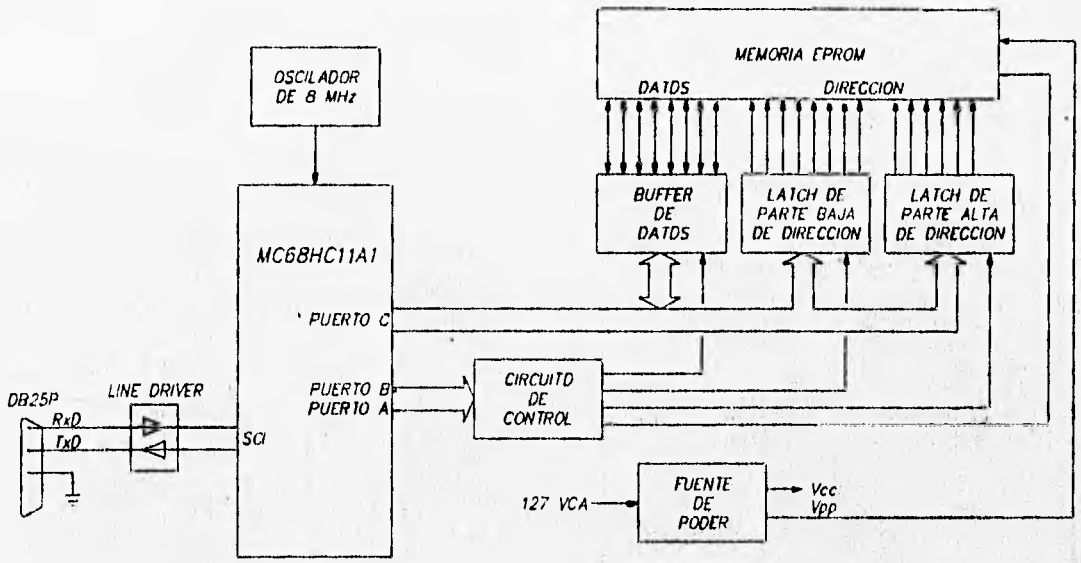


FIG. 4.1 DIAGRAMA DE BLOQUES DEL PROGRAMADOR

La figura 4.1 muestra un diagrama de bloques con los elementos del Programador. Las terminales del microcontrolador que se utilizan son los puertos A, B, C y del puerto D, la interfase de comunicación tipo serie SCI.

La interfase SCI tiene dos líneas: transmisión (Tx) y recepción (Rx), las cuales se conectan a la computadora personal a través de un circuito integrado *Line Driver* o "Manejador de Línea". La interfase SCI y el puerto tipo serie de la computadora deben programarse empleando el mismo protocolo de comunicación.

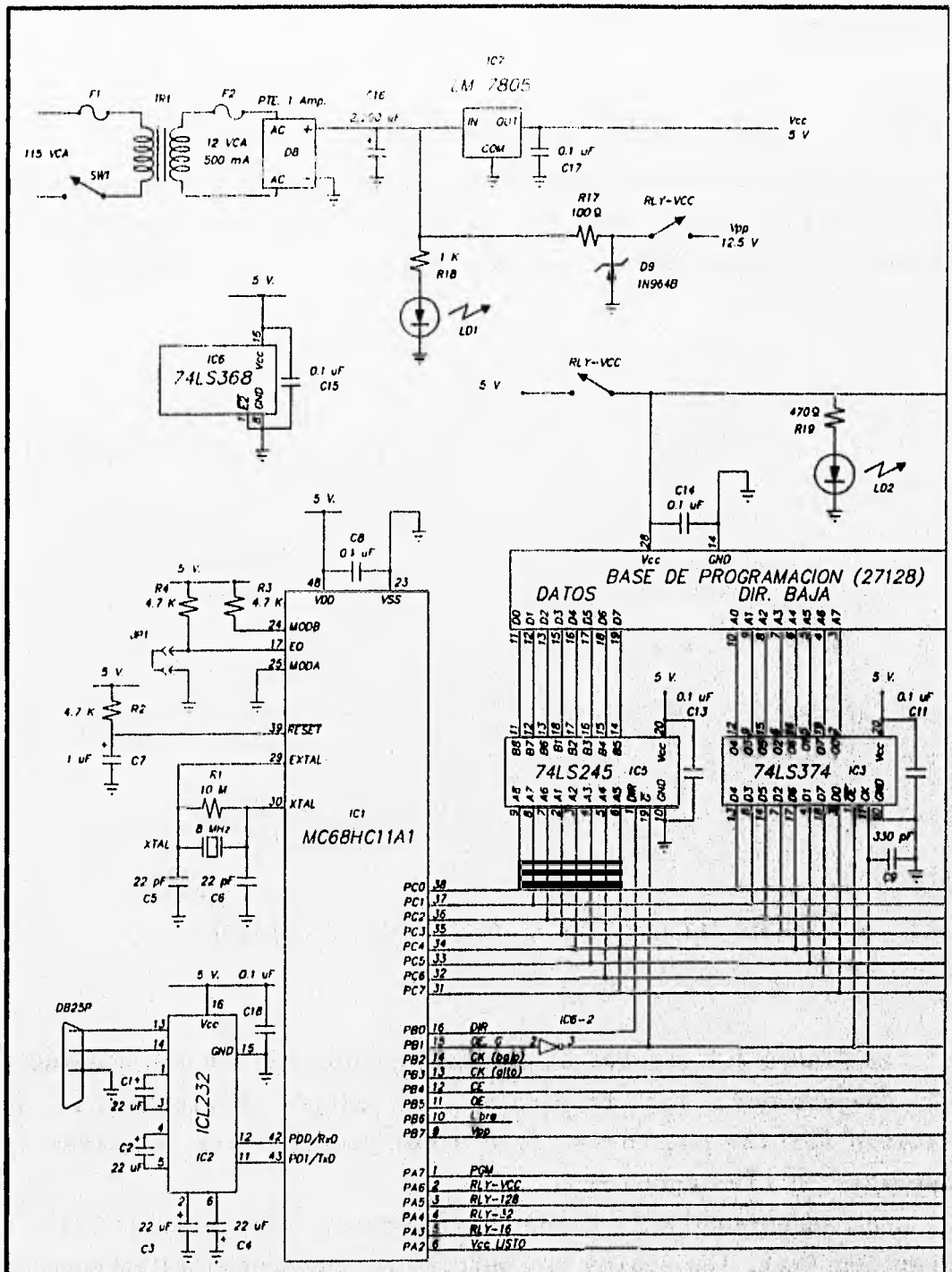


FIG. 4.2a DIAGRAMA ELECTRONICO DEL PROGRAMADOR

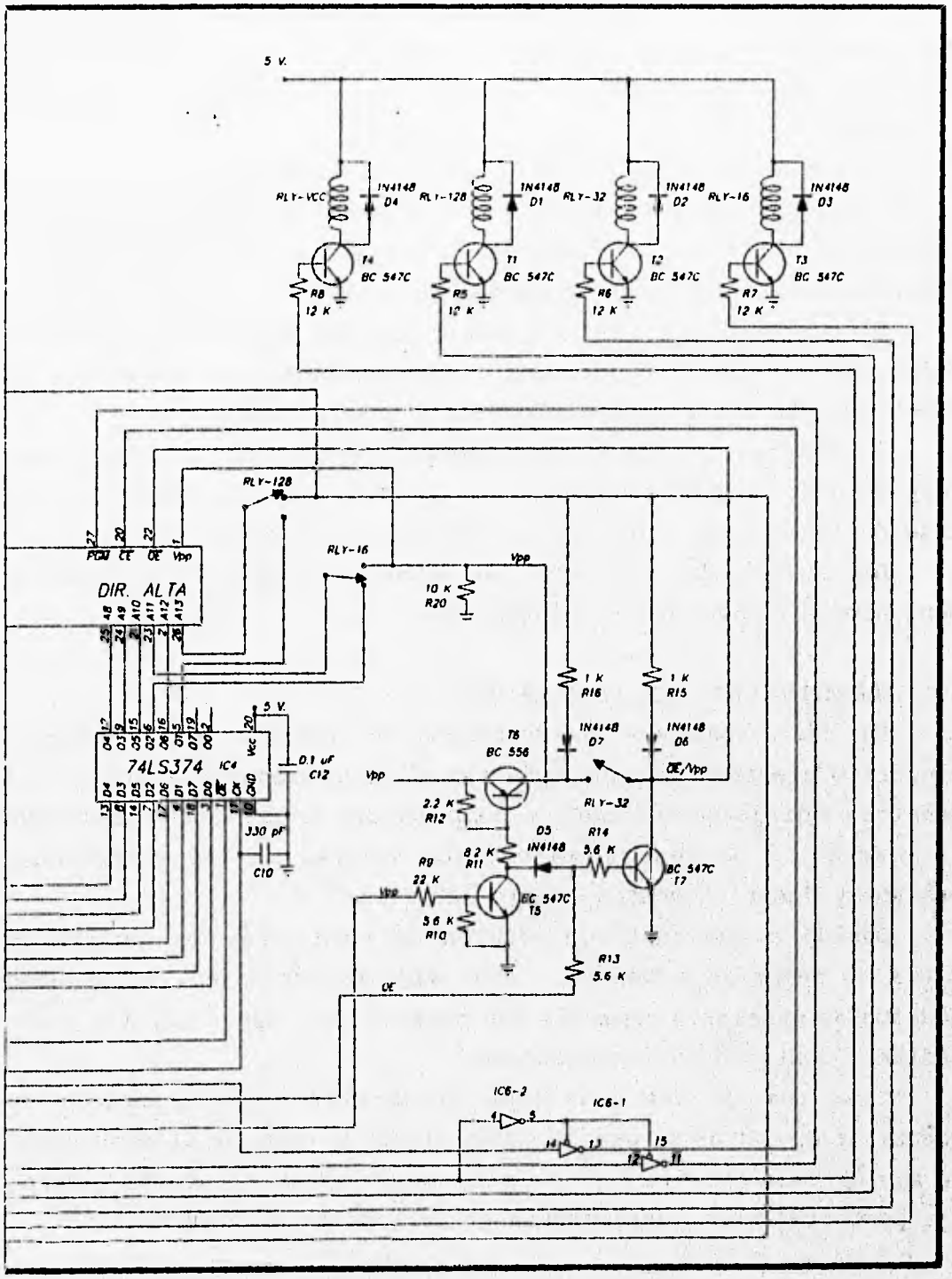


FIG. 4.2b DIAGRAMA ELECTRONICO DEL PROGRAMADOR

La señal de reloj del microcontrolador es proporcionada por un oscilador de cristal de cuarzo de 8 MHz. De esta señal se deriva la velocidad de transmisión de la interfase SCI mediante divisores internos.

La fuente de alimentación general es un modelo típico, basada en un regulador de voltaje fijo de 5 V que se usa para alimentar a todo el circuito y a un diodo Zener que actúa como regulador para obtener la tensión de programación de 12.5 V.

El *bus* de direcciones, formado por dos circuitos integrados *latch*, se usa para proporcionar las direcciones de memoria a la EPROM durante los ciclos de lectura y de escritura.

Se utiliza un circuito integrado "manejador de *bus*", para proporcionar el dato a ser programado en la memoria EPROM, y en el caso de la lectura, para transmitir el dato leído al puerto C.

Las señales de control de las memorias se manejan mediante un conjunto de componentes no integrados.

4.1 FUNCIONAMIENTO DEL PROGRAMADOR

En este apartado se describen en detalle los elementos eléctrico/electrónicos que integran al Programador. La figura 4.2 muestra el diagrama electrónico completo del Programador, indicando la numeración de los circuitos, los valores de los componentes pasivos y demás elementos que lo integran.

Debido a que la configuración de terminales de los cuatro tipos de memorias a manejar tiene algunas variantes, en algunos modelos es necesario conmutar las señales Vcc, Vpp, A13, A11 y $\overline{OE}$, mediante contactos de relevadores.

Cada uno de los circuitos integrados tiene conectado un capacitor de 0.1 uF en paralelo con sus terminales de alimentación, lo más cercano posible a éstas para hacer mínimo el efecto causado por perturbaciones instantáneas en la tensión de alimentación.

4.1.1 Comunicación PC-SCI

La interfase de comunicación tipo serie, SCI, se usa para realizar el intercambio de información entre el microcontrolador y la computadora personal.

El puerto tipo serie de la computadora utiliza la interfase RS-232 ó EIA-232 cuyos niveles de voltaje son de +3 a +12 V para un 0 lógico y entre -3 y -12 V para un 1 lógico para permitir una comunicación hasta 16 metros de distancia. En la interfase SCI se manejan niveles lógicos de voltaje de 0 y 5 V. Esta diferencia de tensiones en los niveles lógicos obliga a emplear un circuito Manejador de Línea ó *Line Driver* para apegarse al estándar RS-232 ó EIA-232.

El circuito *Line Driver* cambia los niveles lógicos de 0 y 5 V a ±12 V por medio de una pequeña fuente basada en la conmutación de los capacitores C1, C2, C3 y C4. La mitad del circuito *Line Driver* realiza la función inversa, es decir, convierte los niveles ±12 V a niveles 0 y 5 V compatibles con el microcontrolador.

La interfase SCI se puede programar para velocidades de transmisión desde 110 hasta 9,600 *bauds*; 7 u 8 *bits* de datos; 1 ó 2 *bits* de paro y 1 ó ningún *bit* de paridad.

El protocolo que se emplea es:

Velocidad: 9,600 *bauds*

Dato: 8 *bits*

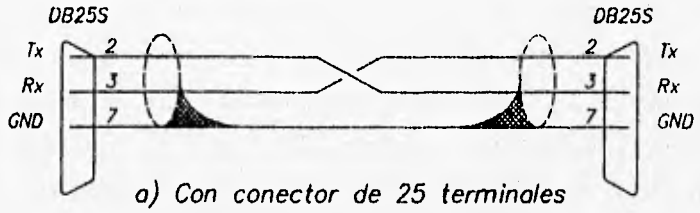
Paro: 1 *bit*

Paridad: ninguna

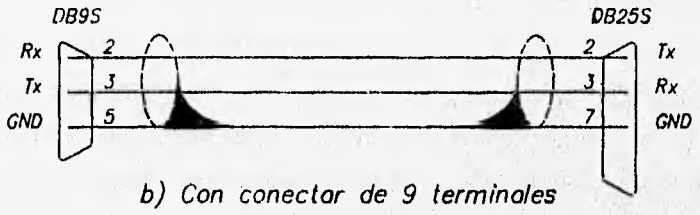
La mayor velocidad de transmisión ayuda directamente a reducir el tiempo en que el programador espera instrucciones de la computadora. Debido a que en el diálogo PC-Programador, se envían sólo caracteres numéricos y alfabéticos en mayúsculas, la información también se puede enviar en paquetes de 7 *bits* de datos, ya que el *bit* de mayor orden no se utiliza.

En el programador se utiliza un conector tipo DB25-P para las líneas de comunicación tipo serie, en el cual se asignan las señales de acuerdo a la interfase DTE (*Data Terminal Equipment*). Asimismo los puertos de comunicación tipo serie de las computadoras

utilizan la interfase DTE. En la figura 4.3 se muestra un diagrama de conexión del cable de comunicación Programador-PC para los casos de conectores de 25 ó 9 terminales.



a) Con conector de 25 terminales



b) Con conectar de 9 terminales

FIG. 4.3 CABLE DE INTERCONEXION

4.1.2 Fuente de alimentación

El Programador tiene una fuente de alimentación con dos voltajes diferentes de salida: una salida de 5 V para todo el circuito, incluyendo la memoria a programar y otra salida para el voltaje de programación de 12.5 V.

La fuente de alimentación se compone de un transformador de 127/12 VCA con una corriente nominal de 500 mA en el devanado secundario, un regulador de voltaje LM7805 y un diodo Zener de 13 V.

El regulador de voltaje utilizado es tipo serie, por lo que la corriente total del circuito puede calcularse de manera global sumando las corrientes típicas de cada uno de los componentes que integran al Programador y despreciando la corriente proporcionada a los transistores y resistores por ser muy pequeña en comparación.

10 mA	Idd Microcontrolador
8 mA	Idd Memoria EPROM (Activada)
47 mA	Icc 74LS245
56 mA	Icc 74LS374 (2 circuitos)
10 mA	Icc 74LS368
76 mA	I_L Relevador (2 relevadores)
3 mA	ICL232
210 mA	TOTAL

La tensión requerida puede calcularse así:

5.0 V	Vcc
1.5 V	(Vo-Vi) Regulador
1.4 V	Vt (dos diodos)
7.9 V	TOTAL

El voltaje RMS del devanado secundario del transformador debe ser mayor que 7.9 V para suministrar la potencia requerida.

La tensión de programación se obtiene a través de las terminales del diodo Zener, el cual se alimenta a través del resistor Rx, con una corriente aproximada de 15 mA, que produce una tensión zener de 12.48 V. Este es el valor que determina la tensión del transformador de alimentación.

Siempre existen factores fuera del control del diseñador, como la variación de la tensión del suministro eléctrico doméstico, la cual afecta directamente la tensión del extremo secundario del transformador, por lo que se debe tomar en cuenta un margen de seguridad.

4.1.3 Oscilador de 8 MHz

Es un oscilador "Pierce" también llamado "oscilador de cristal de resonancia en paralelo", lo componen el cristal XTAL, R1, C5 y C6. El resistor R1 proporciona una tensión de polarización en la entrada EXTAL del microcontrolador para la que la NAND interna opere en la región lineal. Los capacitores C5 y C6 limitan la potencia que maneja el cristal (típicamente 1 uW). El valor de la frecuencia del cristal determina la del oscilador.

4.1.4 *Bus* de dirección

El propósito de hacer un *bus* de dirección es proporcionar un valor hexadecimal a las terminales de dirección de la memoria que será programada o leída.

El *bus* de dirección se divide en dos partes: Alto y Bajo Orden. La parte de alto orden la integran las líneas de la memoria que van desde A13 hasta A8 y la parte de bajo orden las que van desde A7 hasta A0. Debido a la capacidad de cada modelo específico, las memorias 27C64 no utilizan la señal A13, las 27C32 no utilizan las señales A13 ni A12 y las 27C16 no utilizan las señales A13, A12 ni A11, por tener cada una de ellas, la mitad de capacidad que la anterior. De cualquier forma, estas líneas tomarán un valor lógico aunque no sean utilizadas.

Cada parte del *bus* está constituída por un circuito integrado "*latch* óctuple" que "congela" la información que se presenta en la entrada de sus *flip-flops* mediante un flanco ascendente en la terminal de reloj.

El dato requerido en el *bus* se presenta en los *flip-flops* de IC3 e IC4 procedente de las salidas del puerto C del microcontrolador. Enseguida el puerto B presenta una transición de 0 lógico a 1 lógico en la terminal de reloj del *latch* correspondiente por medio de los *bits* B3 y B2 para la parte de alto y bajo orden respectivamente, efectúandose la carga en ambas partes de manera no simultánea.

Los capacitores C9 y C10 conectados a las terminales de reloj tienen la finalidad de evitar falsos "disparos" que provoquen un cambio de estado en los *flip-flops* debido a la interfase CMOS-TTL.

Las terminales de salida de IC3 e IC4 se activan cuando la terminal $\overline{OE}$ es 0 lógico, la cual se conecta a la terminal B1 del microcontrolador mediante una compuerta inversora. Debido a que el microcontrolador pone todas las terminales de sus puertos de salida en 0 lógico al salir del estado de *reset*, justifica la existencia de la compuerta inversora en la señal de B1, de este modo el *bus* de dirección no se activará al encenderse el Programador.

4.1.5 *Bus* de datos

El *bus* de datos está formado por el circuito IC5 que es un *buffer* óctuple bidireccional con salidas de tres estados, para permitir la transferencia de información entre el puerto C del microcontrolador y las terminales de datos de la memoria EPROM. El sentido del flujo de datos está determinado por la terminal de entrada DIR, la cual se controla con el *bit* B0 del puerto B. Las salidas se habilitan mediante la terminal $\overline{G}$ conectada al *bit* B1 a través de un inversor, de forma que la habilitación del *bus* de datos sea simultánea a la del *bus* de dirección.

4.1.6 Circuitos de control general

Los circuitos adicionales a los *buses* de dirección y de datos tienen la finalidad de proporcionar los niveles de voltaje adecuados en las terminales de control de la base de programación. La figura 4.4 muestra un diagrama de tiempos de las señales de control.

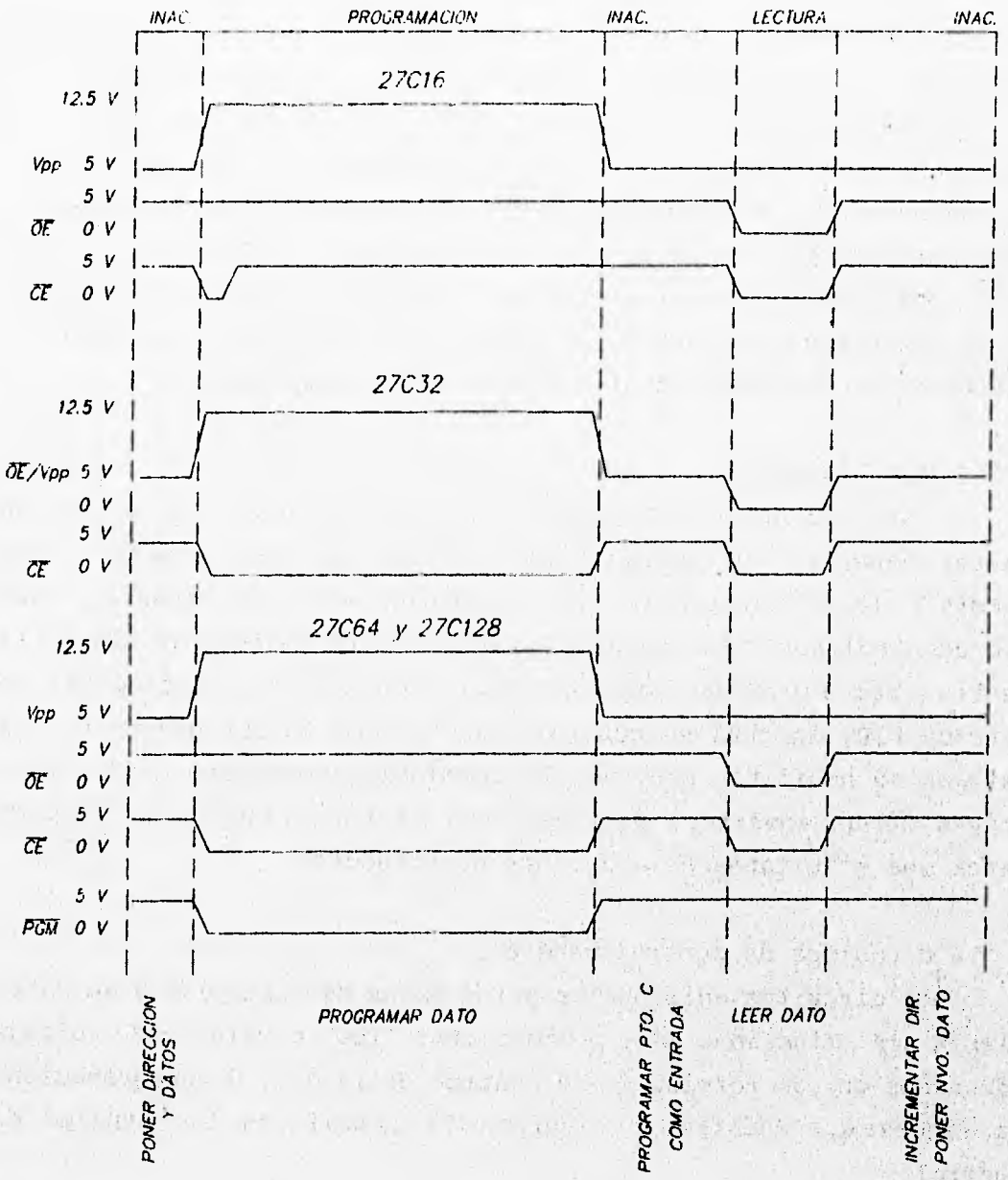


FIG. 4.4 DIAGRAMA ESQUEMATICO DE TIEMPOS

La terminal B7 del puerto B maneja al transistor T5 y éste a su vez al T6. Cuando T6 está en estado de saturación permite obtener 12.5 V de la fuente de programación en la terminal Vpp; en estado de corte, bloquea la tensión de esta fuente y en la terminal Vpp se obtienen 4.3 V proporcionados por la fuente de 5 V a través de R15 y el diodo D7.

Los resistores R9 y R11 tienen la finalidad de contribuir a la descarga de la capacitancia parásita B-E de los transistores T5 y T6 para obtener una conmutación efectiva a alta frecuencia. R8 y R10 determinan la corriente de base de T5 y T6 respectivamente.

El transistor T7 actúa como inversor para suministrar la señal $\overline{OE}$ hacia la memoria a partir del *bit* B5 del puerto B. Se obtiene una tensión de 0.2 ó 4.3 V del colector de T7 para la terminal $\overline{OE}$ cuando se trate de las memorias 27C128, 27C64 y 27C16. Para el caso de la 27C32 se requiere adicionalmente una tensión de 12.5 V en el colector de T7, esto se logra llevando a T6 al estado de saturación y cerrando el contacto del relevador RLY-32. En este caso T7 debe estar siempre en corte, lo cual se asegura mediante el diodo D5 conectado al colector de T5 que en tal caso se encuentra en estado de saturación. Cuando exista una tensión de 12.5 V en el colector de T7, el diodo D6 se polariza inversamente evitando que ésta altere la tensión de alimentación de 5 V.

Existen algunas diferencias en la asignación de señales de las terminales de los tipos de memoria a programar, por ejemplo, la terminal número 26 en la base receptora puede contener las señales Vcc para los tipos 27C16 y 27C32, sin asignar para la 27C64 y A13 para la 27C128, lo que hace apropiado el uso de un relevador (RLY-128) para conmutar la terminal 26 de la base con las señales mencionadas de acuerdo al tipo de memoria que se elija para ser programada.

Asimismo, la terminal 23 de la base receptora, tendrá la señal A11 para los tipos 27C128, 27C64 y 27C32; y Vpp para la 27C16, para lo cual se utiliza el relevador RLY-16.

La terminal 22 de la base receptora proporciona la señal $\overline{OE}$ para los tipos 27C128, 27C64 y 27C16 y para la 27C32 es una combinación de las señales $\overline{OE}$ y Vpp. El relevador RLY-32 realiza esta combinación; en los casos de lectura (Vpp desactivada) los valores que puede tomar esta terminal son 0 ó 5 V, y en el caso de escritura, Vpp se activa proporcionando 12.5 V sin importar el estado de la señal $\overline{OE}$.

Los relevadores RLY-128, RLY-32 y RLY-16 se energizan por medio de los transistores T1, T2 y T3 respectivamente, que a su vez están conectados a las señales A5, A4 y A3 del puerto A. Los diodos D1, D2 y D3 permiten limitar la tensión de colector de los transistores cuando éstos salen del estado de conducción y hacen que las bobinas de los relevadores funcionan como inductores.

Para permitir el intercambio de la memoria EPROM a programar, el relevador RLY-Vcc permite el corte de las tensiones de alimentación y de programación de la memoria. A6 tiene dos funciones: energizar al relevador RLY-Vcc mediante el transistor T4 y habilitar el estado de alta impedancia para las señales $\overline{PGM}$ y $\overline{CE}$ de la memoria a través de dos inversores con salida de tres estados.

El resistor R2 y el capacitor C8 forman un arreglo de desplazamiento de fase, el cual se emplea como señal de $\overline{RESET}$ para el microcontrolador.

El microcontrolador ejecuta al arrancar el programa monitor contenido en ROM, el cual realiza un salto hacia la primera localidad de EEPROM ($B600_H$) dependiendo del nivel lógico presentado en la terminal E0. El puente JP1 permite inhibir el salto para entrar al programa monitor y cargar los códigos de operación del Programador de Memorias.

4.2 CARACTERÍSTICAS ELÉCTRICAS DE LOS CIRCUITOS

En esta sección se mencionan las especificaciones de los circuitos tomadas de los manuales técnicos de los fabricantes.

4.2.1 Memorias EPROM 27C16, 27C32, 27C64 y 27C128

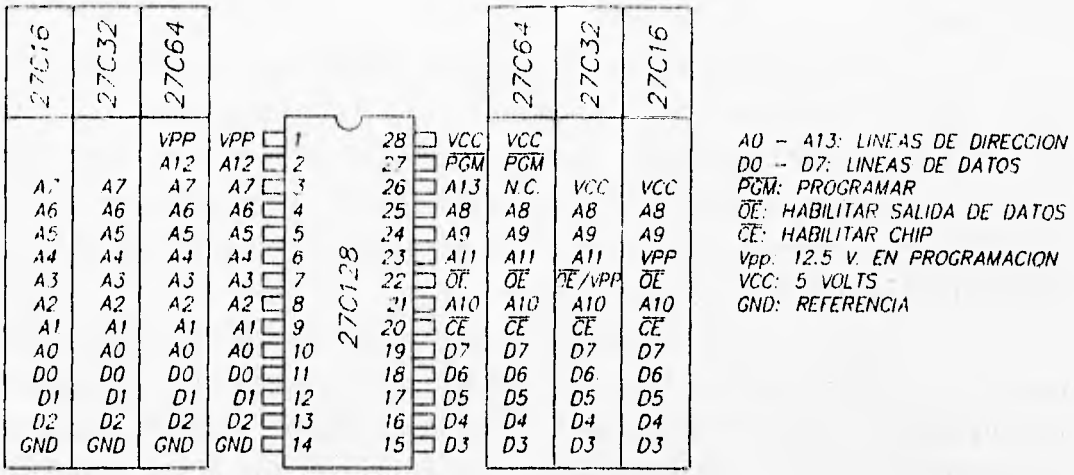


FIG. 4.5 ASIGNACION DE TERMINALES DE LAS MEMORIAS EPROM

Las memorias 27C128, 27C64, 27C32 y 27C16 son memorias de borrado ultravioleta, eléctricamente programables de sólo lectura y fabricadas con tecnología CMOS. Su capacidad de almacenamiento es: 16 KB, 8 KB, 4 KB y 2 KB respectivamente.

La tensión de alimentación requiere una fuente única de 5 V para el modo de lectura, con una tolerancia de 10% que permite diseños más versátiles de la fuente de poder del sistema. Para la programación se requiere de una fuente adicional de 12.5 V. El tiempo de acceso estándar va desde 150 nS hasta 250 nS, el cual es compatible con los microprocesadores de alto rendimiento.

Otra característica importante de estas memorias es el uso de una línea de control de habilitación de salida: *Output Enable* ($\overline{OE}$), la cual se usa para evitar conflictos de *bus* en sistemas con microprocesador. Tienen un modo de "espera" (*standby*) para reducir el consumo de energía sin incrementar el tiempo de acceso. Este

modo se selecciona aplicando una señal de nivel TTL alto a la terminal *Chip Enable* ($\overline{CE}$).

Características de borrado

El borrado de las memorias inicia cuando son expuestas a una luz cuya longitud de onda es menor a 4,000 Angstroms (Å). El procedimiento de borrado recomendado es exponer a una luz ultravioleta cuya longitud de onda sea 2,537 Å (debe notarse que la luz del Sol y ciertas lámparas fluorescentes tienen longitudes de onda entre 3,000 y 4,000 Å).

La dosis acumulada (intensidad de UV por tiempo de exposición) para el borrado debe ser al menos 15 Wseg/cm². El tiempo de borrado es aproximadamente 21 minutos usando una lámpara ultravioleta de 12 mW/cm². Durante el borrado el circuito integrado debe colocarse a una distancia menor a 2.54 cm de la lámpara. Se puede causar un daño permanente cuando la dosis acumulada en la vida del circuito de memoria excede a 7,258 Wseg/cm² o por la exposición a luz ultravioleta de alta intensidad por períodos largos.

Operación del dispositivo

Los seis modos de operación de las memorias se ilustran en la tabla 4.1.

MODO	$\overline{CE}$	$\overline{OE}$	VPP	SALIDAS
LECTURA	VL	VL	VCC	SALIDA DE DATOS
SALIDA DESHABILITADA	VL	VH	VCC	ALTA IMPEDANCIA
EN ESPERA	VH	X	VCC	ALTA IMPEDANCIA
PROGRAMACIÓN	PULSO EN NIVEL ALTO	VH	VPP	ENTRADA DE DATOS
VERIFICACIÓN	VL	VL	VPP	SALIDA DE DATOS
INHIBIR PROGRAMACIÓN	VH	X	VPP	ALTA IMPEDANCIA

a) Memoria 27C16

MODO	$\overline{CE}$	$\overline{OE}$/VPP	$\overline{PGM}$	SALIDAS
LECTURA	VL	VL	VH	SALIDA DE DATOS
SALIDA DESHABILITADA	VL	VH	VH	ALTA IMPEDANCIA
EN ESPERA	VH	X	X	ALTA IMPEDANCIA
PROGRAMACIÓN	VL	VPP	VL	ENTRADA DE DATOS
INHIBIR PROGRAMACIÓN	VH	X	X	ALTA IMPEDANCIA

b) Memoria 27C32

MODO	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	VPP	SALIDAS
LECTURA	VL	VL	VH	VCC	SALIDA DE DATOS
SALIDA DESHABILITADA	VL	VH	VH	VCC	ALTA IMPEDANCIA
EN ESPERA	VH	X	X	VCC	ALTA IMPEDANCIA
PROGRAMACIÓN	VL	VH	VL	VPP	ENTRADA DE DATOS
VERIFICACIÓN	VL	VL	VH	VPP	SALIDA DE DATOS
INHIBIR PROGRAMACIÓN	VH	X	X	VPP	ALTA IMPEDANCIA

c) Memorias 27C64 y 27C128

Tabla 4.1 Modos de operación de las EPROM

Modo de lectura

Las memorias tienen dos terminales de control y ambas deben activarse para obtener en las salidas el contenido de la localidad indicado en las terminales de dirección.

$\overline{CE}$ es la línea de control de energía y debe usarse para tener acceso al dispositivo. $\overline{OE}$ es la línea de control de salidas para habilitar los datos en las terminales, independientemente de la selección del dispositivo.

Modo "en espera"

El modo "en espera" reduce la corriente de alimentación, desde el valor de activación (8 mA) hasta el valor de "espera" (1 mA). Este modo se selecciona aplicando una señal TTL alta a la terminal $\overline{CE}$, en este modo las salidas están en alta impedancia, sin importar el estado de la terminal $\overline{OE}$. Estas dos líneas de control permiten la conexión de grandes arreglos de memoria, disminuir el consumo de energía y evitar conflictos de *bus*.

Para utilizar estas dos líneas de control, $\overline{CE}$ debe ser decodificada y utilizada como función primaria de selección de dispositivo, mientras $\overline{OE}$ debe ser una conexión común para todos los dispositivos en el arreglo y conectada a la línea *READ* del *bus* de control del sistema, esto asegura que todos los dispositivos de memoria que no hayan sido seleccionados estén en modo "en espera" de bajo consumo de energía y que sus terminales de salida estén en alta impedancia.

Cuando la memoria entra o sale del estado "en espera", se presentan efectos transitorios en el suministro de corriente, por lo que se recomienda colocar un capacitor de cerámica entre Vcc y GND lo más cercano posible a cada dispositivo. Adicionalmente, se debe colocar un capacitor electrolítico de 4.7 uF por cada arreglo de ocho dispositivos.

Modo de programación

Después de cada borrado, todos los *bits* de la EPROM están en 1 lógico. Los datos introducidos programan 0's lógicos de modo selectivo en las localidades deseadas y los *bits* que no deban ser afectados permanecerán en 1 lógico. La única forma de convertir 0's lógicos en 1's lógicos es mediante el borrado ultravioleta.

El dato a ser programado se presenta (*bit* a *bit*) antes de seleccionarse el modo de programación en un estado de salida deshabilitada o de inhibir programación en las terminales de datos, que solamente en este modo actúan como entradas. La dirección de memoria debe indicarse previamente a la programación. Los niveles requeridos para la dirección y el dato son niveles de entrada TTL.

En los tipos 27C128 y 27C64, el modo de programación se selecciona fijando la terminal Vpp en 12.5 V, y las señales $\overline{CE}$ y $\overline{PGM}$ en nivel TTL bajo. Los tipos 27C32 y 27C16 carecen de la señal $\overline{PGM}$ y en este último tipo, Vpp se fija a 12.5 V para que posteriormente en $\overline{CE}$ se aplique una transición de niveles de 0 lógico a 1 lógico.

El tiempo de programación en cada localidad de memoria debe ser de 50 mS, con una tolerancia del 10%, pudiéndose programar cualquier localidad en cualquier momento (de modo individual o secuencial).

Programación en paralelo

La programación de múltiples memorias del mismo tipo, con los mismos datos se puede llevar al cabo fácilmente conectando en paralelo las entradas de las memorias.

La programación con diferentes datos se puede llevar al cabo empleando el modo inhibir programación. Un nivel alto en las entradas $\overline{CE}$ o $\overline{PGM}$ inhibe a la memoria que no se desea programar.

Verificación

La verificación debe realizarse en los *bits* programados para determinar si éstos se han grabado correctamente. La verificación se realiza con $\overline{CE}$ y $\overline{OE}$ en nivel bajo, $\overline{PGM}$ en nivel alto y Vpp a 12.5 V. La verificación es semejante a la lectura pero con Vpp en nivel de programación. Este estado no existe en la memoria 27C32.

4.2.2 SN74LS374 *Latch* óctuple

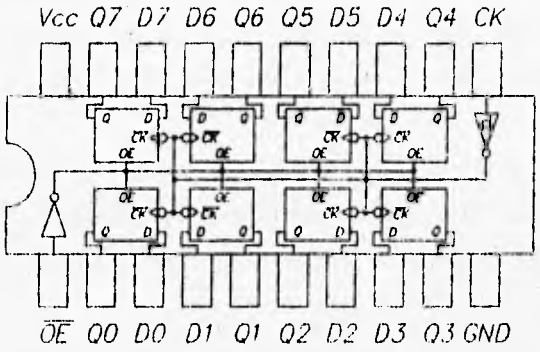


TABLA DE VERDAD

HABIL. $\overline{OE}$	RELOJ CK	DATO Dn	SALIDA On
L	↑	H	H
L	↑	L	L
L	↓	X	Q n-1
H	X	X	Z

FIG. 4.6 CIRCUITO SN74LS374

El circuito integrado SN74LS374 está compuesto por ocho ***flip-flops*** **tipo D, de la familia TTL de bajo consumo de energía, con entradas independientes y salidas de 3 estados, orientado para aplicaciones de** *bus*. **Está diseñado para operar en un rango de temperatura desde 0°C hasta 70°C. La tensión de alimentación es 5 V, con una tolerancia del 10% y el consumo de corriente en el caso máximo (alta impedancia) es 40 mA.**

Tiene señales comunes de reloj CK y habilitación de salidas $\overline{OE}$ para los ocho ***flip-flops***. **Posee histéresis en la señal de reloj para mejorar el margen de ruido y diodos en las terminales de entrada cuya función es limitar los efectos de alta velocidad.**

La figura 4.6 muestra un diagrama de bloques interno, la disposición de terminales y la tabla de verdad de este circuito.

4.2.3 SN74LS245 Manejador de *buses*

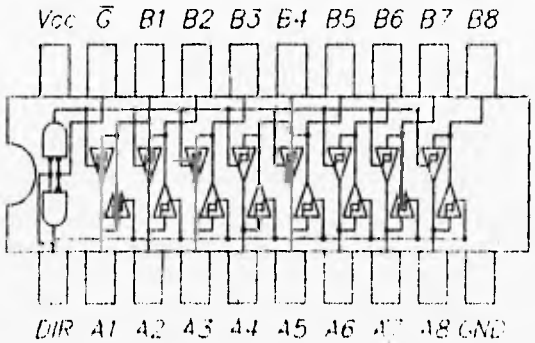


TABLA DE VERDAD

HABIL. $\bar{G}$	CONTROL DE DIRECCION DIR	OPERACION
L	L	DATOS DE BUS A HACIA B
L	H	DATOS DE BUS B HACIA A
H	\	ALTA IMPEDANCIA

FIG. 4.7 CIRCUITO SN74LS245

El circuito integrado SN74LS245 está formado por ocho trans-receptores de *bus*, de la familia TTL con salidas de 3 estados y de bajo consumo de energía. Está destinado para la comunicación asíncrona entre dos *buses* de datos en ambas direcciones y diseñado para operar en un rango de temperatura desde 0°C hasta 70°C. La tensión de alimentación es 5 V ±10% y el consumo de corriente en el caso máximo (alta impedancia) es 95 mA.

Este dispositivo permite la transmisión de datos del *bus* A al *bus* B o viceversa, dependiendo del nivel lógico en la terminal de entrada DIR. El tiempo típico de propagación de datos es 8 nS. La terminal $\bar{G}$ se utiliza para habilitar o deshabilitar las salidas, de modo que los *buses* A y B puedan estar aislados.

La figura 4.7 muestra un diagrama de bloques interno, la disposición de terminales y la tabla de verdad de este circuito.

4.2.4 SN74LS368 Inversor séxtuple de 3 estados

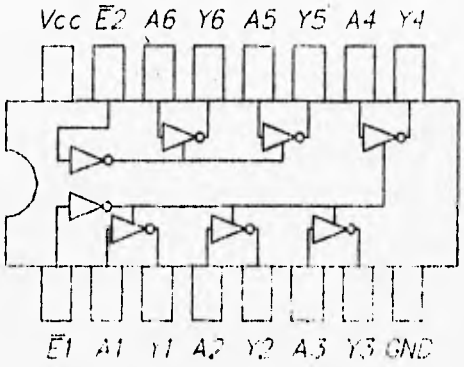


TABLA DE VERDAD

HABIL. $\overline{E}$	ENTRADA A	SALIDA Y
L	L	H
L	H	L
H	X	ALTA IMPEDANCIA

FIG. 4.8 CIRCUITO SN74LS368

El circuito integrado SN74LS368 está formado por seis *buffers* inversores de alta velocidad, de la familia TTL de bajo consumo de energía, con salidas de 3 estados, organizado en dos secciones de 2 y 4 *bits* cada una. Está diseñado para operar en un rango de temperatura desde 0°C hasta 70°C. La tensión de alimentación es 5 V con una tolerancia del 10% y el consumo de corriente en el caso máximo (alta impedancia) es 21 mA.

Cuando la terminal de habilitación $\overline{E}$ es 1 lógico, las salidas de los inversores son forzadas al estado de alta impedancia.

La figura 4.8 muestra un diagrama de bloques interno, la disposición de terminales y la tabla de verdad de este circuito.

4.3 CONSTRUCCIÓN DEL PROTOTIPO

En este apartado se detalla la lista de partes que integran al Programador y se describen e ilustran los procedimientos que deben seguirse para el montaje de los componentes en el circuito impreso y del armado del gabinete.

4.3.1 Lista de partes

En la tabla 4.2 se indican todos los elementos que forman al Programador, indicando nomenclatura, descripción y en su caso marcas.

NOMENCLATURA	DESCRIPCIÓN
R1	Resistor 10 MΩ
R2..R4	Resistor 4.7 KΩ
R5..R8	Resistor 12 KΩ
R9	Resistor 22 KΩ
R10	Resistor 5.6 KΩ
R11	Resistor 8.2 KΩ
R12	Resistor 2.2 KΩ
R13, R14	Resistor 5.6 KΩ
R15, R16, R18	Resistor 1 KΩ
R17	Resistor 100 Ω
R19	Resistor 470 Ω
R20	Resistor 10 KΩ
C1..C4	Capacitor 22 uF 16 V. de tantalio
C5, C6	Capacitor 22 pF de cerámica
C7	Capacitor 1 uF, 16 V. electrolítico
C8, C11..C15, C17, C18	Capacitor 0.1 uF de cerámica
C9, C10	Capacitor 330 pF de cerámica
C16	Capacitor 2,200 uF, 25 V. electrolítico radial marca Siemens o CGE
D1..D7	Diodo 1N4148 Motorola ó T.I.
D8	Puente de diodos 1 Amp.
D9	Diodo Zener 1N964B, 13 V. marca Motorola
IC1	Microcontrolador Motorola MC68HC11A1
IC2	ICL232 Manejador de línea
IC3, IC4	74LS374 *Flip Flop* D óctuple
IC5	74LS245 Transreceptor de *bus*
IC6	74LS368 Buffer inversor séxtuple
IC7	LM7805 Regulador 5 Volts 1.5 Amp. encapsulado TO-220AB

NOTA: Todos con 5% de tolerancia y ¼ W. de potencia

NOMENCLATURA	DESCRIPCION
LD1 LD2	LED color verde, 5 mm. de diámetro de 10 mA LED color rojo, 5 mm. de diámetro de 10 mA
T1..T5, T7 T6	Transistor BC 547C marca National Transistor BC 556 marca National
XTAL	Cristal de cuarzo de 8 MHz
RLY-Vcc, RLY-16, RLY-32, RLY-128	Relevador miniatura 5 V., DF2E-DC5V marca Aromat
DB25P	Conector macho DB25P de 25 terminales
TR1	Transformador 115/12 VCA 500 mA
1 Pza. 1 Pza	Conector puente con paso 2.54 mm. Poste recto duplex con paso de 2.54 mm.
1 Pza.	Circuito impreso doble cara (figura 4.9)
B1 B2	Base ZIF (cero fuerza de inserción) de 28 terminales marca Textool/3M Base de 48 terminales
1 Pza.	Caja metálica marca PRODEL: 13.97 mm. frente, 11.43 mm. fondo, (con tornillos) 6.35 mm. altura
4 Pza. 8 Pza.	Poste roscado de aluminio de 0.4 cm. de diámetro X 0.47 cm. de largo Tornillo de alumnio con cabeza plana de 0.4 cm. de diámetro X 0.47 cm. de largo
1 Pza.	Cable dúplex de 1.5 mts. calibre 16 con clavija y pasacable de hule
5 Pza.	Tornillos de 0.317 cm. X 1.27 cm. con tuerca
1 Pza.	Disipador de calor para encapsulado TO-220
1 Pza.	Interruptor 1P1T
2 Pza. F1 F2	Portafusible de paso Fusible 100 mA de vidrio Fusible 500 mA de vidrio
1 pza	Cable serie macho-macho, DB25-DB25/9

Tabla 4.2 LISTA DE PARTES

4.3.2 Circuito impreso

Los componentes electrónicos del Programador de Memorias se montan mediante soldadura en un circuito impreso de doble cara, cuyas caras superior e inferior se ilustran en la figuras 4.9aa y 4.9bb respectivamente.

La memoria a programar se coloca sobre una base de 28 terminales del tipo "cero fuerza de inserción". Esta base y los leds LD1 y LD2 se colocan sobre la cara superior del circuito impreso y se sueldan por la cara inferior, como se ve en la figura 4.9aa; sobre la cara inferior se localiza el resto de los componentes, como se muestra en la figura 4.9bb.

El microcontrolador se sitúa sobre una base para facilitar su reemplazo, la cual se suelda por la cara superior. Los relevadores se montan directamente y sólo se sueldan por la cara superior.

Las terminales del resto de los componentes: transistores, resistores, capacitores, diodos y circuitos integrados se sueldan en alguna cara o en ambas dependiendo del caso.

Los puentes o pasos de un cara a otra que se construyen con un alambre calibre 22 soldado por ambas caras. El puente JP1 se coloca sobre dos postes o *headers* rectos.

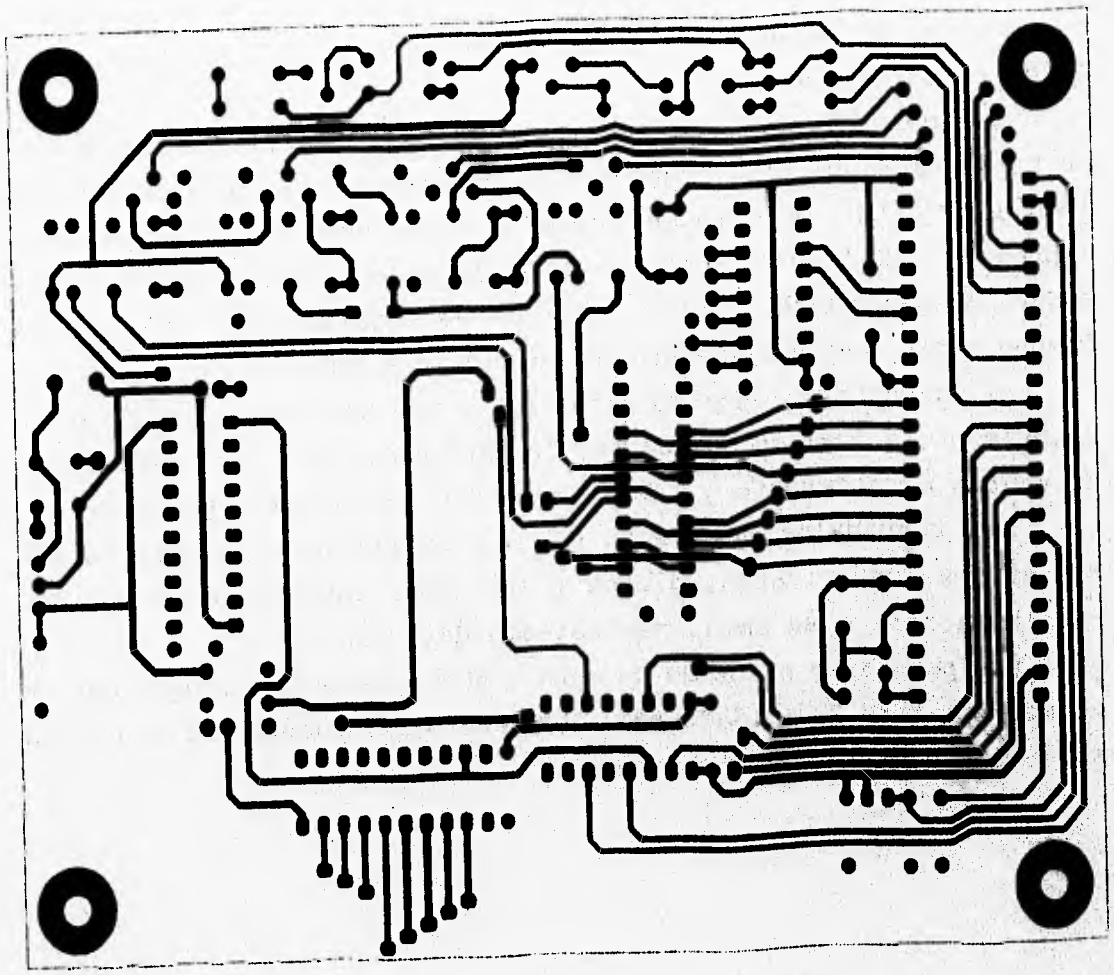


FIG. 4.9a PISTAS DE LA CARA SUPERIOR

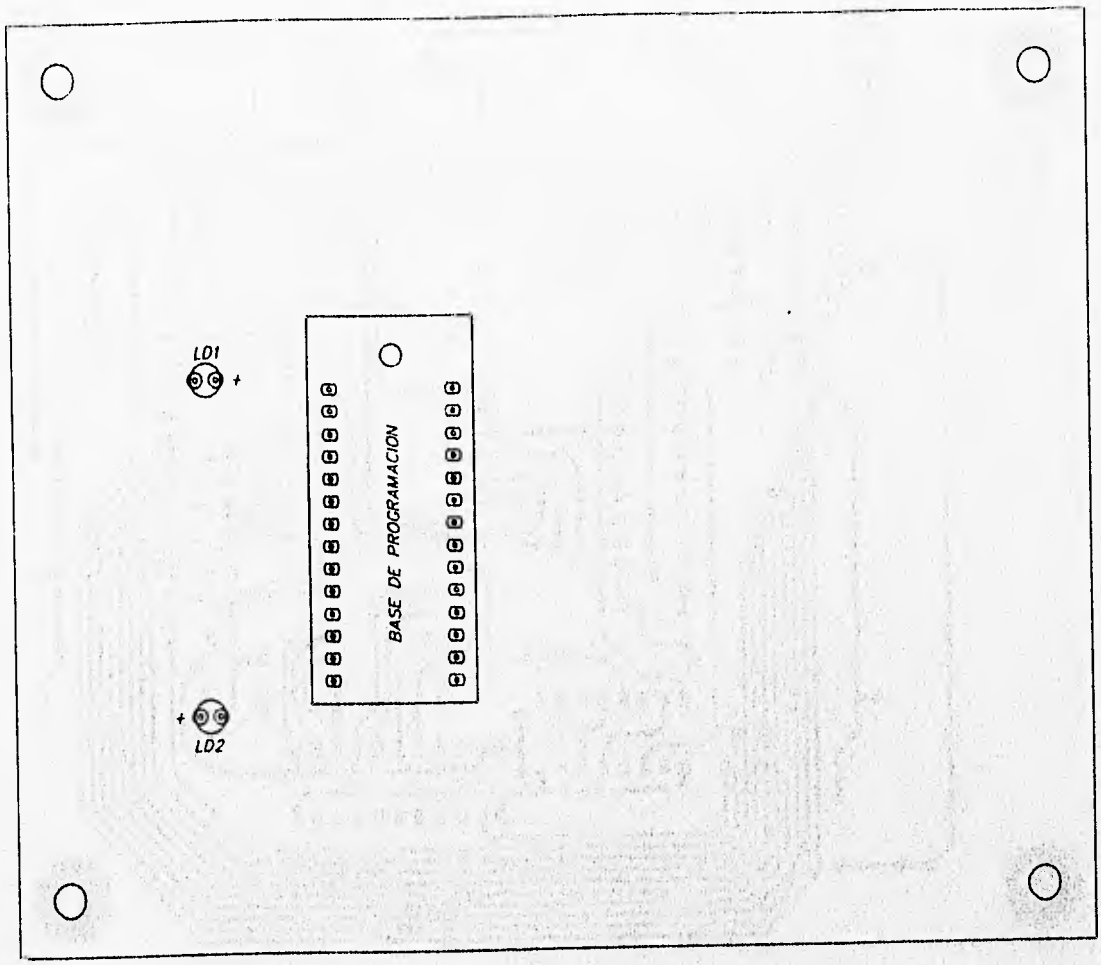


FIG. 4.9aa COMPONENTES DE LA CARA SUPERIOR

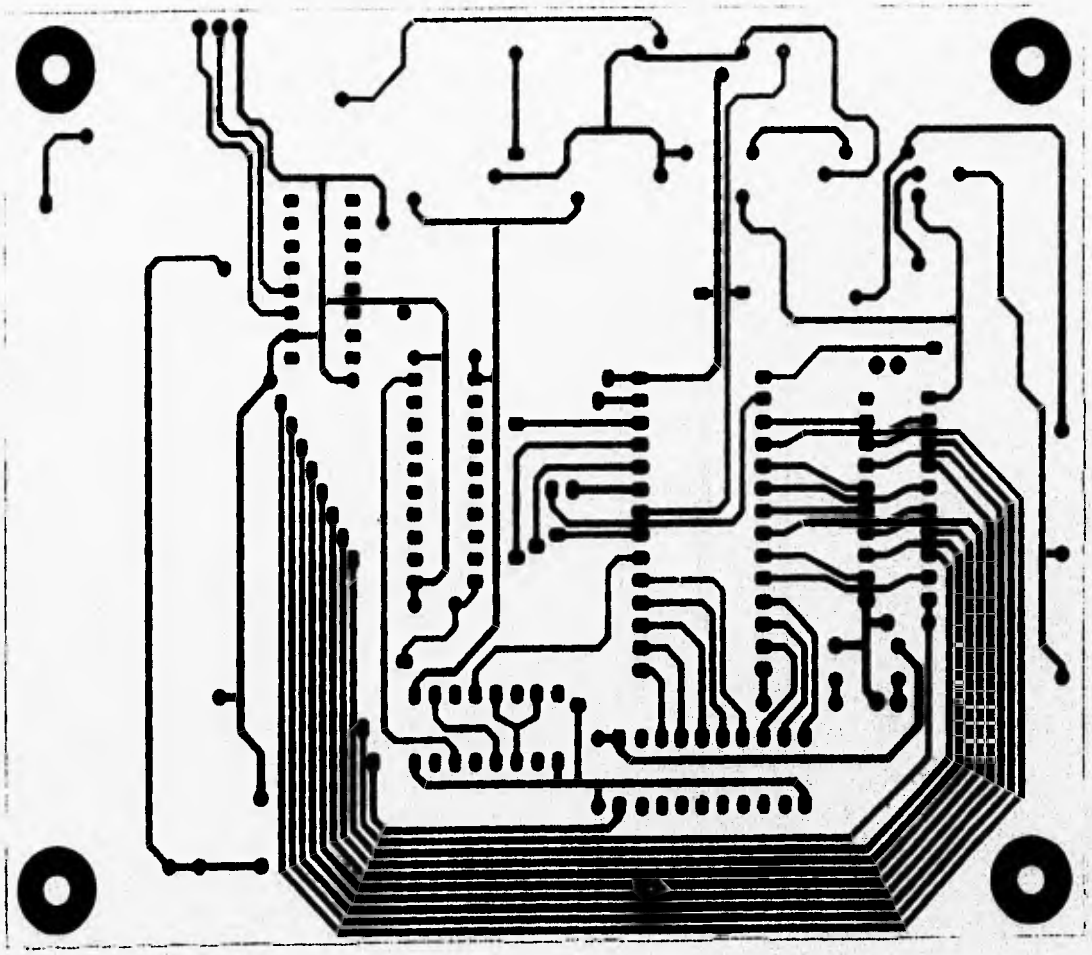


FIG. 4.9b PISTAS DE LA CARA INFERIOR

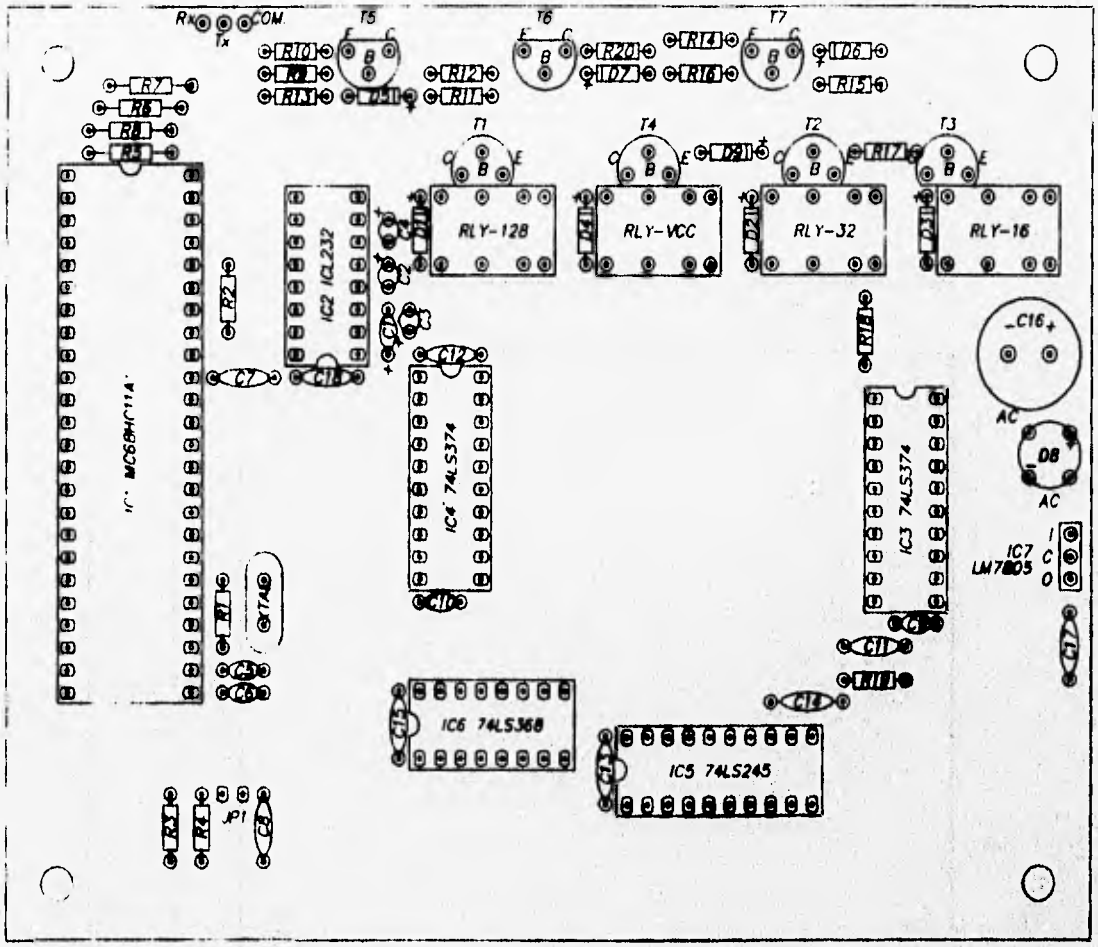


FIG. 4.9bb COMPONENTES DE LA CARA INFERIOR

Las dimensiones de la placa de circuito impreso se muestran en la figura 4.9c.

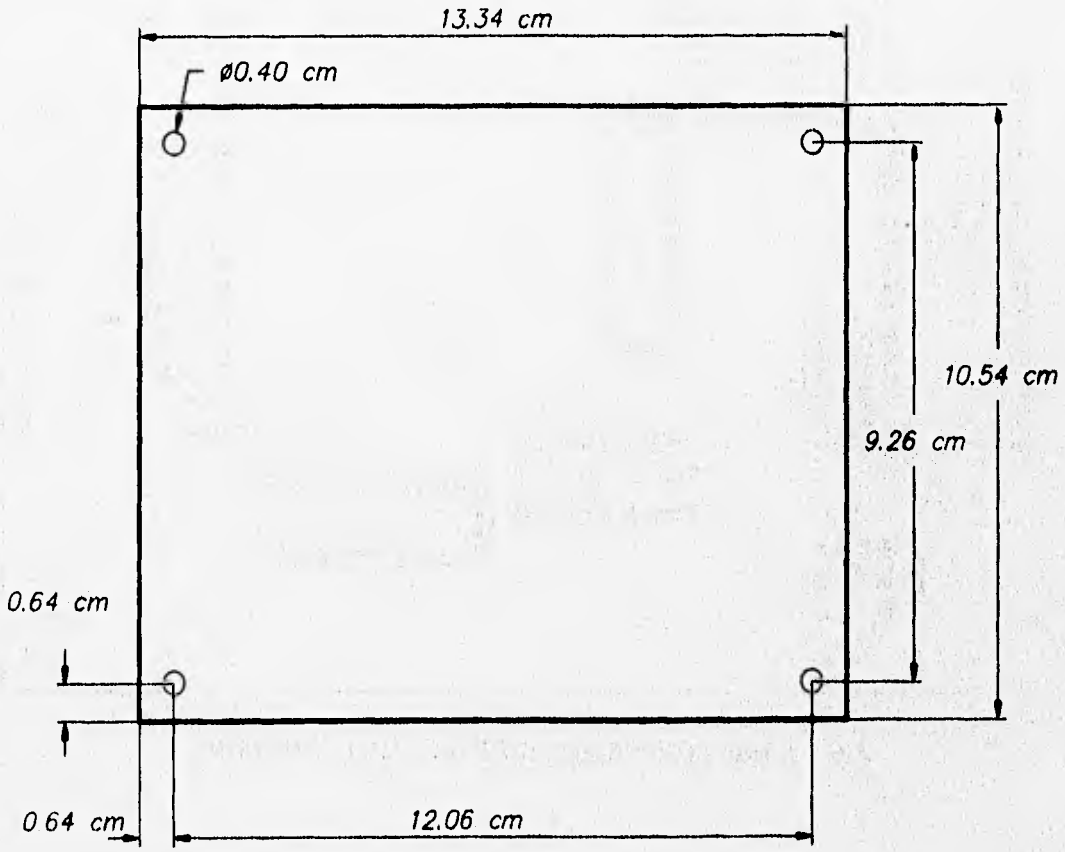


FIG. 4.9c DIMENSIONES DEL CIRCUITO IMPRESO

4.3.3 Montaje en el gabinete

El circuito impreso con los componentes electrónicos se fija mediante espaciadores o postes roscados en un gabinete de aluminio anodizado. Los postes se fijan a la tapa del gabinete por medio de cuatro tornillos, a la vez que otros cuatro fijan el circuito impreso a los postes. Como se ve en la figura 4.10a.

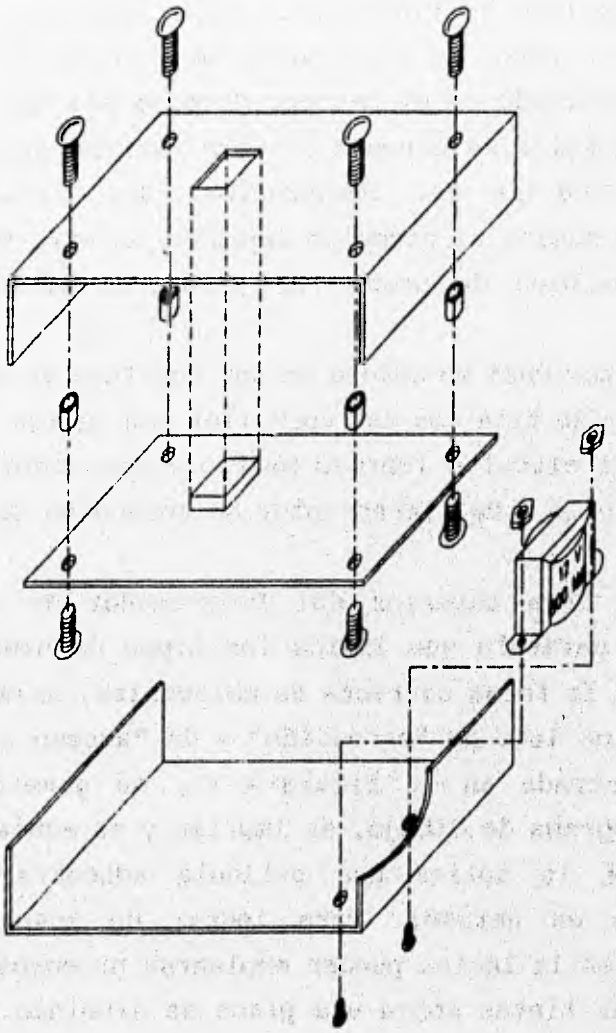


FIG. 4.10a MONTAJE DEL CIRCUITO IMPRESO

La abertura de la tapa debe coincidir en tamaño y localización con la base de programación para permitir que ésta sobresalga de la tapa y que de esta forma el usuario coloque la memoria a programar desde una posición cómoda. Asimismo los leds de indicación de "Encendido" y "Acceso a memoria" deben coincidir con su correspondiente orificio.

En el piso del gabinete se sujeta el transformador con dos tornillos con tuerca. Una punta del devanado primario se conecta al fusible encapsulado de vidrio (F1) tipo americano colocado en un portafusible de paso. La otra punta se conecta al interruptor de encendido localizado en el costado derecho del gabinete. El cable externo con clavija se conecta al otro extremo del portafusible y a la otra terminal del interruptor. Una punta del devanado secundario se suelda al circuito impreso, la otra punta se conecta a un portafusibles de paso (F2) y éste a su vez al circuito impreso.

El conector DB25 se coloca en una abertura en la parte derecha del gabinete y se fija con dos tornillos con tuerca. Sus terminales se conectan al circuito impreso mediante tres cables. La posición del conector DB25 y del interruptor de encendido se muestran en la figura 4.10b.

Para la cara superior del Programador de memorias se ha diseñado una carátula que indica los tipos de memoria que pueden programarse y la forma correcta de colocarlos, asimismo muestra la posición de los leds de "encendido" y de "acceso a memoria". Esta carátula, mostrada en la figura 4.11, se genera por medio de cualquier programa de dibujo, se imprime y se adhiere a la lámina. Finalmente se le aplica una película adhesiva plástica o un recubrimiento en aerosol. Para lograr un acabado más fino y profesional, en la lámina pueden emplearse procesos litográficos a base de varias tintas sobre una placa de aluminio.

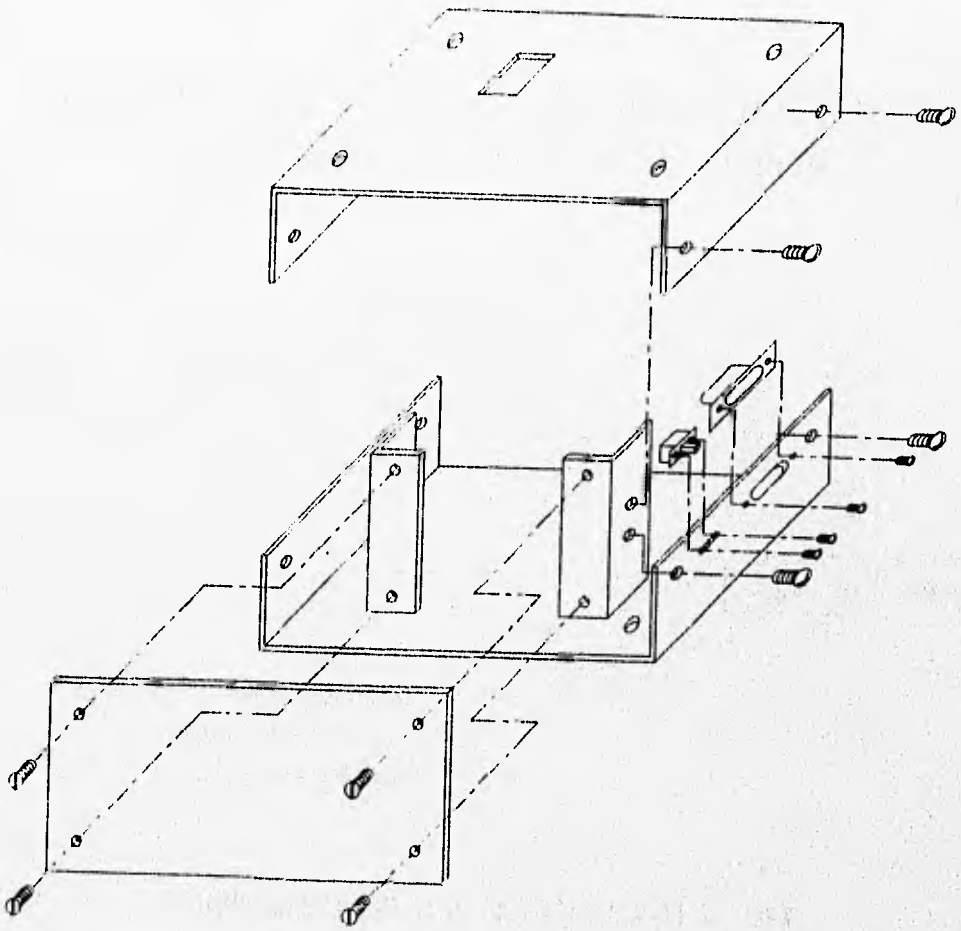


FIG. 4.10b ENSAMBLE DEL GABINETE

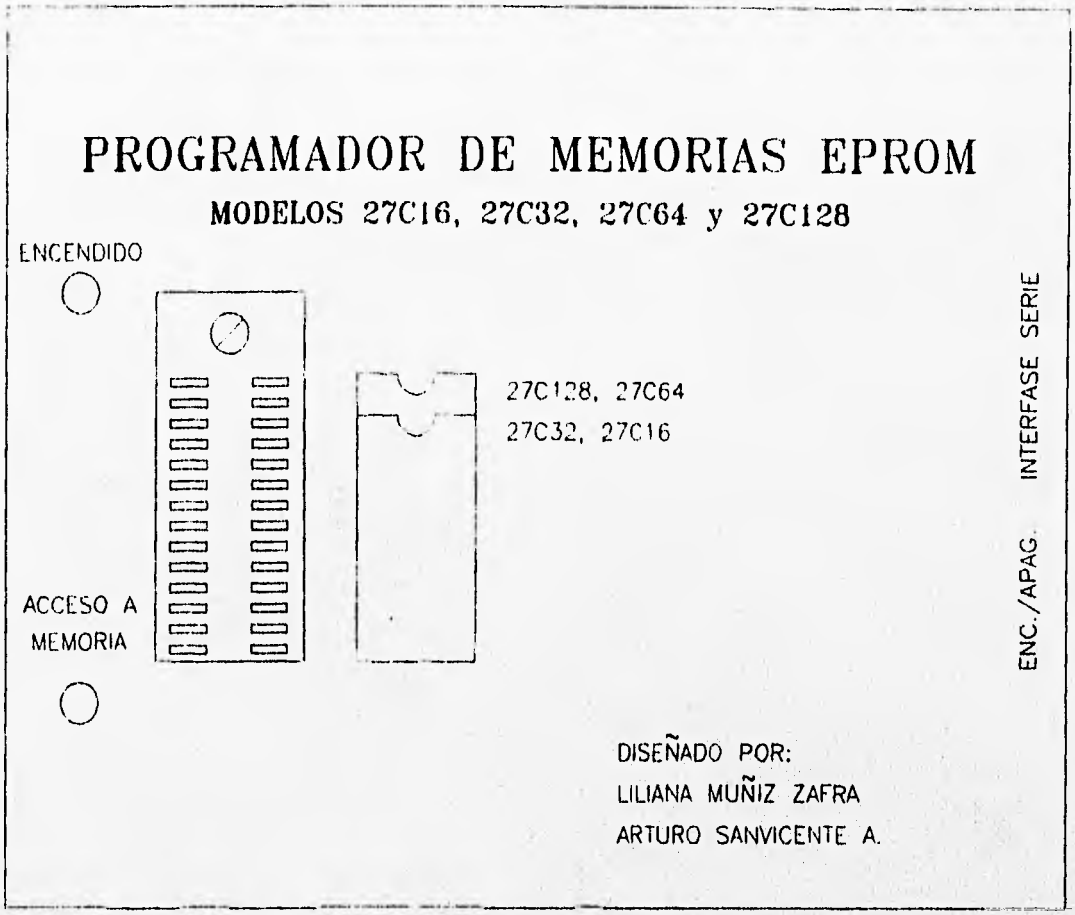


FIG. 4.11 CARATULA DEL PROGRAMADOR

CAPÍTULO V

SOFTWARE: PROGRAMAS DE APLICACIÓN

5.1 PROGRAMA DE OPERACIÓN DEL PROTOTIPO

La operación del prototipo se realiza por medio de la ejecución de un programa principal en el microcontrolador para realizar la lectura y escritura de la memoria EPROM; está codificado en lenguaje ensamblador 68HC11 de Motorola y mediante una transferencia se deja "residente" en la memoria interna EEPROM en el mismo microcontrolador.

Para alojar el programa principal en las localidades de memoria EEPROM se requiere de un programa intermedio o "cargador". Éste es un programa temporal que se almacena en la memoria RAM y únicamente se emplea para grabar el programa principal en la memoria EEPROM interna. Su ejecución sólo es necesaria para grabar por primera vez o modificar un programa en el microcontrolador.

5.1.1 Programa Principal

El programa de aplicación para la operación y funcionamiento completos del programador de memorias reside en la porción de memoria EEPROM interna, desde la localidad $B600_H$ hasta la $B790_H$ y se ejecuta automáticamente al encender el programador.

El programa ha sido desarrollado usando el macroensamblador AS11.EXE de Motorola con el fin de generar un programa objeto en formato S19. En la figura 5.1 se muestra el algoritmo de programación empleado. Lo forman un ciclo de captura de comando con sus parámetros y tres procedimientos principales: escritura, lectura y selección del tipo de memoria.

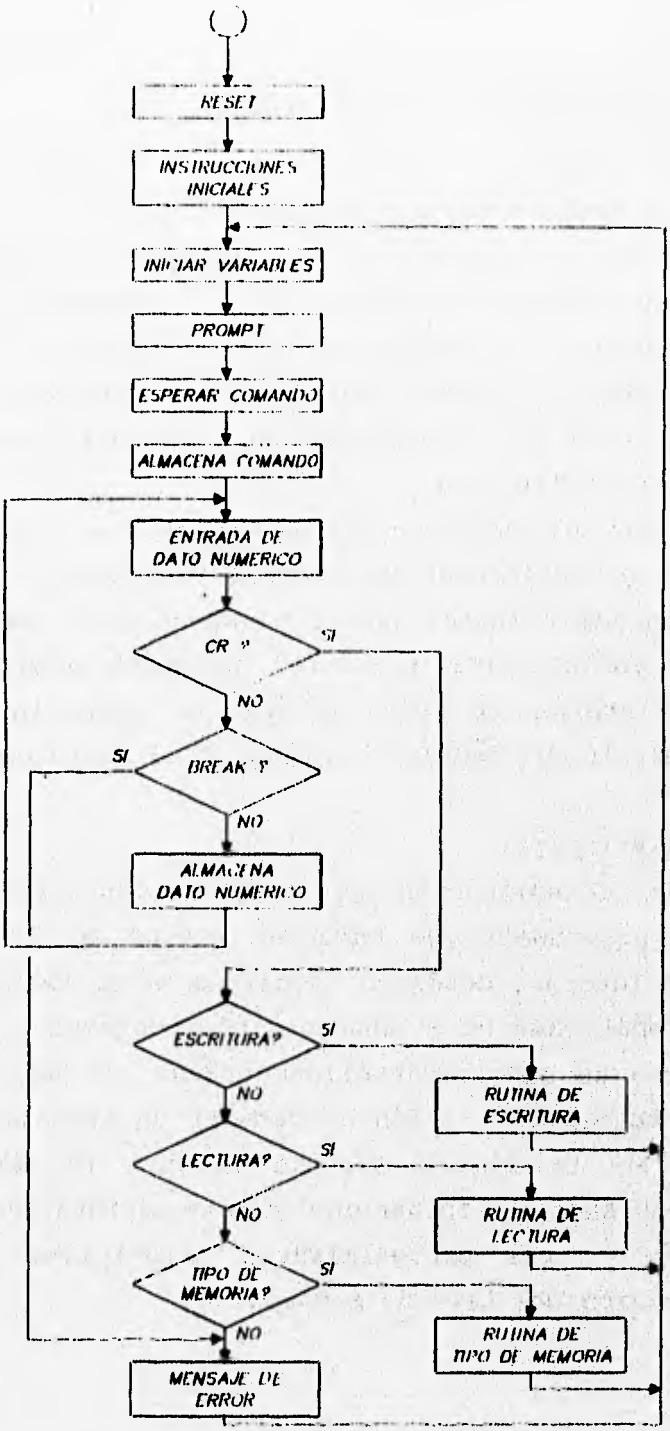


FIG. 5.1 DIAGRAMA DE FLUJO GENERAL

Cuando el microcontrolador sale del estado de *reset*, se programan el protocolo de comunicación serie de la interfase SCI (velocidad de transmisión 9,600 *bauds*, 8 *bits* de datos, un *bit* de paro, sin paridad) y la localización del apuntador de pila.

Se da un valor inicial (00_H) al área de RAM que sirve para alojar al comando y a los parámetros que se esperan recibir. El microcontrolador envía un *prompt* (indicador) señalando que está listo para recibir un comando. Enseguida se espera la llegada de un carácter por medio de la SCI que será tomado como comando, y los siguientes caracteres serán tomados como parámetros hasta la llegada del carácter CR (*Carriage Return* ó *Enter*) que indica la terminación del comando. Si en lugar de recibirse *Enter* o cualquier carácter numérico, se recibe otro distinto, el comando se cancela, se envía un mensaje de error y enseguida el *prompt*.

Al darse por recibido el comando y sus parámetros, se procede a la identificación del mismo para ejecutar el procedimiento correspondiente: lectura, escritura o especificación de tipo de memoria. En caso de un comando no reconocido, se enviará un mensaje de error y el *prompt*.

Los comandos deben tener la siguiente sintaxis:

TIPO DE MEMORIA:

T <tipo><Enter>

Donde:

tipo puede tomar cualquiera de estos valores:

VALOR	TIPO
60	27C128
40	27C64
50	27C32
48	27C16
otro	Alta impedancia

ESCRITURA:

E <Dir> <Dato1> [...<Dato32>]<Enter>

E: Comando de escritura

Dir: Dirección de memoria (de 2 *bytes*) a partir de la cual se graba.

Dato n: El dato a ser grabado (dato1 en dir1, dato2 en dir1+1, ...dato n en dir1+n-1)

LECTURA:

L <Dir1>[<Dir2>]<Enter>

L: Comando de lectura

Dir1: Dirección de memoria (de 2 *bytes*) inicial

Dir2: Dirección de memoria (de 2 *bytes*) final

Se debe indicar el tipo de memoria antes de cualquier operación de lectura o escritura, para seleccionar las líneas apropiadas de la base de programación. Para desenergizar la memoria y poder retirarla de la base de programación es necesario ejecutar la rutina de selección de memoria asignando un parámetro 00_H.

Para indicar que el comando ha terminado, el programa envía el *prompt*.

<Enter>**MILDRED>**

El procedimiento de lectura, que se ilustra en la figura 5.2, pone la dirección Dir1 en el *bus* de direcciones de la base de programación, coloca la memoria EPROM en estado de lectura, toma el valor del *bus* de datos y lo transmite por medio de la interfase SCI. Incrementa el valor de Dir1 y lo compara con Dir2, el procedimiento de lectura sigue en progreso mientras Dir1 es menor o igual a Dir2, o hasta que el usuario interrumpe el proceso al enviar cualquier carácter por la interfase SCI.

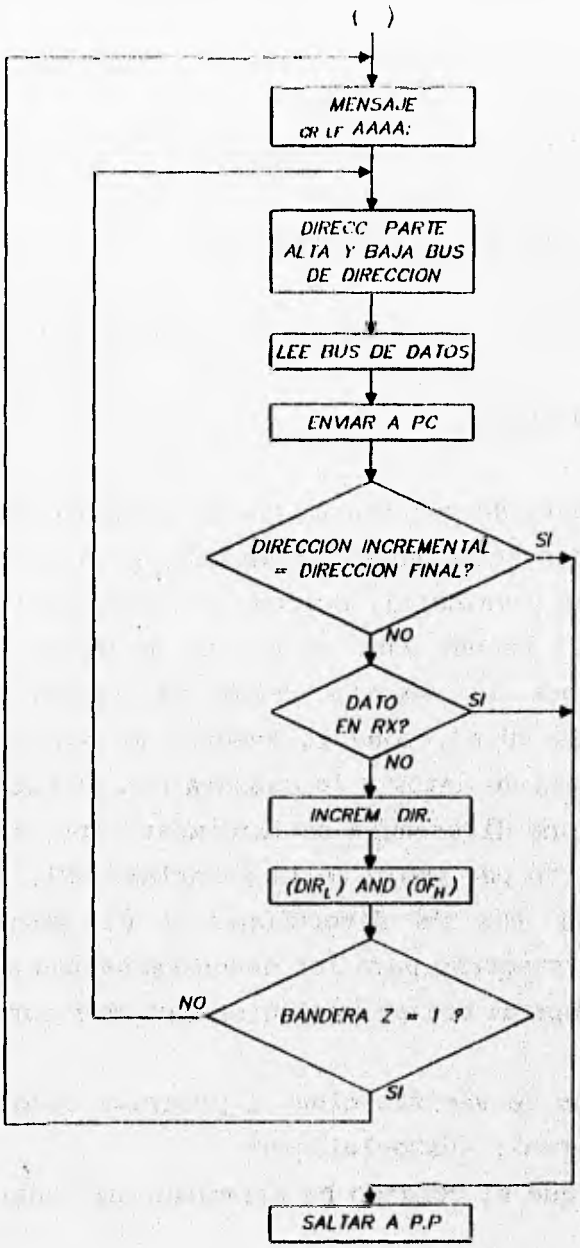


FIG. 5.2 DIAGRAMA DE FLUJO DE LECTURA

La respuesta de este procedimiento tiene la siguiente sintaxis:

<Dir1>: <dato1> [<dato2>...<dato16>]<LF><Enter>

si Dir2 > Dir1+16 envía otra línea

<Dir1+16>: <dato17>[...]<LF><Enter>

si Dir2 > Dir1+32 envía otra línea

<Dir1+32>: <dato33>[...]<LF><Enter>

y así sucesivamente.

El programa envía el *prompt* para indicar que el comando ha terminado.

<Enter>MILDRED>

El procedimiento de escritura, que se ilustra en la fig. 5.3, verifica que existan suficientes argumentos para la grabación (al menos una dirección y un dato), coloca la dirección Dir1 en el *bus* de direcciones y el primer dato en el *bus* de datos de la base de programación, coloca la memoria EPROM en estado de escritura durante un lapso de 50 mS, pone la memoria en estado de lectura, toma el valor del *bus* de datos y lo compara con el dato programado.

Si existiera una diferencia en la comparación, la dirección y el dato se transmiten por medio de la interfase SCI. Se incrementa la dirección del *bus* de direcciones y el procedimiento de escritura sigue en progreso para los datos restantes o hasta que el usuario lo interrumpa al enviar cualquier carácter por la interfase SCI.

Por cada error de verificación el programa desplegará:

<Enter><Dir-n>: <dato-leído-n>

Para indicar que el comando ha terminado el programa envía el *prompt*.

<Enter>MILDRED>

El programa P-PRINC puede ejecutarse para interactuar con el programa especial de usuario MILDRED.EXE o con cualquier *software* de emulación de terminal.

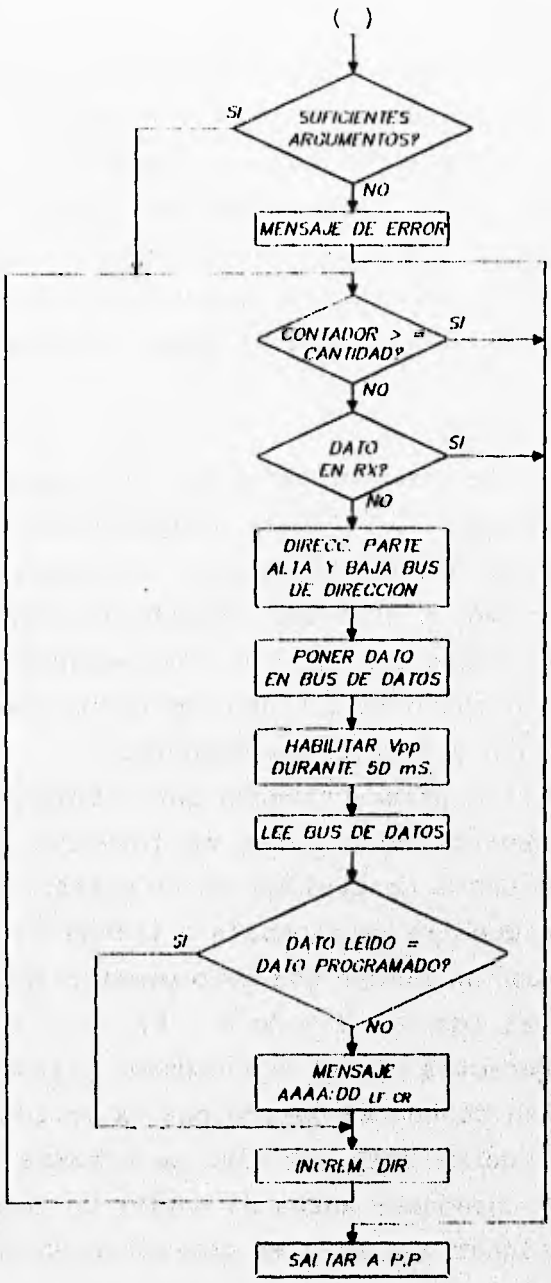


FIG. 5.3 DIAGRAMA DE FLUJO DE ESCRITURA

5.2 PROGRAMA DE APLICACIÓN DE LA PC

En la computadora personal se ejecutan dos programas: el primero es un programa llamado MILDRED.EXE (Manejador Inteligente, Lógico y Dinámico para la pRogración Eficiente de Datos) que sirve exclusivamente para la operación con el Programador, el segundo es un programa llamado PC_HC11.EXE diseñado para transferir el programa que se va a ejecutar en el microcontrolador. Esta transferencia se realiza únicamente cuando el microcontrolador es nuevo o cuando se desee realizar alguna actualización del programa.

5.2.1 Programa MILDRED.EXE

El objetivo de este programa es permitir la edición de datos y enviarlos al Programador, o realizar alguna operación de lectura de memoria y mostrar los datos en pantalla, con algunas facilidades adicionales para el usuario como son: manejo de archivos, edición de encabezado y selección de un tipo de memoria entre cuatro posibles. En un sólo archivo se guardan los datos (contenido de la memoria), el encabezado y el tipo de memoria.

El programa utiliza principalmente una matriz de tipo *string* de 1,024 por 16 elementos de 2 *bytes* de longitud cada uno para almacenar en RAM los datos contenidos en el editor.

El contenido se muestra en pantalla a través de 64 páginas de 16 por 16 elementos con un filtro que sólo permite introducir datos en formato hexadecimal (de 0 - 9 y de A - F).

El programa interactúa con el Programador y le envía comandos de lectura o escritura cuando se selecciona la opción de lectura o grabado en el editor de MILDRED. El tipo de memoria se especifica automáticamente al Programador antes de enviar un comando de estas operaciones, para asignar las señales correctas en las terminales y energizar la memoria.

En el caso de la grabación, todo el rango señalado por las direcciones inicial y final se divide en bloques de 16 *bytes* (como máximo), con el propósito de invocar un comando de escritura en la memoria con un parámetro de dirección y 16 de datos.

En la lectura se invoca un sólo comando al Programador con el rango de direcciones completo (sin dividirlo). Los datos que se reciben vienen en bloques de 16 *bytes* (como máximo) precedidos de una dirección, la cual se extrae e indica la posición en la matriz a partir de la cual se van almacenando los datos.

La figura 5.4 muestra el diagrama de flujo del programa **MILDRED.EXE**.

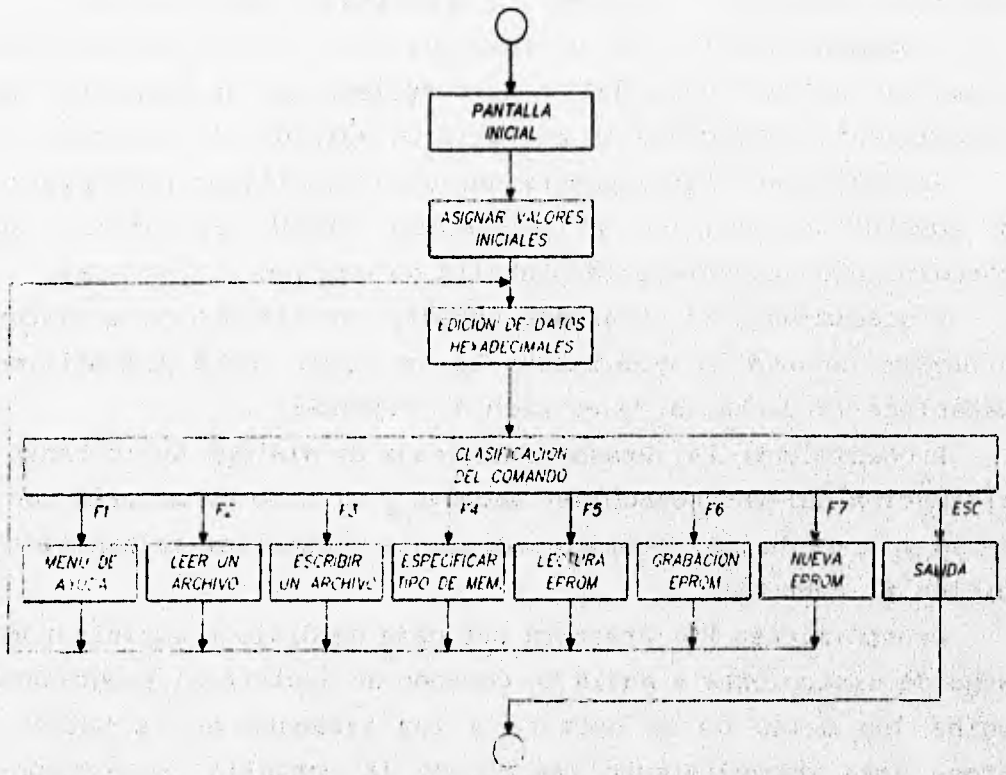


FIG. 5.4 DIAGRAMA DE FLUJO "MILDRED.EXE"

El programa está compuesto por 20 módulos que realizan una función específica, cuyos argumentos son de tipo *string* (alfanumérico) o *integer* (numérico entero).

Procedimiento **CAPTURADIR**: Captura una dirección de memoria de 4 dígitos en formato hexadecimal, usando filtros de validación.

Procedimiento **CAPTURAHEX**: Captura los datos en formato hexadecimal en la pantalla de edición.

Procedimiento **CAPTURATEXT**: Captura los textos del encabezado de archivo y nombres de archivo.

Procedimiento **CUADRITO**: Dibuja un pequeño cuadro resaltado en el centro de la pantalla como base para los procedimientos que requieren desplegar y capturar información con el usuario.

Procedimiento **F1**: Es un procedimiento en el cual se tiene acceso al archivo AYUDA.TXT y lo despliega en la pantalla. Este procedimiento restablece la pantalla de edición al regresar.

Procedimiento **F2**: Presenta una caja de diálogo para capturar un nombre de archivo y recuperarlo desde el disco. Este procedimiento restablece la pantalla de edición al regresar.

Procedimiento **F3**: Presenta una caja de diálogo para capturar un nombre de archivo y almacenarlo en disco. Este procedimiento restablece la pantalla de edición al regresar.

Procedimiento **F4**: Presenta una caja de diálogo solicitando la información del encabezado del archivo y el tipo de memoria con la que se va a trabajar. Este procedimiento restablece la pantalla de edición al regresar.

Procedimiento **F5**: Presenta una caja de diálogo solicitando un rango de direcciones y envía un comando de lectura al Programador. Recibe los datos de la lectura y los almacena en la matriz de datos. Este procedimiento restablece la pantalla de edición al regresar.

Procedimiento **F6**: Presenta una caja de diálogo solicitando un rango de direcciones y envía varios comandos de escritura al Programador proporcionando los argumentos de los datos de la matriz. Este procedimiento restablece la pantalla de edición al regresar.

Procedimiento **F7**: Presenta una caja de diálogo solicitando la confirmación de reasignación de la matriz de datos y del encabezado de archivo con sus valores por omisión. Este procedimiento restablece la pantalla de edición al regresar.

Procedimiento **GETDIR**: Captura las direcciones inicial y final mediante dos llamadas al procedimiento CAPTURADIR y verifica que la dirección final sea mayor o igual a la dirección inicial y que ambas estén comprendidas en el rango correspondiente al tipo de memoria seleccionado.

Procedimiento **GETFILE**: Captura el nombre del archivo para los procedimientos F2 y F3 con el auxilio del procedimiento CAPTURATEXT.

Procedimiento **IMPRIMETODO**: Imprime un campo en las cajas de diálogo o la pantalla de edición.

Procedimiento **IMPRIMEUNO**: Imprime un carácter en video inverso e intermitente, señalando el carácter que se está editando.

Procedimiento **INICIAFF**: Reasigna la matriz de datos con "FF" y el encabezado del archivo con "espacios en blanco". Este procedimiento es llamado al inicio del programa y por el procedimiento F7.

Procedimiento **LEEHASTA**: Realiza la lectura del puerto tipo serie COM1 hasta que recibe el carácter especificado por el argumento.

Procedimiento **PAGINA**: Realiza un refrescamiento de pantalla cada vez que se requiere de un cambio de la página de edición.

Procedimiento **PAGINI**: Despliega la pantalla de presentación del programa.

Procedimiento **PANTALLA**: Despliega el fondo de la pantalla de edición.

5.2.2 Programa cargador PC_HC11 y método de carga

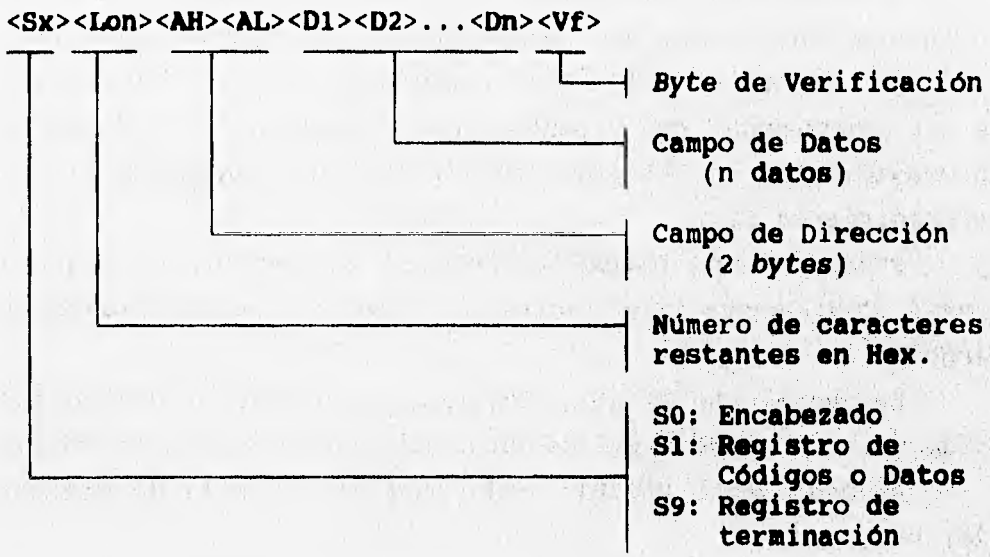
Este programa realiza la transferencia de un conjunto de instrucciones hacia el microcontrolador cuando éste es nuevo y carece de programa o cuando se desea actualizar el programa residente.

El programa del microcontrolador debe estar ensamblado previamente en formato Motorola S19 en un archivo llamado P-PRINC.S19 generado por el macroensamblador AS11.EXE de Motorola.

El formato S19 es desensamblado para encontrar las direcciones de memoria y los códigos de operación del programa P-PRINC.ASM. Enseguida envía un comando de modificación de memoria (**MM**) con los parámetros requeridos de dirección y datos al programa monitor (BUFFALO) del microcontrolador.

Al término de cada línea del archivo en formato S19, el programa cargador envía un carácter CR para finalizar el comando. Cada línea del archivo en formato S19 equivale en el microcontrolador a un comando completo de modificación de memoria.

El formato S19 de Motorola está estructurado como sigue:



Donde:

$$Vf = LSB(NOT(Lon+AH+AL+D1+...+Dn))$$

Los campos de dirección y datos se recuperan del archivo mediante instrucciones de manejo de ***strings***.

El procedimiento para alojar un programa de modo residente en la EEPROM del microcontrolador MC68HC11A1 requiere de un programa de enlace que permita la transferencia de los códigos de operación, realizando los siguientes pasos:

1. Colocar el puente JP1 en la tarjeta (llamar al programa monitor BUFFALO).
2. Ejecutar el programa PC_HC11 el cual espera el *prompt* de BUFFALO.
3. Encender el Programador
4. El programa PC_HC11 establece comunicación por si mismo con el Programador y reconoce el *prompt* del BUFFALO.
5. El programa PC_HC11 desensambla el archivo P-PRINC.S19, envía varios comandos de modificación de memoria (**MM**) y termina su ejecución.
6. Apagar el Programador y retirar JP1.

Al terminar este procedimiento, el microcontrolador se encuentra con el programa P-PRINC residente en memoria EEPROM.

CAPÍTULO VI

MANUAL DE OPERACIÓN

El Programador de Memorias funciona a través del uso de una computadora personal, para la que se ha diseñado un programa de aplicación que permite la introducción de los datos que se almacenarán en las memorias. Este programa envía al Programador los comandos necesarios para tal efecto.

El *software* diseñado para el Programador de Memorias se puede ejecutar sin la conexión al gabinete de programación para la edición de datos y almacenamiento de éstos en archivos en la computadora personal. Posteriormente se podrá conectar el Programador de Memorias para el intercambio de datos con la memoria en operaciones de lectura y/o de escritura.

6.1 ENCENDIDO DEL PROGRAMADOR DE MEMORIAS

Se debe conectar el cable de comunicación entre el Programador y el puerto COM1 de la computadora antes de encenderlos, eligiendo el cable respectivo para la interfase de 9 ó 25 terminales. El encendido del Programador y la computadora puede realizarse en cualquier orden. Este cable sólo podrá retirarse cuando el Programador y la computadora estén apagados.

La memoria a grabar o leer puede colocarse en cualquier momento en la base de programación, siempre y cuando el led de "Acceso a memoria" esté apagado, inclusive puede colocarse antes del encendido.

El Programador de Memorias se enciende accionando el interruptor colocado del lado derecho del gabinete y verificando que el led de "encendido" se ilumine.

6.2 EJECUCIÓN DEL PROGRAMA MILDRED.EXE

El software del Programador de Memorias está compuesto por un programa ejecutable: MILDRED.EXE y un archivo auxiliar: AYUDA.TXT, para ejecutar este programa se requiere de:

- Una computadora personal XT/AT con al menos 512 KB de memoria.
- Una unidad de disco flexible o fijo
- Un puerto de comunicación tipo serie COM1
- Un monitor color o monocromático con escala de grises
- Sistema operativo DOS 3.2 ó superior

El usuario puede crear un directorio de trabajo en el disco fijo o bien crear un disco flexible de trabajo, el cual debe contener a los archivos mencionados. En este mismo directorio se almacenarán los archivos de datos (*.MEM) de las memorias a ser programadas o leídas y el archivo de errores encontrados en las operaciones de escritura (MILDRED.ERR). Los archivos de datos pueden modificarse usando cualquier editor de texto tipo DOS.

```
FACULTAD DE INGENIERIA, U.N.A.M.
PROGRAMADOR DE MEMORIAS EPROM 27C16, 27C32, 27C64 y 27C128
AUTORES: ARTURO SANVICENTE A. Y LILIANA MUÑIZ ZAFRA
                                                          PAG. 1
DIRECCION:          CONTENIDO HEXADECIMAL                 ASCII
 0000  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0010  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0020  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0030  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0040  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0050  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0060  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0070  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0080  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 0090  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 00A0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 00B0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 00C0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 00D0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 00E0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................
 00F0  FF FF FF FF   FF FF FF FF   FF FF FF FF   FF FF FF FF ................

F1:AYUDA  F2:RECUPERAR F3:GUARDAR F4:TIPO MEMORIA F5:LEER F6:GRABAR F7:NUEVO
ESC = Salir    Utilice las flechas de navegación [HOME] [END] [PgUp] [PgDn]
```

Al ejecutar el programa MILDRED.EXE, momentáneamente se despliega una pantalla de presentación y posteriormente se tiene acceso directo a la primera página de la pantalla de edición del contenido de las localidades de memoria comprendidas entre 0000_H y $3FFF_H$. Los datos se introducen en formato hexadecimal y en la parte derecha de la pantalla se muestra el equivalente en código ASCII.

El cursor puede desplazarse utilizando las flechas de navegación y las teclas [Home], [End], [PgUp] y [PgDn]. La salida de cualquier función así como la del programa, se realiza por medio de la tecla [ESC].

Tecla	Función
PgUp	Retrocede una página de edición
PgDn	Avanza una página de edición
Home	Va a la primera localidad de la página
End	Va a la última localidad de la página
↑	Desplaza el cursor a la localidad superior. Cambia de página en caso de que el cursor esté en la primera línea.
↓	Desplaza el cursor a la localidad inferior. Cambia de página en caso de que el cursor esté en la última línea.
→	Desplaza el cursor a la localidad derecha. Cambia de línea en caso de que el cursor esté en la última localidad de la línea. Cambia de página cuando el cursor esté en la última localidad de la última línea.
←	Desplaza el cursor a la localidad izquierda. Cambia de línea en caso de que el cursor esté en la primera localidad de la línea. Cambia de página cuando el cursor esté en la primera localidad de la primera línea.
ESC	Sale del programa. Aborta cualquier opción.

6.2.1 Ayuda

El programa cuenta con una ayuda disponible en la pantalla de edición, la cual se invoca por medio de la tecla [F1]

```
F1:    Esta pantalla
F2:    Recuperar los datos desde un archivo
F3:    Guardar los datos hacia un archivo en un disco
F4:    Tipo de Memoria
       Definición del tipo de memoria a grabar y encabezado del archivo
F5:    Lectura de los datos contenidos en la memoria
       Despliegue de datos, dadas las direcciones de memoria inicial y final
F6:    Programación de la memoria
       Escritura de datos, dados: Dirección Inicial, dirección final y datos
F7:    Nuevo
       Restablece los datos originales y el encabezado del archivo
PgUp:  Ir a la página anterior
PgDn:  Ir a la siguiente página
Home:  Ir a la primera posición de la página
End:   Ir a la última posición de la página
ESC:   Salida del programa
```

6.2.2 Recuperar archivo

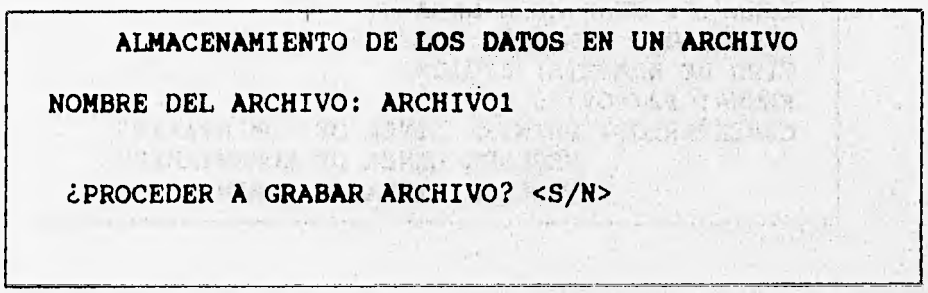
Para recuperar los datos almacenados previamente en un archivo, se presiona la tecla [F2], enseguida aparecerá la siguiente pantalla y se debe proporcionar el nombre del archivo que se desea recuperar, respetando la sintaxis propia del sistema operativo. La extensión del archivo siempre será .MEM y no será posible especificar otra.

```
        RECUPERAR DATOS DESDE UN ARCHIVO

NOMBRE DEL ARCHIVO: ARCHIVO1

LOS DATOS MOSTRADOS EN LA PANTALLA SE PERDERAN

 ¿PROCEDER A RECUPERAR ARCHIVO? <S/N>
```

Al recuperar el archivo, los datos nuevos se sobreescriben en los actuales en aquellas localidades que abarque el rango del tipo de memoria especificado en el archivo leído. Se debe confirmar esta operación para que pueda ser ejecutada. Si el archivo requerido no existe, el programa envía un mensaje de "Archivo no encontrado".

6.2.3 Guardar archivo

Los datos contenidos en las páginas de edición se pueden almacenar en un archivo presionando la tecla [F3], enseguida aparecerá la siguiente pantalla y en ella se debe proporcionar el nombre del archivo que contendrá dichos datos, debiéndose confirmar el grabado. Si el archivo especificado no se grabó (disco: sin formato, lleno, protegido contra escritura, ausente o unidad de disco no lista), el programa envía un mensaje de "Archivo no grabado".



6.2.4 Tipo de memoria

Esta opción permite especificar el tipo de memoria que se va a utilizar, para que el Programador asigne a las terminales de la base de programación las señales correspondientes. La capacidad del tipo de memoria especificado determinará el rango de localidades que se pueden modificar en el editor.

Se debe especificar el tipo de memoria antes de efectuar cualquier operación de lectura o de grabación. En este campo sólo se permite especificar los siguientes tipos de memoria: 27C128, 27C64, 27C32 y 27C16.

Esta opción también permite agregar la información del usuario necesaria para la identificación del contenido de la memoria. Los campos son únicamente de texto y se pueden modificar por medio de cualquier editor de texto tipo DOS.

```
     DEFINICION DEL TIPO DE MEMORIA A GRABAR
           Y ENCABEZADO DEL ARCHIVO

USUARIO: JUAN HERNANDEZ N.
NOMBRE DEL PROGRAMA: ENSAMBLADOR HC11
TIPO DE MEMORIA: 27C128
FECHA: 12/NOV/95
COMENTARIOS: PRIMERA LINEA DE COMENTARIOS
             SEGUNDA LINEA DE COMENTARIOS
             TERCERA LINEA DE COMENTARIOS
```

6.2.5 Lectura de la EPROM

Para efectuar un operación de lectura es necesario que la memoria ya esté instalada en la base de programación y que ésta no sea removida mientras el led de "Acceso a memoria" esté encendido. La memoria se debe colocar siguiendo el esquema indicado en la carátula del Programador. Se debe especificar el tipo de memoria antes de efectuar esta operación.

La lectura de la memoria EPROM se realiza presionando la tecla [F5]. Se debe indicar la dirección inicial así como la dirección final. Si la dirección final es menor que la dirección inicial, el programa indicará el error y volverá a preguntar por ambas. Es necesario confirmar esta operación para que pueda efectuarse.

```
LECTURA DE DATOS CONTENIDOS EN LA MEMORIA EPROM


          DIRECCION INICIAL:
          DIRECCION  FINAL :
¿PROCEDER CON LA LECTURA? <S/N>


PRESIONE CUALQUIER TECLA PARA INTERRUMPIR LA LECTURA
```

6.2.6 Grabado de la EPROM

Para efectuar una operación de grabado es necesario que la memoria ya esté instalada en la base de programación y que ésta no sea removida mientras el led de "Acceso a memoria" esté encendido. La memoria se debe colocar siguiendo el esquema indicado en la carátula del Programador. Se debe especificar el tipo de memoria antes de efectuar esta operación.

El grabado de la memoria EPROM se realiza presionando la tecla [F6]. Se debe indicar la dirección inicial así como la dirección final. Si la dirección final es menor que la dirección inicial, el programa indica el error y vuelve a preguntar por ambas. Es necesario confirmar esta operación para que pueda efectuarse. En la pantalla se indica la dirección que se está grabando al momento.

```
GRABADO DE DATOS CONTENIDOS EN LA MEMORIA EPROM

          DIRECCION INICIAL:
          DIRECCION  FINAL :

¿PROCEDER CON EL GRABADO? <S/N>

          GRABANDO DIRECCION:

PRESIONE CUALQUIER TECLA PARA INTERRUMPIR EL GRABADO
```

La grabación siempre se realiza con los datos que se encuentran en las páginas de edición, ya sea que provengan de un archivo o de una edición manual, es decir, para grabar la EPROM no es necesario que los datos provengan de un archivo en disco y tampoco es necesario guardarlos después de realizada la grabación de la EPROM.

Se pueden realizar múltiples duplicados del contenido de una memoria si se efectúa una lectura de ésta, enseguida se retira de la base de programación, se coloca una nueva memoria y se procede a la función de grabado. Estos datos no necesariamente se deben almacenar en un archivo. La fuente de datos también puede ser un archivo o la introducción manual en el editor.

Cuando se especifica el tipo de memoria no se alteran los datos contenidos en el editor, por lo tanto es posible realizar un copiado entre diferentes tipos de memorias, teniendo en cuenta las limitaciones de capacidad de cada uno.

El programa almacena en un archivo solamente las localidades de memoria que corresponden al tipo que se ha especificado, de este modo aunque el usuario edite datos en las direcciones desde 0800_H hasta $3FFF_H$, estos no serán almacenados si posteriormente se cambia el tipo de memoria especificado por una 27C16.

El usuario podrá hacer una comparación del contenido de dos memorias (aunque no sean del mismo tipo) si realiza una lectura de cada una, los datos se almacenan en sendos archivos y empleando el comando FC (*File Compare*) del DOS realiza la comparación de ambos archivos. Mediante el uso de un editor de texto DOS se pueden realizar combinaciones del contenido de dos archivos o realizar la repetición de rangos en un sólo archivo.

6.2.7 Archivo nuevo

En caso de que se desee la edición de nuevos datos para otra memoria, se selecciona esta opción presionando la tecla [F7] para de este modo reasignar a todos los datos y al encabezado del archivo con los valores por omisión.

```
ASIGNAR VALORES ORIGINALES A LOS DATOS MOSTRADOS
EN PANTALLA Y AL ENCABEZADO DEL ARCHIVO

¿PROCEDER A RE-INICIAR LOS DATOS DE LA PANTALLA?<S/N>
```

Es necesario confirmar esta operación para que pueda efectuarse. Se debe especificar nuevamente el tipo de memoria antes de una operación de lectura o grabado.

El tiempo requerido en este proceso varía de acuerdo a la velocidad y tipo de procesador de la computadora, por ejemplo, en una computadora 386SX toma alrededor de 4 segundos.

CONCLUSIONES

Para el diseño y construcción de un Programador de Memorias EPROM, objeto de la presente tesis, se decidió utilizar como base un microcontrolador, lo que permitiría un mayor rendimiento de su funcionamiento a la vez que facilitaría su propio diseño y construcción.

Para este efecto, se decidió utilizar un microcontrolador marca Motorola, porque dicha empresa presentó, explicó y proporcionó la documentación técnica de sus microcontroladores en un *Simposium* efectuado en la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, en 1991.

Motorola ofrece dos familias de microcontroladores: los modelos 'HC11 y los 'HC05 de 8 *bits*. De entre las dos familias se seleccionó al MC68HC11A1 debido a que posee las siguientes características:

- Permite la comunicación tipo serie asíncrona
- Tiene memorias RAM y EEPROM internas
- Posee un programa monitor residente en ROM
- Posee cinco puertos de entrada/salida
- Maneja un sistema de *timers*

Otros factores que se consideraron fueron la fácil adquisición en el comercio especializado local, su relativo bajo precio y la disposición de sus terminales en doble fila que facilitan la soldadura manual.

Para facilitar el desarrollo del prototipo, se adquirió un módulo de evaluación M68HC11EVBU con microcontrolador MC68HC11E9, cuyas características son muy parecidas a las del MC68HC11A1.

Las facilidades en el desarrollo del *software* del microcontrolador a que se hacen referencia son la posibilidad de ejecutar el programa paso a paso, la de ejecutar bloques separados y la de revisar o modificar el contenido de los registros. La ayuda en *hardware* consiste en la posibilidad de modificar directamente los datos en los puertos A, B y C y de verificar las señales en los *buses*.

Durante la etapa de diseño se armó una versión inicial del prototipo en tres tarjetas de inserción rápida, controlado con ayuda del módulo de evaluación citado. Posteriormente se llegó a una versión definitiva sobre un circuito impreso, en la que se sustituyó el módulo por el microcontrolador elegido: MC68HC11A1.

Las memorias más comunes en los proyectos que requieren utilizar un microprocesador, tienen capacidades que van desde 2 KB hasta 16 KB, para tal efecto, el prototipo del Programador de Memorias que se desarrolló es capaz de grabar memorias EPROM 27C16, 27C32, 27C64 y 27C128, cuyas capacidades están dentro del rango mencionado. La serie C (CMOS) se eligió porque requiere el mismo voltaje de programación en todos sus modelos, así como un bajo consumo de corriente en las fuentes de alimentación y de programación.

En el transcurso del desarrollo del prototipo se pensó en programar otros modelos de memorias EPROM que requerían tensiones de programación distintas entre sí: 25 volts para la memoria 2716 y 21 volts para las 2732A, 2764 y 27128. Esta decisión habría obligado al programador a manejar un cambio de tensión en el generador de "voltaje de programación" (25/21 V). Las soluciones que se estudiaron para resolver este problema fueron:

1.- Una fuente de voltaje independiente con un regulador de voltaje ajustable.
2.- Un multiplicador de tensión basado en capacitores y diodos.
3.- Una fuente por conmutación controlada por el mismo controlador y utilizando una entrada del convertidor A/D.
4.- Una fuente por conmutación utilizando el circuito LM497 a partir de la fuente de alimentación de 5 V.

Cuando se optó por emplear las memorias CMOS, se evitó este problema de manejo de diferentes tensiones.

Para permitir el intercambio de *chips* de memoria de la base de programación es necesario cortar la tensión de alimentación Vcc, lo cual puede hacerse mediante transistores BJT, FET o algún tipo de tiristor, con el inconveniente de disminución en el voltaje entregado (4.8 V. en lugar de 5 V.) y, consecuentemente, caer en un nuevo problema para elevar este voltaje. Se decidió utilizar un relevador con doble juego de contactos para desconectar también la tensión de programación Vpp (adicional a T6).

Durante el desarrollo del prototipo también se pensó usar el sistema interno de *timers* para lograr el pulso de programación en la terminal $\overline{\text{PGM}}$ de la memoria, por lo que se eligió la terminal OC1 para el manejo de esta señal, por ser la más poderosa de las funciones de salidas de este sistema. Sin embargo, la función de OC1 resultó ser ineficiente ya que requiere aproximadamente 80 *bytes* de memoria de programa para fijar el vector de interrupción y elegir el rango base; programar la interrupción para 50 mS; activarla y desactivarla; además de destinar una porción de memoria para la rutina de servicio de interrupción. La ventaja que tiene es que el programador puede recibir más datos al tiempo que tiene un ciclo de programación en progreso.

En consecuencia, en vez de manejar la interrupción para el pulso de programación se optó por hacerlo mediante un ciclo en el cual se decrementa el valor de un registro y requiere 7 *bytes* de memoria de programa.

Para programar las 16,384 localidades de una memoria 27C128 se requirieron 14.5 minutos, logrando un promedio de 53 mS por cada localidad. La duración del pulso de programación por localidad es de 50 mS, por lo que el tiempo restante se emplea para la comunicación Computadora-Programador y en la actualización de los *buses* de dirección y datos. Por lo tanto, la eficiencia en el aprovechamiento del tiempo es de 94% aproximadamente.

En el diseño del *software* se debe equilibrar entre la cantidad de memoria empleada y el tiempo de ejecución del programa. Es decir, un programa eficiente en tiempo de ejecución puede requerir más memoria que otro que tarde un poco de tiempo de más en ejecutarse. En este proyecto, el microcontrolador se utiliza en modo "un solo *chip*" y se dió preferencia a reducir la cantidad de memoria utilizada, para evitar utilizar memoria externa, limitando únicamente al uso de la EEPROM interna de 512 *bytes*, de la cual se emplean 440 *bytes* para el código del programa.

En cuanto a los aspectos materiales y costos del proyecto, todos los componentes fueron adquiridos en la Ciudad de México, con lo que se cumple uno de los requisitos primordiales planteados al inicio de este trabajo. El costo aproximado de los componentes al mes de enero de 1996 es de $ 520, sin embargo, existen además gastos indirectos (como pequeños tramos de cable, soldadura, pasta para soldar, solventes, etc.).

Hay que destacar, dentro del renglón de gastos adicionales para poder desarrollar el prototipo: un módulo de evaluación ($ 950), asi como el material fotográfico necesario para la manufactura del circuito impreso ($ 100).

Existen en el mercado varios tipos de programadores: Unos requieren insertar un tarjeta electrónica dentro de la computadora, otros modelos poseen un pantalla y teclado propios y funcionan de modo autónomo pero no poseen la ventaja del almacenamiento en disco, un tercer tipo, muy parecido al de este prototipo, se conecta únicamente al puerto serie de la computadora, pero presentan el inconveniente del precio (alrededor de $350 USD), aunque también programan toda la familia 25XX, 27XX, PALs, GALs, microcontroladores, y lo ponen fuera del alcance del estudiante o del principiante.

El prototipo desarrollado podría mejorarse adicionando al diseño básico las siguientes funciones:

- Copiado múltiple en forma simultánea utilizando varias bases de programación.

- Programación de otros modelos de EPROMs.

- Programación de otros dispositivos: PROMs, EEPROMs, PALs, GALs, PLDs y MPUs.

- Manejo de diferentes formatos binarios de información: INTEL Hex, Motorola S Hex, Tektronix Hex y JEDEC.

Este trabajo ha dejado, a los autores, los conocimientos y la experiencia para la realización de otros proyectos en el campo de la industria, basados en el microcontrolador MC68HC11.

El estudiante y el profesional podrán realizar aplicaciones con microprocesadores y prácticas de laboratorio, basados en la programación de una memoria no volátil.

BIBLIOGRAFÍA

IC Memories Data Book #M10
Hitachi, San José, Cal., USA., 1982.

IC Memories Data Book #M11
Hitachi, San José, Cal., USA., 1983.

Memory Components Handbook
Intel, Santa Clara, Cal., USA., 1983.

MC68HC11E9 Technical data /rev 1
Motorola Inc.
Phoenix, Arizona, USA., 1991.

M68HC11 Reference manual /rev 2
Motorola Inc.
Phoenix, Arizona, USA., 1991.

MC68HC11E9 Programming reference guide
Motorola Inc.
Phoenix, Arizona, USA., 1991.

M68HC11EVBU
Universal Evaluation Board User's Manual
Motorola Inc.
Phoenix, Arizona, USA., 1992.

MC68HC11 EEPROM Programming from a Personal Computer
Motorola Semiconductor Application Note AN1010
Motorola Inc.
Phoenix, Arizona, USA., 1988.

Motorola Schottky TTL Devices
Motorola, Phoenix, Arizona, USA., 1983.

The TTL Data Book for Design Engineers
Second Edition
Texas Instruments Inc., Dallas, Tx. 1976.

NAIS Relay Technical Data Book
Aromat Corporation, Japan 1993

Electrónica, teoría de circuitos
Robert Boylestad y Louis Nashelsky
Prentice Hall Hispanoamericana, Madrid, España, 1984.

Diccionario de computación
Alan Freedman
Mc Graw Hill, USA., 1994.

Redes de Computadoras. Protocolos Normas e Interfaces
Uyless Black
Macrobit Editores, Mexico, D.F., 1992.