UNIVERSIDAD LA SALLE

ESCUELA DE INGENIERIA
INCORPORADA A LA U.N.A.M.

CARACTERISTICAS Y ARQUITECTURA DEL MICROCONTROLADOR 8052AH-BASIC

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:
INGENIERO MECANICO ELECTRICISTA.
CON ESPECIALIDAD EN:
INGENIERIA ELECTRONICA.

PRESENTA:

GABRIELA VAZQUEZ AGUILERA

Asesor de Tesis: Ing. Germán Villalobos Alarcón

México, D.F.

1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

TABLA DE FIGURAS

TABLA DE FIGURAS

INTRODUCCION

INTRODUCCION

Dentro de la industria de hoy en día, la automatización es una herramienta indispensable para el buen funcionamiento y eficiencia de la misma, logrando con esto que el trabajo se realice de forma más rápida y eficiente compensando de esta forma los salarios de los trabajadores con los costos de producción.

Debido a esto es por lo que la tarea de los ingenieros es buscar la **optimización del control automático** por medio de la utilización de un controlador de fácil capacitación y manejo, ya que el obrero al recibir una instrucción adecuada debe de ser capaz de saber manejar y resolver cualquier problema que se le presente al estar trabajando con estos.

Por este motivo se han desarrollado en el mercado **MICROCONTROLADORES** de distintos tipos, algunos con **aplicaciones específicas**, otros con **aplicaciones generales**, pero la mayoría de fácil manejo. Dentro de esta variedad de microcontroladores, se ha fabricado uno de aplicación general y de fácil manejo, el circuito **8052AH-BASIC**, como su nombre lo indica, tiene un **interprete de BASIC** (el **MCS BASIC-52**) el cual es de fácil aprendizaje y por lo tanto su manejo no requiere mucha experiencia.

Algunas de las características de este circuito, que lo hacen ser una herramienta efectiva para el control automático son:

- **Una Unidad Central de Proceso (C.P.U.)** la cual está constituida por 8 "bits".
- **Circuitería de reloj y oscilador** integrado en el circuito.
- **32 líneas de entrada/salida (I/O).**
- **64K "bytes" de direcciones** para la memoria externa de datos.
- **64K "bytes" de direcciones** para la memoria de programa.
- **3 "TIMER/COUNTER" de 16 "bits"** cada uno.

- **6 interrupcciones** con dos niveles de prioridad cada una.
- **Un puerto serial "full duplex"**, lo cual significa que puede transmistir y recibir al mismo tiempo.
- **Procesador Booleano.**
- Como se comentó anteriormente tiene **residente un interprete de BASIC** con todos los comandos, operadores y declaraciones standard. También permitiendo al usuario realizar tareas que generalmente requieren la utilización de un lenguaje ensamblador.
- **MCS BASIC-52** permite el manejo de programas residentes en memoria RAM, PROM, EPROM y EEPROM.
- **Genera todos los tiempos necesarios para programar** cualquier PROM, EPROM ó EEPROM standard.
- Se pueden accesar desde el **programa codificado en BASIC** subrutinas realizadas en ensamblador por medio de una instrucción.

Como se puede observar todas estas características hacen de este circuito una excelente ayuda en la empresa ya que se pueden desarrollar proyectos bastante complejos con la ayuda de unos cuantos circuitos y el conocimiento del lenguaje BASIC.

El desarrollo de esta investigación va dividirse básicamente en **5 capítulos**:

-En el **PRIMER CAPITULO** se hará una pequeña remembranza de la aparición de los **microprocesadores** y su evolución hasta los **microcontroladores**.

-El **SEGUNDO CAPITULO** describe internamente al circuito **8052AH-BASIC** que es la base de esta investigación hablando de **toda su estructura interna (tiempos, estructuras de puertos, acceso a memorias externas, etc.)**.

-El **CAPITULO TERCERO** se enfocará en la **programación del circuito** mencionando las instrucciones en **BASIC** y en **ensamblador.**

-En este **CAPITULO (CUARTO)** se dedicará básicamente hablar de **algunas aplicaciones** que se le pueden dar al circuito **8052AH-BASIC.**

-En el **APENDICE A** se describirán en forma básica los circuitos que van a formar parte del controlador principal (**disposición de terminales y principales características eléctricas**), algunos de los cuales son: el **8052AH-BASIC, compuertas, memorias, etc..**Adicionalmente se anexará un **breve GLOSARIO DE TERMINOS .**

CAPITULO I

QUE ES UN MICROPROCESADOR Y QUE UN MICROCONTROLADOR

Este al ser el primer capítulo de este trabajo de investigación, va a estar dedicado a dar una breve explicación de las **diferencias básicas entre microprocesadores y microcontroladores**, y por decirlo así, la **justificación de la utilización de estos últimos.**

EL MICROPROCESADOR Y EL MICROCONTROLADOR

Como se puede observar en las tareas que realizamos a diario, la aparición del **Microprocesador** ha ayudado al **avance de la tecnología**, ya que se pudieron realizar máquinas más pequeñas y con una gran variedad de funciones que las primeras.

Con esa evolución, los microprocesadores parecía que podían hacerlo todo, pero los investigadores se dieron cuenta de que había que crear **dispositivos con propósito específico de control**, que ocuparan menos espacio que los sistemas basados en microprocesadores y que hicieran una utilización más eficiente de sus memorias. **Debido a la alta escala de integración** pudieron realizar esta tarea apareciendo así los **Microcontroladores.**

Los **Microcontroladores son circuitos electrónicos inteligentes** ya que pueden ejecutar un programa o una secuencia de instrucciones que se encuentran grabados internamente, ya que estan constituidos con los elementos necesarios como lo es la memoria (RAM y EPROM), unidad lógico-aritmética y puertos de Entrada/Salida, con los cuales se comunicará con el exterior para así poder controlar dispositivos externos.

Fueron creados como "**CIRCUITOS DE CONTROL**" para las calculadoras, hoy en día la evolución y la necesidad de reducir sistemas complejos en tarjetas más pequeñas, donde sea utilizado el menos número de circuitos integrados, realizando las

mismas funciones o un número mayor de las mismas, dando así a los diseños una gran ventaja en competitividad en el mercado.

Estos circuitos electrónicos inteligentes pueden **sustituir cierta lógica** o ser **parte integral de un sistema electrónico complejo** de alto funcionamiento.

Como se puede observar, estos circuitos integrados nos proporcionan una serie de ventajas, como son:

- **Reducir el espacio físico.**
- **Reducir tiempo de diseño del mismo.**
- **Reducir costos en el sistema.**

Y la más importante:

- **El de ser el más eficiente en funcionamiento que el de los sistemas que está sustituyendo asegurando así una mejor opción en utilización desplazando las técnicas tradicionales de diseño de circuitos lógicos.**

Hoy en día los microcontroladores son utilizados en diferentes áreas donde se requiera o se necesite que los sistemas esten sujetos a ser controlados, como por ejemplo:

- **Industria Automotriz.**
- **Sistemas de Aire Acondicionado.**
- **Sistemas Computacionales.**
- **Sistemas de Comunicación.**
- **Sistemas de Defensa de los Paises.**

- **Industria Aeronáutica.**
- **Biomedicina, etc..**

COMO SON INTERNAMENTE UN MICROPROCESADOR Y UN MICROCONTROLADOR

Una **Unidad Central de Proceso o Microprocesador** es aquel elemento de una computadora el cuál se encarga de llevar a cabo todas las **operaciones y el control de las mismas.** Se compone fundamentalmente de tres partes:

- **1) A. L. U.** (Unidad Aritmetica Lógica) .- Esta unidad es la que ejecuta realmente el trabajo de procesamiento. Puede recibir datos y efectuar con ellos operaciones aritméticas lógicas, de comparación y corrimiento, entre otras. También tiene algunos registros para almacenar los datos sobre los que va a realizar las operaciones; el número exacto de estos registros depende de cada computadora en particular. El registro principal de la **A.L.U.** se conoce como **Acumulador**. Generalmente, al inicio de una operación, el **Acumulador** contiene uno de los operandos, y al final de la operación, contiene el resultado.

- **2) Unidad de Control** .- Es la encargada de dirigir todas las operaciones dando tiempo y señales de control entendibles para todos los periféricos que conforman un sistema. Se compone de tres partes que son: **Un Registro de Instrucciones** encargado de decodificar la instrucción y determinar cuantos ciclos se lleva; **Un Decodificador de Instrucciones y Codificador de Ciclos Máquina** encargado de determinar los pasos para ejecutar una instrucción y por último **Un Bloque de Control** encargado de generar las señales apropiadas.

- 3) **Registros** .- Se dividen en dos: **De Propósito General**, los cuales están disponibles para el usuario como son generalmente el registro **A** o **Acumulador** y el **B**. Por otro lado los de **Propósito Específico**, que son los que contienen información valiosa para el microprocesador como son el **Contador de Programa "PROGRAM COUNTER"** o el **Apuntador de Pila "STACK POINTER"**.

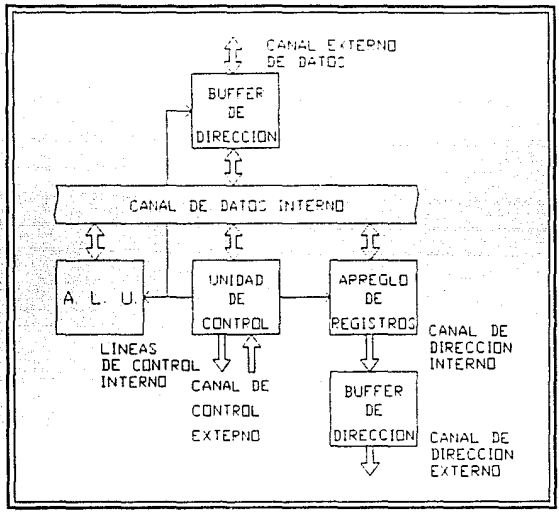

FIG. 1.1 Partes Fundamentales de un Microprocesador

Por otra parte un Microcontrolador está constituido por los siguientes elementos:

- 1) **A. L. U.** (Unidad Aritmética Lógica) .- Explicada con anterioridad.
- 2) **Unidad de Control** .- La cual realiza las mismas funciones que en un microprocesador.
- 3) **Registros** .- Contando también con los de **propósito específico** y de **propósito general**.

- **4) Memoria** .- Todos los microcontroladores cuentan con una memoria **RAM** interna y algunos hasta con memoria **ROM**. Los memorias son elementos en los cuales se almacena información. La primera es de acceso aleatorio, de lectura/escritura y volátil (esto es que necesita suministro contínuo de energía), y la segunda, es una memoria de solo lectura la cual ya viene pregrabada y es no volátil.

- **5) Puertos de Entrada/Salida** .- Son aquellos que sirven de **interfaz** con el **medio exterior.**

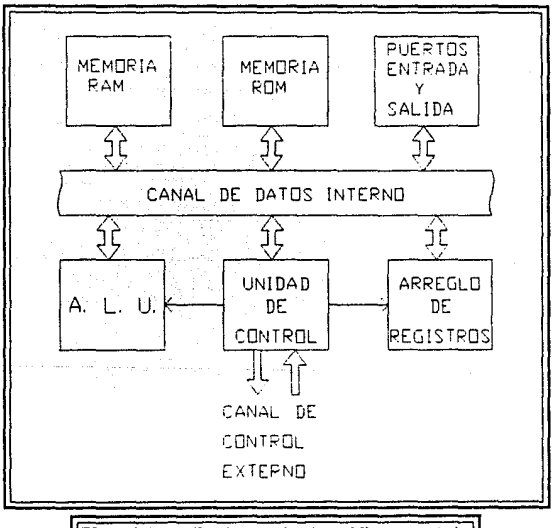

FIG. 1.2 Partes Fundamentales de un Microcontrolador

COMO SELECCIONAR UN MICROCONTROLADOR

Durante el proceso de una investigación no siempre concuerdan absolutamente los requerimientos del investigador con las características que nos ofrecen los circuitos disponibles en el mercado. Una pequeña alteración en los requerimientos o la adición de un circuito periférico puede crear la solución más efectiva y de bajo costo aunque no la ideal que se esperaba al principo de la investigación.

El proceso de evaluación deberá de iniciar con una lista de requerimientos en orden descendente de prioridad, como sea determinado por la aplicación. El planteamiento incial del proyecto no restringirá la selección del dispositivo o el fabricante, pero identificará suficientemente las características del **Microcontrolador**.

El procedimiento en este punto es comparar cada candidato con su asignación e periféricos y otros requerimientos, reteniendo para un estudio futuro cualquier dispositivo que concuerde con la mayoría de los requerimientos.

Si ninguno es el apropiado, la preguntas serían:

¿Cuál criterio causa la mayoría de los problemas para encontrar algo que concuerde?

¿Pueden ser alterados los requerimientos?

Y si no fuera así,

¿Se puede volver a plantear el problema de forma diferente?

Una serie de preguntas son realizadas hasta que se encuentra un circuito que satisfaga las necesidades.

En la siguiente figura (FIG. 1.3), se muestra un diagrama de flujo que sirve de gran ayuda para la elección de un microcontrolador.

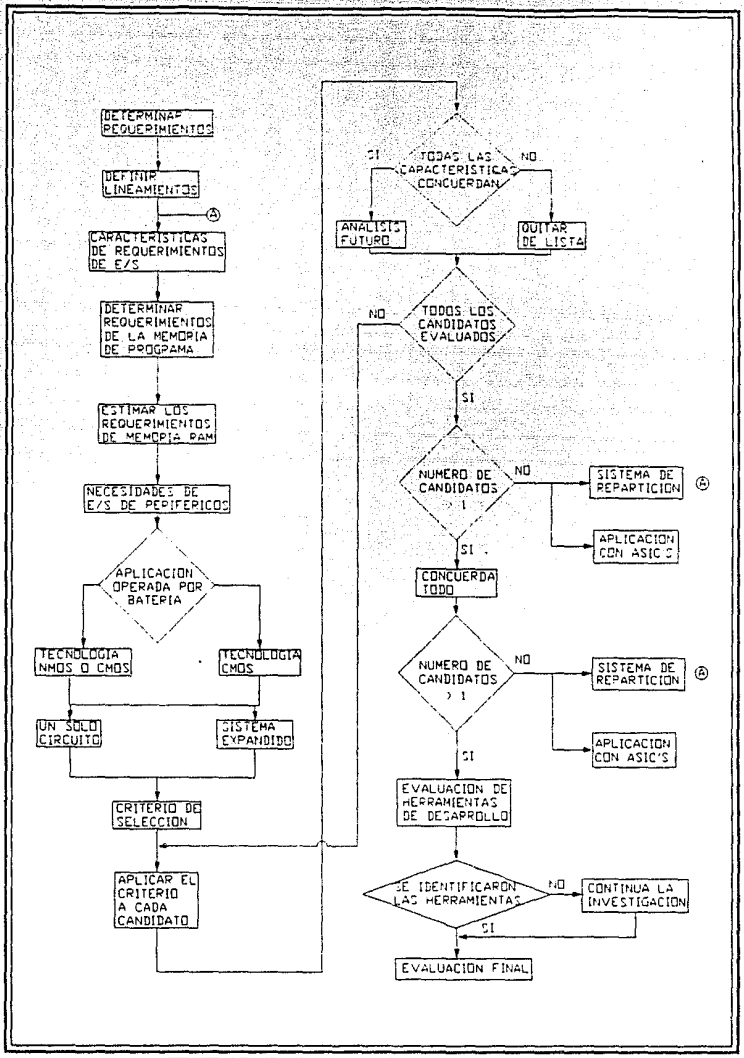

FIG. 1.3 Pasos para la Selección de un Microcontrolador

CAPITULO II

CONFIGURACION DEL "HARDWARE" DEL 8052AH

Dentro de este capítulo se hablará acerca de la **estructura interna** del **8052AH-BASIC.**

El **8052AH-BASIC** es un **Microcontrolador** el cual cuenta con un **interprete de BASIC** en su memoria **ROM**, constituido internamente como sigue:

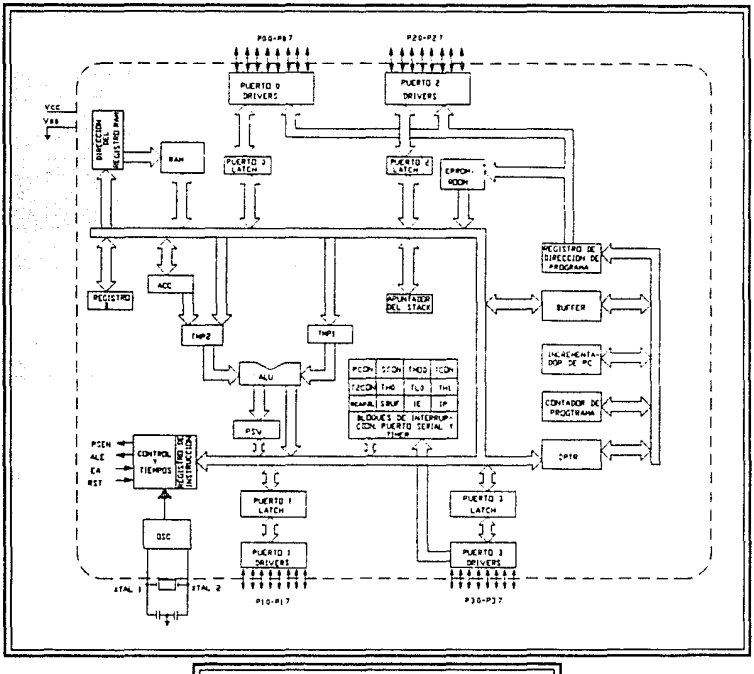

FIG. 2.1 Arquitectura Interna del 8052AH BASIC

El **MCS BASIC-52** requiere por lo menos **1K "byte"** de memoria externa. Después de dar el **"Reset"** éste dimensiona la memoria externa y **si no estuviera disponible al menos 1K "byte"**, el **MCS BASIC-52** no responderá, por lo tanto quedará en un ciclo interno por un tiempo indefinido (hasta que se vuelva a dar el **"Reset"**).

El **MCS BASIC-52** dimensiona consecutivamente las localidades externas de memoria desde la **0000H** (por si una falla dentro de la memoria es detectada). La operación de dimensionamiento es realizada simplemente escribiendo un **5AH** a una localidad de memoria externa, entonces es probada dicha localidad. Si la localidad de memoria en particular pasa dicha prueba, el **BASIC** entonces escribe un **00H** a esa localidad y la vuelve a checar.

El **MCS BASIC-52** solamente dimensiona las localidades externas de memoria desde la **0000H** hasta la **0DFFFH**. Las localidades de memoria desde la **0E000H** hasta la **0FFFFH** son reservadas para las entradas/salidas **(I/O)** del usuario y/o para programas en lenguaje ensamblador.

El programa **MCS BASIC-52** reside en los 8K "bytes" de memoria ROM disponible en el circuito **8052AH** y como resultado requiere que la memoria externa sea particionada en una manera específica.

La arquitectura del **8052AH** no es del tipo **Von Neumann**. Esto significa que la memoria para los programas y la de los datos no residen en el mismo espacio físico de direcciones del **8052AH**.

Específicamente el **/RD** (terminal 17) y el **/WR** (terminal 16) en el **8052AH** son utilizadas para habilitar la memoria de datos y **/PSEN** (terminal 29) es utilizado para la habilitación de la memoria de programas.

Dependiendo de la configuración del "Hardware", el **MCS BASIC-52** opera en dos modos distintos de **MEMORIA** los cuales a continuación se explicarán:

MODO DE OPERACION CON RAM SOLAMENTE

En este modo de operación, la memoria de lectura y escritura (RAM) es conectada al circuito iniciando desde la dirección de memoria **0000H**, la memoria puede llegar hasta la localidad **0FFFFH**.

En este modo de operación el decodificador de direcciones es utilizado para generar la señal de "**Chip Select**" (**CS**) para las memorias **RAM**; la terminal de **/RD** en el **8052AH** es utilizada para generar el pulso de habilitación de salida ("**Output Enable**" **/OE**) y la terminal **/WR** genera el pulso de habilitación de escritura ("**Write Enable**" **/WE** o **/WR**); la terminal del **/PSEN** ("**Program Store ENable**") no es utilizada en este modo de operación.

Este modo ofrece una configuración muy simple de "**Hardware**" para el circuito con **MCS BASIC-52** (desde este momento se hará referencia a este circuito con el nombre de **8052AH-BASIC**).

Como la terminal de **/PSEN** no es utilizada en este modo, el usuario no puede utilizar rutinas en lenguaje ensamblador, así mismo no puede programar un **EPROM**.

En general, el modo de operación con **RAM** solamente, es utilizado solo para checar el circuito durante el desarrollo incial del sistema.

MODO DE OPERACION CON RAM / EPROM

Este modo de operación permite la implantación del sistema completo para el **MCS BASIC-52**. Este modo de operación requiere que la memoria externa sea mapeada de cierta manera. La configuración de la memoria con **RAM** y **EPROM** es como sigue:

- 1) Las terminales de /RD y de /WR en el 8052AH-BASIC son utilizadas para habilitar la memoria RAM la cual es direccionada desde la localidad 0000H a la 7FFFH. Las direcciones son utilizadas para decodificar el /CS de las memorias RAM, y las terminales /RD y /WR son utilizadas para habilitar a las terminales /OE y a /WE (/WR) respectivamente.

- 2) La terminal de /PSEN en el 8052AH-BASIC es utilizada para habilitar la memoria EPROM la cual es direccionada desde la localidad 2000H a 7FFFH. Las direcciones son utilizadas para decodificar la terminal de /CS de las memorias EPROM y la teminal /PSEN es utilizada para la habilitación de la terminal /OE.

- 3) Para direcciones entre la 8000H y la 0FFFFH las terminales /RD y /PSEN en 8052AH-BASIC son utilizadas para habilitar la memoria. La memoria RAM o la EPROM pueden ser colocadas en este espacio de direcciones. Para permitir que las terminales de /RD y de /PSEN puedan habilitar las direcciones en este espacio, debe de haber una operación lógica AND entre la terminal de /RD y la de /PSEN. Esto puede ser realizado con un 74LS08 (compuerta de tecnología TTL "Transistor Transistor Logic"). La terminal de /WR en el 8052AH-BASIC es utilizada para escribir en la memoria RAM en este mismo espacio de direcciones. No se tiene que realizar la operación lógica de AND entre las señales de /RD y /PSEN más allá de la dirección 7FFFH para habilitar al MCS BASIC-52 para programar el EPROM. Esto es una sugerencia ya que de este modo el usuario podrá ejecutar rutinas en lenguaje ensamblador con la misma eficiencia que la ejecutadas, utilizando MCS BASIC-52.

Este esquema de direccionamiento de memoria actualmente permite al MCS BASIC-52 direccionar hasta 96K "bytes" de memoria, esto es:

- 32 "Kbytes" de circuitos de RAM.
- 32 "Kbytes" de circuitos ROM/EPROM.
- 32 "Kbytes" de circuitos combinados de RAM/ROM/EPROM.

Esta última se lleva acabo por medio de una operación lógica de AND con /RD y /PSEN para el direccionamiento desde la localidad 8000H hasta la 0FFFFH, dentro de ésta área de memoria el 8052AH se comporta como un circuito con arquitectura Von Neumann.

Cuando para la programación de un EPROM es utilizado el MCS BASIC-52, este asume que las direcciones deben de iniciar en la localidad 8000H. El MCS BASIC-52 puede solamente programar EPROM's direccionados entre la localidad 8000H y la 0FFFFH.

Cuando el comando PROG es utilizado por primera vez en un EPROM que ha sido borrado, el MCS BASIC-52 almacena este programa iniciando en la direción 8010H. Las localidades desde la 8000H hasta la 800FH son utilizadas para salvar la información del "Baud Rate" (velocidad de transmisión), junto con la información de la configuración.

TIEMPOS Y CONFIGURACION DE LA PROGRAMACION DEL EPROM

Con un "Hardware" adecuado el 8052AH-BASIC puede programar casi cualquier cibcuito EPROM o EEPROM. El único requerimiento para la programación de un EPROM es que el circuito que vaya a ser programado debe de iniciar su direccionamiento en la localidad 8000H. El MCS BASIC-52 requiere poco "Hardware" para programación de memorias EPROM.

Todos los tiempos críticos para la programación de este tipo de memorias son generados por tres terminales del puerto de I/O en el 8052AH-BASIC. Esta tres terminales generan las siguiente señales:

PUERTO 1.3. DESHABILITACION DEL HABILITADOR DEL "LATCH" DE DIRECCIONES ("ADRESS LATCH ENABLE")

El **"bit" 3 del puerto 1 (terminal 4** del **8052AH)** es utilizado para **DESHABILITAR** la señal del **ALE** para el **"latcheo"** requerido por el **8052AH** cuando la memoria externa es direccionada. Con esta terminal y con la de **ALE** debe de realizarse una operación lógica de **AND**, una compuerta **TTL** simple como un **74LS08** puede ser utilizada para realizar dicha operación.

Bajo operaciones normales **P1.3** se encuentra en un **estado lógico alto** (1), solamente durante la programación del **EPROM** se encuentra en un **estado lógico bajo** (0). La deshabilitación de la señal de **ALE** del **"latcheo"** externo es requerida para la programación el **EPROM** pues es la forma en que el **MCS BASIC-52** incia el proceso de programación.

Durante la programación, el **MCS BASIC-52** trata a los **Puertos 0** y **2** como **Puertos para entrada y salida, no** como **puertos de direcciones y de datos**. Primero escribe la parte baja de direcciones que van a ser programadas en el **Puerto 0**. Los datos en el **Puerto 0** son entonces **"latcheados"** en el "latch" de direcciones externas y el **MCS BASIC-52** deshabilita la señal de ALE del **"latch"** limpiando el **"bit" P1.3**, así, la parte baja de direcciones es "permanentemente" almacenada en el **"latch"** externo.

El **MCS BASIC-52** entonces escribe la parte alta de la dirección al **Puerto 2** y los datos que van a ser programados al **Puerto 0**. Por lo tanto, el **"latch"** de direcciones externo contiene la parte baja de las direcciones, el **Puerto 2** contiene la parte alta de las direcciones y el **Puerto 0** contiene los datos, al estar en proceso la programación el **EPROM**.

NOTA IMPORTANTE: Cuando el Puerto 0 en el 8052AH-BASIC es utilizado como puerto de Entrada/Salida, la estructura de salida es una configuración de "drenaje abierto", esto requiere que se coloquen unas resistencias de "pull-up" en el Puerto 0 para que permita que se programe el EPROM. En la versión 1.1 del circuito, la terminal INT0 debe de estar en estado lógico alto (1) cuando se este llevando a cabo la programación del circuito.

* PUERTO 1.4. ANCHO DE PULSO DE PROGRAMA

El **"bit" 4 del puerto 1 (terminal 5** del **8052AH-BASIC)** es utilizado para suministrar los pulsos de programación de 50 milisegundos ó el de 1 milisegundo. La longitud del pulso de programación es determinada dependiendo del modo de programación seleccionado (el **"normal"** o el **"INTELligent"**).

El **MCS BASIC-52** calcula la longitud del pulso de programación en base al valor de cristal (XTAL) asignado, la presición de este pulso es de 10 ciclos de reloj de la **CPU**. Esta terminal se encuentra normalmente en estado lógico alto (1), encontrándose en estado bajo (0) cuando se programa el **EPROM**. Dependiendo de cual **EPROM** va a ser programado esta señal va a ser utilizada de distintas formas.

* PUERTO 1.5. HABILITADOR DE VOLTAJE DEL PROGRAMA

El **"bit" 5 del puerto 1 (terminal 6** del **8052AH-BASIC)** es utilizado para habilitar el voltaje de programación del **EPROM**. Esta terminal se encuentra normalmente en estado lógico alto (1).

Es necesario para la operación de programación que sea llevado a un nivel lógico bajo (0). Esta terminal es utilizada para el **Encendido/Apagado** del alto voltaje (12.5 volts a 25 volts, dependiendo del circuito) requerido para programar los **EPROM**.

Los **tiempos de las terminales para la programación** de los **EPROM** son mostrados en la figura 3.2.

NOTA IMPORTANTE: El MCS BASIC-52 calcula el ancho del pulso de programación cuando el valor del XTAL es asignado y realiza el cálculo del ancho del pulso de programación durante 5 ciclos de reloj, así que la exactitud del pulso de programación se encuentra dentro de los límites de cualquier EPROM.

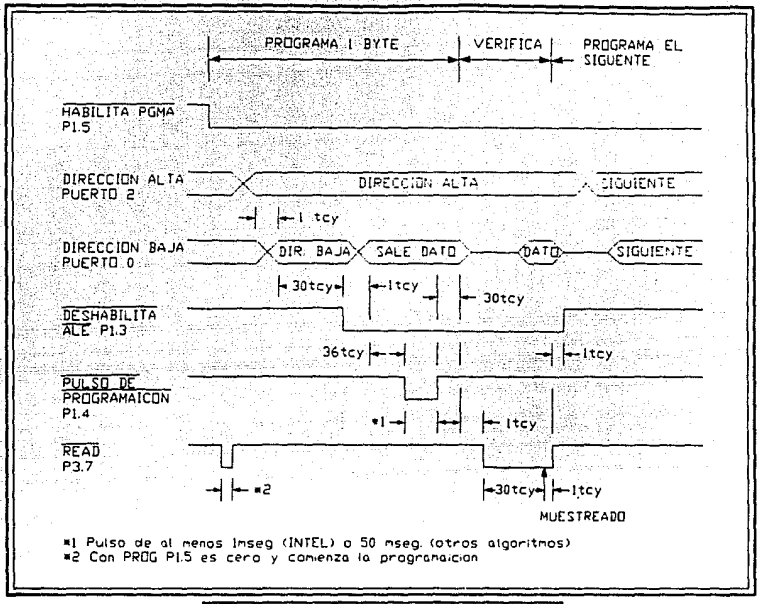

FIG. 2.2 Tiempos de Programación del EPROM

IMPLANTACION DEL PUERTO SERIAL

Las señales del puerto serial de entrada/salida en el 8052AH son compatibles con las señales **TTL**, pero son típicamente incompatibles con la mayoría de los sistemas de monitoreo. El puerto serial es inicializado por el **MCS BASIC-52** para el modo de **"UART" ("Universal Asincronous Receiver Transmiter")** de **8 "bits"**. En este modo **8 "bits" de datos, más un "bit" de inicio y otro de termino es transmitido**, pero el **"bit"** de paridad no es utilizado.

ORGANIZACION DE LA MEMORIA

El **8052AH-BASIC** está organizado en dos diferentes áreas de direcciones tanto para la **memoria que va a contener los programas** como para la **memoria que va a manejar los datos**. El hecho de contar solo con **16 direcciones** (de **A0** a **A15**) hace que se puedan direccionar hasta **65,536 "bytes"** o sea **64K "bytes"**. Es por esto que la memoria para los programas puede ser de dos formas:

- **64K "bytes"** completos de memoria externa.
- **8K "bytes"** que pueden residir en el circuito y los **56K "bytes"** restantes en memoria externa.

Visto esto de manera de bloques:

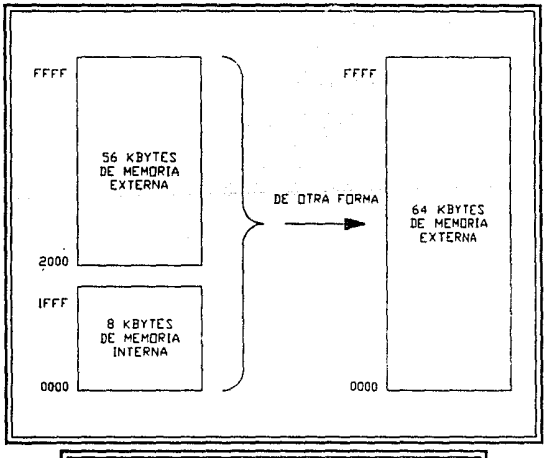

FIG. 2.3 Diagrama de Bloques de la Memoria de Programa

La memoria de datos también consiste en **64K "bytes"** como máximo de memoria **RAM externa, 256 "bytes"** de **RAM interna**, un área de **direccionamiento indirecto** solamente, un área de **direccionamiento directo e indirecto** y una serie de **Registros de Funciones Especiales.**

Visto en forma de bloques esta memoria se estructura como sigue:

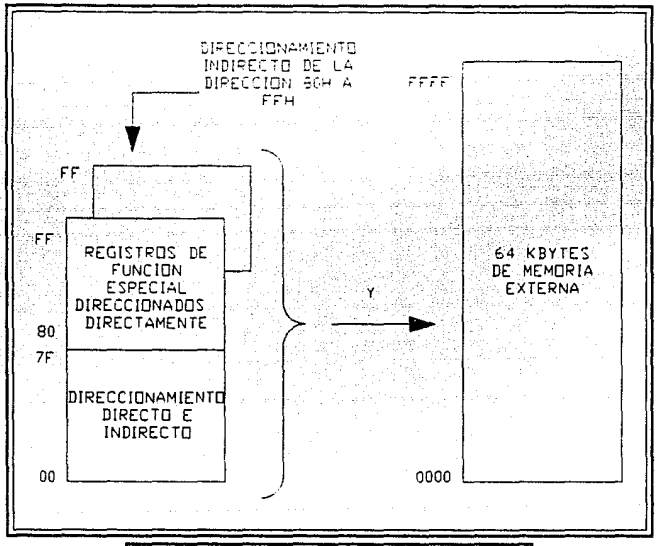

FIG. 2.4 Diagrama de Bloques de la Memoria de Datos

* AREA DE ACCESO INDIRECTO

Nótese que las direcciones de los registros de funciones especiales y las del área de **RAM** interna de acceso indirecto son las mismas (**080H-0FFH**). Esto no ocasiona ningún tipo de problema ya que como son dos áreas separadas, son accesadas en dos formas diferentes por medio de las instrucciones correspondientes (ver Capítulo III).

AREA DE DIRECCIONAMIENTO DIRECTO E INDIRECTO

Son **256** "**bytes**" de **RAM** los cuales pueden ser direccionados de manera directa o indirecta, y pueden ser divididos en tres segmentos:

- **1.-Bancos de Registros 0-3** : Estos bancos están formados por las localidades desde la **0** hasta la **1FH** (siendo un total de **32** "**bytes**") contando cada banco con **8 registros** de un solo "**byte**". El **Ensamblador** y el **Controlador** después del "**Reset**" toman por "default" el **banco de registros 0**. Para la utilización de los otros bancos de registros el usuario los debe de seleccionar por medio de "**Software**" (para poder accesar estos bancos, ver los **Registros de Funciones Especiales RS0** y **RS1**). Si se piensa utilizar más de un banco de **registros**, el "**Stack Pointer**" (**Apuntador de Pila**) debe de inicializarse en una localidad diferente de **RAM** la cual no sea utilizada para almacenamiento de datos (por ejemplo, la parte alta de la **RAM**), ya que al inicializarse lo hace en el **primer registro del segundo banco.**

- **2.-Area direccionable por un "bit"** : Para esta sección el diseñador del circuito asignó **16** "**bytes**" desde **20H** hasta **2FH**. Cada uno de los **128** "**bits**" de este segmento pueden ser directamente direccionados (**0-7FH**), estos "**bits**" pueden ser referidos de dos formas diferentes las cuales son aceptadas por el **Ensamblador**. Una de estas es referirse a ellos por medio de sus direcciones, por ejemplo, **0** a **7FH**, la otra es referirse por medio de los "**bytes**" **20H** a **2FH**. De esta manera los "**bits**" **0** a **7** pueden ser referidos como los "**bits**" **20.0** a **20.7**, y los "**bits**" **8** a **FH** son los mismos que **21.0** a **21.7**, y así sucesivamente. Cada uno de los **16** "**bytes**" en este segmento también pueden ser direccionados como un "**byte**".

- **3.-Area de trabajo auxiliar**: Los **"bytes" 30H** hasta **7FH** están disponibles para el usuario para ser utilizados como **RAM** para datos. De cualquier manera si el **"Stack Pointer"** ha sido inicializado en esta área, un número suficiente de **"bytes"** deben de ser apartados para preveer la destrucción de los datos.

Una vez vista esta división se pueden definir los **Registros de Funciones Especiales.**

REGISTROS DE FUNCIONES ESPECIALES

A continuación se verá una tabla que nos muestra los **Registros de Funciones Especiales** con sus direcciones, indicando cuales son direccionados solamente por medio de un **"byte"** (estos registros no tienen marca), y cuales son direccionados en ambos modos, o sea por medio un **"bit"** o un **"byte"** (marcadas con un asterisco). Posteriormente se explicará cada uno de los Registros, dando la nomenclatura de cada **"bit"** utilizado.

SIMBOLO	NOMBRE	DIRECCION
*ACC	Acumulador	0E0H
*B	Registro B	0F0H
*PSW	Palabra de Estado del Programa	0D0H
SP	Apuntador de Pila "STACK POINTER"	081H
DPTR	Apuntador de Datos de 2 "BYTES" "BYTE" Bajo (DPL) "BYTE" Alto (DPH)	 083H 082H
*P0	Puerto Cero	080H
*P1	Puerto Uno	090H

SIMBOLO	NOMBRE	DIRECCION
*P2	Puerto Dos	0A0H
*P3	Puerto Tres	0B0H
*IP	Control de Prioridad de Interrupciones	0B8H
*IE	Control de Habilitación de Interrupciones	0A8H
TMOD	Control Modo "TIMER/COUNTER"	089H
*TCON	Control del "TIMER/COUNTER"	088H
*T2CON	Control del "TIMER/COUNTER" 2	0C8H
TH0	"TIMER/COUNTER" 0 ("BYTE" Alto)	08CH
TL0	"TIMER/COUNTER" 0 ("BYTE" Bajo)	08AH
TH1	"TIMER/COUNTER" 1 ("BYTE" Alto)	08DH
TL1	"TIMER/COUNTER" 1 ("BYTE" Bajo)	08BH
TH2	"TIMER/COUNTER" 2 ("BYTE" Alto)	0CDH
TL2	"TIMER/COUNTER" 2 ("BYTE" Bajo)	0CCH

SIMBOLO	NOMBRE	DIRECCION
RCAP2H	Registro de Captura del "TIMER/COUNTER" 2 ("BYTE" Alto)	0CBH
RCAP2L	Registro de Captura del "TIMER/COUNTER" 2 ("BYTE" Bajo)	0CAH
*SCON	Control Serial	098H
SBUF	Almacenador Serial de Datos ("BUFFER")	099H
PCON	Control de Poder	087H

ACUMULADOR

ACC denominado también como el **registro acumulador**. Los mnemónicos para instrucciones específicas de este registro siempre se refieren a él como A.

REGISTRO B

El registro **B** es utilizado durante las operaciones de **multiplicación** y **división**. Para otras instrucciones puede ser utilizado como cualquier otro registro de trabajo **auxiliar.**

PALABRA DE ESTADO DEL PROGRAMA

Este **"byte"** contiene toda la información del **estado del programa** como se detalla a continuación:

CY	AC	F0	RS1	RS0	OV	X	P

PSW.7 PSW.0

SIMBOLO, NOMBRE Y DESCRIPCION

- **CY BANDERA DE ACARREO** .- Un estado alto en esta posición (1) indica que la operación que se realizó generó un **"bit"** de acarreo, esto es que el resultado de la operación de **8 "bits"** no puede ser expresado en la misma cantidad de **"bits"** sino que se tiene que ocupar una posición más.

Por ejemplo:

```
  1101 1111
+ 1100 0000
-----------
1 1001 1111
```

el **uno** que quedó a la izquierda de la operación es conocido como **Acarreo.**

- **AC BANDERA DE ACARREO AUXILIAR** .- (Para operación **BCD**) Un **1** en esta posición indica que se generó un **acarreo** en la operación **BCD** (**"Binary Coded Decimal"**) realizada.

- **F0 BANDERA DE 0** .- Esta bandera es de **propósito general** y es disponible para el usuario.

- **RS0, RS1** .- Estos dos **"bits"** seleccionanel **banco de registros,** siendo esta selección controlada por medio de **"Software"**. Considerando a **RS1** como el **"bit"** más significativo y a **RS0** como el menos significativo, se determinará el banco a utilizar por medio de la combinación siguiente:

(0,0)	BANCO 0	(00H-07H)
(0,1)	BANCO 1	(08H-0FH)
(1,0)	BANCO 2	(10H-17H)
(1,1)	BANCO 3	(18H-1FH)

- **OV** .- Un **1** en esta posición indica que ha habido un sobreflujo en la operación que ha sido realizada. Esto es que al realizar una operación el resultado de la misma sobrepasa el número máximo que puede ser representado por el microcontrolador (esto es distinto al **"bit"** de **acarreo**).

- **X RESERVADO** .- Esta posición está reservada para utilización futura del diseñador.

- **P BANDERA DE PARIDAD** .- Se actualiza por medio de **"Hardware"** en cada **ciclo de instrucción** para indicar si es **par** o **non** el número de **"bits"** en estado alto en el **acumulador.**

APUNTADOR DE PILA "STACK POINTER"

Se inicializa en la localidad **07H** después del **"Reset"**, esto causa que el **"Stack"** inicie en la localidad **08H.**

APUNTADOR DE DATOS

Este registro consiste en un **"byte"** en estado alto (**DPH**) y otro **"byte"** en estado bajo (**DPL**). Una de las formas como se puede utilizar este registro es almacenando una dirección de **16 "bits"**, así como también se puede utilizar como un registro de **16 "bits"** o dos registros independientes de **8 "bits"**.

PUERTOS 0 al 3

P0 al **P3** son los **"Latches"** de los **Registros de Funciones Especiales** de los Puertos **0, 1, 2,** y **3,** respectivamente. Los **"Latches"** y la estructura interna de los puertos así como su funcionamiento serán explicados mas adelante.

"BUFFER" SERIAL

El **"Buffer" Serial** (**SBUF**) se divide en dos registros distintos, el primero de ellos es un registro para el **"buffer" de transmisión** y el otro es para el **"buffer" de recepción.** Cuando un dato es almacenado en **SBUF**, este va a al **"byte"** que corresponde al **"buffer" de transmisión**, en donde es almacenado para la **transmisión serial.** Almacenando un **"byte"** en **SBUF** es lo que **inicia la transmisión**, de manera contraria, cuando un dato es obtenido de **SBUF** viene del **"byte"** del **"buffer" de recepción.**

REGISTROS "TIMERS"

Los registros pares (TH0, TL0), (TH1, TL1), y (TH2, TL2) son registros contadores de 16 "bits" para los "TIMER/COUNTER" 0, 1, y 2 respectivamente.

REGISTROS DE CAPTURA

El registro par (RCAP2H, RCAP2L) son los registros de captura para el "TIMER" 2. En este modo en respuesta a la transición en la terminal T2EX, TH2 y TL2 son copiados en los registros RCAP2H y RCAP2L respectivamente.

REGISTROS DE CONTROL

Los **Registros de Funciones Especiales** IP, IE, TMOD, TCON, T2CON, SCON y PCON contienen los "bits" de **control** y de **estado** para las **interrupciones del sistema**, los "TIMER/COUNTER" y el puerto **serial**.

A continuación se dará una breve explicación de cada uno de ellos, explicando en primer lugar que son las **Interrupciones** dentro de un circuito, posteriormente se darán los **registros** que las manejan, así como también los **registros** que tienen que ver directamente con los "TIMER/COUNTER" y con el **puerto serial** :

* INTERRUPCIONES

Las **INTERRUPCIONES** son un grupo de señales que tienen por objeto pedir la atención del circuito. Esto es, cuando uno de los circuitos exteriores requiere la atención del microcontrolador, manda una señal que "**interrumpe**" la **secuencia** que está siguiendo el programa y posteriormente **regresa** el control al programa principal. Cuando se carga una señal de interrupción, lo que hace es que carga en el "**Program Counter**" (Contador de Programa) una dirección específica y predefinida, dicha

dirección se va a utilizar como referencia para realizar un programa o programas conocidos como **RUTINAS DE SERVICIO** de o de los dispositivos.

Para la utilización de cualquiera de las **INTERRUPCIONES** del **8052AH-BASIC**, el usuario deberá seguir los tres pasos siguientes:

- **1.** Poner un **1** en el "**bit**" **EA** (para habilitar todo) del registro **IE**.
- **2.** Poner un **1** en el "**bit**" correspondiente a la **habilitación de la interrupción** requerida en el registro **IE**.
- **3.** Iniciar la **rutina de servicio de esa interrupción** en su correspondiente **Vector de Direcciones**.

FUENTE DE INTERRUPCION	VECTOR DE DIRECCION
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI&TI	0023H
TF2&EXF2	002BH

Adicionalmente, para **INTERRUPCIONES EXTERNAS**, debe de haber un **1** en las terminales **/INT0** e **/INT1** (**P3.2** y **P3.3**), y dependiendo de cual sea activada por nivel ó por transición, los "**bits**" **IT0** o **IT1** en el registro **TCON** podrían necesitar tener un **1**.

ITx = 0 ACTIVADA EN EL NIVEL
ITx = 1 ACTIVADA EN LA TRANSICION

* ASIGNACION DE MAYOR PRIORIDAD A UNA O MAS INTERRUPCIONES

La asignación de prioridades a las interrupciones, significa que se le va a dar más importancia a la ocurrencia de una que a la de otra, es decir, si se esta ejecutando una interrupción y de repente se presenta una de mayor prioridad, se detiene la interrupción en proceso y se ejecuta la que se presentó al final.

Para la asignación de la mayor prioridad a una interrupción el "bit" correspondiente en el registro **IP** debe de estar en 1. Para esto se debe de recordar que cuando un servicio de interrupción se encuentra en proceso, no puede ser detenido por otra interrupción del mismo o menor nivel de prioridad.

IE CONTROL DE HABILITACION DE INTERRUPCIONES

Si el **"bit"** es cero la interrupción correspondiente será deshabilitada, de manera contraria si el **"bit"** es uno la interrupción correspondiente será habilitada.

EA	X	ET2	ES	ET1	EX1	ET0	EX0

IE.7 IE.0

SIMBOLO, NOMBRE Y DESCRIPCION

- **EA** .- Deshabilita todas las interrupciones. Si **EA = 0** ninguna interrupción podrá ser pedida, pero si **EA = 1**, cada interrupción puede ser habilitada o deshabilitada individualmente colocando un **cero** o un **uno** en la posición correspondiente.
- **X** .- Esta posición no está implementada.
- **ET2** .- Habilita o deshabilita la interrupción de sobreflujo del **"TIMER" 2** o la de captura.

- **ES** .- Habilita o deshabilita la **interrupción del puerto serial.**
- **ET1** .- Habilita o deshabilita la interrupción de sobreflujo del **"TIMER" 1.**
- **EX1** .- H bilita o deshabilita la **Interrupción Externa 1.**
- **ET0** .- Habilita o deshabilita la interrupción de sobreflujo del **"TIMER" 0.**
- **EX0** .- Habilita o deshabilita la **Interrupción Externa 0.**

IP REGISTRO DE PRIORIDAD DE LAS INTERRUPCIONES

Si el **"bit"** es **0**, la interrupción correspondiente tiene la menor prioridad, y si el **"bit"** correspondiente es 1 la interrupción tiene una mayor prioridad.

X	X	PT2	PS	PT1	PX1	PT0	PX0

IP.7 IP.0

SIMBOLO, NOMBRE Y DESCRIPCION

- **X** .- Estas posiciones no están implementadas.
- **PT2** .- Define el nivel de prioridad de la interrupción del **"TIMER" 2.**
- **PS** .- Define el nivel de prioridad de la interrupción del **puerto serial.**
- **PT1** .- Define el nivel de prioridad de la interrupción del **"TIMER" 1.**
- **PX1** .- Define el nivel de prioridad de la **Interrrupción Externa 1.**

- **PT0** .- Define el nivel de prioridad de la interrupción del **"TIMER" 0.**
- **PX0** .- Define el nivel de prioridad de la **Interrupción Externa 0**

TCON REGISTRO DE CONTROL DEL "TIMER/COUNTER"

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON.7 TCON.0

SIMBOLO, NOMBRE Y DESCRIPCION

- **TF1** .- **BANDERA DE SOBREFLUJO DEL "TIMER" 1.** Se enciende por "**Hardware**" cuando hay un sobreflujo en el **"TIMER/COUNTER" 1.** Se apaga por "**Hardware**" como un vector de procesamiento para la rutina de servicio de la interrupción.
- **TR1** .- **"BIT" DE CONTROL DEL FUNCIONAMIENTO DEL "TIMER" 1.** Encendida y apagada por "**Software**" para que el **"TIMER/COUNTER" 1** se encuentre en **Encendido/Apagado** respectivamente.
- **TF0** .- **BANDERA DEL SOBREFLUJO DEL "TIMER" 0.** Se enciende por "**Hardware**" cuando el **"TIMER/COUNTER" 0** presenta un sobreflujo. Se apaga por "**Hardware**" como un **vector de procesamiento** para la rutina de servicio de la interrupción.
- **TR0** .- **"BIT" DE CONTROL DEL FUNCIONAMIENTO DEL "TIMER" 0.** Encendida y apagada por "**Software**" para que

"TIMER/COUNTER" 0 se encuentre en ON/OFF respectivamente.

- **IE1** .- BANDERA DE FLANCO DE LA INTERRUPCION EXTERNA 1. Encendida por "Hardware" cuando el flanco de la interrupción externa es detectado, apagada por "Hardware" cuando la interrupción es procesada.

- **IT1** .- "BIT" DE CONTROL DE LA INTERRUPCION TIPO 1.

- **IE0** .- BANDERA DE FLANCO DE LA INTERRUPCION EXTERNA 0. Encendida por "Hardware" cuando el flanco de la interrupción externa es detectado, apagada por "Hardware" cuando la interrupción es procesada.

- **IT0** .- "BIT" DE CONTROL DE LA INTERRUPCION TIPO 0.

TMOD REGISTRO DE CONTROL DEL MODO "TIMER/COUNTER"

GATE	C ó /T	M1	M0	GATE	C ó /T	M1	M0

TIMER 1 TIMER 0

SIMBOLO, NOMBRE Y DESCRIPCION

- **GATE** .- Cuando **TRx** (en el registro **TCON**) está encendida y el valor de **GATE** es igual a **1**, el **"TIMER/COUNTER" X** funcionará solamente cuando la terminal **INTx** está en estado alto (control por medio de "**Hardware**"). Cuando el valor de **GATE** es igual a **0**, el **"TIMER/COUNTER" X**

funcionará solamente cuando TRx sea igual a 1 (control por medio de "Software").

- **C ó /T .- SELECTOR DEL "TIMER" O "COUNTER".** Este selector se encuentra en estado bajo para operación como "TIMER" (entrada desde el sistema interno de reloj), y en estado alto para operación como "COUNTER" (entrada desde la terminal de entrada Tx).

- **M1 y M0 .- "BITS" DE MODO DE SELECCION.** La combinación de estos "bits" nos indican de que modo va a funcionar el "TIMER/COUNTER".

M1	M0		MODO DE OPERACION
0	0	0	"TIMER" DE 13 "BITS"
0	1	1	"TIMER/COUNTER" DE 16 "BITS"
1	0	2	"TIMER/COUNTER" DE 8 "BITS" DE AUTO-RECARGA
1	1	3	("TIMER" 0). TL0 ES UN "TIMER/COUNTER" DE 8 "BITS" CONTROLADO POR LOS "BITS" DE CONTROL STANDARD DEL "TIMER" 0
1	1	4	("TIMER" 1). PARADA DEL "TIMER/COUNTER" 1

ESTABLECIMIENTO DEL "TIMER"

Las siguientes tablas dan algunos valores para **TMOD** los cuales pueden ser utilizados para hacer que el "**TIMER**" **0** funcione de distintas maneras.

Se asume que solamente uno de los "**TIMER**" está siendo utilizado. Si se desea que ambos estén funcionando simultáneamente, en cualquier Modo, debe de realizarse una operación **OR** lógica entre el valor en **TMOD** para el "**TIMER**" **0** y el valor mostrado para el "**TIMER**" **1** en las **Tablas**.

Por ejemplo, si se desea que el "**TIMER**" **0** funcione en el **Modo 1** como **GATE** (con control externo), y el "**TIMER**" **1** en **Modo 1** como "**COUNTER**", entonces el valor que debe de ser cargado en **TMOD** es **69H** (realizando la operación **OR** lógica entre el **09H** de la **Tabla 1** y el **60H** de la **Tabla 4**).

- "TIMER/COUNTER" 0

COMO "TIMER"(TABLA1)

TMOD

MODO	FUNCION "TIMER" 0	CONTROL INTERNO	CONTROL EXTERNO
0	"TIMER" de 13 "BITS"	00H	08H
1	"TIMER" de 16 "BITS"	01H	09H
2	Autocarga de 8 "BITS"	02H	0AH
3	2 "TIMERS" de 8 "BITS"	03H	0BH

COMO "COUNTER" (TABLA 2)

TMOD

MODO	FUNCION "TIMER" 0	CONTROL INTERNO	CONTROL EXTERNO
0	"TIMER" de 13 "BITS"	04H	0CH
1	"TIMER" de 16 "BITS"	05H	0DH
2	Autocarga de 8 "BITS"	06H	0EH
3	1 "COUNTER" de 8 "BITS"	07H	0FH

En **Modo de Control Interno**, el **"TIMER"** es desactivado o activado, convirtiendo **0 ó 1** el **"bit"** en **TR0** por medio de **"Software"**. En **Modo de Control Externo** el **"TIMER"** es activado o desactivado por la transición de **1** a **0** en la terminal **/INT0 (P3.2)** cuando **TR0 = 1**, todo esto es controlado por medio de **"Hardware"**.

- **"TIMER/COUNTER" 1**

COMO "TIMER" (TABLA 3)

TMOD

MODO	FUNCION "TIMER" 0	CONTROL INTERNO	CONTROL EXTERNO
0	"TIMER" de 13 "BITS"	00H	80H

MODO	FUNCION "TIMER" 0	CONTROL INTERNO	CONTROL EXTERNO
1	"TIMER" de 16 "BITS"	10H	90H
2	Autocarga de 8 "BITS"	20H	A0H
3	No funciona	30H	B0H

COMO "COUNTER" (TABLA 4)

TMOD

MODO	FUNCION "TIMER" 0	CONTROL INTERNO	CONTROL EXTERNO
0	"TIMER" de 13 "BITS"	40H	C0H
1	"TIMER" de 16 "BITS"	50H	D0H
2	Autocarga de 8 "BITS"	60H	E0H
3	No disponible	-----	-----

En el **Modo de Control Interno** de ambas funciones y en el **Externo** de la función **"COUNTER"**, el **"TIMER"** es desactivado o activado poniendo en **1** ó **0** el **"bit" TR1** por medio de **"Software"**. En el **Modo de Control Externo** de la función **"TIMER"**, este es activado o desactivado por la transición de **1** a **0** en la terminal **/INT1 (P3.3)** cuando **TR1 = 1**, todo esto es controlado por medio de **"Hardware"**.

T2CON REGISTRO DE CONTROL DEL "TIMER/COUNTER" 2

TF2	EXF2	RCLK	TCNL	EXEN2	TR2	C ó /T2	CP ó /RL2

T2CON.7 TCLK T2CON.0

SIMBOLO, NOMBRE Y DESCRIPCION

- **TF2 .- BANDERA DE SOBREFLUJO DEL "TIMER" 2.** Se enciende por medio de "**Hardware**" y se apaga por medio de "**Software**". **TF2** no se encenderá cuando **RCLK = 1** ó **CLK = 1**.

- **EXF2 .- BANDERA EXTERNA DEL "TIMER" 2.** Se enciende cuando se causa una captura o una recarga por una transición negativa en **T2EX** y **EXEN2 = 1**. Cuando la interrupción del "**TIMER**" **2** es habilitada y **EXF2 = 1**, causará que la **C.P.U.** (Unidad Central de Proceso) carge el **vector de la rutina de interrupción** del "**TIMER**" **2**. **EXF2** debe de ser apagada por "**Software**".

- **RCLK .- BANDERA DE RECEPCION DE RELOJ (CLK).** Cuando está encendida, causa que el **Puerto Serial** utilice los pulsos de sobreflujo del "**TIMER**" **2** para la recepción del reloj en los **Modos 1** y **3**. **RCLK = 0** causa que el sobreflujo del "**TIMER**" **1** sea utilizado para la recepción del reloj.

- **TCLK .- BANDERA DE TRANSMISION DE RELOJ.** Cuando está encendida, causa que el **Puerto Serial** utilice los pulsos de sobreflujo del "**TIMER**" **2** para la transmisión del reloj en los **Modos 1** y **3**. **TCLK = 0** causa que el sobreflujo del "**TIMER**" **1** sea utilizado para la transmisión del reloj.

- **EXEN2 .- BANDERA DE HABILITACION EXTERNA DEL "TIMER" 2.** Cuando está encendida, permite que ocurra una captura o una recarga como resultado de una transición negativa en T2EX si el "TIMER" 2 no está siendo utilizado como reloj para el Puerto Serial. EXEN2 = 0 causa que el "TIMER" 2 ignore todo lo que pase en T2EX.

- **TR2 .- CONTROL DE "START/STOP" DEL "TIMER" 2.** Un 1 en esta posición hace que el "TIMER" inicie su funcionamiento.

- **C ó /T2 .- SELECTOR DEL "TIMER" O "COUNTER".** Se seleccionan por medio de los siguientes valores: un **0** selecciona el "TIMER" interno, un 1 selecciona el contador de eventos externos (disparado en el flanco de bajada).

- **CP ó /RL2 .- BANDERA DE CAPTURA/RECARGA.** Cuando está encendida, las capturas ocurrirán en las transiciones negativas del T2EX si EXEN2 = 1. Cuando se apaga, la **Auto-Recarga** ocurrirá con el sobreflujo del "TIMER" 2 o con la transición negativa del T2EX cuando EXEN2 = 1. Cuando RCLK = 1 ó TCLK = 1 este "bit" es ignorado y el "TIMER" es forzado a **Auto-Recargarse** en el sobreflujo del "TIMER" 2.

* ESTABLECIMIENTO DEL "TIMER/COUNTER" 2

Excepto para el modo de generador de **Velocidad de Transmisión**, los valores dados de T2CON no incluyen el establecimiento del "bit" TR2. Por esto, el "bit" TR2 debe de ser activado, separadamente, para hacer funcionar el "TIMER".

COMO "TIMER" (TABLA 1)

T2CON

MODO	CONTROL INTERNO	CONTROL EXTERNO
Auto-Recarga de 16 "BITS"	00H	08H
Captura de 16 "BITS"	01H	09H
Generador de velocidad de Transmision, Recepción y Transmision del mismo	34H	36H
Recepción solamente	24H	26H
Transmision solamente	14H	16H

COMO "COUNTER" (TABLA 2)

T2CON

MODO	CONTROL INTERNO	CONTROL EXTERNO
Auto-Recarga de 16 "BITS"	02H	0AH
Captura de 16 "BITS"	03H	0BH

En **Modo de Control Interno** la **Captura/Recarga** ocurre solamente cuando el **"TIMER/COUNTER"** tiene un sobreflujo. En **Modo de Control Externo** la **Captura/Recarga** ocurre cuando el **"TIMER/COUNTER"** tiene un sobreflujo y hay una transición de **1** a **0** en la terminal **T2EX (P1.1)** excepto cuando el **"TIMER" 2** es utilizado en el **Modo de Generador de Velocidad de Transmisión.**

PCON REGISTRO DE CONTROL DE PODER

SMOD	X	X	X	GF1	GF0	PD	IDL

PCON.7 PCON.0

SIMBOLO, NOMBRE Y DESCRIPCION

- **SMOD .- "BIT" DE DOBLE VELOCIDAD DE TRANSMISION.** Cuando el "TIMER" 1 es utilizado para generar la **Velocidad de Transmision** y **SMOD = 1**, la velocidad es duplicada cuando el **Puerto Serial** es utilizado en los **Modos 1, 2 o 3.**

- **X .-** Estas posiciones no están implantadas.

- **GF1 .- BANDERA DE PROPOSITO GENERAL.**

- **GF0 .- BANDERA DE PROPOSITO GENERAL.**

- **PD .- "BIT" DE DESACTIVADO.** Este " bit" no funciona para este tipo de circuito pues está disponible en circuitos con tecnología **CHMOS.**

- **IDL .- "BIT" DE MODO DESOCUPADO.** Al igual que el anterior solo está implantados en circuitos con tecnología **CHMOS.**

CCON REGISTRO DE CONTROL DEL PUERTO SERIAL

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SCON.7 SCON.0

SIMBOLO, NOMBRE Y DESCRIPCION

- **SM0 y SM1 .- ESPECIFICADOR DEL MODO DE PUERTO SERIAL.** Se especifica del modo siguiente. Las iniciales **UART ("Universal Asincronous Receiver Transmiter")** se refieren al **Modo de Transmisión del Puerto.**

SM0	SM1	MODO	DESCRIPCION	VELOCIDAD DE TRANSMISION
0	0	0	Registro de Corrimiento	Fosc/12
0	1	1	UART de 8 "BITS"	Variable
1	0	2	UART de 9 "BITS"	Fosc/64 o Fosc/32
1	1	3	UART de 9 "BITS"	Variable

NOTA: Fosc = Frecuencia de Oscilación

* ESTABLECIMIENTO DEL PUERTO SERIAL

MODO	SCON	VARIACION SM2
0	10H	Medio de Procesamiento Individual (SM2 = 0)
1	50H	
2	90H	
3	D0H	
0	NA	Medio de Multiprocesamiento (SM2 = 1)
1	70H	
2	B0H	
3	F0H	

* GENERACION DE VELOCIDAD DE TRANSMISION

El **Modo 0** tiene determinada una **Velocidad de Transmision** la cual es **1/12** de la frecuencia del oscilador. Para que funcione el **Puerto Serial** en este **Modo** ninguno de los **"TIMER/COUNTERS"** necesitan estar activados. Solamente el registro **SCON** necesita estar definido.

$$\text{Velocidad de Transmisión} = \frac{\text{Frecuencia de Oscilación}}{12}$$

* PUERTO SERIAL EN MODO 1

Este **Modo** tiene determinada una **Velocidad de Transmision** variable. Esta puede ser generada por el **"TIMER" 1** o el **"TIMER" 2.**

* UTILIZANDO EL "TIMER/COUNTER" 1 PARA GENERAR VELOCIDADES DE TRANSMISION

Para este propósito el "TIMER" 1 es utilizado en Modo 2 (Auto-Recarga).

$$\text{Velocidad de Transmisión} = \frac{(K)\,(\text{Frecuencia de Oscilación})}{(32)\,(12)\,[256 - (TH1)]}$$

Si:

SMOD = 0, entonces **K = 1**

Si:

SMOD = 1, entonces **K = 2**. (SMOD se encuentra en el registro PCON).

La mayoría del tiempo el usuario conoce la **Velocidad de Transmision** y necesita conocer el valor de recarga para **TH1**. Por esto la ecuación se puede expresar de la siguente forma:

$$TH1 = 256 - \frac{(K)\,(\text{Frecuencia de Oscilación})}{(384)\,(\text{Velocidad de Transmisión})}$$

Como el registro **PCON** no es direccionable por medio de un "**bit**", una manera de dar el valor a este "**bit**" es realizando una operación lógica con este registro.

* UTILIZANDO EL "TIMER/COUNTER" 2 PARA GENERAR VELOCIDADES DE TRANSMISION

Para este propósito el "**TIMER**" **2** debe ser utilizado en el **Modo de Generación de Velocidad de Transmision**. Si el "**TIMER**" **2** está recibiendo la señal de reloj a través de la terminal **T2 (P1.0)** la Velocidad de Transmisión es:

$$\text{Velocidad de Transmisión} = \frac{\text{Rango de Sobreflujo del "TIMER" 2}}{16}$$

pero si está recibiendo la señal de reloj internamente la **Velocidad** es:

$$\text{Velocidad de Transmisión} = \frac{\text{Frecuencia de Oscilación}}{(32)\,[\,65536 - (\,RCAP2H,RCAP2L\,)\,]}$$

Para obtener el valor de recarga para **RCAP2H, RCAP2L** la ecuación anterior puede ser reescrita como sigue:

$$RCAPH,RCAPL = 65536 - \frac{(\,\text{Frecuencia de Oscilación}\,)}{(32)\,(\,\text{Velocidad de Transmisión}\,)}$$

- PUERTO SERIAL EN MODO 2

Este modo tiene determinado un **Velocidad de Transmision** de **1/32** o **1/64** de la frecuencia del oscilador dependiendo del valor del "**bit**" **SMOD** del registro **PCON.**

En este modo ninguno de los "**TIMERS**" es utilizado y el reloj viene de la fase interna **2.** Y se determinan como sigue:

SMOD = 1, Velocidad de Transmision = 1/32 de la frecuencia de oscilación.

SMOD = 0, Velocidad de Transmision = 1/64 de la frecuencia de oscilación.

Como se mencionó anteriormente el registro **PCON** no es direccionable por medio de un "**bit**", una manera de dar el valor a este "**bit**" es realizando una operación lógica con este registro.

- **PUERTO SERIAL EN MODO 3**

La **Velocidad de Transmision** en **Modo 3** es variable y se establece exactamente igual al **Modo 1.**

OSCILADOR Y CIRCUITO DE RELOJ

Las terminales con la etiqueta **XTAL1** y **XTAL2** son la entrada y la salida (respectivamente) de un inversor lineal de un solo estado que se encuentra dentro del circuito, con el fin de ser utilizado como un oscilador de reactancia positiva controlado por medio del cristal, como se puede ver en la siguiente figura:

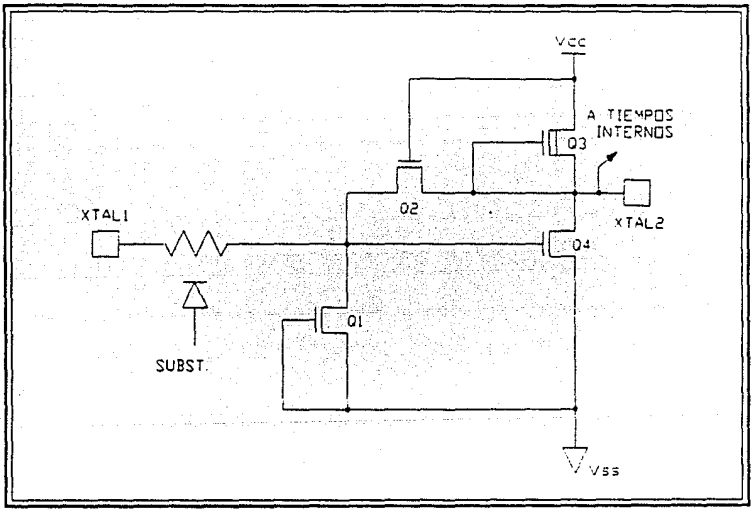

FIG. 2.5 Configuración del Oscilador Interno

Este oscilador puede ser también configurado con componentes externos como un **Oscilador Pierce**. El diagrama de este oscilador es como sigue:

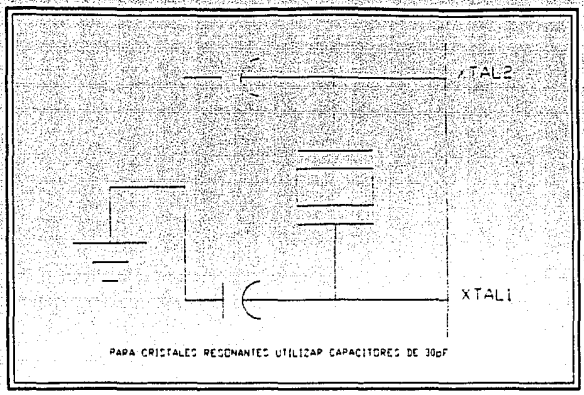

FIG. 2.6 Oscilador Cristal/Cerámico

Los componentes externos que se necesitan agregar (como se puede observar) son: una reactancia positiva (normalmente un cristal o un resonador cerámico) y dos capacitores **Cx1** y **Cx2**. Otra forma con un reloj externo, es aplicando la señal a la terminal **XTAL2**, y conectando a tierra la terminal **XTAL1**, como es mostrado en la figura siguiente:

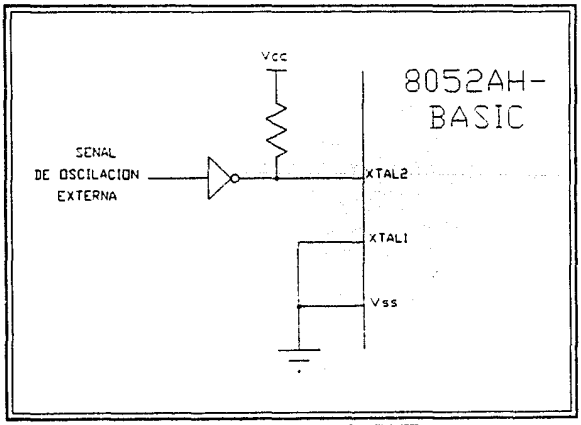

FIG. 2.7 Utilizando Reloj Externo

La decisión de utilizar el oscilador interno o uno externo está basada en los requerimientos tanto técnicos como económicos, en este caso se utilizará un oscilador interno ya que en la mayoría de los casos el amplificador interno con los apropiados componentes externos provee la solución más económica para el problema de reloj. Las excepciones radican en que en un medio ambiente severo las tolerancias de las frecuencias son casi del **0.01%**.

Una resistencia de "**pull-up**" puede ser utilizada (para incrementar el margen de ruido), pero es opcional si VOH de la compuerta manejadora excede la especificación de VIH MIN de XTAL2.

La frecuencia de oscilación está determinada en un **99.5%** por el cristal y hasta cerca del **0.5%** por la circuiteria externa a este. El amplificador interno tiene un efecto muy pequeño sobre la frecuencia.

El oscilador en cualquier caso maneja el generador de reloj interno. Este generador provee las señales de reloj necesarias para el circuito. Las señales internas son a la mitad de la frecuencia del oscilador y definen las fases internas, estados y ciclos de máquina, los cuales son descritos en "Diagrama de Tiempo para la C.P.U.".

DIAGRAMA DE TIEMPO PARA LA C.P.U.

Un ciclo de máquina consiste en **6 estados (12 períodos de oscilacion)**; cada estado se encuentra dividido en 2 fases, dentro de cada una de ellas hay un estado bajo y un estado alto de reloj denominado como **Período de Oscilación.** Por lo tanto un ciclo de máquina consta de **12 períodos de oscilación**, numerados desde **S1P1 (estado 1, fase 1)** hasta **S6P2 (estado 6, fase 2)**. Cada fase dura un período de oscilación, por lo tanto cada estado dura dos períodos de oscilación (como se puede ver en la siguiente figura).

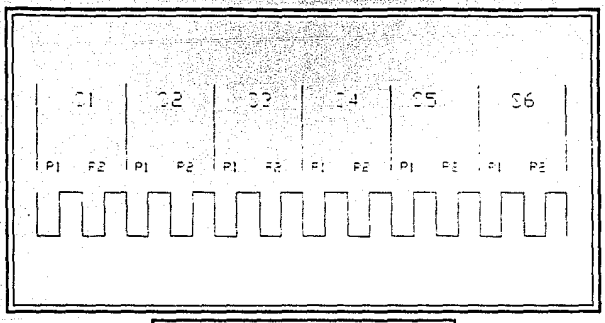

FIG. 2.8 Estados y Fases del Oscilador

Típicamente, operaciones aritméticas y lógicas se ejecutan durante l! **fase 1** mientras que las tranferencias internas de registro a registro son ejecutadas durante la **fase 2.**

En la figura 2.9 muestra los **Tiempos de Búsqueda y Ejecución** referidos a los **estados** y a las **fases.** A pesar de que estas señales internas del reloj no son accesibles al usuario, la señal de oscilación **XTAL2** y la señal de **ALE ("Address Latch Enable")** son mostradas para referencias externas. El **ALE** es normalmente activado dos veces durante **cada ciclo de máquina**, una vez durante **S1P2** y nuevamente en **S5P1.**

La ejecución de una instrucción de **un ciclo** inicia en **S1P2**, que es cuando el **código de operación** es transferido al **Registro de Instrucción.** Si es una instrucción de **dos "bytes", el segundo "byte"** es leído durante **S4** en el **mismo ciclo de máquina.** Si es una instrucción de **un "byte"**, de cualquier manera hay una búsqueda en **S4**, pero el **"byte"** leído (el cual podría ser el **próximo código de operación**) es ignorado, y el **Contador del Programa** no es incrementado.

En cualquier caso, al final de **S6P2**, la ejecución ha sido llevada a cabo en su totalidad. En los incisos **a** y **b** de la figura se muestran los diagramas de tiempo de una instrucción de **un "byte"** la cual dura **un ciclo de instrucción** y el diagrama para una instrucción de **dos "bytes"** la cual también dura **un ciclo de instrucción** respectivamente. La mayoría de las intrucciones del **8052AH-BASIC** son ejecutadas en **un solo ciclo.** La únicas instrucciones que duran más de un ciclo son **MUL (Multiplica)** y **DIV (Divide)**, a las cuales les toma **cuatro ciclos de máquina** completar su ejecución.

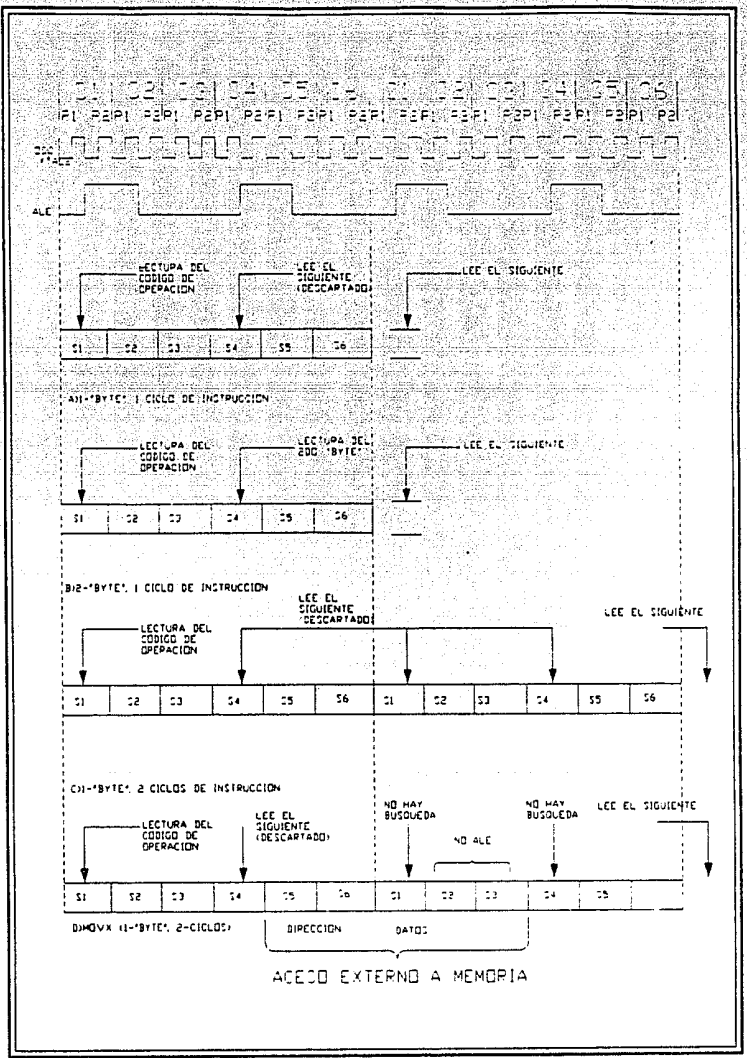

FIG. 2.9 Secuencia de Busqueda y Ejecución

Normalmente, **dos "bytes" de código** son buscados desde la memoria de los programas durante **cada ciclo de máquina**. La única excepción de esto es cuando se ejecuta la instrucción **MOVX**, la cual está compuesta por **un "byte" (dos ciclos)** y es la que **Accesa la Memoria Externa de Datos**. Durante la instrucción **MOVX dos búsquedas** son ignoradas mientras que la **Memoria de Datos Externa** está siendo direccionada. En los incisos **c** y **d** se muestran los diagramas de tiempo de una instrucción normal de **1 "byte" 2 ciclos**, y de una instrucción **MOVX** respectivamente.

ESTRUCTURA DE LOS PUERTOS Y OPERACION

El **8052AH-BASIC** tiene **cuatro Puertos bidireccionales de 8 "bits"** cada uno. Estos Puertos estan constituidos por un **"latch" (Registros de Funciones Especiales desde P0 hasta P3)**, un **"driver" de salida** y un **"buffer" de entrada.**

Los **"drivers" de salida** de los **Puertos 0** y **2**, y los **"buffers" de entrada** del **Puerto 0**, son utilizados en el **Acceso a Memoria Externa**. En esta aplicación, el **Puerto 0** coloca en la salida el **"byte" bajo** de la **dirección de la Memoria Externa, multiplexado** al mismo tiempo con el **"byte"** que está siendo leído o escrito. El **Puerto 2** coloca en la salida el **"byte" alto** de la **dirección de la Memoria Externa**, solamente cuando ésta dirección es de **16 "bits"** de longitud. De otra manera las terminales que constituyen el **Puerto 2** continuan mandando el contenido del **Registro de Función Especial P2.**

Todas las terminales del **Puerto 3** y dos del **Puerto 1** son multifuncionales. Estas no son solamente terminales para los **Puertos**, también tienen funciones específicas las cuales son listadas a continuación:

PURT.TERMINAL	FUNCION ALTERNATIVA
P1.0	T2 (Entrada Externa del "TIMER/COUNTER" 2)
P1.1	T2EX ("TRIGGER" de Captura y Recarga del "TIMER/COUNTER" 2)

PURT.TERMINAL	FUNCION ALTERNATIVA
P3.0	RXD (Puerto Serial de Entrada)
P3.1	TXD(Puerto Serial de Salida)
P3.2	/INT0(Interrupcion Externa)
P3.3	/INT1 (Interrupción Externa)
P3.4	T0 (Entrada Externa del "TIMER/COUNTER" 0)
P3.5	T1 (Entrada Externa del "TIMER/COUNTER" 1)
P3.6	/WR (Pulso de Escritura de la Memoria de Datos Externa)
P3.7	/RD (Pulso de Lectura de la Memoria de Datos)

Las **Funciones Alternativas** pueden ser activadas solamente si el **"bit"** del **"latch"** correspondiente al **Registro de Función Especial** de cada Puerto contiene un **1**, de otro modo la terminal del Puerto se mantiene en **0**.

* CONFIGURACIONES DE ENTRADA/SALIDA

La figura 2.10 muestra un diagrama funcional típico de un **"latch"** de un **"bit"**, y el **"buffer"** de **Entrada/Salida** de cada uno de los **cuatro Puertos**.

El "bit" del "latch" (un "bit" en los **Registros de Función Especial** de los **Puertos)** es representado por un **"Flip-Flop" tipo D**, el cual dará el reloj en un valor desde el canal interno en respuesta a una señal de **escribir en el "latch"** venida desde la C.P.U. La **salida Q del "Flip-Flop"** es colocada en el canal interno en respuesta a una señal de **leer del "latch"** venida desde la C.P.U. El nivel de la terminal del Puerto es colocado en el canal interno en respuesta a una señal de **leer la terminal** venida desde la C.P.U. Algunas instrucciones que **leen el Puerto** activan la señal de **leer el "latch"**, y algunas otras activan la de **leer la terminal.**

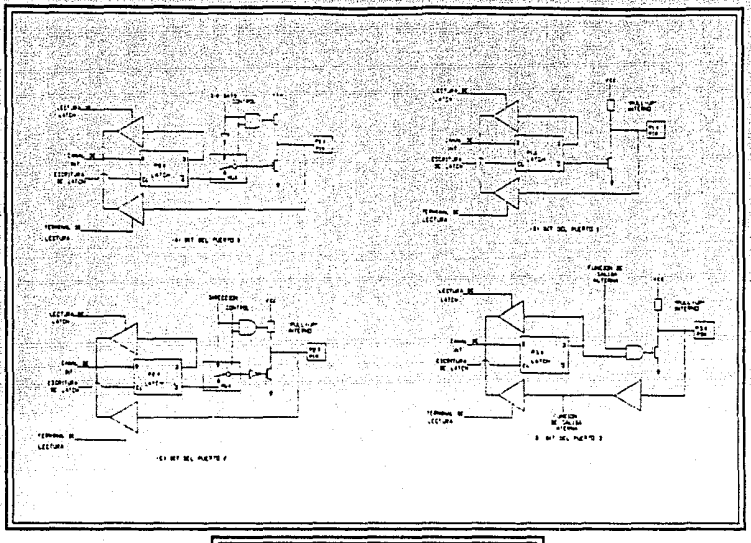

FIG. 2.10 Laches y Buffers de Entrada Salida

Como es mostrado en el diagrama anterior, los **"drivers"** de salida de los **Puertos 0 y 2** son **"switcheados"** a un **canal de Direcciones** y de **Direcciones/Datos** por una señal interna de **CONTROL** para utilizarse en el **acceso a la memoria externa**. Durante el **acceso a la memoria externa**, el **Registro de Función Especial del Puerto 2** se mantiene sin cambio, pero el del **Puerto 0** tienen un **1** escrito en él. También se muestra, que si el **"bit"** del **"latch"** del **Puerto 3** contiene un **1**, entonces el nivel de salida es controlado por la señal llamada **"Función Alternativa de Salida"**. El nivel actual de la terminal **P3.X** está siempre disponible para la **Función Alternativa de Entrada**.

Los **Puertos 1, 2** y **3** tienen un **"pull-up" interno** cada uno, mientras que el **Puerto 0 tiene salida de drenaje abierto**. Cada una de las líneas de **Entrada/Salida** pueden ser utilizadas independientemente como **una entrada ó una salida**. El **Puerto 0** y el **2** pueden no ser utilizados como líneas de **Entrada/Salida de propósito general**, esto ocurre cuando estan siendo utilizadas como el **canal de Direcciones/Datos**. Para poder ser utilizadas, el **"bit"** del **"latch"** del **Puerto** debe de contener un **1**, el cual desactiva el **FET ("Field Effect Transistor")** del **"driver"** de salida. Entonces, para los **Puertos 1, 2**

y 3, la terminal es llevada a un estado lógico alto por medio de los "pull-up" internos, pero pueden ser puestas en un estado bajo por medio de una fuente externa.

El Puerto 0 difiere en que no tiene "pull-up" interno. El FET que trabaja como "pull-up" en el "driver" de salida del P0 (ver inciso a de la figura anterior) es utilizado solamente cuando el Puerto está emitiendo 1's durante el acceso a la memoria externa, de otra manera el FET de "pull-up" se encuentra desactivado. Consecuentemente las líneas de P0 que están siendo utilizadas como líneas del Puerto de salida son de drenaje abierto. Escribiendo un 1 en el "bit" del "latch" deja a ambos FET's de salida desactivados, así que la terminal queda inactiva también. En esta condición puede ser utilizada como una entrada de alta impedancia.

El hecho de que los Puertos 1, 2 y 3 tengan un "pull-up" interno ya determinado hace que sean llamados algunas veces Puertos "CUASI-BIDIRECCIONALES". Cuando son configuradas como entradas se mantienen en estado alto y pueden suministrar corriente (IIL) cuando son llevadas a un estado bajo de manera externa. El Puerto 0 en otro modo, es considerado "verdaderamente" Bidireccional, porque cuando es configurado como una entrada queda desactivado.

Todos los "latches" de los Puertos tienen un 1 escrito después de la fución de "Reset". Si un 0 es subsecuentemente escrito a el "latch" del Puerto puede ser reconfigurado como una entrada escribiendo un 1 en el.

* ESCRIBIENDO A UN PUERTO

En la ejecución de una instrucción que cambia el valor en el "latch" de un Puerto, el nuevo valor llega al "latch" durante S6P2 (ver figura 2.9) del final del ciclo de instrucción. De cualquier modo, los "latches" del Puerto son realmente muestreados por sus "buffers" de salida solamente durante la fase 1 de cualquier período de reloj. Durante la fase 2 el "buffer" de salida contiene el valor que obtuvo en la fase 1 previa. Consecuentemente, el nuevo valor en el "latch" del Puerto no aparecerá en la terminal de salida hasta la próxima fase 1, la cual será en S1P1 en el siguiente ciclo de máquina.

Si el cambio requiere una transición de 0 a 1 en los Puertos 1, 2 o 3, un "pull-up" adicional es activado durante S1P1 y S1P2 en ciclo en el cual la transición ocurre. Esto es realizado para incrementar la velocidad de transición. El "pull-up" extra puede suministrar cerca de 100 veces la corriente que pueda suministrar un "pull-up" normal.

Debe de notarse que los **"pull-up" internos** son **Transistores de Efecto de Campo (FET)** y **no transistores lineales.** Los arreglos de **"pull-up"** son mostrados en la siguiente figura:

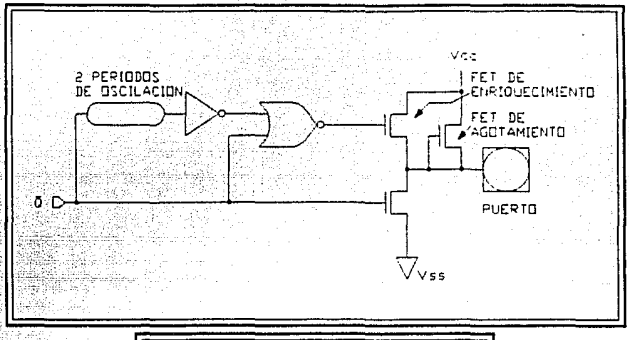

FIG. 2.11 Arreglo de Pull-Ups para la Transición

La parte fija del **"pull-up"** es un **Transistor en Modo de Agotamiento** con la **compuerta alambrada** a la **fuente.** Este **transistor** permitirá a la terminal suministrar poco más o menos **0.25 mA** cuando es conectada a **tierra.** En paralelo con el **"pull- up"** fijo hay un **Transistor en Modo de Enriquecimineto,** el cual es activado en **S1** siempre que el **"bit" del Puerto** realiza una transición de **0** a **1.** Durante este intervalo, si la **terminal del Puerto** es conectada a **tierra,** este **transistor** extra permitirá a la terminal suministrar **30 mA** adicionales.

• CARGA E INTERFAZ DE LOS PUERTOS

Los **"buffers" de salida** de los **Puertos 1, 2 y 3** pueden manejar cada uno **4 entradas** con tecnología **LS TTL.** Estos **Puertos** pueden manejar de una manera normal por cualquier circuito **TTL** o **NMOS,** así mismo las terminales pueden ser manejadas por **salidas a Colector Abierto** o a **Drenaje Abierto** pero hay que notar que una **transición de 0** a **1** no será rápida. Si la terminal es manejada por una **salida a Colector Abierto** la **transición de 0** a **1** deberá de ser manejada por el **FET** que se encuentra en **modo de Agotamiento** en el inciso a de la figura anterior.

Los "buffer" de salida del Puerto 0 pueden manejar cada uno 8 entradas de tecnología LS TTL. De cualquier manera se requieren "pull-up" externos para manejar entradas NMOS, excepto cuando está siendo utilizado como el canal de Datos/Direcciones.

* CARACTERISTICA DE LECTURA-MODIFICACION-ESCRITURA

Algunas instrucciones que leen el puerto leen el "latch" y otras leen la terminal. Las instrucciones que leen el "latch" en vez que de la terminal, son aquellas que leen un valor con la posibilidad de cambiarlo, y entonces reescribirlo en el "latch". Estas son llamadas Instrucciones de "LECTURA-MODIFICACIÓN- ESCRITURA". Las instrucciones listadas a continuación son de este tipo. Cuando el operando de destino es un Puerto, o el "bit" de un Puerto, estas instrucciones leen el "latch" en vez de la terminal:

ANL	Operación lógica de AND
ORL	Operación lógica de OR
XRL	Operación lógica de OR-EXCLUSIVA
JBC	Salta si el "BIT" = 1 y lo Limpia
CPL	Complemento del "BIT"
INC	Incremento
DEC	Decremento
DJNZ	Decrementa y salta si no es Cero
MOV PX,Y,C	Mueve el "BIT" del Acarreo al "BIT" Y del Puerto X

CLR PX,Y	Limpia el "BIT" Y del Puerto X
SET PX,Y	Enciende el "BIT" Y del Puerto X

No es obvio que las **últimas tres instrucciones** en la lista son instrucciones de **"LECTURA-MODIFICACIÓN-ESCRITURA". Leen el "byte" del Puerto, modifican el "bit" direccionado,** y entonces **escriben el nuevo "byte" en el "latch".**

La razón por la cual las instrucciones "LECTURA-MODIFICACIÓN-ESCRITURA" son dirigidas al "latch" en vez a la terminal es para evitar una posible mala interpretación del nivel de voltaje en la terminal. Por ejemplo, el "bit" del Puerto puede ser utilizado para manejar la base de un transistor. Cuando un 1 es escrito en el "bit", el transistor es activado. Si la C.P.U. entonces lee el mismo "bit" del Puerto en la terminal en vez de en el "latch", leerá el voltaje de base del transistor y los interpretará como un 0. Leyendo el "latch" en vez de en la terminal regresará el valor correcto de 1.

ACCESO A LA MEMORIA EXTERNA

El **Acceso a la Memoria Externa** es de dos tipos:

- **Acceso a la Memoria Externa de Programas**
- **Acceso a la Memoria Externa de Datos**

El **Acceso a la Memoria Externa de Programas** utiliza la señal **/PSEN ("Program Store ENable", Habilitación de Almacenamiento del Programa)** como el **pulso de lectura.** El **Acceso a la Memoria Externa de Datos** utiliza **/RD** o **/WR (Funciones Alternativas** de **P3.7** y **P3.6** respectivamente) para dar **un pulso y activar la memoria.**

Las **búsquedas** desde la **Memoria Externa de Programas** siempre utilizan una dirección de **16 "bits"**, mientras que el **Acceso a la Memoria Externa de Datos** puede utilizar una dirección del **16 "bits"** o **una de 8 "bits".**

Siempre que una dirección de 16 "bits" es utilizada, el "byte" alto de la dirección tiene salida por el Puerto 2, donde es tomado durante la duración del ciclo de lectura o escritura. Nótese que los "drivers" del Puerto 2 utilizan varios "pull-up" durante todo el tiempo en el cual estén emitiendo "bits" de dirección que sean 1, esto es durante la ejecución de la instrucción MOVX @DPTR (ver Capítulo 3). Durante este tiempo el "latch" del Puerto 2 (el Registro de Función Especial) no tiene que contener un 1, y el contenido del Registro del Puerto 2 no es modificado. Si el ciclo de memoria externa no es inmediatamente seguido por otro ciclo de Memoria Externa, el contenido del Registro de Función Especial del Puerto 2 reaparecerá íntegro en el próximo ciclo.

Si una dirección de 8 "bits" es utilizada (por ejemplo MOVX @Ri), el contenido del Registro de Función Especial del Puerto 2 se mantiene en las terminales del Puerto 2 durante todo el ciclo de Memoria Externa, facilitando así la paginación de la misma.

En cualquier caso, el "byte" bajo de la dirección es multiplexado en el tiempo con el "byte" de datos en el Puerto 0. La señal de Direcciones/Datos maneja ambos FET's en los "buffers" de salida del Puerto 0. Entonces, en esta aplicación las terminales del Puerto 0 no son salidas a drenaje abierto y no requieren un "pull-up" externo. La señal de ALE ("Adress Latch Enable", Habilitador del "Latch" de Direcciones) debe de ser utilizada para capturar el "byte" de direcciones en el "latch" externo. El "byte" de direcciones es válido en la transición negativa del ALE. Entonces, en un ciclo de escritura, el "byte" de datos que debe de ser escrito aparecerá en el Puerto 0 justo después de que /WR es activado, y se mantiene hasta después de que /WR es desactivado. En un ciclo de lectura, el "byte" que viene entrando, es aceptado en el Puerto 0 justo después que el pulso de lectura es desactivado.

Durante cualquier Acceso a la Memora Externa, la C.P.U. escribe un 0FFH en el "latch" del Puerto 0, de este modo borrando cualquier información del Registro de Función Especial del Puerto 0 que pueda estar siendo cargada.

La Memoria Externa de Programa es accesada bajo dos condiciones:

- 1) Siempre que la señal de /EA es activada
- 2) Siempre que el Contador de Programa (PC) contenga un número que sea más largo que 1FFFH.

Cuando la C.P.U. está trabajando con la **Memoria Externa de Programas**, los 8 "bits" del **Puerto 2** están dedicados a sus **funciones de salida** y no pueden ser utilizados como **Entrada/Salida de Propósito General.**

Durante las búsquedas de **Programas Externos** dan salida al "byte" **alto** de el **Contador del Programa.** Durante este tiempo los "drivers" **del Puerto 2** utilizan los "pull-up" para emitir los "bits" del **Contador del Programa** que son **1.**

* /PSEN ("PROGRAM STORE ENABLE" HABILITADOR DE ALMACENAMIENTO DEL PROGRAMA)

El **pulso de lectura** para **búsquedas externas** es llamado /PSEN, el cual **no es activado para búsquedas internas.** Cuando la C.P.U. se encuentra **Accesando la Memoria Externa de Programas**, la señal de /PSEN es **activada dos veces durante cada ciclo (excepto durante una instrucción de MOVX)** sea o no, la **búsqueda del** "byte" es actualmente necesitada para la instrucción que está siendo llevada a cabo. Cuando /PSEN es activada su diagrama de tiempo no es el mismo que el de /RD.

Un **ciclo completo** de /RD, incluyendo la **activación y desactivación** de ALE y /RD, toma **doce períodos de oscilación. Un ciclo completo** de /PSEN, incluyendo la **activación y desactivación** de ALE y /PSEN, toma seis períodos de oscilación.

La **secuencia de oscilación** de estos **dos tipos de ciclos de lectura** son mostrados en la siguiente figura para comparación.

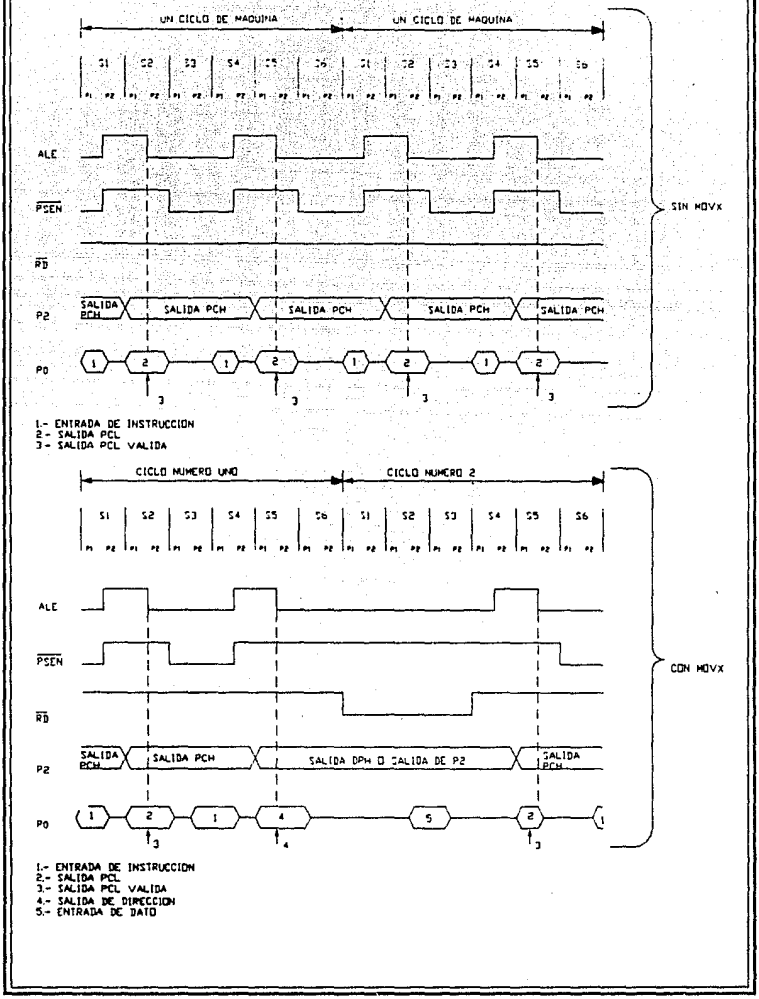

FIG. 2.12 Ciclos de Lectura de Memoria Externa

• ALE ("ADRESS LATCH ENABLE" HABILITADOR DEL "LATCH" DE DIRECCIONES)

La función principal del ALE es proveer la señal de tiempo apropiada para **"latchear" el "byte" bajo de una dirección** desde P0 a un **"latch" externo** durante búsquedas desde la **Memoria Externa de Programas**. Para ese propósito la señal de **ALE** es activada **dos veces cada ciclo de máquina**. Esta activación toma lugar siempre que no involucre una **búsqueda externa**, la única vez que el pulso de ALE no es emitido es durante un **Acceso a la Memoria Externa de Datos**. La primera señal de **ALE** del **segundo ciclo** durante una instrucción de MOVX es perdida (ver figura anterior). Por lo tanto, en cualquier sistema que no utiliza la **Memoria Externa de Datos**, la señal de **ALE** es activada en un rango constante de **1/6 de la frecuencia del oscilador**, y puede ser utilizada para dar un **reloj externo** o **propósitos de tiempo**.

• TRASLAPAMIENTO DE LOS ESPACIOS DE LAS MEMORIAS EXTERNAS DE PROGRAMAS Y DE DATOS

En algunas aplicaciones es deseable ejecutar un programa desde la misma **Memoria Física** que está siendo utilizada para **almacenar los datos**. En el **8052AH-BASIC**, los espacios de **Memoria Externa de Programas** y de **datos** pueden ser combinados realizando una operación lógica de **AND** entre **/PSEN** y **/RD**. Una operación de **AND con lógica positiva** de estas dos señales produce un pulso de **lectura** que se activa en estado bajo el cual puede ser utilizado para la **Memoria Física combinada**. Como el ciclo de /PSEN es más rápido que el ciclo de /RD, la **Memoria Externa** necesita ser lo suficientemente rápida para acomodar el ciclo de /PSEN.

"TIMER/COUNTERS"

El **8052AH-BASIC** tiene **tres registros "TIMER/COUNTER" de 16 "bits"** cada uno, los cuales pueden ser configurados como "TIMER" o como "COUNTER".

En su función como "TIMER" el registro es incrementado **cada ciclo de máquina**. De este modo, el usuario podría pensar que es un **contador de ciclos de máquina**.

Como **un ciclo de máquina** consiste en **12 períodos de oscilación**, el rango de conteo es de **1/12 de la frecuencia de oscilación.**

En su función como "**COUNTER**", el registro es incrementado en respuesta a una transición de **1** a **0** en su correspondiente terminal externa de entrada, **T0**, **T1** ó **T2**. En esta función, la entrada externa es muestreada durante **S5P2** de cada ciclo de máquina. Cuando el muestreo señala un estado alto en un ciclo de máquina y un estado bajo en el próximo ciclo, la cuenta es incrementada. El nuevo valor de la cuenta aparece en el registro durante **S3P1** del ciclo de máquina siguiente al ciclo en el cual transición fue detectada. Como toma dos ciclos de máquina (**24 períodos de oscilación**) para reconocer una transición de **1** a **0**, el máximo rango de conteo es **1/24 de la frecuencia de oscilación**. No hay restricciones para el rendimiento del ciclo de la señal de entrada externa, pero para asegurar que el nivel dado es muestreado al menos una vez antes de que cambie, debe de ser retenido por lo menos durante un ciclo de máquina completo.

Además de la selección del "**TIMER**" o "**COUNTER**", los "**TIMER**" **0** y **1** tienen **cuatro modos de operación** mientras que el "**TIMER**" **2** tiene solo **tres modos**. Todos estos modos serán explicados a continuación.

* "TIMER" 0 Y "TIMER" 1

La función como "**TIMER**" o como "**COUNTER**" es seleccionada por los "**bits**" de control **C o /T** en el **Registro de Función Especial TMOD**. Estos dos "**TIMER/COUNTER**" tienen **cuatro modos de operación**, los cuales son seleccionados por un par de "**bits**" en el mismo **Registro de Función Especial**. A continuación se explicarán cada uno de los cuatro modos, partiendo de la base que los tres primeros son iguales para ambos y el unico diferente es el cuarto.

° **MODO 0** -. Si trabaja cualquiera de los dos en este modo, su funcionemiento es el de un contador de **8** "**bits**" con un prescalador que divide en **32**. En este modo el "**TIMER**" es configurado como un **registro de 13** "**bits**". Si la cuenta vuelve a **cambiar** todos los **unos** por **ceros**, se establece la **bandera de interrupción** correspondiente al "**TIMER**" **TF1**. La entrada del contador es habilitada para el "**TIMER**" cuando

TR1 = 1 y cualquiera de los "**bits**" de **GATE = 0** o /**INT1 = 1** (el establecer **GATE = 1** permite a l "**TIMER**" ser controlado por la entrada externa /INT1, para facilitar la medición del ancho del pulso). **TR1** es un "**bit**" **de control** en el **Registro de Función Especial** TCON, mientras que **GATE** se encuentra en el **registro** MOD. El registro de 13 "**bits**" está formado por los **8** "**bits**" **de TH1** y los **5** "**bits**" **menos significantes de TL1**. Los **3** "**bits**" **más significativos de TL1** son indeterminados y deberán ser ignorados. La siguiente figura muestra su operación.

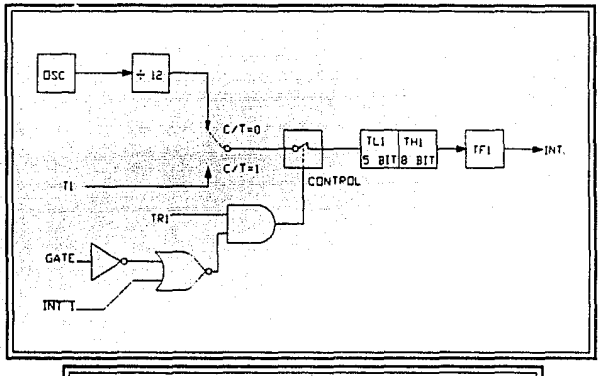

FIG. 2.13 "TIMER/COUNTER" l Modo 0 (Contador de 13 "bits")

- **MODO 1** -. Este modo es el mismo que el **Modo 0** con la diferencia de que el registro del "**TIMER**" correrá con los **16** "**bits**".

- **MODO 2** -. En este modo es configurado el **registro del** "**TIMER**" como un **contador de 8** "**bits**" (**TL1**) con recarga automática. Un sobreflujo generado en **TL1** no solo establece **TF1**, también recarga **TL1** con el contenido de **TH1**, el cual es establecida con anterioridad por medio de "**Software**", esta recarga deja a **TH1** sin cambio, como es mostrado en la siguiente figura.

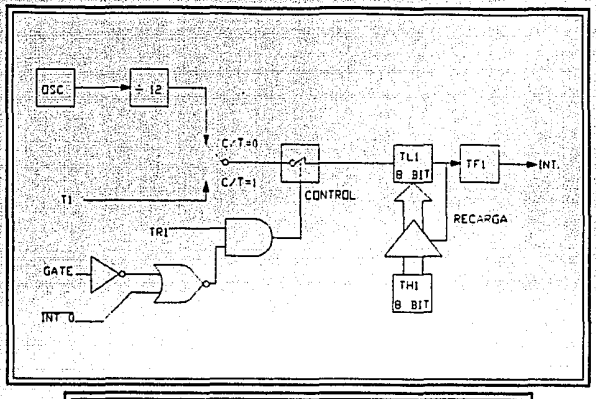

FIG. 2.14 "TIMER/COUNTER"1 Modo 2 (Contador de 8 "bits")

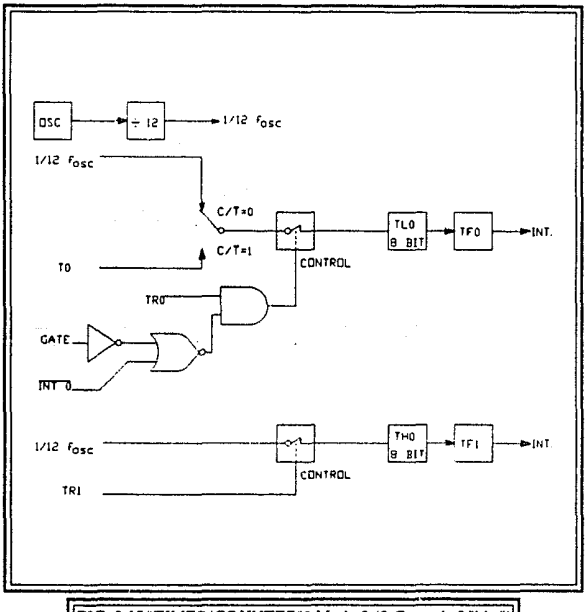

FIG. 2.15 "TIMER/COUNTER"0 Modo 3 (2 Cont. de 8 "bits")

- **MODO 3 -.** El "TIMER" 1 en este modo simplemente retiene su cuenta. El efecto es el mismo como el establecimiento de **TR1 = 0.** El "TIMER" 0 en este modo establece **TL0** y **TH0** como dos contadores separados. **TL0** utiliza los "bits" de **control C** y /T, **GATE, TR0, /INT0** y **TF0** de "TIMER" 0. **TH0** se queda encadenado en una función del "TIMER" (contando ciclos de máquina) y deja de utilizar **TR1** y **TF1** del "TIMER" 1, entonces **TH0** controla ahora la **interrupción de** "TIMER" 1. La lógica para este modo es mostrada en la figura 2.15 anteriormente expuesta.

El **modo 3** está hecho para aplicaciones que requieran un "TIMER/COUNTER" de **8 "bits" extra.** Con el "TIMER" 0 en **modo 3** el **8052AH-BASIC** parecerá **tener 4** "TIMER/COUNTER". Cuando el "TIMER" 0 se encuentra en **modo 3**, el "TIMER" 1 puede ser encendido y apagado activandolo y desactivandolo en su propio **modo 3**, o pueder seguir siendo utilizado por el puerto serial como un **generador de velocidad de transmisión**, o en su defecto en cualquier aplicación en la cual no se requiera una interrupción.

- "TIMER" 2

Este es un "TIMER/COUNTER" de **16 "bits"**. Como los "TIMER" 0 y 1, puede ser operado como un "TIMER" o como un **Contador de Eventos**. Esto es seleccionado por el "bit" C o /T2 en el **Registro de Función Especial T2CON**. Tiene solo **3 modos de operación**, las cuales son seleccionadas por los "bits" en el registro **T2CON**, como es mostrado a continuación:

RCLK + TCLK	CP o /RL2	TR2	MODO
0	0	1	Autorecarga de 16 "bits"
0	1	1	Captura de 16 "bits"
1	X	1	Generador de Velocidad de Transmisión
X	X	0	Apagado

- **Modo de Captura** .- Donde existen **dos opciones** las cuales pueden ser seleccionadas por el "**bit**" **EXEN2** en el **registro T2CON**. Si **EXEN2 = 0**, entonces el "**TIMER**" **2** es un "**TIMER**" o **Contador de 16 "bits"**, el cual cuando tiene un sobreflujo establece el "**bit**" **TF2** que es **un "bit" de sobreflujo del "TIMER" 2** el cual puede ser utilizado para generar una interrupción. Si **EXEN2 = 1**, entonces el "**TIMER**" **2** trabajará como anteriormente se dijo, con la adición de que si es detectada una transición de **1** a **0** en la entrada externa **T2EX** el valor actual de los registros **TL2** y **TH2** será capturado en los registros **RCAP2L** y **RCAP2H** respectivamente. Adicionalmente la transición en **T2EX** causa que el "**bit**" **EXF2** en **T2CON** sea establecido, y **EXF2**, como **TF2**, puede generar una **interrupción**.

- **Modo de Autorecarga** .- Nuevamente existe **dos opciones**, los cuales son seleccionados por el "**bit**" **EXEN2** en **T2CON**. Si **EXEN2 = 0**, entonces cuando el "**TIMER**" **2** presenta un sobreflujo no solamente establece **TF2** también causa que los **registros sean recargados** con el valor de **16 "bits" de los registros RCAP2L** y **RCAP2H**, los cuales son establecidos con anterioridad por medio de "**Software**". Si **EXEN2 = 1**, entonces el "**TIMER**" trabajará como se dijo anteriormente, pero adicionalmente cuando es detectada una transición de **1** a **0** en la entrada externa **T2EX** se disparará la recarga de los **16 "bits"** y se establecerá **EXF2**.

- **Modo Generador de Velocidad de Transmisión** .- Es seleccionado por **RCLK = 1** y/o por **TCLK = 1**. Este **Modo** será descrito junto con el **Puerto Serial**.

INTERFAZ SERIAL

El Puerto Serial el del tipo "**full duplex**", lo que quiere decir, que puede **transmitir y recibir de manera simultánea**. También reciben de manera de "**almacenamiento**",

es decir que pueden iniciar la recepción de un segundo "byte" antes de que el que ha sido previamente recibido sea leido en su totalidad. Los registros de recepción y transmisión del Puerto Serial son accesado por el Registro de Función Especial SBUF. Escribiendo en SBUF carga el registro que se va a transmitir, de manera contraria leyendo de SBUF accesa un registro receptor físicamente separado. El Puerto Serial puede operar de cuatro modos distintos:

- MODO 0 .- En este modo los datos seriales entran y salen a través de RXD, siendo TXD el que da la salida al Reloj de Corrimiento. Ocho "bits" de datos son transmitidos y recibidos (transmitiendo o recibiendo los "bits" menos significativos primero). La velocidad de transmisión es establecida a 1/12 de la frecuencia del oscilador.

- MODO 1 .- En este modo son transmitidos 10 "bits" a través de TXD y recibidos a través de RXD. Estos de dividen de la siguiente manera: Un "bit" de inicio (0), ocho "bits" de datos (primero los menos significativos), y un "bit" de parada (1). Cuando está recibiendo, el "bit" de parada se encuentra en RB8 en el Registro de Función Especial SCON. En este modo la velocidad de transmisión es variable.

- MODO 2 .- En este modo son transmitidos 11 "bits" a través de TXD y recibidos a través de RXD. Estos de dividen de la siguiente manera: Un "bit" de inicio (0), ocho "bits" de datos (primero los menos significativos), un noveno "bit" de dato programable y un "bit" de parada (1). Cuando está transmitiendo el noveno "bit" de dato (TB8 en el registro SCON) puede ser asignado con el valor de 1 o 0. Por ejemplo el "bit" de paridad (P en el registro PSW) puede ser copiado a TB8. En la recepción el noveno "bit" de dato está en RB8 en el registro SCON cuando el "bit" de parada es ignorado. La velocidad de transmisión es programada a 1/32 o a 1/64 de la frecuencia de oscilación.

- **MODO 3**.- En este modo son transmitidos **11 "bits"** a través de **TXD** y recibidos a través de **RXD**. Estos de dividen de la siguiente manera: **Un "bit" de inicio (0), ocho "bits" de datos (primero los menos significativos), un noveno "bit" de dato programable y un "bit" de parada (1)**. De hecho, como se puede ver, el **Modo 3 es igual al 2** en todos los aspectos a excepción de la **velocidad de transmisión** la cual en este caso **es variable.**

En los **Cuatro Modos**, la **transmisión es iniciada** por **cualquier instrucción** que utiliza a **SBUF** como un **registro destino**. La **recepción es iniciada en Modo 0** por la condición **RI = 0** y **REN = 1**, mientras que para los **demás modos es iniciada por la llegada del "bit" de inicio si REN = 1.**

* INTERCOMUNICACION CON OTROS MULTIPROCESADORES

Los **Modos 2** y **3** tienen una característica adicional para la **intercomunicación** con otros **multiprocesadores**. En estos **Modos 9 "bits" de datos** son recibidos, quedando el **noveno en RB8**, entonces viene **un "bit" de parada** ocasionando con esto que el puerto sea programado cuando el **"bit"** antes mencionado es recibido, la interrupción del **Puerto Serial** será activada solamente si **RB8 = 1**. Esta característica es activada estableciendo el **"bit" SM2** en **SCON**. Una manera de utilizar esta característica con multiprocesadores es como sigue:

- Cuando el procesador **"maestro"** quiere transmitir **un bloque de datos** a uno de varios **"esclavos"**, primero manda **un "byte" de dirección** el cual va a identificar la **tarjeta "esclava"**. **Un "byte" de dirección** difiere de **un "byte" de datos** en que el **noveno "bit" es 1 en el "byte" de direcciones y 0 en el de datos.**

- Con **SM2 = 1**, ningún **"esclavo"** será interrumpido por **un "byte" de datos. Un "byte" de direcciones** de cualquier manera, interrumpirá a todos los **"esclavos"**, solo que cada **"esclavo"** podrá examinar el **"byte"** recibido y verá si es el que está siendo direccionado.

- El "esclavo" que finalmente ha sido direccionado limpiará su "bit" SM2 y se preparará a recibir los"bytes" de datos que han de venir. Los "esclavos" que no fueron direccionados dejarán su "bit" SM2 establecido y seguirán con su trabajo ignorando los "bytes" de datos que vengan.

SM2 no tiene ningún efecto en el Modo 0, y en el Modo 1 solo puede ser utilizado para checar la validad del "bit" de parada. En la recepción del Modo 1, si SM2 = 1, la interrupción recibida no será activada hasta que el "bit" de parada valido sea recibido.

* REGISTRO DE CONTROL DEL PUERTO SERIAL

El Registro de Ccontrol y de Estado del Puerto Serial es el Registro de Función Especial SCON. Este registro contiene no solamente los "bits" de Modo de Selección, sino también el noveno "bit" de transmisión y recepción (TB8 y RB8), y los "bits" de Interrupción del Puerto Serial (TI y RI).

* VELOCIDADES DE TRANSMISION

La Velocidad de Transmisión en el Modo 0 es establecida por medio de la ecuación:

$$\text{Velocidad de Transmisión} = \frac{\text{Frecuencia del Oscilador}}{12}$$

La Velocidad de Transmisión en el Modo 2 depende del valor del "bit" SMOD en el Registro de Función Especial PCON. Si SMOD = 0 (el cuál es su valor después del "Reset"), la velocidad de transmisión es 1/64 de la frecuencia del oscilador. Si SMOD = 1, la velocidad de transmisión es 1/32 de la frecuencia del oscilador.

$$\text{Velocidad de Transmisión} = \frac{2^{\text{SMOD}}}{64} - (\text{Frecuencia de Oscilación})$$

En este circuito la velocidad de transmisión en el Modo 1 y 3 estan determinada por el "TIMER" 1, o por el "TIMER" 2, o por ambos (uno para transmisión y otro para recepción).

UTILIZANDO EL "TIMER" 1 PARA GENERAR VELOCIDADES DE TRANSMISION

Cuando el "**TIMER**" 1 es utilizado para generar la **velocidad de transmisión**, esta es determinada en los **Modos 1** y **3** por el rango de sobreflujo del "**TIMER**" y por el valor de **SMOD** como sigue:

$$\text{Velocidad de Transmisión} = \frac{2^{SMOD}}{32} \times (\text{Rango de Sobreflujo de el TIMER1})$$

La interrupción del "**TIMER**" 1 deberá ser deshabilitada en esta aplicación. Este "**TIMER**" por sí solo puede ser configurado ya sea para que funcione como "**TIMER**" o como **Contador**, y en cualquiera de los tres **modos de operación**. En la mayoría de las aplicaciones típicas, se configura para que opere como "**TIMER**", en el **modo de autorecarga** (en el "**nibble**" **alto de TMOD = 0010B**). En este caso, la **velocidad de transmisión** está dada por la fórmula:

$$\text{Velocidad de Transmisión} = \frac{2^{SMOD}}{32} \times \frac{\text{Frecuencia de Oscilación}}{(12)[256 - (TH1)]}$$

El usuario podrá obtener **velocidades de transmisión** bastante bajas dejando la interrupción del "**TIMER**" **1 habilitada**, y configurandolo para que funcione como un "**TIMER**" **de 16 "bits"** (en el "**nibble**" **alto de TMOD = 0001B**), y utilizando la interrupción del "**TIMER**" **1** para que trabaje como una recarga desde "**Software**" **de 16 "bits"**.

A continuación se verá una lista de varias **velocidades de transmisión** comunmente utilizadas y como pueden ser obtenidas desde el "**TIMER**" **1**.

VEL. DE TRANSMISION (BAUD)	Fosc	SMOD	C ó /T	MODO	VALOR DE AUTORECARGA
Modo 0 Max : 1M	12Mhz	X	X	X	X
Modo 2 Max : 375K	12Mhz	1	X	X	X
Modos 1, 3 : 62.5K	12Mhz	1	0	2	FFH

VEL. DE TRANSMISION (BAUD)	Fosc	SMOD	C ó /T	MODO	VALOR DE AUTORECARGA
19.2K	11.059Mhz	1	0	2	FDH
9.6K	11.059Mhz	0	0	2	FDH
4.8K	11.059Mhz	0	0	2	FAH
2.4K	11.059Mhz	0	0	2	F4H
1.2K	11.059Mhz	0	0	2	E8H
137.5	11.059Mhz	0	0	2	1DH
110	6Mhz	0	0	2	72H
110	12Mhz	0	0	1	FEEBH

* UTILIZANDO EL "TIMER" 2 PARA GENERAR VELOCIDADES DE TRANSMISION

El **"TIMER" 2** es seleccionado como un **Generador de Velocidades de Transmisión** estableciendo **TCLK** y/o **RCLK** en **T2CON**. Notese entonces que las **velocidades de transmisión y recepción** pueden ser simultáneamente diferentes.

El modo de **Generador de Velocidad de Transmisión** es similar al de **Autorecarga**, en que un regreso en **TH2** causa que el **registro del "TIMER" 2** sea recargado con un valor de **16 "bits"** en los **registros RCAP2H** y **RCAP2L**, los cuales son preestablecidos por medio de **"Software"**.

Ahora las **Velocidades de Transmisión** en los **Modos 1** y **3** son determinadas por el rango de sobreflujo del **"TIMER" 2** como sigue:

$$\text{Velocidad de Transmisión} = \frac{\text{Rango de Sobreflujo de el TIMER 2}}{16}$$

Este **"TIMER"** puede ser configurado para que opere ya sea como un **Contador** o como un **"TIMER"**. En la mayoría de las aplicaciones típicas, es configurado como un **"TIMER" (C o /T2 = 0)**. La operación como **"TIMER"** es un poco diferente para este caso cuando está siendo utilizado como un **Generador de Velocidades de Transmisión.**

Normalmente como un "**TIMER**" se incrementará cada ciclo de máquina (esto es a **1/12** de la frecuencia del oscilador). Como un **Generador de Velocidad de Transmisión**, de cualquier modo, se incrementará cada tiempo de estado (esto es a **1/12** de la frecuencia del oscilador). En este caso la **Velocidad de Transmisión** está dada por la formula:

$$\text{Velocidad de Transmisión} = \frac{\text{Frecuencia de Oscilación}}{(32)\,[\,65536 - (\,\text{RCAP2H,RCAP2L}\,)\,]}$$

donde (**RCAP2H, RCAP2L**) es el contenido de **RCAP2H** y **RCAP2L** tomado como un entero de **16 "bits"**.

Notese que un regreso en **TH2** no establece **TF2**, y por lo tanto no generará una interrupción. Entonces, la interrupción del "**TIMER**" **2** no tiene que ser deshabilitada cuando es utilizado en el modo de **Generador de Velocidad de Transmisión**. Nótese también, que si **EXEN2** está establecido, una transisión de **1** a **0** en **T2EX** establecerá a **EXF2** pero no causará una recarga desde (**RCAP2H, RCAP2L**) a (**TH2, TL2**). Entonces cuando el "**TIMER**" **2** está siendo utilizado como un **Generador de Velocidad de Transmisión**, **T2EX** puede ser utilizado como una **Interrupción Externa** extra si es deseado.

Deberá de ser notado que cuando el "**TIMER**" **2** está operando (**TR2 = 1**) en la función de "**TIMER**" en el modo de **Generador de Velocidad de Transmisión**, el usuario no deberá tratar de leer o escribir en **TH2** o **TL2**. Bajo estas condiciones el "**TIMER**" será incrementado cada estado de tiempo, y el resultado de un lectura o una escritura puede no ser preciso. Los **registros RCAP** pueden ser leidos, pero no deberán de ser escritos, pues una escritura puede traslapar y causar un error de escritura y/o de recarga. Se debe apagar el "**TIMER**" ("limpiar" **TR2**) antes del acceso al "**TIMER**" **2** o a los **registros RCAP** en este caso.

* MAS ACERCA DEL MODO 0

La transmisión es iniciada por cualquier intrucción que utiliza a **SBUF** como un **registro destino**. La señal de "escribe en **SBUF**" en **S6P2** también carga un **1** en el **noveno "bit"** del **registro de corrimiento** transmitido y le indica al bloque de control **TX** que comience la transmisión. Los tiempos internos serán tales que, un ciclo de máquina completo será intercalado entre "escribe en **SBUF**" y la activación del **SEND**.

SEND habilita la salida del **registro de corrimiento** a la línea de función de salida alternada **P3.0**, y también establece el Reloj de corrimiento de la línea de función de salida alternada **P3.1**. El **Reloj de Corrimiento** permanece en estado bajo (**0**) durante **S3**, **S4** y **S5** de cada ciclo de máquina, y en estado alto (**1**) durante **S6**, **S1** y **S2**. En **S6P2** de cada ciclo de máquina en el cual **SEND** está activo, el contenido del **registro de corrimiento** transmitido es modificado una posición a la derecha.

Así como los **"bits" de datos** son mandados recorriendolos hacia la derecha, los ceros entran desde la izquierda. Cuando el **"bit" más significativo del "byte" de datos** está en la posición de salida en el **registro de corrimiento**, entonces el **1** que estaba inicialmente cargado en la novena posición, se encuentra justo a la izquierda del **"bit" más significativo**, y todas las posiciones a la izquierda contienen ceros. Estas condiciones dan como resultado que el **bloque de control TX** haga un ultimo corrimiento desactivando por ultimo **SEND** y estableciendo **T1**. Ambas de estas acciones ocurren en **S1P1** del décimo ciclo de máquina después de "escribe en **SBUF**".

La recepción es iniciada por la condición **REN** = 1 y **RI** = 0. En **S6P2** del siguiente ciclo de máquina, la unidad de control **RX** escribe los "**bits**" **11111110 al registro de corrimiento** recibido, y la siguiente fase de reloj activa a **RECEIVE**.

RECEIVE habilita al **Reloj de Corrimiento** a la línea de función de salida alternada de **P3.1**. Este reloj realiza la transisión en **S3P1** y **S6P1** de cada ciclo de máquina. En **S6P2** de cada ciclo de máquina en el cual **RECEIVE** está activo, el contenido del **registro de corrimiento** recibido es recorrido a la izquierda una posisión. El valor que viene de la derecha es el valor que fue muestreado en la terminal **P3.0** durante **S5P2** del mismo ciclo de máquina.

Como el **"bit" de datos** viene desde la derecha, puros 1 se van recorriendo hacia la salida desde la izquierda. Cuando el **0** que estaba inicialmente cargado en la posisión de la extrema derecha llega a la posisión de la extrema izquierda en el **registro de corrimiento**, da como resultado que el bloque de control **RX** haga un ultimo corrimiento y carge **SBUF**. En **S1P1** del décimo ciclo de máquina después de escribir en **SCON** lo cual "limpia" **RI**, **RECEIVE** lo limpia y a **RI** establecido.

* MAS ACERCA DEL MODO 1

En este modo la **Velocidad de Transmisión** es determinada ya sea por el rango de sobreflujo del "**TIMER**" **1**, el del "**TIMER**" **2**, o de ambos (uno para la **transmisión** y otro para la **recepción**).

La **transmisión** se inicia por cualquier instrucción que utiliza a **SBUF** como un **registro destino**. La señal de "escribe en SBUF" también carga un **1** en el **noveno** "**bit**" **del registro de corrimiento transmitido** y le indica al **bloque de control TX** que una **transmisión** está siendo pedida. La **transmisión** actualmente comienza en **S1P1** de cada ciclo de máquina siguiendo el proximo regreso en el contador que está dividido en **16**. (Entonces los tiempos están sincronizados con el contador dividido en **16**, no con la señal de "escribe en SBUF")..

La **transmisión** comienza con la activación de /**SEND**, el cuál coloca el "**bit**" **de inicio en TXD**. Un tiempo después, **DATA** es activado, y habilita el "**bit**" **de salida del registro de corrimiento transmitido** a **TXD**. El primer pulso de corrimiento ocurre un tiempo despúes de esto.

Así como los "**bits**" **de datos** son mandados recorriendolos hacia la derecha, los **ceros** entran desde la izquierda conforme al reloj. Cuando el "**bit**" **más significativo del** "**byte**" **de datos** está en la posición de salida en el **registro de corrimiento**, entonces el **1** que estaba inicialmente cargado en la **novena posición**, se encuentra justo a la izquierda del "**bit**" **más significativo**, y todas las posiciones a la izquierda contienen **ceros**. Estas condiciones dan como resultado que el **bloque de control TX** haga un ultimo corrimiento desactivando por ultimo /**SEND** y estableciendo **T1**. Esto ocurre en el **décimo** regreso del contador después del "escribe en **SBUF**".

La **recepción** es iniciada por la detección de una transisión de **1** a **0** en **RXD**. Para este propósito **RXD** es muestreada en un rango de **16** veces en cualquier **velocidad de transmisión** que haya sido establecida. Cuando una transisión es detectada, el contador es inmediatamente reestablecido, y un **1FFH** es escrito en el **registro de corrimiento** de la entrada.

Los **16 estados del contador** dividen cada tiempo en **16** partes. En el **séptimo, octavo y noveno estados** del contador de cada tiempo, un detector de **"bits"** muestrea el valor de **RXD**. El valor aceptado es aquel que ha sido visto por lo menos en **dos** o **tres muestreos**, esto es realizado para **reducción de ruido**. Si el valor aceptado durante el primer tiempo no es **0**, los circuitos de recepción son reestablecidos y la unidad regresa revisar otra transisión de **1** a **0**. Esto provee una reducción de **"bits" falsos de inicio**. Si el **"bit" de entrada** prueba su validez, es recorrido al **registro de corrimiento** de entrada, y la recepción se llevará a cabo.

Como el **"bit" de datos** viene desde la derecha, puros **1** se van recorriendo hacia la salida desde la izquierda. Cuando el **"bit" inicial** llega a la posisión de la extrema izquierda en el **registro de corrimiento**, (el cual en el **Modo 1** es el **noveno "bit" del registro**), da como resultado que el **bloque de control RX** haga un ultimo corrimiento, carge **SBUF**, a **RB8** y establesca **RI**. La señal de carga en **SBUF** y **RB8**, y establece **RI**, será generada si, y solo si, las siguientes condiciones son encrontradas al mismo tiempo que el ultimo pulso de corrimiento es generado.

- **1) RI = 0.**
- **2) SM2 = 0 o es recibido un "bit" de parada = 1.**

Si cualquiera de estas dos condiciones no es encontrada, la información que está siendo recibida es irremediablemente perdida. Si ambas condiciones son encontradas, el **"bit" de parada** entra en **RB8**, los **8 "bits" de datos** van a **SBUF**, y **RI** es activada. En este tiempo, no importando si las condiciones antes mencionadas son encontradas o no, la unidad regresa a revisar otra transisión de **1** a **0** en **RXD**.

* MAS ACERCA DE LOS MODOS 2 Y 3

La porción recibida es exactamente igual a la del **Modo 1**. La porción transmitida difiere del **Modo 1** solamente en el **noveno "bit" del registro del corrimiento** de transmisión.

La transmisión es iniciada por cualquier instrucción que utiliza a **SBUF** como un **registro destino**. La señal de "escribe en **SBUF**" también carga un **1** en el **noveno "bit" del registro de corrimiento** transmitido y le indica al **bloque de control TX** que una transmisión está siendo pedida. La transmisión actualmente inicia en **S1P1** de cada ciclo de máquina siguiendo el próximo regreso en el contador que está dividido en **16**. (Entonces los tiempos están sincronizados con el contador dividido en **16**, no con la señal de "escribe en **SBUF**").

La transmisión comienza con la activación de **SEND**, el cuál coloca el **"bit" de inicio** en **TXD**. Un tiempo después, **DATA** es activado, y habilita el **"bit" de salida** del **registro de corrimiento** transmitido a **TXD**. El primer pulso de corrimiento ocurre un tiempo despúes de esto. El primer recorrimiento conforme al reloj coloca un **1** (que es el **"bit" de parada**) en la **novena posición del registro de corrimiento**. Después de esto, solamente **ceros** son introducidos. Entonces, como los **"bits" de datos** son mandados recorriendolos hacia la derecha, los **ceros** entran desde la izquierda conforme al reloj. Cuando **TB8** se encuentra en la posisión de salida del **registro de corrimiento**, entonce el **"bit" de parada** se encuentra a la izquierda de **TB8**, y todas la posisiones a la izquierdade el contienen **ceros**. Estas condiciones dan como resultado que el **bloque de control TX** haga un último corrimiento desactivando por último **SEND** y estableciendo **T1**. Esto ocurre en el **onceavo** regreso del contador después del "escribe en **SBUF**".

La recepción es iniciada por la detección de una transisión de **1** a **0** en **RXD**. Para este propósito **RXD** es muestreada en un rango de **16** veces en cualquier **velocidad de transmisión** que haya sido establecida. Cuando una transisión es detectada, el contador es inmediatamente reestablecido, y un **1FFH** es escrito en el **registro de corrimiento** de la entrada.

En el **séptimo, octavo y noveno estados del contador de cada tiempo**, un detector de **"bits"** muestrea el valor de **RXD**. El valor aceptado es aquel que ha sido visto por lo menos en dos o tres muestreos. Si el valor aceptado durante el primer tiempo no es **0**, los circuitos de recepción son reestablecidos y la unidad regresa revisar otra transisión de **1** a **0**. Si el **"bit" de entrada** prueba su validez, es recorrido al **registro de corrimiento de entrada**, y la recepción se llevará a cabo.

Como el **"bit" de datos** viene desde la derecha, puros **1** se van recorriendo hacia la salida desde la izquierda. Cuando el **"bit" inicial** llega a la posisión de la extrema

izquierda en el **registro de corrimiento**, (el cual en los **Modos 2** y **3** es el **noveno "bit" del registro**), da como resultado que el **bloque de control RX** haga un ultimo corrimiento, carge **SBUF**, a **RB8** y establesca **RI**. La señal de carga en **SBUF** y **RB8**, y establece **RI**, será generada si, y solo si, las siguientes condiciones son encrontradas al mismo tiempo que el ultimo pulso de corrimiento es generado.

- **1) RI = 0.**
- **2) SM2 = 0 o es el noveno "bit" recibido = 1.**

Si cualquiera de estas dos condiciones no es encontrada, la información que está siendo recibida es irremediablemente perdida y **RI** no será establecido. Si ambas condiciones son encontradas, el **noveno "bit"** entra en **RB8**, y los **8 primeros "bits" de datos** van a **SBUF**. Un tiempo después, no importando si las condiciones antes mencionadas son encontradas o no, la unidad regresa a revisar otra transisión de 1 a **0** en la entrada **RXD**.

Notese que el valor recibido del **"bit" de parada** es irrelevante para **SBUF**, **RB8** y **RI**.

INTERRUPCIONES

Las seis interrupciones que se encuentran en este circuito son mostradas a en la figura 2.16.

Las interrupciones externas /INT0 e **/INT1** pueden cada una ser activadas ya sea por **transisión** o por nivel, dependiendo de los **"bits" IT0** e **IT1 del registro TCON**. Las **banderas** que actualmente generan estas **interrupciones** son los **"bits" IE0** e **IE1** en **TCON**. Cuando una **interrupción externa** es generada, la **bandera** que la generó es **"limpiada"** por **"Hardware"** cuando la rutina de servicio es direccionada por medio de su vector solamente si la **interrupción es activada** por transisión. Si la interrupción es

activada por nivel, entonces la fuente externa que está requiriendo la **interrupción** es la que controla la **bandera** de petición, más que el "**Hardware**" interno.

Las **interrupciones** del "**TIMER**" **0** y **1** son generadas por **TF0** y **TF1**, las cuales son establecidas por el regreso en sus respectivos **registros** de "**TIMER/COUNTER**" (excepto el "**TIMER**" **0** en el **Modo 3**). Cuando la **interrupción** de un "**TIMER**" es generada, la **bandera** que la generó es "**limpiada**" por el "**Hardware**" interno cuando la **rutina de servicio es direccionada**.

La **interrupción del Puerto Serial** es generada por una **operación lógica** de **OR** entre **RI** y **TI**. Ninguna de estas **banderas** es "**limpiada**" por "**Hardware**" cuando la rutina de servicio es direccionada. De hecho, la rutina de servicio normalmente determinará cual de las dos (**RI** o **TI**) ha generado la **interrupción**, y el "**bit**" tendrá que ser puesto en **cero** por "**Software**".

La **interrupción** del "**TIMER**" **2** es generada por medio de una **operación lógica** entre **TF2** y **EXF2**. Ninguna de estas **banderas** es "**limpiada**" por "**Hardware**" cuando la rutina de servicio es direccionada. De hecho, la rutina de servicio normalmente determinará cual de las dos (**TF2** o **EXF2**) ha generado la **interrupción**, y el "**bit**" tendrá que ser puesto en **cero** por "**Software**".

Todos los "**bits**" que generan la **interrupción** pueden ser establecidos o puestos en **cero** por "**Software**", con el mismo resultado obtenido si son establecidos o puestos en **cero** por medio de "**Hardware**". Esto es, las interrupciones pueden ser generadas o la **interrupciones pendientes** pueden ser canceladas por medio de "**Software**".

Cada uno de estos **servicios de interrupción** pueden ser habilitados individualmente colocando en **cero** o en **uno** su "**bit**" correspondiente en el **Registro de Función Especial IE** (ver "**IE CONTROL DE HABILITACION DE INTERRUPCIONES**"). Nótese que **IE** contiene también un "**bit**" de deshabilitación global, **EA**, el cual deshabilita todas las **interrupciones** al mismo tiempo.

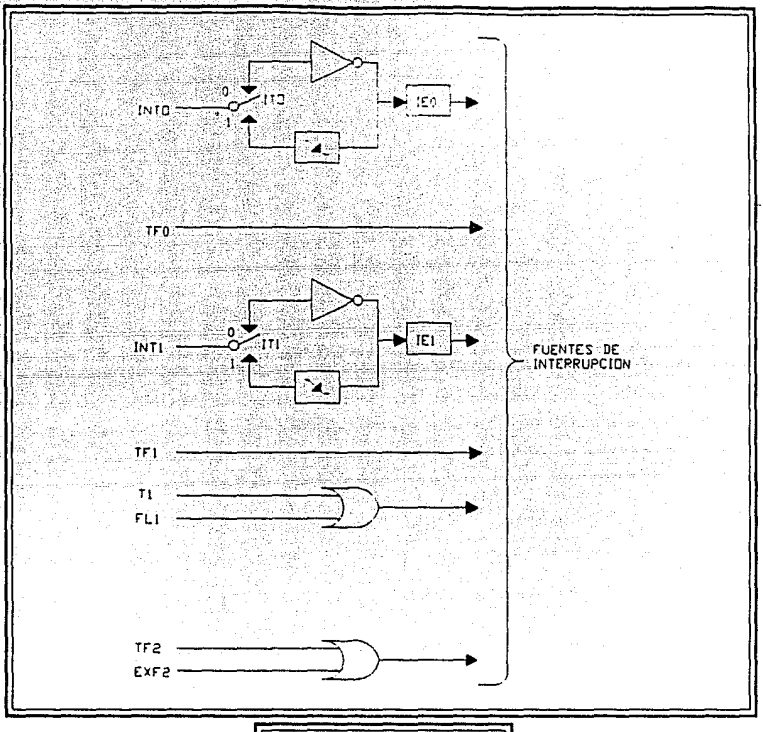

FIG. 2.16 Fuentes de Interrupción

* ESTRUCTURA DE NIVELES DE SEGURIDAD

Cada una de las **interrupciones** puede ser individualmente programada en **uno** o **dos niveles de prioridad** colocando en **uno** o en **cero** su **"bit"** en el **Registro de Función Especial IP** (ver **"IP REGISTRO DE PRIORIDAD DE LAS INTERRUPCIONES"**).

Una interrupción de baja prioridad puede ser interrumpida por una de alta prioridad, pero no por otra de la misma prioridad. Una interrupción de alta prioridad no podrá ser interrumpida por ningún otro recurso de interrupción.

Si dos requisiciones de diferentes tipos de prioridad son recibidas de manera simultánea, la de mayor prioridad es atendida. Si las requisiciones son de la misma prioridad una secuencia de revisión interna va a determinar cual es la que va a ser atendida.

Entonces aparte de cada nivel de prioridad hay una segunda estructura de prioridades determida, por dicha secuencia de revisión como sigue:

RECURSO	PRIORIDAD
IE0	MAYOR
TF0	|
IE1	|
TF1	|
RI + TI	|
TF2 + EXF2	MENOR

Nótese que la estructura de nivel de prioridad es solamente utilizada para resolver requisiciones simultáneas de interrupciones de la misma prioridad.

* COMO SON OBTENIDAS ESTAS INTERRUPCIONES

Las **banderas de interrupción** son muestreadas en **S5P2** de cada ciclo de máquina. Los muestreos son revisados durante el siguiente ciclo de máquina.

Si una de las **banderas** se encuentra en la condición de establecida en **S5P2** del ciclo precedente, el ciclo de revisión la encontrará y el **sistema de interrupción** generará un **LCALL** a la rutina de servicio apropiada, una vez provista este **LCALL** generado por medio de **"Hardware"** no será bloqueada por cualquiera de las siguientes condiciones:

- 1.- Una interrupción de igual o mayor nivel de prioridad está ya en proceso.

- 2.- El ciclo (de revisión) actual no es el ciclo final en la ejecución del la instrucción en proceso.
- 3.- La instrucción en proceso es RETI o cualquier acceso a los registros IE e IP.

Cualquiera de estas tres condiciones bloqueará la generación de **LCALL** para la **rutina de servicio de interrupción**. La **condición 2** asegura que la instrucción en proceso será completada antes de que sea direccionada cualquier rutina de servicio. La **condición 3** asegura que si la instrucción en proceso es **RETI** o cualquier acceso a **IE** o **IP**, entonces al menos una instrucción más será ejecutada antes de que una **interrupción** sea direccionada.

El ciclo de revisión es repetido con cada ciclo de máquina, y los valores revisados son los que se presenten en **S5P2** del ciclo de máquina previo.

Nótese entonces que si una **bandera de interrupción** es activada pero no ha sido respondida por cualquiera de las condiciones antes mencionadas, si la **bandera** sigue inactiva cuando la condición de bloqueo es removida, entonces la **interrupción** que ha sido rechazada no será atendida. En otras palabras, el hecho de que la **bandera** fue una vez activada pero no atendida no es recordado. Cada ciclo de revisión es nuevo.

El ciclo de búsqueda y la secuencia de **LCALL** es mostrado en la figura 2.17.

Nótese que si una **interrupción de mayor nivel de prioridad** se encuentra activo en **S5P2** del ciclo de máquina etiquetado como **C3** en la figura anterior, entonces de acuerdo a las reglas anteriores será direccionada durante **C5** y **C6**, sin que ninguna instrucción de la **rutina de baja prioridad** estuviera siendo ejecutada.

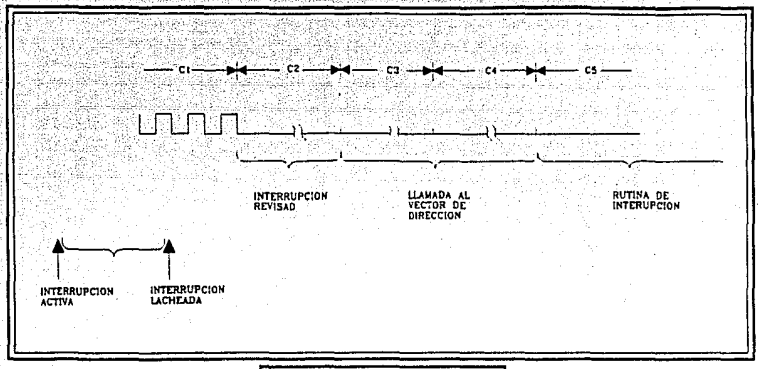

FIG. 2.17 Tiempos de Respuesta

Esto hace que el procesador habilite una **requisición de una interrupción** ejecutando un **LCALL** generado por **"Hardware"** a la **rutina de servicio** apropriada. En algunos casos también coloca un **cero** a la **bandera** que ha generado la **interrupción** y en otros casos no. Nunca pondrá en **cero** la **bandera** del **Puerto Serial** y la del **"TIMER" 2.**

Esto tiene que ser realizado por medio del **"Software"** del usuario. Coloca un **cero** en la **bandera** de **interrupción externa (IE0** o **IE1)** solamente si es activada en la transisión. El **LCALL** generado por **"Hardware"** coloca el contenido del **Contador de Programa ("Program Counter", PC)** dentro del **"Stack"** (ver **"ESTRUCTURA DE LOS PUERTOS Y OPERACIONES"** pero eso no salva la **Palabra de Estado del Programa PSW)** y recarga el **PC** con una dirección que depende de cual es la **interrupción** que está siendo direccionada, como se muestra a continuación:

RECURSO	VECTOR DE DIRECCION
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH

RECURSO	VECTOR DE DIRECCION
RI + TI	0023H
TF2 + EXF2	002BH

La ejecución procede desde esa localidad hasta que una instrucción de **RETI** es encontrada. La instrucción de **RETI** informa al procesador que esta **rutina de interrupción** ya no está en proceso, entonces saca el **"bit"** que se encuentra hasta arriba del **"Stack"** y recarga el **Contador de Programa** (**"Program Counter"**). La ejecución del programa interrumpido continua desde el lugar en donde se quedó.

Nótese que una simple instrucción de **RET** también regresará al programa que estaba siendo ejecutado, pero podrá haber dejado el **Sistema de Control de Interrupciones** pensando que una **interrupción** sigue en proceso aún.

* INTERRUPCIONES EXTERNAS

Los recursos externos pueden ser programados para ser activados por nivel o por transisión poniendo un uno o un cero en el **"bit" IT1** o **IT0** en el **registro TCON**. Si **ITx = 0**, la **interrupción externa** x es disparada por una detección de baja en la terminal **/INTx**. Si **ITx = 1**, la **interrupción externa x** es disparada en el flanco. En este **modo** si los muestreos sucesivos en la terminal **/INTx** muestran un estado alto en un ciclo y uno bajo en el siguiente, la **bandera de requisición IEx** en **TCON** se encuentra con un uno. La **bandera del "bit" IEx** entonces requiere la **interrupción**.

Desde que las terminales de las **interrupciones externas** son muestreadas una vez cada ciclo de máquina, una entrada alta o baja podrá ser retenida por lo menos **12** períodos de oscilación para asegurar el muestreo. Si la **interrupción externa** es activada por transisión, el recurso externo tiene que retener el estado alto de requisición en la terminal por lo menos un ciclo, y entonces retener el estado bajo por lo menos un ciclo para asegurar que la transisión sea vista de modo tal que la **bandera de requisición** de la **interrupción IEx** será colocada en **uno**. **IEx** automáticamente será colocada en **cero** por la **CPU** cuando la **rutina de servicio** es llamada.

Si la **interrupción externa** es activada en el nivel, el recurso externo debe de retener la **requisición** activa hasta que la **interrupción** requerida es generada. Entonces tiene

que ser desactivada la **requisición** antes de que la **rutina de servicio de la interrupción** es completada, o cualquier otra **interrupción** será generada.

* TIEMPO DE RESPUESTA

Los niveles de /INT0 e /INT1 son invertidos y "latcheados" en **IE0** e **IE1** en **S5P2** de cada ciclo de máquina. Los valores no son revisados por la circuitería en este momento sino hasta el siguiente ciclo de máquina. Si una **requisición** es activada y las condiciones son verdaderas para ser habilitada, una subrutina de "**Hardware**" hace un llamado a la **rutina de servicio** requerida y va a ser la siguiente instrucción a ejecutar. La llamada por sí misma toma dos ciclos. Entonces, por lo menos 3 ciclos completos de máquina transcurren entre la activación de una **requisición** de una **interrupción externa** y el comienzo de la ejecución de la primera instrucción de la **rutina de servicio.**

Un tiempo largo de respuesta será como resultado si la **requisición** es bloqueada por una de las 3 condiciones listadas con anterioridad. Si una **interrupción** de igual o mayor nivel de prioridad está en proceso, el tiempo adicional de espera obviamente depende de la naturaleza de la otra **rutina de servicio** de la **interrupción.** Si la instrucción en proceso no se encuentra en su ciclo final, el tiempo de espera adicional no puede ser de más de 3 ciclos, partiendo del hecho de que la instrucciones más largas (**MUL** y **DIV**) son de 4 ciclos solamente, y si la instrucción en proceso es **RETI** o un acceso a **IE** o **IP**, el tiempo adicional de espera no puede ser mayor a **5** ciclos (un máximo de **un** ciclo más para completar la instrucción en proceso, más 4 ciclos para completar la siguiente instrucción si es que esta es **MUL** o **DIV**).

Entonces, en un **sistema de una sola interrupción**, el tiempo de respuesta es siempre mayor a 3 ciclos y menor que 9.

OPERACION PASO A PASO ("SINGLE-STEP")

La **estructura de interrupciones** del **8052AH-BASIC** permite la operación en "**Single-Step**" con muy poco "**Software**". Como se notó previamente, una **requisición de una interrupción** no será atendida cuando una **interrupción** de igual nivel de prioridad está en proceso, y no será atendida hasta después de un **RETI** y por lo menos que alguna otra instrucción esté siendo ejecutada.

Entonces, si una **rutina de interrupción** ha entrado no puede ser introducida nuevamente hasta que por lo menos una instrucción del programa interrumpido es ejecutada. Una manera de utilizar esta característica para la operación de **"Single-Step"** es programando una **interrupción** externa (por ejemplo **/INT0**) para que sea activada por nivel. La **rutina de servicio** para la **interrupción** terminará con el siguiente código:

JNB	P3.2,$	;Espera aquí hasta que INT0 se encuentre en estado alto
JB	P3.2,$	;Espera aquí hasta que se encuentra en estado bajo
RETI		;Regresa y ejecuta una instrucción

Ahora si la terminal **/INT0**, la cual también es la terminal **P3.2**, es sostenida normalmente en estado bajo, la **C.P.U.** estará en estado verdadero en la **rutina de interrupción externa 0** y permanecerá ahí hasta que **/INT0** es pulsada (de bajo a alto y nuevamente a bajo). Entonces ejecutará un **RETI**, regresará al programa, ejecutará una instrucción, e inmediatamente volverá a la **rutina de interrupción externa 0** para la espera del próximo pulso de **P3.2**. Un paso del programa es ejecutado cada vez que **P3.2** es pulsado.

"RESET"

La entrada del **"Reset"** es la terminal **RST**, la cual es la entrada a un **"Schmitt Trigger"**.

Este **"Reset"** es completado sosteniendo un estado alto en la terminal **RST** por lo menos **2** ciclos de máquina (**24** períodos de oscilación), cuando el oscilador se encuentra funcionando.

La señal externa del "Reset" es **asincrona** con respecto al reloj interno. La terminal **RST** es muestreada durante **S5P2** de cada ciclo de máquina. Las terminales del puerto mantendrán sus corrientes activas por **19** períodos de reloj después de que un **1** lógico haya sido muestreado en la terminal **RST**; esto es, de **19** a **31** períodos de oscilación

después de la señal externa de "**Reset**" ha sido aplicada a la terminal **RST**.

Esto también configura las terminales de **ALE** y/**PSEN** como entradas (siendo estas cuasi-bidireccionales). El "**Reset**" interno es ejecutado durante el **segundo** ciclo en el cual **RST** se encuentra en estado alto y es repetido en cada ciclo de máquina hasta que **RST** se encuentra en estado bajo. Esto deja los **registros internos** como sigue:

REGISTRO	CONTENIDO
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	0FFH
IP	XX000000B
IE	0X000000B
TMOD	00H
TCON	00H
T2CON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
TH2	00H
TL2	00H
RCAP2H	00H
RCAP2L	00H

REGISTRO	CONTENIDO
SCON	00H
SBUF	INDETERMINADO
PCON	0XXXXXXXB

La **RAM interna** no es afectada por el "Reset".

* "RESET" DURANTE EL ENCENDIDO

Un **"Reset"** automático puede ser obtenido cuando es alimentado el **"Kit"** conectando la terminal **RST** a **Vcc** a través de un capacitor de **10 microfaradas** y a tierra a través de una resistencia de **8.2 KE**, previniendo así que el tiempo de subida no **exceda un** milisegundo y el tiempo de inicio del oscilador no exceda **10** milisegundos. Este circuito de **"Reset"** de encendido es mostrado en la siguiente figura:

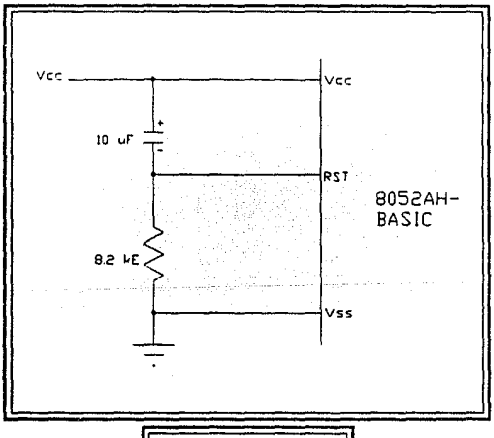

FIG. 2.18 Figura del "Reset"

Cuando se inicia la alimentación del circuito este mantiene la teminal de **RST** en un estado alto durante un tiempo que depende del valor del capacitor y el rango durante el cual se carga. Para asegurar un buen **"Reset"** la terminal **RST** debe de

mantenerse en un estado alto lo suficiente para permitir al oscilador iniciar (normalmente pocos milisegundos) mas dos ciclos de máquina.

Nótese que las terminales de los puertos estarán en un estado aleatorio hasta que el oscilador haya comenzado y un algoritmo de "**Reset**" interno haya escrito un **1** en ellos.

Con este circuito, reduciendo Vcc rápidamente a **0** cuasa que el voltaje en la terminal **RST** momentáneamente baje de **0** volts. De cualquier manera, este voltaje es internamente limitado, y no daña al circuito.

CAPITULO III

PROGRAMACION

En el siguiente capítulo se hablará de las instrucciones que nos sirven para la programación del **8052AH-BASIC**. Este será dividido en dos partes:

- **1) Juego de Instrucciones en Ensamblador**
- **2) Juego de Instrucciones en BASIC.**

JUEGO DE INSTRUCCIONES EN ENSAMBLADOR

Este juego de instrucciones está compuesto básicamente por **111 instrucciones**, las cuales se dividen de la siguiente manera:

- **A) 49 de un solo "byte".**
- **B) 45 de dos "bytes".**
- **C) 17 de tres "bytes".**

El **formato del código de operación** consiste de un **mnemónico propio de la función** y de **un campo de operando** el cual va a ser destino. Este campo especifica el tipo de dato y el o los métodos de direccionamiento que deben de utilizarse.

Asi mismo este **juego de instrucciones** se divide en cuatro grupos de acuerdo a su funcionamiento. Estos grupos son:

- **a) Transferencia de Datos.**
- **b) Aritmético.**
- **c) Lógico.**
- **d) Transferencia de Control.**

Dentro de cada uno de estos tipos de funcionamiento hay divisiones las cuales se verán a continuación con sus respectivas instrucciones. La nomenclatura a seguir es:

> **Rn - Registro R7-R0 del Banco de Registros** seleccionado actualmente.

> **direct** - Dirección de la localidad interna de datos de **8 "bits"**. Esta puede ser una localidad interna de la **RAM (0-127)** o un **Registro de Función Especial (128-255)**.

> **@Ri** - Localidad interna de datos de la **RAM de 8 "bits" (0-255)** direccionada indirectamente por el **registro R1** o **R0**.

> **#data** - Constante de **8 "bits"** incluida en la instrucción.

> **#data 16** - Constante de **16 "bits"** incluida en la instrucción.

> **addr 16** - Dirección destino de **16 "bits"**. Utilizada por **LCALL** y **LJMP**.

> **addr 11** - Dirección destino de **11 "bits"**. Utilizada por **ACALL** y **AJMP**.

> **rel - "Byte" de relacionamiento con signo.** Utilizado por **SJMP** y todos los saltos condicionales. El rango va desde **-128** hata **+127** **"bytes"** relativos al **primer "byte"** de la siguiente instrucción.

> **bit - "Bit" de direccionamiento directo** en la **RAM de datos** o **Registro de Función Especial.**

* TRANSFERENCIA DE DATOS

Las operaciones de la transferencia de datos se dividen en 3 clases:

° **1) Propósito General.**

° **2) Específicas del Acumulador.**

° **3) Dirección-Objeto.**

Ninguna de estas operaciones afecta el establecimiento de la **bandera de la Palabra de Estado del Programa (PSW)**, excepto un **POP** o **MOV** directos a **PSW**.

- **MOV A,Rn**
  OPERACION : Mueve el registro Rn al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 1 0 1 r r r
  BANDERAS AFECTADAS : PARIDAD

- **MOV A,direct**
  OPERACION : Mueve contenido de esa dirección al Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 1 1 1 0 0 1 0 1 DIRECCION
  BANDERAS AFECTADAS : PARIDAD

- **MOV A,@Ri**
  OPERACION : Mueve lo que se encuentra en la localidad de la RAM al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 1 0 0 1 1 i
  BANDERAS AFECTADAS : PARIDAD

- **MOV A,#data**
  OPERACION : Mueve el dato al Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 1 1 0 1 0 0 DATO
  BANDERAS AFECTADAS : PARIDAD

- **MOV Rn,A**
  OPERACION : Mueve el contenido del Acumulador al registro Rn.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 1 1 1 r r r
  BANDERAS AFECTADAS : NINGUNA

- **MOV Rn,direct**
  OPERACION : Mueve el contenido de esa dirección al registro Rn.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 1 0 1 r r r DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **MOV Rn,#data**
  OPERACION : Mueve el dato al registro Rn.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 1 1 1 r r r DATO
  BANDERAS AFECTADAS : NINGUNA

- **MOV direct,A**
  OPERACION : Mueve el Acumulador a esa dirección.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 1 1 1 1 0 1 0 1 DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **MOV direct,Rn**
  OPERACION : Mueve el registro Rn a esa dirección.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 0 0 1 r r r DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **MOV direct,direct**
  OPERACION : Mueve el contenido de la segunda dirección a la primera dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 1 0 0 0 0 1 0 1 DIR(FUENTE) DIR(DESTINO)
  BANDERAS AFECTADAS : NINGUNA

- **MOV direct,@Ri**
  OPERACION : Mueve el contenido de una dirección de la RAM a la dirección.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 0 0 0 1 1 i DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **MOV direct,#data**
  OPERACION : Mueve el dato al"byte"de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 1 1 1 0 1 0 1 DIRECCION DATO
  BANDERAS AFECTADAS : NINGUNA

- **MOV @Ri,A**
  OPERACION : Mueve el Acumulador a una dirección de RAM.
  No. DE "BYTES" : 1

CICLOS : 1
CODIFICACION : 1 1 1 1 0 1 1 i
BANDERAS AFECTADAS : NINGUNA

- **MOV @Ri,direct**
  OPERACION : Mueve el contenido de una dirección a una dirección de RAM.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 1 0 0 1 1 i DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **MOV @Ri,#data**
  OPERACION : Mueve el dato inmediato a una dirección de RAM.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 1 1 0 1 1 i DATO
  BANDERAS AFECTADAS : NINGUNA

- **MOV DPTR,#data16**
  OPERACION : Carga el Apuntador de Datos con una constante de 16 "bits".
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 1 0 0 1 0 0 0 0 DATOS_15-8 DATOS_7-0
  BANDERAS AFECTADAS : NINGUNA

- **MOV A,@A+DPTR**
  OPERACION : Suma el Acumulador con el Apuntador de Datos y mueve el contenidodeesecódigodedirección al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 1 0 0 1 0 0 1 1
  BANDERAS AFECTADAS : PARIDAD

- **MOVC A,@A+PC**
  OPERACION : Suma el Acumulador con el Contador de Programa y mueve el contenido de ese código de dirección al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 1 0 0 0 0 0 1 1
  BANDERAS AFECTADAS : PARIDAD

- **MOVX A,@Ri**
  OPERACION : Mueve el contenido de una dirección de 8 "bits" de la RAM externa direccionada por el Registro al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 2

CODIFICACION : 1 1 1 0 0 0 1 i
BANDERAS AFECTADAS : PARIDAD

- **MOVX A,@DPTR**
  OPERACION : Mueve el contenido una dirección de 16 "bits" de la RAM externa direccionada por el Apuntador de Datos al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 1 1 1 0 0 0 0 0
  BANDERAS AFECTADAS : PARIDAD

- **MOVX @Ri,A**
  OPERACION : Mueve el contenido del Acumulador a una dirección de la RAM externa de 8 "bits" direccionada por el Registro.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 1 1 1 1 0 0 1 i
  BANDERAS AFECTADAS : NINGUNA

- **MOVX @DPTR,A**
  OPERACION:Mueve el contenido del Acumulador a una dirección de la RAM externa de 16 "bits" direccionada por el Apuntador de Datos.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 1 1 1 1 0 0 0 0
  BANDERAS AFECTADAS : NINGUNA

- **PUSH direct**
  OPERACION : Incrementa el Apuntador del "Stack" (SP) y coloca el contenido de la dirección en la parte más alta del "Stack".
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 1 0 0 0 0 0 0 DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **POP direct**
  OPERACION : Coloca el contenido de la parte más alta del "Stack" en la dirección señalada y decrementa el SP.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 1 0 1 0 0 0 0 DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **XCH A,Rn**
  OPERACION : Intercambia el contenido del Registro con el del Acumulador.
  No. DE "BYTES" : 1

CICLOS : 1
CODIFICACION : 1 1 0 0 1 r r r
BANDERAS AFECTADAS : PARIDAD

- **XCH A,direct**
  OPERACION : Intercambia el contenido de la dirección con el del Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 1 1 0 0 0 1 0 1 DIRECCION
  BANDERAS AFECTADAS : PARIDAD

- **XCH A,@Ri**
  OPERACION : Intercambia el contenido de una dirección de la RAM con el del Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 0 0 0 1 1 i
  BANDERAS AFECTADAS : PARIDAD

- **XCHD A,@Ri**
  OPERACION : Intercambia la parte menos significativa ("bits" 3-0) del contenido de una dirección de la RAM con la parte menos significativa del contenido del Acumulador ("bits" 3-0).
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 0 1 0 1 1 i
  BANDERAS AFECTADAS : PARIDAD

* ARITMETICO

Aquí se cuenta con cuatro operaciones matemáticas básicas:

- Suma.
- Resta.
- Multiplicación.
- División.

Solamente operaciones de 8 "bits" utilizando aritmética sin signo son soportadas directamente. La **bandera de sobreflujo**, de cualquier manera, permite la **suma** y la **resta** sirviendo así a enteros binarios con o sin signo. Esta aritmérica puede ser realizada ya sea directamente o en forma de representación **BCD**.

- **ADD A,Rn**
  OPERACION : Le suma el contenido del Registro al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 1 0 1 r r r
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR.

- **ADD A,direct**
  OPERACION : Suma el contenido de la direlción al Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 0 1 0 0 1 0 1 DIRECCION
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **ADD A,@Ri**
  OPERACION : Suma lo que se encuentra en la localidad de la RAM al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 1 0 0 1 1 i
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **ADD A,#data**
  OPERACION : Suma el dato inmediato al Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 0 1 0 0 1 0 0 DATO
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **ADDC A,Rn**
  OPERACION : Suma el Acarreo y el Registro al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 1 1 1 r r r
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **ADDC A,direct**
  OPERACION : Suma el Acarreo y el contenido de esa dirección al Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 0 1 1 0 1 0 1 DIRECCION
  BANDERAS ATADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **ADDC A,@Ri**
  OPERACION:Suma el Acarreo y el contenido de la RAM al Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 1 1 0 1 1 i
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **ADDC A,#data**
  OPERACION : Suma el Acarreo y dato al Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 0 1 1 0 1 0 0 DATO
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **SUBB A,Rn**
  OPERACION : Resta el registro Rn del Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 0 0 1 1 r r r
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **SUBB A,direct**
  OPERACION:Resta el contenido de es dirección del Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 1 0 0 1 0 1 0 1
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **SUBB A,@Ri**
  OPERACION: Resta el contenido de una dirección de la RAM del Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 0 0 1 0 1 1 i

BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **SUBB A,#data**
  OPERACION : Resta el dato del Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 1 0 0 1 0 1 0 0 DATO
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO, ACARREO Y ACARREO AUXILIAR

- **INC A**
  OPERACION : Incrementa el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 0 0 1 0 0
  BANDERAS AFECTADAS : PARIDAD

- **INC Rn**
  OPERACION : Incrementa el registro.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 0 1 r r r
  BANDERAS AFECTADAS : NINGUNA

- **INC direct**
  OPERACION : Incrementa el contenido de esa dirección.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 0 0 0 0 1 0 1 DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **INC @Ri**
  OPERACION : Incrementa el contenido de una dirección de RAM.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 0 0 1 1 i
  BANDERAS AFECTADAS : NINGUNA

- **DEC A**
  OPERACION : Decrementa el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 1 0 1 0 0
  BANDERAS AFECTADAS : PARIDAD

- **DEC Rn**
  OPERACION : Decrementa el Registro.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 1 1 r r r
  BANDERAS AFECTADAS : NINGUNA

- **DEC direct**
  OPERACION : Decrementa el contenido de una dirección.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 0 0 1 0 1 0 1 DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **DEC @Ri**
  OPERACION : Decrementa el contenido de una dirección de RAM.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 1 0 1 1 i
  BANDERAS AFECTADAS : NINGUNA

- **INC DPTR**
  OPERACION : Incrementa el Apuntador de Datos.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 1 0 1 0 0 0 1 1
  BANDERAS AFECTADAS : NINGUNA

- **MUL AB**
  OPERACION : Multiplica A & B.
  No. DE "BYTES" : 1
  CICLOS : 4
  CODIFICACION : 1 0 1 0 0 1 0 0
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO Y ACARREO

- **DIV AB**
  OPERACION : Divide A por B.
  No. DE "BYTES" : 1
  CICLOS : 4
  CODIFICACION : 1 0 0 0 0 1 0 0
  BANDERAS AFECTADAS : PARIDAD, SOBREFLUJO Y ACARREO

- **DA A**
  OPERACION: Es es ajuste decimal del Acumulador después de una suma BCD.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 0 1 0 1 0 0
  BANDERAS AFECTADAS : PARIDAD Y ACARREO

* LOGICO

Se realizan operaciones lógicas básicas en operandos ya sea de un "**bit**"o un "**byte**".

- **ANL A,Rn**
  OPERACION:Realiza una operación lógica de AND entre el Registro y el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 1 0 1 1 r r r
  BANDERAS AFECTADAS : PARIDAD

- **ANL A,direct**
  OPERACION:Realiza una operación lógica de AND entre el contenido de la dirección y el Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 0 1 0 1 0 1 DIRECCION
  BANDERAS AFECTADAS : PARIDAD

- **ANL A,@RI**
  OPERACION:Realiza una operación lógica de AND entre el contenido de la RAM y el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 1 0 1 0 1 1 i
  BANDERAS AFECTADAS : PARIDAD

- **ANL A,#data**
  OPERACION:Realiza una operación lógica de AND entre el dato y el Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 0 1 0 1 0 0 DATO
  BANDERAS AFECTADAS : PARIDAD

- **ANL direct,A**
  OPERACION:Realiza una operación lógica de AND entre el Acumulador y el

contenido de la dirección.
No. DE "BYTES" : 2
CICLOS : 1
CODIFICACION : 0 1 0 1 0 0 1 0 DIRECCION
BANDERAS AFECTADAS : NINGUNA

- **ANL direct,#data**
  OPERACION:Realiza una operación lógica de AND entre el dato y el contenido de la dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 1 0 1 0 0 1 1 DIRECCION DATO
  BANDERAS AFECTADAS : NINGUNA

- **ORL A,Rn**
  OPERACION:Realiza una operación lógica de OR entre el Registro y el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 1 0 0 1 r r r
  BANDERAS AFECTADAS : PARIDAD

- **ORL A,direct**
  OPERACION:Realiza una operación lógica de OR entre el contenido de la dirección y el Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 0 0 0 1 0 1 DIRECCION
  BANDERAS AFECTADAS : PARIDAD

- **ORL A,@Ri**
  OPERACION:Realiza una operación lógica de OR entre el contenido de la dirección de RAM y el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 1 0 0 0 1 1 i
  BANDERAS AFECTADAS : PARIDAD

- **ORL A,#data**
  OPERACION : Realiza una operación lógica de OR entre el dato y el Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 0 0 0 1 0 0 DATO
  BANDERAS AFECTADAS : PARIDAD

- **ORL direct,A**
  OPERACION:Realiza una operación lógica de OR entre el Acumulador y el

contenido de la dirección.
No. DE "BYTES" : 2
CICLOS : 1
CODIFICACION : 0 1 0 0 0 0 1 0 DIRECCION
BANDERAS AFECTADAS : NINGUNA

- **ORL direct,#data**
  OPERACION : Realiza una operación lógica de OR entre el dato y el contenido de la dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 1 0 0 0 0 1 1 DIRECCION DATO
  BANDERAS AFECTADAS : NINGUNA

- **XRL A,Rn**
  OPERACION :Realiza una operación lógica de OR Exclusiva (XOR) entre el Registro y el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 1 1 0 1 r r r
  BANDERAS AFECTADAS : PARIDAD

- **XRL A,direct**
  OPERACION:Realiza una operación lógica de OR Exclusiva (XOR) entre el contenido de la dirección y el Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 1 0 0 1 0 12 DIRECCION
  BANDERAS AFECTADAS : PARIDAD

- **XRL A,@Ri**
  OPERACION:Realiza una operación lógica de OR Exclusiva (XOR) entre el contenido de la dirección de RAM y el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 1 1 0 0 1 1 i
  BANDERAS AFECTADAS : PARIDAD

- **XRL A,#data**
  OPERACION:Realiza una operación lógica de OR Exclusiva (XOR) entre el dato y el acumulador.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 1 0 0 1 0 0 DATO
  BANDERAS AFECTADAS : PARIDAD

- **XRL direct,A**
  OPERACION :Realiza una operación lógica de OR Exclusiva (XOR) entre el Acumulador y el contenido de la dirección.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 0 1 1 0 0 0 1 0 DIRECCION
  BANDERAS AFECTADAS : NINGUNA

- **XRL direct,#data**
  OPERACION:Realiza una operación lógica de OR Exclusiva (XOR) entre el dato y el contenido de la dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 1 1 0 0 0 1 1 DIRECCION DATO
  BANDERAS AFECTADAS : NINGUNA

- **CLR A**
  OPERACION : Carga con 0 el Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 1 0 0 1 0 0
  BANDERAS AFECTADAS : PARIDAD

- **CPL A**
  OPERACION : Complementa cada "bit" del Acumulador.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 1 1 0 1 0 0
  BANDERAS AFECTADAS : PARIDAD

- **RL A**
  OPERACION:Rota el contenido del Acumulador hacia la izquierda una posición.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 1 0 0 0 1 1
  BANDERAS AFECTADAS : NINGUNA

- **RLC A**
  OPERACION: Rota el contenido del Acumulador y el Acarreo hacia la izquierda una posición.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 1 1 0 0 1 1
  BANDERAS AFECTADAS : PARIDAD Y ACARREO

- **RR A**
  OPERACION:Rota el contenido del Acumulador hacia la derecha una posición.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 0 0 0 1 1
  BANDERAS AFECTADAS : NINGUNA

- **RRC A**
  OPERACION : Rota el contenido del Acumulador y el Acarreo hacia la derecha una posición.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 1 0 0 1 1
  BANDERAS AFECTADAS : PARIDAD Y ACARREO

- **SWAP A**
  OPERACION : Intercambia la parte baja del Acumulador ("bits" 3-0) con la parte alta ("bits" 7-4).
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 0 0 0 1 0 0
  BANDERAS AFECTADAS : NINGUNA

* TRANSFERENCIA DE CONTROL

Hay tres clases de operaciones de **transferencia de control**:

- **1) Incondicionales.**
- **2) Regresos.**
- **3) Saltos, Saltos Condicionales e Interrupciones.**

Todas las operaciónes de **transferencia de control** causan que continue la ejecución del programa en una localidad no secuencial en la memoria de programa.

- **ACALL addr11**
  OPERACION : Guarda el Contador de Programa en el "stack" y reemplaza los 11 "bits" de la parte baja con los 11 "bits" de la parte baja del codigo de dirección.
  No. DE "BYTES" : 2
  CICLOS : 2

CODIFICACION : a10 a9 a8 1 0 0 0 1 a7 a6 a5 a4 a3 a2 a1 a0
BANDERAS AFECTADAS : NINGUNA

- **LCALL addr16**
  OPERACION: Guarda el Contador de Programa en el "stack" y reemplaza el valor completo del Contador de Programa con el código de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 0 0 1 0 0 1 0 addr15-addr8 addr7-addr0
  BANDERAS AFECTADAS : NINGUNA

- **RET**
  OPERACION : Es el regreso de una llamada a una subrutina.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 0 0 1 0 0 0 1 0
  BANDERAS AFECTADAS : NINGUNA

- **RETI**
  OPERACION:Es el regreso de una rutina que ha sido interrumpida.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 0 0 1 1 0 0 1 0
  BANDERAS AFECTADAS : NINGUNA

- **AJMP addr11**
  OPERACION:Reemplaza los 11 "bits" de la parte baja del Contador de Programa con los 11 "bits" de la parte baja del codigo de dirección.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : a0 a9 a8 0 0 0 0 1 a7 a6 a5 a4 a3 a2 a1 a0
  BANDERAS AFECTADAS : NINGUNA

- **LJMP addr16**
  OPERACION:Realiza un salto "largo" al código de dirección.
  No. E "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 0 0 0 0 0 1 0 addr15-addr8 addr7-addr0
  BANDERAS AFECTADAS : NINGUNA

- **SJMP rel**
  OPERACION : Realiza un salto "corto" al código de dirección.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 0 0 0 0 0 0 DIRECCION_RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **JMP @A + DPTR**
  OPERACION : Suma el Acumulador con el Apuntador de Datos y realiza un salto a ese código de dirección.
  No. DE "BYTES" : 1
  CICLOS : 2
  CODIFICACION : 0 1 1 1 0 0 1 1
  BANDERAS AFECTADAS : NINGUNA

- **JZ rel**
  OPERACION :Si el Acumulador es cero salta al código de dirección.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 0 1 1 0 0 0 0 0 DIRECCION_RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **JNZ rel**
  OPERACION: Si el Acumulador es diferente de cero salta al código de dirección.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 0 1 1 1 0 0 0 0 DIRECCION_RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **CJNE A,direct,rel**
  OPERACION : Si el contenido de la dirección y el Acumulador no son iguales, salta al código de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 1 0 1 1 0 1 0 1 DIRECCION DIR._RELATIVA
  BANDERAS AFECTADAS : ACARREO

- **CJNE A,#data,rel**
  OPERACION :Si el dato y el Acumulador no son iguales, realiza un salto al código de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 1 0 1 1 0 1 0 0 DATO DIRECCION_RELATIVA
  BANDERAS AFECTADAS : ACARREO

- **CJNE Rn,#data,rel**
  OPERACION : Si el dato y el Registro no son iguales, realiza un salto al código de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 1 0 1 1 1 r r r DATO DIRECCION_RELATIVA
  BANDERAS AFECTADAS : ACARREO

- **CJNE @Ri,#data,rel**
  OPERACION : Si el dato y el contenido de la dirección de RAM no son iguales, salta al código de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 1 0 1 1 0 1 1 i DATO DIRECCION_RELATIVA
  BANDERAS AFECTADAS : ACARREO

- **DJNZ Rn,rel**
  OPERACION: Decrementa el Registro, si no es igual a cero, salta al código de operación.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 1 0 1 1 r r r DIRECCION_RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **DJNZ direct,rel**
  OPERACION : Decrementa la Dirección y si es cero, entonces salta al código de operación.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 1 1 0 1 0 1 0 1 DIRECCION DIR._RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **NOP**
  OPERACION : No hace nada, se utiliza como un retardo.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 0 0 0 0 0 0 0 0
  BANDERAS AFECTADAS : NINGUNA

- **CLR C**
  OPERACION : Coloca ceros en la bandera de Acarreo.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 0 0 0 0 1 1
  BANDERAS AFECTADAS : ACARREO

- **CLR "bit"**
  OPERACION : Coloca un cero en el "bit".
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 1 1 0 0 0 0 1 0 DIRECCION_DEL_"BIT"
  BANDERAS AFECTADAS : NINGUNA

- **SETB C**
  OPERACION : Coloca un uno en la bandera de Acarreo.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 1 0 1 0 0 1 1
  BANDERAS AFECTADAS : ACARREO

- **SETB "bit"**
  OPERACION : Coloca un uno en el "bit".
  No. DE "BYTES" : 2
  CICLOS : 1
  BANDERAS AFECTADAS : NINGUNA

- **CPL C**
  OPERACION : Complementa C.
  No. DE "BYTES" : 1
  CICLOS : 1
  CODIFICACION : 1 0 1 1 0 0 1 1
  BANDERAS AFECTADAS : ACARREO

- **CPL "bit"**
  OPERACION:Complementa el "bit".
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 1 0 1 1 0 0 1 0 DIRECCION_DEL_"BIT"
  BANDERAS AFECTADAS : NINGUNA

- **ANL C,"bit"**
  OPERACION:Realiza una operación lógica de AND entre el "bit" y el Acarreo.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 0 0 0 0 1 0 DIRECCION_DEL_"BIT"
  BANDERAS AFECTADAS : ACARREO

- **ANL C,/"bit"**
  OPERACION :Realiza una operación lógica de AND entre el complemento del "bit" y el Acarreo.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 1 1 0 0 0 0 DIRECCION_DEL_"BIT"
  BANDERAS AFECTADAS : ACARREO

- **ORL C,"bit"**
  OPERACION : Realiza una operación lógica de OR entre el "bit" y el Acarreo.
  No. DE "BYTES" : 2

CICLOS : 2
CODIFICACION : 0 1 1 1 0 0 1 0 DIRECCION_DEL_"BIT"
BANDERAS AFECTADAS : ACARREO

- **ORL C,/"bit"**
  OPERACION :Realiza una operación lógica de OR entre el complemento del "bit" y el Acumulador.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 1 0 0 0 0 0 DIRECCION_DEL_"BIT"
  BANDERAS AFECTADAS : ACARREO

- **MOV C,"bit"**
  OPERACION : Mueve el valor del "bit" al Acarreo.
  No. DE "BYTES" : 2
  CICLOS : 1
  CODIFICACION : 1 0 1 0 0 0 1 0 DIRECCION_DEL_"BIT"
  BANDERAS AFECTADAS : ACARREO

- **MOV "bit",C**
  OPERACION : Mueve el valor del Acarreo al "bit".
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 1 0 0 1 0 0 1 0 DIRECCION_DEL_"BIT"
  BANDERAS AFECTADAS : NINGUNA

- **JC rel**
  OPERACION: Si la bandera de Acarreo es igual a uno, salta al código de dirección.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 0 1 0 0 0 0 0 0 DIRECCION_RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **JNC rel**
  OPERACION : Si la bandera de Acarreo es igual a cero, salta al código de dirección.
  No. DE "BYTES" : 2
  CICLOS : 2
  CODIFICACION : 0 1 0 1 0 0 0 0 DIRECCION_RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **JB "bit",rel**
  OPERACION:Si el "bit" es igual a 1, salta al código de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 0 1 0 0 0 0 0 DIR._"BIT" DIR._RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **JNB "bit",rel**
  OPERACION : SI el "bit" es igual a cero, salta al código de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 0 1 1 0 0 0 0 DIR._"BIT" DIR._RELATIVA
  BANDERAS AFECTADAS : NINGUNA

- **JBC "bit",rel**
  OPERACION: SI el "bit" es igual a uno, limpia el "bit" y salta al código de dirección.
  No. DE "BYTES" : 3
  CICLOS : 2
  CODIFICACION : 0 0 0 1 0 0 0 0 DIR._"BIT" DIR._RELATIVA
  BANDERAS AFECTADAS : NINGUNA

JUEGO DE INSTRUCCIONES EN BASIC

El **MCS BASIC-52** como ya se mencionó con anterioridad es un **intérprete de BASIC** con todos los comandos y declaracion. estandard que tiene cualquier **BASIC normal.**

El poderío del **MCS BASIC-52** radica básicamente en que es ideal para la realización de tareas que no pueden tener una terminal permanente para ejecutar algún programa, ya que tiene la habilidad de almacenar y ejecutar programas que se encuentren previamente almacenados en algún **EPROM** y, procesar interrupciones desde el mismo programa en **BASIC**. Adicionalmente **rutinas aritméticas** y de E/S pueden ser accesadas desde la **rutinas en Ensamblador**, esto ayuda al programador a evitar la escritura de difíciles y a veces hasta tediosos programas.

Este juego de instrucciones está compuesto por **114 instrucciones**, las cuales se dividen básicamente en comandos y sentencias.

A continuación veremos algunas definiciones de terminos utilizados dentro de este punto.

> **COMANDO** .- El **MCS BASIC-52** opera en **dos modos**, el **Modo de COMMAND** o **Directo** y el **Modo de Interprete** o de **RUN**. Los comandos solamente pueden ser introducidos cuando el

procesador se encuentra en **Modo Directo**, y realiza la acción inmediatamente después de que el comando ha sido introducido.

> **SENTENCIA** .- Un programa en **BASIC** está siempre compuesto por **sentencias**. Cada sentencia comienza con un **número de línea**, seguido por el "**cuerpo**" **de la sentencia** y termina con un ("**Carriage Return**", **Regreso de Carro**), o con **dos puntos** (:) en el caso de tener varias sentencias en una sola línea. Algunas sentencias pueden ser ejecutadas en el **Modo Directo**, y algunas otras no.

Hay **tres tipos generales de sentencias** en el **MCS BASIC-52**:

- **1) Asignaciones.**
- **2) Entrada/Salida.**
- **3) Control.**

Dentro de la estructura del programa debemos indicar que cada línea del programa debe de tener un número asignado el cual va desde **0** hasta **65,535** inclusive (y que debe de ser utilizado una sola vez), los cuales son utilizados para ordenar las sentencias de forma secuencial.

Las sentencias no deben de ser necesariamente introducidas en forma secuencial, pues el **BASIC** las ordena automáticamente en forma ascendente.

Dependiendo de la versión del **MCS BASIC-52**, una sentencia no puede constar de más de **72 (versión** 1.0) o de **79 (version 1.1)** caracteres. Aquí cabe hacer notar que los espacios en blanco son ignorados por **BASIC** y automáticamente los inserta cuando un comando de **LIST** es realizado.

SENTENCIAS DE FORMATO .- Estas **sentencias** pueden ser utilizadas solamente con las sentencias de **PRINT**. Estas icluyen **TAB([expr])**, **SPC([expr])**, **USING** (simbolos especiales), y **CR** ("**Carriage Return**", **Regreso de Carro** sin que salte a la siguiente línea).

> **FORMATO DE LOS DATOS** .- El rango de números que pueden ser representados por el **MCS BASIC-52** es:

+ - 1 E-127 hasta + - 0.99999999 E + 127

Tenemos **8 dígitos significativos.** Los números son internamente redondeados a esta presición, y pueden ser introducidos y desplegados de **cuatro formas:**

1) **Enteros.**
2) **Decimales.**
3) **Hexadecimales.**
4) **Exponenciales.**

ENTERO .- En el **MCS BASIC-52**, los **enteros** son números que se encuentran en el rango de **0** a **65,535** o **0FFFFH**. Todos los enteros pueden ser introducidos ya sea en formato **decimal** o **hexadecimal**, y todos los números hexadecimales deben de ser introducidos anteponiendo un dígito válido (por ejemplo el número **AF43H** debe de ser introducido como **0AF43H**). Cuando un operador como la operación lógica de **AND** (**.AND.**) requiere de un entero, el **MCS BASIC-52** truncará la parte franccionaria del número y así lo dejará en un formato de entero. Todos los números de línea utilizados por el **BASIC** son enteros. En la explicación nos referiremos a los enteros y a los números de línea como sigue:

Entero	[integer]
Número de Línea	[ln num]

> **CONSTANE** .- Una **constante** es un número real que se encuentra en el rango de **+ - 1 E-127** a **+ - 0.99999999 E + 127.** Una constante puede ser entera y nos referiremos a ella como **[const]**.

> **OPERADORES** .- Un **operador** como su nombre lo indica, realiza una operación ya predefinida con variables y/o constantes, y requieren al menos uno o dos operandos. Típicamente dos operandos son incluidos en las siguientes sentencis: **ADD (+), SUBSTRACT (-), MULTIPLY (*),** y **DIVIDE** (/). Los operadores que requieren de un solo operando son referidos contínuamente como **Operadores Unitarios** y son: **SIN (seno), COS (coseno)** y **ABS (valor absoluto).**

> **VARIABLES** .- Una variable para el **MCS BASIC-52** puede ser definida como una letra (p.e. **S, H, A**), una letra seguida por un número (p.e. **S5, H2, A8**), una letra seguida por una expresión de una sola dimensión (p.e. **S(5), H(2), A(8)**), o una letra seguida por un número y una expresión de una sola dimensión (p.e. **A1(8), P7(DBY(9)), W8(A+B)**). De cualquier manera las variables pueden contener hasta **8 caracteres o número** incluyendo los caracteres de subrayado (_). Esto permite al usuario utilizar un nombre más descriptivo para una variable dada.

Cuando se utilizan nombres expandidos de variables es importante notar que:

1) Le toma al MCS BASIC-52 un poco de tiempo el procesarlos.

2) El usuario no puede utilizar cualquier palabra como el nombre de una variable (esto es que por ejemplo no se puede utilizar el nombre de DIET o TABLA pues TAB e IE son palabras reservadas).

Errores de mala sintaxis pueden ser generados si el usuario utiliza un nombre que contenga una palabra reservada.

Las variables que incluyen **Expresiones de una Dimensión** son referidas contínuamente como **Dimensionamientos** o **Arreglos**. Variables que solo involucran una letra o una letra y un número son llamadas como **Escalares**. Para hacer referencia a las variables las nombraremos [**var**].

Cada vez que el **BASIC** utiliza una variable le asigna una porción de memoria (**8 "bytes"**) específicos, y que son transparentes al usuario, esto es, que este espacio físico no es disponible. La única manera de que un usuario pueda "**limpiar**" este espacio es ejecutando una sentencia de **CLEAR**.

Al **MCS BASIC-52** le toma menos tiempo el localizar una **variable escalar** que una **Dimensionada**, esto es por que no es necesario evaluar ninguna expresión en una variable escalar.

> **EXPRESIONES** .- Una **expresión** es una fórmula matemática lógica que incluye operadores (unitario o con varios operandos), constantes, y variables, y pueden ser simples o con algo de complejidad. Una sola variable o constante también es considerada como una expresión, nos referiremos a ellas como **[expr]**.

> EXPRESIONES RELACIONALES .- La **expresiones relacionales** incluyen operadores como **Igual (=)**, **Diferente (< >)**, **Mayor Que (>)**, **Menor Que (<)**, **Mayor o Igual Que (> =)** y **Menor o Igual Que (< =)**. Son utilizadas en sentencias de control para probar una condición. Siempre requieren de dos operandos y nos referiremos a ellas como **[rel expr]**.

> **OPERADORES DE FUNCION ESPECIAL** .- Virtualmente todos los **Registros de Función Especial** en el **8052AH** pueden ser accesados utilizando **operadores de función especial.** Las excepciones son los **Puertos 0, 2 y 3**, y los registros que no se encuentran asociados a las entradas y salidas como son el **Acumulador, el registro B**, y la **Palabra de Estado del Programa (PSW)**. Otros operadores de función especial son **XTAL** y **TIME.**

> VALORES DEL CONTROL DEL SISTEMA .- Aquí se encuentra los siguientes valores de control del sistema:

° **1) LEN** - El cual da la longitud del programa.

° **2) FREE** - Que designa cuantos "bytes" de RAM que ha sido asignada a BASIC no estan siendo utilizados.

° **3) MTOP** - La cual es la última localidad de memoria que ha sido asignada a BASIC.

> **ESTRUCTURA DEL "STACK"** .- EL **MCS BASIC-52** reserva los primeros **512 "bytes"** de la memoria de datos externa para

implemetar dos "Stack" de "Software", los cuales son el de Control y el Aritmético o de Argumento. Ademas de estos dos tipos de "Stack", cuenta con uno Interno.

- 1) "STACK" DE CONTROL - Este "Stack" ocupa las localidades desde la **96** (**60H**) hasta **254** (**0EFH**) en la memoria **RAM externa**. Esta área es utilizada para almacenar toda la información asociada con el control de lazos y subrutinas básicas. El "Stack" es inicializado en la **254** (**0EFH**) y va **decreciendo**.

- 2) "STACK" DE ARGUMENTO - Este "Stack" ocupa desde la localidad **301** (**12DH**) hasta la **510** (**1FEH**) en la memoria **RAM externa**. Este "Stack" almacena todas las constantes que el **MCS BASIC-52** utiliza contínuamente. Las operaciones como **ADD, SUBTRACT, MULTIPLY,** y **DIVIDE** simpre realizan su operación con los dos primeros elementos que se encuentren en este "Stack" y regresan el resultado a esta misma área. Este "Stack" es inicializado en la localidad **510** (**1FEH**) y va decreciendo conforme son almacenados más valores. Todo número con punto flotante requiere de **6** "**bytes**" para su almacenamiento.

- 3) "STACK" INTERNO - El Apuntador de "Stack" en el **8052AH** (**Registro de Función Especial, SP**) se inicializa en la localidad **77** (**4DH**). Este apuntador crece conforme se van almacenando los valores. En el **MCS BASIC-52** el usuario tiene la opción de colocar el **Apuntador de "Stack"** es donde quiera (**arriba de la localidad 77**) en la **memoria interna**.

  > **EDITOR DE LINEA** .- El **MCS BASIC-52** tiene un editor de línea de bajo nivel. Una vez que la línea es tecleada el usuario no puede cambiarla a menos de que vuelva a ser tecleada. De cualquier manera, es posible borrar caracteres cuando hay un error encontrandoce aún en esa línea con la tecla de **DELETE** o con el caracter (**7FH**). Adicionalmente al oprimir **CRTL-D** se borrará la línea entera. En la **versión 1.1 del MCS BASIC-52** se

tiene habilitados **CTRL-S** y **CTRL-Q** los cuales sirven para pausar la consola y activarla nuevamente respectivamente.

A continuación se mencionarán las intrucciones en **BASIC** y una breve explicación de cada una de ellas.

• COMANDOS

- **RUN**
  Inicia con la ejecución del programa siempre en la primera línea. Puede ser utilizado en Modo de Comando o de Interprete. La ejecución puede sepuede interrumpir presionando simultaneamente las teclas de **CTRL** y **C** (**CTRL-C**) desde la consola.

- **CONT**
  Si el programa es detenido por un **CTRL-C** o con la ejecución de la sentencia **STOP**, tecleando éste comando continúa la ejecución.

- **LIST**
  Este comando imprime el programa en la consola, y puede ser interrumpido por un **CTRL-C** desde la misma.

- **LIST#**
  Este comando imprime el programa por medio de un periférico (generalmente una Impresora Serial).

- **LIST@**
  Este comando realiza la misma función que el anterior con la excepción de que el dispositivo de impresión es uno designado por el mismo usuario.

- **NEW**
  Con este comando es borrado el programa que actualmente se encuentra almacenado en la memoria **RAM**, adicionalmente todas las variables son cargadas con cero, y todas la interrupciones que habían sido llamadas son ignoradas.

- **NULL**
  Este comando determina cuantos caracteres nulos el **MCS BASIC-52**, el cual sera colocado después de un regreso de carro. Después de la inicialización **NULL = 0**.

* COMANDO DE ARCHIVO DEL EPROM

Una de las poderosas habilidades del **MCS BASIC-52** es la de salvar y ejecutar programas que se encuentren almacenados en un **EPROM**, ya que genera todas los tiempos necesarios para la programación de los mismos.

- **RAM y ROM [integer]**
  Con estos comandos se va a llamar el modo de **RAM** o **ROM** para así poder seleccionar los programas actuales que se enuentran en las memorias **RAM**, o **ROM/EPROM** respectivamente.

- **XFER**
  Este comando transfiere el programa que ha sido seleccionado en el **EPROM** a la **RAM** para poder ser editado y modificado, y selecciona el Modo de Interprete.

- **PROG**
  Con este comando se graba el programa actual en el **EPROM** que está residente. Después de tecleado este comando el **MCS BASIC-52** despliega el número de programa que ocupa dentro de la memoria.

- **PROG1 y PROG2**
  PROG1 salva el programa actual en el **EPROM** y almacena la velocidad de transmisión, mientras que **PROG2** salva el programa actual en **EPROM** almacena también la velocidad de transmisión y después de un "**Reset**" se ejecuta automáticamente.

- **FPROG, FPROG1 y FPROG2**
  FPROG, FPROG1 y **FPROG2**, realizan la misma función que **PROG, PROG1** y **PROG2**, respectivamente con la diferencia de que el algoritmo utilizado para la programación es el algoritmo

de rapida progamación de **INTEL "Intelligent"**. Para esta función el usuario debe de preveer una alimentación de 6 volts para el **EPROM**.

- **PROG3, PROG4, FPROG3 y FPROG4**
  La función de **PROG3** es igual que la de **PROG1** con la excepción de que también salva el valor **MTOP**, cuando este es accesado. **PROG4** realiza la misma función que **PROG2** solo que también salva la información de MTOP.

- **PROG5, PROG6, FPROG5 y FPROG6**
  PROG5 y FPROG5
  El comando **PROG5** y **FPROG5** realizan la misma función que **PROG4** y **FPROG4** excepto que la memoria **RAM** externa no es "limpiada" si tiene un **0A5H** en la localidad **5EH**.
  PROG6 y FPROG6
  Realiza la misma operación que **PROG5** y **FPROG5** excepto que hace un llamado a la localidad de memoria externa **4039H** después de un **RESET**.

* DESCRIPCION DE SENTENCIAS

- **BAUD [expr]**
  Esta sentencia trabaja ya sea en Modo Directo y/o en Modo de Interprete, y es utilizada par establecer el rango de la velocidad de transmisión a la cual va a trabajar la línea 7 del puerto 1 (que es exclusivamente para la impresora) residente en el **8052AH-BASIC**.

- **CALL [integer]**
  Trabaja en Modo Directo y/o de Interprete. Esta sentencia es utilizada para llamar un programa que se encuentre en lenguaje Ensamblador.

- **CLEAR**
  Trabaja en Modo Directo y/o de Interprete. Esta sentencia coloca cero en todas las variables, inicializa las interrupciones y el "**Stack**".

- **DATA, READ, RESTORE**
  Solo trabaja en Modo de Interprete.

° **DATA** - especifica expresiones que van a ser accesadas por la sentencia **READ**.

° **READ** - Llama la expresiones especificadas en **DATA** y asigna el valor de la expresión a la variable dada en **READ**.

° **RESTORE** - Inicializa el apuntador del **READ** e inicia a leer los datos nuevamente.

- **DIM**
  Trabaja en Modo Directo y/o de Interprete. **DIM** reserva un área para el almacenamiento de matrices.

- **DO UNTIL [rel expr]**
  Trabaja en Modo de Interprete. Esta sentencia provee un control de lazo dentro de un programa del **MCS BASIC-52**. Todas las sentencias que se encuentren dentro de estas dos serán ejecutadas hasta que la expresión relacional que sigue al **UNTIL** sea verdadera.

- **DO WHILE [rel expr]**
  Trabaja en Modo de Interprete. Su operación es similar a la de la sentencia anterior, excepto que todas la sentencias que se encuentren dentro de estas dos serán ejecutadas mientras que la expresión relacional sea verdadera.

- **END**
  Trabaja en Modo de Interprete. Esta sentencia termina con la ejecución del programa, por lo tanto debe de ser la última línea dentro del mismo.

- **FOR[var] = [integer1] TO [integer2] {STEP}NEXT[var]**
  Trabaja en Modo de Interprete. Este es otro tipo de lazo en el cual se ejecutan las líneas siguientes a la sentencia **FOR TO** y la sentencia **NEXT**. Entonces, el contador se incrementa (o decrementa según el signo) con la cantidad especificada por el valor **STEP**. Si no se especifica un valor para **STEP**, se supondrá que el incremento es uno.

- **GOSUB [ln num] RETURN**
  Trabaja en Modo de Interprete. La sentencia **GOSUB [ln num]** transfiere el control del programa directamente al número de línea indicado y guarda la dirección en el "**Stack**". **RETURN** regresa el control al programa principal en la siguiente sentencia que se encuentre después del **GOSUB** que fue ejecutado.

- **IF, THEN, ELSE**
  Trabaja en Modo de Interprete. La sentencia **IF** establece una prueba condicional. La forma generalizada es como sigue:
  [ln num] IF [rel expr] THEN sentencia valida ELSE sentencia valida.

- **INPUT**
  Trabaja en Modo de Interprete. Esta sentencia permite al usuario introducir datos desde la consola durante la ejecución de un programa.

- **LET**
  Trabaja en Modo Directo y/o de Interprete. Esta sentencia es utilizada para asignar a una variable el valor de una expresión.

- **ONERR [ln num]**
  Trabaja en Modo de Interprete. Esta sentencia premite al programador revisar los errores aritméticos, que pueden ocurrir durante la ejecución de un programa. El código de error es como sigue:

Código de Error = 10	División por Cero
Código de Error = 20	Sobreflujo Aritmético
Código de Error = 40	Mal Argumento

- **ONEX1 [ln num]**
  Trabaja en Modo de Interprete. Esta sentencia permite al usuario el manejo de interrupciones en la terminal **INT1** del **8052AH-BASIC** por medio de un programa en **BASIC**. El número de línea indica a donde se debe de transferir el control al detectarse una interrupción. El usuario debe de ejecutar un **RETI** para salir de esta rutina de interrupción.

- **ONTIME [expr], [ln num]**
  Trabaja en Modo de Interprete. La sentencia **ONTIME [expr], [ln num]** genera una interrupción cada vez que el operador **TIME** es igual o mayor que la expresión que sigue a la sentencia, y con esto forza a transferir el control al número de línea que los sigue.

- **PRINT, P., ?**
  Trabaja en Modo Directo y/o de Interprete. Esta sentencia dirige la salida a una consola. El valor de expresiones, literales, variables, pueden ser "impresas" en la consola. La sentencia especiales para el formato de impresión son las siguientes:

° **TAB([expr])** - Esta función le dice al **MCS BASIC-52** exactamente en que posición iniciará a imprimir el siguiente valor en la lista de impresión.

° **SPC([expr])** - Es utilizada para que se "impriman" un cierto número de espacios en blanco especificados por la expresión.

° **CR** - Esta función es utilizada para forzar un regreso de carro, pero no hay un avance de línea.

- **USING**(caracteres especiales). Esta da el formato en el cuál van a ser desplegados los valores que se van a imprimir. Las opciones son las siguientes:

  * **USING(Fx)** - Imprimirá utilizando el formato de punto flotante.

  * **USING(#,#)** - Imprimirá utilizando formato de número entero o fraccional.

  * **USING(0)** - Este argumento permite al **MCS BASIC-52** determinar cual formato utilizar, siguiendo las siguientes reglas: Si el número se encuentra entre **+- 999999999** y **+- 0.1**, el **BASIC** desplegará enteros y fracciones. Si se encuentra fuera de este rango el **BASIC** utilizará el formato **USING(F0)**. Después del "**RESET**" el **MCS BASIC-52** se encuentra en formato **USING(0)**.

- **PRINT#, P.#, ?#**
  Estas sentencias realizan la misma función que las anteriores solo que la salida la direccionan a una impresora.

- **PH0., PH1., PH0.#, PH1.#**
  Trabaja en Modo Directo y/o de Interprete. **PH0.** y **PH1.** realizan la misma función que **PRINT** solo que los valores son impresos en formato hexadecimal.

- **PRINT@, PH0.@, PH1.@**
  Trabaja en Modo Directo y/o de Interprete. Las sentencias **PRINT@**, **PH0.@** y **PH1@** realizan la misma función que **PRINT**, **PH0.**, y **PH1**. respectivamente con la excepción de que la salida es dirigida a un dispositivo designado por el usuario.

- **PUSH [expr]**
  Trabaja en Modo Directo y/o de Interprete. Las expresiones aritméticas son evaluadas y secuencialmente colocadas dentro del "**Stack**" de Argumento. Esta sentencia junto con la de **POP**

proveen una manera simple de traspaso de parámetros durante una rutina en lenguaje Ensamblador.

- **POP [var]**
  Trabaja en Modo Directo y/o de Interprete. La parte más alta del "**Stack**" de Argumento, es asignada a una variable que va a ser la que va a seguir a esta sentencia, y de esta manera extraer la información del "**Stack**".

- **PWM [expr], [expr], [expr]**
  Trabaja en Modo Directo y/o de Interprete. **PWM** viene de las siglas "**Pulse Width Modulation**" (**Modulación del Ancho del Pulso**). La primera expresión va a definir el número de ciclos de reloj en el cual se va a mantener en estado alto. La segunda expresión va a ser el número de ciclos de reloj en el cual se va a mantener en estado bajo, y para finalizar la tercera expresión va a definir el número total de ciclos que el usuario quiere a la salida. El mínimo valor para las dos primeras expresiones es de **25** siendo el máximo de 65535..

- **REM**
  Trabaja en Modo Directo y/o de Interprete. Sirve para colocar comentarios dentro de un programa y es transparente en el momento de correr el mismo.

- **RETI**
  Trabaja en Modo de Interprete. Sirve para salir de interrupciones manejadas por el **MCS BASIC-52** por ejemplo **ONTIME** y **ONEX1**.

- **STOP**
  Trabaja en Modo de Interprete. Esta sentencia se utiliza para detener el programa en un punto específico y después continuarlo con la instrucción **CONT**.

- **STRING [expr], [expr]**
  Trabaja en Modo Directo y/o de Interprete. Un "**String**" representa una cadena de caracteres alfanuméricos. Esta sentencia designa localidades de memoria en las cuales van a estar almacenados los "**String**". La primera expresión define el número de "**bytes**" que se van a utilizar para el almacenamiento, la segunda define el número máximo de "**bytes**" que hay en cada "**String**".

- **UI1 y UI0**
  Trabaja en Modo Directo y/o de Interprete. La sentencia **UI1** permite al usuario escribir a dispositivos de entrada específicos (como consola). La sentencia **UI0** regresa la rutina de la consola a el "software" residente en el **MCS BASIC-52**.

- **UO1 y UO0**
  Trabaja en Modo Directo y/o de Interprete. La sentencia **UO1** permite al usuario escribir a dispositivos de salida específicos (como consola). La sentencia **UO0** regresa la rutina de la consola a el "**Software**" residente en el **MCS BASIC-52**.

- **IDLE**
  Trabaja en Modo de Interprete. Esta sentencia forza al **8052AH-BASIC** a permanecer en espera de un modo de interrupción.

- **RROM [integer]**
  Trabaja en Modo Directo y/o de Interprete. Ejecuta un programa que se encuentra en el **EPROM**.

- **LD@ [expr] y ST@ [expr]**
  Trabaja en Modo Directo y/o de Interprete. La sentencia **ST@** permite especificar donde se van a almacenar números de punto flotante pertenecientes al **MCS BASIC-52**. La expresión especifica la dirección del "**Stack**" donde se almacenarán. En conjunto la sentencia **LD@** permite llamar los números que han sido almacenados previamente.

- **PGM**
  Trabaja en Modo Directo y/o de Interprete. Permite grabar un **EPROM** o **EEPROM** mediante la ejecución de un programa en **BASIC**. Al finalizar se tiene que checar que las localidades **1EH** y **1FH** contengan cero si es así el **EPROM** ha sido programado correctamente.

- **CLEARS**
  Trabaja en Modo Directo y/o de Interprete. Borra todo el **"Stack"**.

- **CLEARI**
  Trabaja en Modo Directo y/o de Interprete. Borra todas la interrupciones que habían sido requeridas.

- **CLOCK1**
  Trabaja en Modo de Interprete. Habilita el Reloj de Tiempo Real que se encuentra en el **MCS BASIC-52**.

- **CLOCK0**
  Trabaja en Modo de Interprete. Deshabilita el Reloj de Tiempo Real que se encuentra en el **MCS BASIC-52**.

- **GOTO [ln num]**
  Trabaja en Modo de Interprete. Realiza una salto al número de línea especificado.

- **ON [var] GOTO [ln num]**
  Trabaja en Modo de Interprete. Realiza una salto al número de línea especificado cuando la variable sea verdadera.

- **ON [var] GOSUB [ln num]**
  Trabaja en Modo de Interprete. Realiza una transferencia de control al número de línea especificado cuando la variable sea verdadera.

* DESCRIPCION DE EXPRESIONES Y OPERADORES LOGICOS, ARITMETICOS

- **[expr] + [expr]**
  Operador para adición.

- **[expr]/[expr]**
  Operador para división.

- **[expr]**[expr]**
  Operador de exponente. La primera expresión es elevada a la potencia designada por la segunda expresión. El mayor número de exponente es 255.

- **[expr]*[expr]**
  Operador demultiplicación.

- **[expr]-[expr]**
  Operador de sustracción.

- **[expr].AND.[expr]**
  Realiza una operación lógica de **AND**.

- **[expr].OR.[expr]**
  Realiza una operación lógica de **OR**.

- **[expr].XOR.[expr]**
  Realiza una operación lógica de **OR** exclusivo.

- **ABS([expr])**
  Regresa el valor absoluto de la expresión.

- **NOT([expr])**
  Regresa el complemento de la expresión, es decir, los unos los convierte en ceros y viceversa.

- **INT([expr])**
  Regresa la parte entera de la expresión.

- **SGN([expr])**
  Regresará el valor de 1 si la expresión es mayor a cero, de cero si es igual y de -1 si es menor a cero.

- **SQR([expr])**
  Regresa la raíz cuadrada de la expresión.

- **RND**
  Regresa un número aleatorio.

- **PI**
  Es la única constante almacenada y su valor es de **3.1415926.**

- **LOG([expr])**
  Regresa el logaritmo natural de la expresión.

- **EXP([expr])**
  Eleva el número e (**2.7182818**) a la expresión.

- **SIN([expr])**
  Regresa el seno de la expresión la cual debe de estar en radianes.

- **COS([expr])**
  Regresa el coseno de la expresión la cual debe de estar en radianes.

- **TAN([expr])**
  Regresa la tangente de la expresión la cual debe de estar en radianes.

- **ATN([expr])**
  Regresa el arcotangente de la expresión la cual debe de estar en radianes.

- **ASC([expr])**
  Regresa el valor **ASCII** de la expresión.

- **CHR([expr])**
  Regresa el caractér **ASCII** al cual corresponde la expresión.

* OPERADORES ESPECIALES

- **CBY([expr])**
  Este operador es utilizado para leer dator de la memoria de programas del **8052AH-BASIC**.

- **DBY([expr])**
  Este operador es utilizado para leer o asignar un valor de la memoria interna de datos del **8052AH-BASIC**.

- **XBY([expr])**
  Este operando es utilizado para leer o asignar un valor de la memoria externa de datos.

- **GET**
  Lee de la consola y asigna el valor leído a una constante.

- **IE**
  Este operando es utilizado para leer o asignar un valor al Registro de Función Especial **IE**.

- **IP**
  Este operando es utilizado para leer o asignar un valor al Registro de Función Especial **IP**.

- **PORT1**
  Este operando es utilizado para leer o asignar un valor al Puerto de Entrada/Salida 1 del **8052AH-BASIC**.

- **PCON**
  Este operando es utilizado para leer o asignar un valor al Registro de Función Especial **PCON**.

- **RCAP2**
  Este operando es utilizado para leer o asignar un valor al Registro de Función Especial **RCAP2**.

- **T2CON**
  Este operando es utilizado para leer o asignar un valor al Registro de Función Especial **T2CON**.

- **TCON**
  Este operando es utilizado para leer o asignar un valor al Registro de Función Especial **TCON**.

- **TMOD**
  Este operando es utilizado para leer o asignar un valor al Registro de Función Especial **TMOD**.

- **TIME**
  Este operando es utilizado para leer o asignar un valor al Reloj de Tiempo Real residente en el **MCS BASIC-52**.

- **TIMER0**
  Este operando es utilizado para leer o asignar un valor a los Registros de Función Especial **TH0** y **TL0**.

- **TIMER1**
  Este operando es utilizado para leer o asignar un valor a los Registros de Función Especial **TH1** y **TL1**.

- **TIMER2**
  Este operando es utilizado para leer o asignar un valor a los Registros de Función Especial **TH2** y **TL2**.

- **XTAL**
  Este operando informa al **MCS BASIC-52** a que frecuencia está operando el sistema.

- **MTOP**
  Después del "**RESET**" el **MCS BASIC-52** dimensiona la memoria externa y asigna el valor de la última dirección a **MTOP**.

- **LEN**
  Informa cuantos "bytes" ocupa el programa actual.

- **FREE**

  Informa cuantos "bytes" de la memoria **RAM** están disponibles para el usuario.

CAPITULO IV

APLICACIONES

Este capítulo lo dedicaremos a dar algunos ejemplos que nos muestren algunas de las características sobresalientes del circuito como son su **salida serial**, su **lectura a puerto** directamente y otras mas. solo son aplicaciones en BASIC debido a que es la característica más sobresaliente que contiene el circuito **8052AH-BASIC**.

ORGANO PROGRAMABLE

Una de las aplicaciones más sencillas que se puede encontrar de la **instrucción PWM** es la de crear un **Organo Programable**. La forma es como sigue: teniendo un teclado se realiza un barrido a todas las teclas y se comprueba que una de ellas está oprimida, se lee este dato por medio del puerto y se ejecuta el programa. Tomando en cuanta la característica de poder tener programas almacenados en una memoria EPROM se pueden tener de antenmano algunas melodias. Por otra parte tomando en cuenta la **salida serial** del **8052AH-BASIC** se pueden conectar varios en cascada logrando con esto un organo de varias voces.

El programa para la asignación de los valores de la frecuencias es el siguiente:

```
10 REM EL SIGUIENTE PROGRAMA ASIGNA EL VALOR DE UNA NOTA A UNA TECLA
 20 PORT1
 30 A=PORT1
 40 IF A=1  THEN B=881
 50 IF A=2  THEN B=831
 60 IF A=3  THEN B=785
 70 IF A=4  THEN B=741
 80 IF A=5  THEN B=699
 90 IF A=6  THEN B=660
100 IF A=7  THEN B=623
110 IF A=8  THEN B=588
120 IF A=9  THEN B=555
130 IF A=10  THEN B=524
140 IF A=11 THEN B=494
150 IF A=12 THEN B=467
```

```
160 IF A=13 THEN B=440
170 IF A=0 THEN STOP
180  PWM B,B,100
190 GOTO 30
```

Línea 10.- Comentario que indica la función que realiza el programa.

Línea 20.- Lee el valor que se encuentra en el Puerto 1 del circuito el cuál es de 8 "bits".

Línea 30.- Asigna el valor antes leído a una constante A.

Línea 40-160.- Condicionales que asignan un valor de frecuencia determinado a cada uno de los valores leídos en el Puerto 1.

Línea 170.- Condicional que al detectar un cero en el puerto detiene la ejecución del programa.

Línea 180.- La instrucción PWM requiere tres datos: Tiempo de estado bajo, tiempo de estado alto y duración del pulso, los dos primeros dependen del valor leído y el último depende del diseñador.

Línea 190.- Regresa a leer el siguiente valor del puerto.

El programa para tener de antemano una melodía es el siguiente:

```
10 READ A
20 IF A=0 THEN STOP
30 PWM A,A,100
40 DATA (FRECUENCIAS QUE CORRESPONDEN A LAS NOTAS DE LA MELODIA)
```

En donde:

10 y 40.- Como se vió en el capítulo 3 estas instrucciones tienen como objetivo la primera leer secuencialmente los valores que almacena la segunda.

20.- Si el valor de A = 0 indica que se han acabado los datos de la instrucció40 y por lo tanto se detiene la ejecución del programa.

30.- Se asignan los valores a la instrucción PWM.

Viendo ambas aplicaciones a manera de bloques:

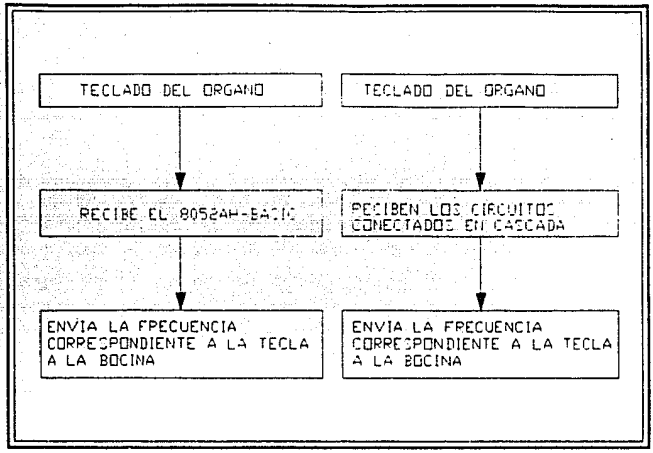

FIG. 4.1 Diagrama de Bloques del Organo

GENERADOR DE FUNCIONES PROGRAMABLE

Por medio de un teclado y aprovechando la característica del 8052AH-BASIC de generar frecuencias por medio de la instrucción PWM, se genera una onda cuadrada a la cual se le determina cuanto tiempo va a estar en estado alto, cuanto en estado bajo y cuantos ciclos vamos a utilizar.

Posteriormente esta señal se pasa a través de un circuito de propósito específico el cual va a cambiar la forma y la amplitud de dicha onda dando como resultado la señal deseada. Esto en forma de bloques es como sigue:

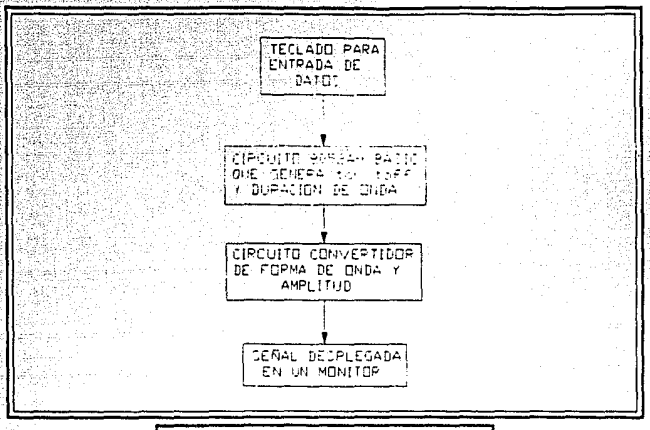

FIG. 4.2 Diagrama de Bloques del Generador

El programa para la realización de dicha función es:

```
  5 STRING 100,20
 10 REM EL SIGUIENTE PROGRAMA VA A PEDIR LOS TIEMPOS EN ESTADO
 20 REM ALTO, ESTADO BAJO Y DURACION DE UNA ONDA PARA
 30 REM POSTERIORMENTE SER MODIFICADA
 40 P. "PROGRAMA GENERADOR DE TIEMPOS DE SUBIDA, BAJADA Y"
 50 P. SPC(14) "DURACION"
 60 INPUT "DEME EL TIEMPO EN ESTADO ALTO",A
 70 INPUT "DEME EL TIEMPO EN ESTADO BAJO",B
 80 INPUT "DEME LA DURACION DEL PULSO",C
 90 P. "EL ESTADO ALTO ES DE ",A," SEGUNDOS."
100 P. "EL ESTADO BAJO ES DE ",B," SEGUNDOS."
110 P. "LA DURACION DEL CICLO ES DE ",C
120 INPUT "SON CORRECTOS ESTO VALORES?",$(1)
130 IF $(1) = "SI" THEN 150
140 GOTO 60
150  A = A/0.000001085
160  B = B/0.000001085
170 PWM A,B,C
180 INPUT "DESEA INTRODUCIR NUEVOS DATOS?",$(2)
190 IF $(2) = "SI" THEN 60
200 P. "GRACIAS"
210 END
```

Donde:

Línea 5.- Se abre un "string" ya que se van a pedir respuestas alfanuméricas como SI y NO.

Línea 10-30.- Comentarios sobre la función que desempeña este programa.

Línea 40-50.- Impresión de un título que describe la Función.

Línea 60.- Pide la duración del estado bajo del pulso en segundos.

Línea 70.- Pide la duración del estado alto del pulso en segudos.

Línea 80.- Pide la duración del ciclo.

Línea 90-110.- Imprime los valores en pantalla con el fin de que sean rectificados por el usuario.

Línea 120.- Pregunta si son correctos los valores desplegados.

Línea 130.- Si es así va a la líinea 150, si no salta a la siguiente línea.

Línea 140.- Regresa a la línea 60 para volver a preguntar los tiempos.

Línea 150-160.- Como se vió un ciclo se calcula 12/valor del cristal el cual en este caso es un cristal de 11.059 MHz, y da un resultado de 1.085 microsegundos, por eso el dato proporcionado por el usuario tiene que ser dividido por este valor para dar calculos reales.

Línea 170.- Se asigna el valor a la instrucción PWM.

Línea 180.- Pregunta si se desea introducir nuevos datos.

Línea 190.- Si es afirmativo va a la línea 60 si no contiua.

Línea 200.- Imprime un mensaje.

Línea 210.- Termina la ejecución del programa.

ENCRIPTADOR DE INFORMACION

Cuando se hay una transmisión o recepción serial para protegerla de intercepciones puede ser convertida a otro tipo de código por medio de un programa que recibe el nombre de ecriptador y se retransmite, al llegar a su destino es desencriptada llegando al receptor de manera normal. Con esto se logra que la información en caso de que sea interceptada no pueda ser interpretada, esta medida es de seguridad para todo tipo de aplicaciones que lo requiera v.gr. Bancos y/o similares.

Este sistema visto en forma de bloques es como sigue:

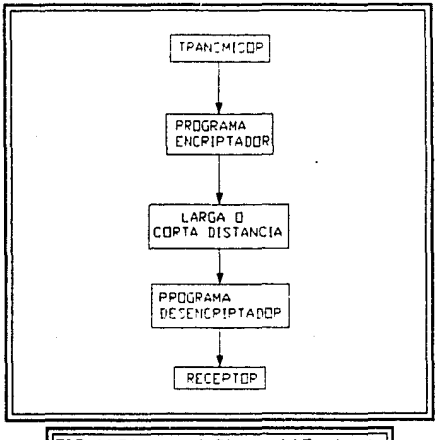

FIG. 4.3 Diagrama de Bloques del Encriptador

Los progragamas que realizan dichas funciones son los siguientes:

```
10 REM EL SIGUIENTE PROGRAMA ENCRIPTA LA INFORMACION RECIBIDA
 20 A=GET
30 IF A=0 THEN GOTO 20
40  A=A*2-60
50  P. CHR(A)
60 GOTO 20
```

```
10 REM EL SIGUIENTE PROGRAMA DESENCRIPTA LA INFORMACION RECIBIDA
20 A=GET
30 IF A=0 THEN GOTO 20
40 A=(A+60)/2
50 P. CHR(A)
60 GOTO 20
```

Donde:

Línea 10.- Comentario donde se especifica la función de cada uno de los programas.

Línea 20.- Se le asigna a la constante A el valor que se encuentra en ese momento desplegado en la pantalla.

Línea 30.- Compara y si el valor es cero vuelve a leerlo.

Línea 40.- Realiza las conversiones correspondientes, esto depende del usuario ya que son arbitrarios los valores que aquí se utilicen.

Línea 50.- Lee nuevamente el dato desplegado en la pantalla.

ANALIZADOR DE TRANSMISIONES

Al momento de estar realizando algun trabajo en el cual se necesite una transmisión serial (por ejemplo una caja registradora) es necesario tener una impresión para verificar si la transmisión esta siendo llevada a cabo satisfactoriamiente. Esto se logra utilizando la salida serial del circuito desplegando dicha información en un dispositivo establecido por el usuario (ya sea una terminal o una impresora).

El programa para la realización de dicha función es:

```
5 REM PROGRAMA ANALIZADOR DE TRANSMISIONES
10 A=GET
20 IF A=0 GOTO 10
30  PRINT@ A
40 GOTO 10
```

Donde:

Línea 5.- Comentario donde se indica la función que realiza el programa.

Línea 10.- Asigna el valor leído en la pantalla a la constante A.

Línea 20.- Si ese valor es cero vuelve a leerlo.

Línea 30.- Esta instrucción como se vió en el capítulo anterior sirve para imprimir en el dispositivo que desee el usuario.

Línea 40.- Regresa a leer un nuevo valor.

Esto visto en forma de bloques:

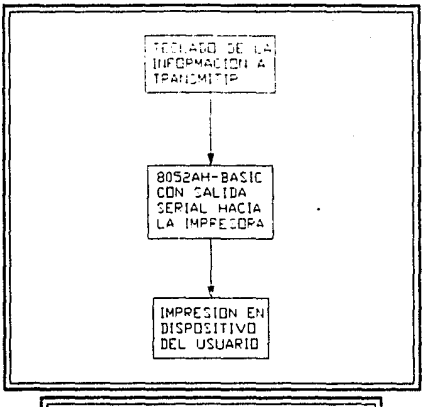

FIG. 4.4 Diagrama de Bloques del Analizador

CONCLUSIONES

CONCLUSIONES

A lo largo de este trabajo, se ha demostrado que el avance de la tecnología ha logrado una alta escala de integración, haciendo que lo que antes era construido con un número considerable de circuitos se reduzca a solo unos cuantos y con mayor poderío.

Gracias a la aparición de los microcontroladores se ha podido hacer una ingeniería de control más precisa, rápida y fácil de manejar.

El hecho de que el 8052AH-BASIC tenga residente un intérprete de BASIC, hace de este circuito uno de los más versátiles, ya que, como se observa en el capítulo 4, hay aplicaciones muy complejas pero fáciles de realizar con la ayuda de este lenguaje pues es de manejo y aprendizaje accesible.

Los microcontroladores son en su mayoría circuitos programados en lenguaje Ensamblador el cuál dependiendo del circuito (marca o número) que se esté manejando cambian las inrucciones. El BASIC es un lenguaje bastante fácil de aprender y de manejo sencillo por lo tanto los ejemplos presentados pueden ser realizados en cualquier microcontrolador con la diferencia de que solo utilizamos unas cuantas líneas para esto.

Dentro de las aplicaciones podemos observar que van desde cosas fáciles como lo es el organo hasta cosas si bien no muy complicadas si de útil manejo ya que son programas para seguridad de información y de tranmisiún de datos.

Al tener instrucciones tales como son las de PROG y FPROG, con todas sus variantes, lo hacen de gran ayuda, pues no siempre se cuenta con el equipo necesario para la programación de EPROM's o EEPROM's, y es muy necesario almacenar ciertos programas de suma importancia. Esto es posible ya que en la terminal 5 el 8052AH- BASIC calcula (de forma interna) el ancho de pulso apropiado con el valor del cristal utilizado para asegurar el tiempo adecuado para la programación. También, el hecho de que, sin tener que contar con una terminal, se pueda comenzar a ejecutar un programa determinado es muy útil, ya que, en ciertas aplicaciones, por ejemplo la del encriptador, sólo se conecta después del transmisor o antes del receptor y no es

necesario la compra de una terminal sólo para ejecutar el programa. Por otra parte el Organo Programable también es un ejemplo de esta aplicación ya que se podría utilizar como una especie de caja de música, o de otra forma ejecutar una melodía determinada pues sabemos con que número es almacenado cada programa pudiendo agregar al teclado del organo una teclas para que al ser leídas sea ejecutada.

La instrucción XFER que realiza la transferencia del programa desde el EPROM o EEPROM hacia a RAM, es muy útil, ya que se pueden modificar los programas sin tener que teclearlos nuevamente y posteriormente se vuelve a grabar el circuito.

En cuanto a "Hardware" se comprobó que, al tener una falla dentro del reconocimiento inicial de memoria, el circuito no opera y se queda en un ciclo de búsqueda, lo cual lleva al usuario a revisar primero si todos los circuitos se encuentran en buen funcionamiento, y posteriormente si las conexiones desde el 8052AH-BASIC hacia el "latch", y de ahí hacia la memoria, son correctas, de no ser así se corrige el error y funciona el cirucuito en óptimas condiciones.

El hecho de tener un oscilador externo causó un poco de problema en la realizacion del prototipo inicial, ya que es un tanto inestable la oscilación del mismo, lo cual se elimina al soldarlo. En lo que se refiere a la transmisión serial no hay problema, ya que se puede transmitir a la velocidad que se desee, con sólo ejecutar la sentencia BAUD con la velocidad de transmisión requerida.

APENDICE A

DESCRIPCION DE CIRCUITOS

En este APENDICE se explicará básicamente la **fución de los circuitos que conforman un sistema básico de trabajo**. La explicación va a consistir en: el diagrama del circuito, el número de terminales, una breve descripción de cada una, y algunas características eléctricas importantes, así como también un diagrama de conexión del mismo.

Se iniciara con una breve explicación de la terminología utilizada en las tablas de características eléctricas de los circuitos, especificando en algunos de ellos, los niveles estandard con los que trabaja la familia TTL.

- **V_{IH} Voltaje de Entrada en Nivel Alto** .- Es el voltaje requerido para que la entrada se considere en nivel alto (1), siendo el voltaje mínimo de 2.0 volts.

- **V_{IL} Voltaje de Entrada en Nivel Bajo** .- Es el voltaje requerido para que la entrada se considera en nivel bajo (0), donde el voltaje máximo es de 0.8 volts.

- **V_{OH} Voltaje de Salida a Nivel Alto** .- Es el voltaje de salida que se toma como nivel alto (1). Garantiza un mínimo de 2.4 volts.

- **V_{OL} Voltaje de Salida a Nivel Bajo** .- Es el voltaje de salida que se toma como nivel bajo (0), garantizando un máximo de voltaje de 0.4 volts.

- **I_{IH} Corriente de Entrada en Nivel Alto** .- Corriente que fluye a una entrada cuando se le aplica un nivel alto a la entrada.

- **I_{IL} Corriente de Entrada en Nivel Bajo** .- Corriente que fluye desde una entrada cuando se le aplica nivel bajo a la entrada.

- **I_{OH} Corriente de Salida en Nivel Alto** .- Corriente que fluye desde una salida cuando esta salida tiene nivel alto.

> **IOL Corriente de Salida en Nivel Bajo** .- Corriente que fluye a una salida cuando esta salida tiene nivel bajo.

8052AH-BASIC

El **8052AH-BASIC** es el circuito principal de la investigación.

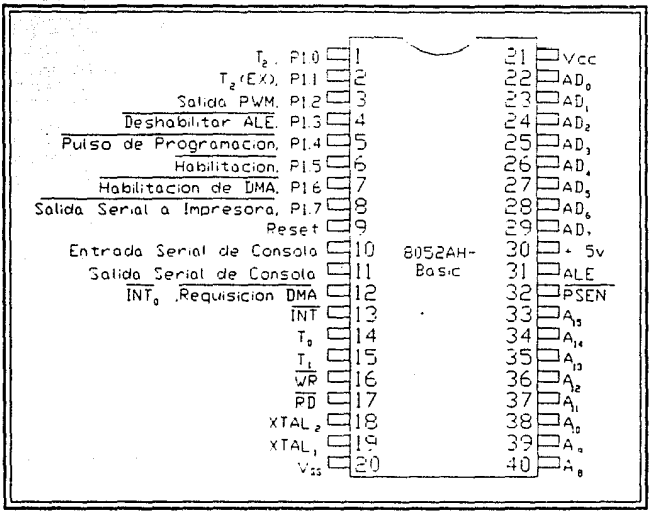

FIG. A.1 Disposición de Terminales del 8052AH BASIC

- **a) Descripción de Terminales :**

1.- Utilizado como disparo de entrada para el **"TIMER/COUNTER"**. Debe de haber un 1 en la salida del "latch" de esta terminal para que se active este modo de operación.

2.- Utilizado como entrada externa para el **"TIMER/COUNTER"**. Debe de haber un 1 en la salida del "latch" de esta terminal para que se active este modo de operación.

3.- Esta terminal es utilizada como el **puerto de salida** de la instrucción **PWM.**

4.- Utilizada para **deshabilitar la señal de ALE** para el "latch" de direcciones externas cuando se está programando un EPROM.

5.- Proporciona los **pulsos** cuando se va a **programar un EPROM.** Calcula el ancho de pulso apropiado con el valor del cristal para asegurar el tiempo apropiado y se activa con un 0.

6.- **Habilita el voltaje de programación** y se mantiene en estado bajo (0) mientras dura la misma.

7.- **Junto con la terminal anterior implanta el Acceso Directo a Memoria** y funciona con lógica negada.

8.- **Puerto 1.7 (Salida Serial de Impresora).** Funciona como puerto serial cuando se utilizan las intrucciones **LIST#** o **PRINT#.**

9.- Es el **"Reset"** el cual es provocado con un voltaje de 2.5 volts (aproximadamente) durante 2 ciclos de máquina cuando el oscilador se encuentra funcionando.

10.- **Entrada serial de la consola.**

11.- **Salida serial de la consola.**

12.- Puede ser **programada** para que **funcione como entrada** para la requisición de un acceso directo a memoria necesario cuando se **programa un EEPROM.**

13.- Trabaja como una **interrupción externa** y se activa en estado bajo (0).

14.- Puede ser programada como una **entrada externa** para el "TIMER/COUNTER" 0.

15.- Puede ser programada como una **entrada externa** para el "TIMER/COUNTER" 1.

16.- **Habilita la escritura de datos en la memoria externa.** Se activa en estado bajo o sea cuando se encuentra un cero en su entrada.

17.- **Habilita la lectura de datos de la memoria externa.** Se activa en estado bajo o sea en cero 0.

18.- **Salida del amplificador inversor** que forma el amplificador y entrada del generador interno del reloj. Para utilizar esta terminal debe de recibir una señal de oscilación externa.

19.- **Entrada al amplificador inversor** que forma el oscilador.

20.- **Tierra.**

21 a la 28.- **Parte alta del canal de direcciones** utilizado para el acceso a memoria externa.

29.- **Señal de control para la habilitación de programas** que se encuentren en la memoria externa. Se activa con lógica negada y permanece inactivo mientras el usuario se encuentra ejecutando un programa en ensamblador en la memoria externa.

30.- Utilizado para el "latch" de las direcciones de la parte baja durante procesos de lectura o escritura u operaciones de búsqueda de programas a memoria externa.

31.- Si se conecta a tierra funciona como un 8052AH normal.

32 a la 39.- Canal de direcciones y de datos multiplexados los cuales son utilizados durante el acceso a memoria externa. Una resistencia de "pull-up" es necesaria en estas terminales cuando se utilizan para programar dispositivos.

40.- Voltaje de alimentación (5 volts).

- **b) Características Eléctricas DC.**

(T_A = 0°C a 70°C, V_{CC} = 4.5V a 5.5V, V_{SS} = 0V)

SIM.	PARAMETROS	MIN.	MAX.	UNID.	CONDICION
V_{IL}	Entrada de Bajo Voltaje	-0.5	0.8	V	
V_{IH}	Entrada de Alto Voltaje (Excepto RST y XTAL2)	2.0	Vcc + 0.5	V	
V_{IH1}	Entrada de Alto Voltaje a RST por "Reset", XTAL2	2.5	Vcc + 0.5	V	XTAL1 a V_{SS}
V_{OL}	Salida de Bajo Voltaje Puerto1, A8-15, Funciones de Control		0.45	V	I_{OL} = 1.6mA
V_{OL1}	Salida de Bajo Voltaje ALE, /PSEN (ver Nota)		0.45	V	I_{OL} = 3.2mA

SIM.	PARAMETROS	MIN.	MAX.	UNID.	CONDICION
V_{OH}	Salida de Alto Voltaje Puerto1, A8-15, Funciones de Control	2.4		V	I_{OH} = -80uA
V_{OH1}	Salida de Alto Voltaje AD0-7, ALE, /PSEN	2.4		V	I_{OH} = -400uA
I_{IL}	Entrada de Corriente con Lógica 0 Puerto 1, A8-15, Funciones de Control		-800	uA	V_{IN} = 0.45V
I_{IL2}	Entrada de Corriente con Lógica 0 XTAL2		-2.5	mA	XTAL1 al V_{SS}, V_{IN} = 0.45V
I_{L1}	Entrada de Corriente de Fuga a AD0-7, EA		+/-10	uA	0.45V < V_{IN} < V_{CC}
I_{IH1}	Entrada de Alta Corriente a RST/VPD para "Reset"		500	uA	V_{IN} = V_{CC}-1.5V
I_{CC}	Fuente de Corriente		175	mA	Todas las Salidas Desconectadas
C_{IO}	Capacitancia de los "Buffers" de Entrada/Salida		10	pF	f_c = 1MHZ, T_A = 25°C

NOTA: V_{OL} es degradado cuando el 8032AH/8052AH tiene un cambio rápido en forma externa de capacitancia.

6416

Memoria **RAM estática** de **2 "Kbytes"** siendo de **8 "bits"** cada una de sus localidades.

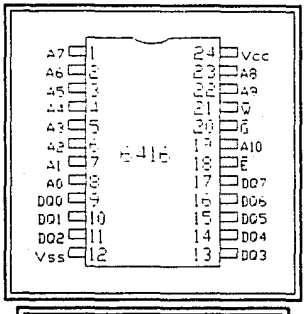

FIG. A.2 Disposición de Terminales

- **a) Descripción de Terminales.**

1 a la 8.- "Bits" de direcciones, iniciando con A7 y terminando con A0.

9 a la 11.- Son las **entradas y salidas de datos D0, D1 y D2** respectivamente.

12.- Tierra.

13 a la 17.- Son las **entradas y salidas de datos** D3 al **D7** respectivamente.

18.- Habilitador de Circuito, activandose con lógica negada.

19.- "Bit" de dirección A10.

20.- Habilitador de Salida, se activa con lógica negada.

21.- Habilitador de Escritura, se activa con lógica negada.

22 y 23.- "Bits" de dirección A9 y A8.

24.- Voltaje de alimentación (5 volts).

- **b) Características Eléctricas DC.**

PARAMETROS	SIM.	MIN.	MAX.	UNID.
Entrada de Corriente de Fuga (V_{CC} = 5.5V, V_{IN} = GND a V_{CC})	I_{LI}	-10	10	uA
Salida de Corriente de Fuga (/E = V_{IH} o /G = V_{IH}, $V_{I/O}$ = GND a V_{CC})	I_{LO}	-50	50	uA
Operación de la Fuente de Corriente (/E = V_{IL}, $I_{I/O}$ = 0mA)	I_{CC1}	---	120	mA
Fuente de Corriente en estado de "Standby" (/E = V_{IH})	I_{SB}	---	20	mA
Bajo Voltaje de Salida (I_{OL} = 8.0mA)	V_{OL}	---	0.4	V
Alto Voltaaje de Salida (I_{OH} = -4.0mA)	V_{OH}	2.4	---	V

1488

Es el "Driver" que maneja la transmisión desde el microcontrolador hacia la terminal.

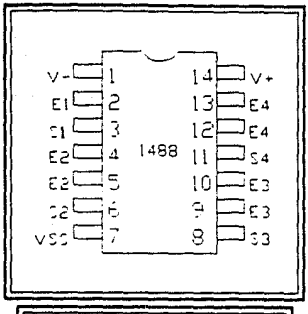

FIG. A.3 Disposición de Terminales

- **a) Descripción de Terminales.**

1.- Voltaje de alimentación (-12 volts).

2.- Entrada al "Driver" 1.

3.- Salida del "Driver" 1.

4 y 5.- Entradas al "Driver" 2.

6.- Salida del "Driver" 2.

7.- Tierra.

8.- Salida del "Driver" 3.

9 y 10.- Entradas al "Driver" 3.

11.- Salida del "Driver" 4.

12 y 13.- Entradas al "Driver" 4.

14.- Voltaje de alimentación (12 volts).

b) Características Eléctricas DC.

$T_A = 0^oC$ a $+70^oC$, $V^+ = +4.5V$ a 12V, GND = 0V, $V^- = -4.5V$ a -12V

SIM.	PARAMETROS	COND.	MIN.	MAX.	UNID.
I_{IL}, I_{IH}	Corriente de Entrada Máxima	V_{IN} = GND o V^+	-10	10	uA
V_{IH}	Entrada de Alto Voltaje		2.0	V_{DD}	V
V_{IL}	Entrada de Bajo Voltaje	$V^+ > 7V$, $V^- < -7V$ $V^+ < 7V$, $V^- > -7V$	GND GND	0.8 0.6	V V
V_{OH}	Salida de Alto Voltaje	$V_{IN} = V_{IL}$ R_L = 3KE o 7KE $V^+ = +4.5V$, $V^- = -4.5V$ $V^+ = +9V$, $V^- = -9V$ $V^+ = +12V$, $V^- = -12V$	3.0 6.5 9.0		V V V
V_{OL}	Salida de Bajo Voltaje	VIN = V_{IH} R_L = 3KE o 7KE $V^+ = +4.5V$,			

SIM.	PARAMETROS	COND.	MIN.	MAX.	UNID.
V_{OL}	Salida de Bajo Voltaje	$V^- = -4.5V$ $V^+ = +9V$, $V^- = -9V$ $V^+ = +12V$, $V^- = -12V$		-3.0 -6.5 -9.0	V V V
I_{OS+}	Salida de Corriente de Corto Circuito en Alto Nivel	$V_{IN} = V_{IL}$ $V_{OUT} = GND$ $V^+ = +12V$, $V^- = -12V$		+45	mA
I_{OS-}	Salida de Corriente de Corto Circuito en Bajo Nivel	$V_{IN} = V_{IH}$ $V_{OUT} = GND$ $V^+ = +12V$, $V^- = -12V$		-45	mA
R_{OUT}	Salida de Resistencia	$V^+ = V^- = 0V$ $-2V < Vout < 2V$	300		E
I_{CC+}	Fuente de Corriente Positiva	$V_{IN} = V_{IL}$, R_L = abierta $V^+ = +4.5V$, $V^- = -4.5V$ $V^+ = +9V$, $V^- = -9V$ $V^+ = +12V$, $V^- = -12V$ $V_{IN} = V_{IH}$, R_L = abierta $V^+ = +4.5V$, $V^- = -4.5V$ $V^+ = +9V$, $V^- = -9V$ $V^+ = +12V$, $V^- = -12V$		 10 30 60 50 300 500	 uA uA uA uA uA uA

SIM.	PARAMETROS	COND.	MIN.	MAX.	UNID.
ICC-	Fuente de Corriente Negativa	$V_{IN} = V_{IL}$, R_L = abierta $V^+ = +4.5V$, $V^- = -4.5V$		-10	uA
		$V^+ = +9V$, $V^- = -9V$		-10	uA
		$V^+ = +12V$, $V^- = -12V$		-10	uA
		$V_{IN} = V_{IH}$, R_L = abierta $V^+ = +4.5V$, $V^- = -4.5V$		-30	uA
		$V^+ = +9V$, $V^- = -9V$		-30	uA
		$V^+ = +12V$, $V^- = -12V$		-60	uA

1489

Es el **"Driver"** que maneja la recepción desde la terminal hacia el microcontrolador.

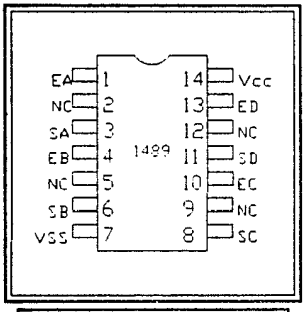

FIG. A.4 Disposición de Terminales

- **a) Descripción de Terminales.**

1, 4, 10 y 13.- Entradas a los "Drivers" A, B, C y D, respectivamente.

2, 5, 9 y 12.- No conectar.

3, 6, 8 y 11.- Salidas de los "Drivers" A, B, C y D, respectivamente.

7.- Tierra.

14.- Voltaje de alimentación (5 volts).

- **b) Características Eléctricas DC.**

$T_A = 0^oC$ a $+70^oC$, $+4.5 < V_{CC} < 5.5V$, $GND = 0V$

SIM.	PARAMETROS	COND.	MIN.	TIPO	MAX.	UNID.
V_{TH}	Entrada de Alto Voltaje en el Umbral		1.3		2.5	V
V_{TL}	Entrada de Bajo Voltaje en el Umbral		0.5		1.7	V
V_H	Tipica Entrada de Histeresis			1.0		V
I_{IN}	Entrada de Corriente	$V_{IN} = +25V$ $V_{IN} = -25V$ $V_{IN} = +3V$ $V_{IN} = -3V$	3.6 -3.6 +0.43 -0.43		8.3 -8.3 +1.0 -1.0	mA mA mA mA
V_{OH}	Salida de Alto Voltaje	$V_{IN} = V_{TLmn}$ $I_{out} = -3.2mA$	2.8			V
V_{OL}	Salida de Bajo Voltaje	$V_{IN} = V_{THmx}$ $I_{out} = -3.2mA$			0.4	V
I_{CC}	Fuente de Corriente	R_L = abierto $V_{IN} = V_{THmx}$ o V_{TLmn}			+900	uA

74373

Este circuito se encuentra constituido por 8 "LATCH" tipo D.

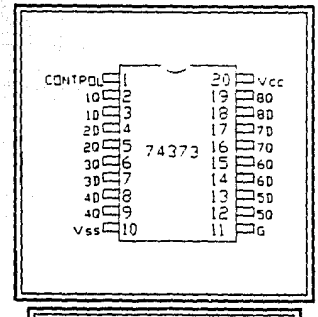

FIG. A.5 Disposición de Terminales

- **a) Descripción de Terminales.**

1.- Control de la salida.

2, 5, 6, 9, 12, 15, 16 y 19.- Salidas de los "Latch".

3, 4, 7, 8, 13, 14, 17 y 18.- Entradas a los "Latch".

10.- Tierra.

11.- Habilitador.

20.- Voltaje de alimentación (5 volts).

- **b) Características Eléctricas.**

SIM.	PARAMETROS	COND. PRUEB.*	MIN.	TIPO **	MAX.	UNID.
V_{IH}	Entrada de Alto Voltaje		2			V
V_{IL}	Entrada de Bajo Voltaje				0.8	V
V_{IK}	Entrada de Voltaje ("Clamp")	V_{CC} = Min, I_I = -18mA			-1.5	V
V_{OH}	Salida de Alto Voltaje	V_{CC} = Min, V_{IH} = 2V, V_{IL} = V_{ILmx}, I_{OH} = Max	2.4	3.1		V
V_{OL}	Salida de Bajo Voltaje	V_{CC} = Min, V_{IH} = 2V, V_{IL} = V_{ILmx} I_{OL} = 12mA, I_{OL} = 24mA		0.25 0.35	0.4 0.5	V V
I_{OZH}	Salida de Corriente en Estado de Apagado, aplicando Alto Voltaje	V_{CC} = Max, V_{IH} = 2V, V_O = 2.7V			20	uA
I_{OZL}	Salida de Corriente en Estado de Apagado, aplicando Bajo Voltaje	V_{CC} = Max, V_{IH} = 2V, V_O = 0.4V			-20	uA
I_I	Entrada de Corriente a una Entrada de Voltaje Maximo	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	Entrada de Alta Corriente	V_{CC} = Max, V_I = 2.7V			20	uA
I_{IL}	Entrada de Baja Corriente	V_{CC} = Max, V_I = 0.4V			-0.4	mA

SIM.	PARAMETROS	COND. PRUEB.*	MIN.	TIPO **	MAX.	UNID.
I_{OS}	Salida de Corriente en Corto Circuito	V_{CC} = Max	-30		-130	mA
I_{CC}	Fuente de Corriente	V_{CC} = Max, Salida de Control a 4.5V		24	40	mA

* Para Min o Max, utilizar apropiadamente las especificaciones de condición de operación.
** Todos los valores típicos de V_{CC} = 5V, T_A = 25°C.

A continuación se muestra el diagrama de conexión básico del controlador principal:.

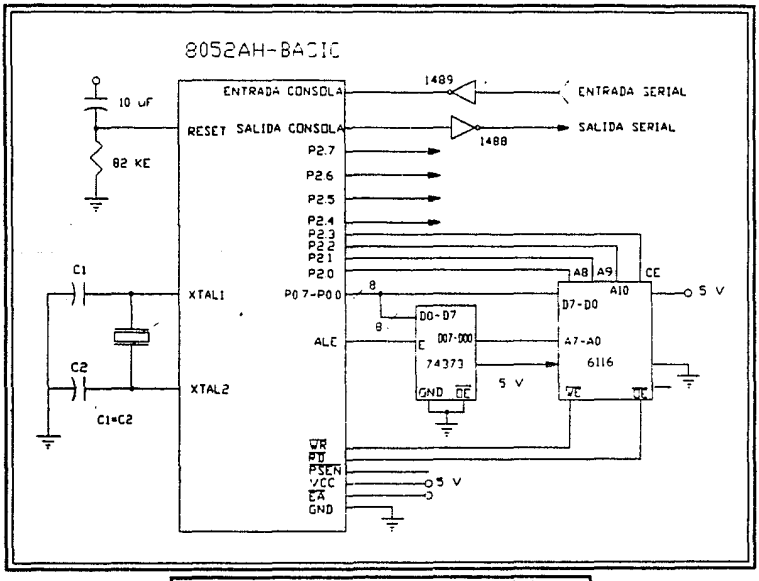

FIG. A.6 Diagrama Básico de Conexión del Controlador.

GLOSARIO

GLOSARIO

ACCESO DIRECTO A MEMORIA. Mecanismo que permite un dispositivo de Entrada/Salida tomar el control por uno o más ciclos de memoria para escribir o leer en la memoria. El orden de ejecución de los pasos del programa permanece sin ningún cambio.

ACUMULADOR. Registro el cual contiene uno de los operandos para operaciones lógicas y aritméticas y al final contiene el resultado.

A.L.U.. Unidad Lógico Aritmética. La cual ejecuta sumas, restas, corrimientos, operaciones lógicas, etc.

APUNTADOR DE REGISTROS. Registro que contiene una dirección de memoria correspondiente a un dato el cual va a ser utilizado en por una instrucción.

APUNTADOR DE PILA ó "STACK POINTER". El "Stack Pointer" (ó Apuntador de Pila) es un registro que contiene una dirección de memoria RAM (dirección en donde se encuentra el "Stack"), a partir de la cual, se pueden salvar en forma descendente los contenidos de un par de registros ó, en forma ascendente, se obtienen los últimos dos datos almacenados para utilizarlos por ejemplo para cargar a un par de registros.

BASIC. Siglas de "Beginner's All purpose Simbolic Instruction Code", lenguaje de propósito general y de fácil manejo.

B.C.D.. En aplicaciones como frecuencímetros, vóltmetros digitales, o calculadoras, en donde la salida es desplegada en forma decimal frecuentemente el código BCD es utilizado. Este código BCD representa cada dígito decimal del 0 al 9 con una palabra binaria de 4 "bits". Las palabras en BCD son equivalentes a las representadas en forma binaria, y no pueden representar un dígito mayor a 9, como resultado de esto tenemos un acarreo si el resultado de una operación es mayor a 1001 binario ó 9 decimal. Por ejemplo si se tiene la siguiente operación:

```
   7        0111
 + 5      + 0101
  12       1100
```

el resultado en binario es correcto pero es una representación ilegal en código BCD y esto provoca que se ponga en 1 el bit de esta posición. Para convertir este resultado al formato BCD, es sumado un factor de corrección de 6.

BIT. Abreviación de Dígito Binario. Caracteres solos en un número binario.

BYTE. Una secuencia de n "bits" la cual es operada como una unidad es llamada como un "byte" de n "bits". La medida más frecuente de un "byte" es de 8 "bits".

CANAL. Grupo de "caminos" que permite el intercambio de "palabras".

CICLO DE INSTRUCCION. Es el proceso que va desde la búsqueda de una instrucción desde la memoria hasta su ejecución.

CICLO DE MAQUINA. Es el ciclo de la Unidad Central de Proceso básico. En un ciclo de máquina una instrucción puede ser mandada a memoria y una palabra (dato o instrucción) leída o escrita, o, en un ciclo de máquina la búsqueda de una instrucción puede ser ejecutada.

CICLO DE TIEMPO. Intervalo de tiempo al cual cualquier juego de instrucciones es repetida con regularidad y con la misma secuencia.

CONTADOR DE PROGRAMA. Es un Registro el cual especifica la dirección de la siguiente instrucción que será buscada y ejecutada. Normalmente se incremente en forma automática cada vez que hay una búsqueda de instrucción.

DATO INMEDIATO. Dato que sigue inmediatamente a una instrucción en memoria, y es utilizado como un operando para una instrucción.

DIRECCION. Número utilizado por la C.P.U. para especificar una localidad en la memoria.

DIRECCIONAMIENTO DIRECTO. La dirección de una instrucción u operando es completamente especificada en una instrucción sin ninguna referencia a un registro base o un registro indexado.

DIRECCIONAMIENTO INDIRECTO. Un tipo de direccionamiento en el cual la dirección del operando es especificada por un registro auxiliar o una localidad de memoria especificada por la instrucción más que por los "bits" que se encuentren en la misma.

EEPROM (Electrically Erasable Programmable Read Only Memory). Igual que la EPROM con la única diferencia de que se borra eléctricamente.

EJECUTAR. Proceso de interpretación de una instrucción y realizar las operaciones indicadas.

EPROM (Erasable Programmable Read Only Memory). Memoria que puede ser grabada y borrada un número indeterminado de veces y que una vez grabada funciona prácticamente como una ROM. Se borra por medio de rayos ultravioleta.

"FLIP-FLOP". Un elemento de memoria es cualquier dispositivo que puede almacenar niveles lógicos 0 y 1 ("bits"), de tal manera que un nivel o un grupo de niveles pueden ser accesados posteriormente o alterado al recibir otros niveles. Un "flip-flop" es un circuito que puede almacenar dos estados lógicos y tiene la capacidad de cambiar de un estado (cuando el estado de entrada es diferente al que tiene almacenado) a otro con la aplicación de una señal de control (reloj), y permanecer en ese estado después de recibir la señal de control.

HABILITACION DE INTERRUPCIONES. Mecanismo que permite al programa especificar si una requisición de interrupción deberá de ser aceptada o no.

"HARDWARE". Equipo físico que forma un sistema determinado.

HEXADECIMAL. Sistema numérico que utiliza los siguientes dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, y F, para representar todos los posibles valores de una palabra de 4 "bits". El equivalente decimal va del 0 al 15. Dos dígitos hexadecimales pueden ser utilizados para la especificación de un "byte".

INSTRUCCION. Un juego de "bits" que definen una operación, y es un comando básico entendido por el dispositivo controlador. Puede realizar el movimiento de un dato, funciones lógicas y aritméticas, control de dispositivos de Entrada/Salida, o tomar decisiones como la de cual instrucción va a ser ejecutada a continuación.

INTERPRETE. Es un programa el cual busca y ejecuta instrucciones o pseudo instrucciones escritas en algún lenguaje de alto nivel.

JUEGO DE INSTRUCCIONES. Son todas las instrucciones de propósito general disponibles en cualquier microcontrolador, microprocesador o microcomputador.

"LATCH". Frecuentemente la información se debe conservar durante cierto tiempo antes de que la microcomputadora la lea o que un dispositivo periférico la reciba. En situaciones como esta se deben de utilizar ciertos dispositivos conocidos como "latch".

LAZO. Series de instrucciones anidadas en las cuales la última instrucción puede causar una repetición de las instrucciones hasta que una condición terminal es utilizada, continuando así con la secuencia del programa principal.

LENGUAJE ENSAMBLADOR. Lenguaje de programación el cual salva al programador del problema de recordar los patrones de "bits" de cada instrucción, esto es debido a que cada frase del lenguaje se traslada directamente en una palabra de lenguaje máquina específica.

LENGUAJE MAQUINA. Es la forma numérica de especificar instrucciones, listas para ser "cargadas" en la memoria y ser ejecutadas por la máquina. Este es el nivel de lenguaje más bajo para poder escribir un programa. El valor de cada "bit" en todas las instrucciones debe de ser especificado.

LONGITUD DE UNA INSTRUCCION. Es el número de palabras necesarias para almacenar una instrucción.

LONGITUD DE UNA PALABRA. Es el número de "bits" en una palabra.

MICROCONTROLADOR. Circuito integrado el cuál se encuentra formado por A.L.U., Registros, Circuito Temporizador, Puertos de Entrada/Salida, y algunos cuentan con Memoria ROM, el cual es utilizado para realizar tareas de control.

MICROPROCESADOR. Circuito integrado el cual se encuentra formado por una A.L.U., Registros y Unidad de Control.

MNEMONICO. Nombres simbólicos o abreviaciones para las instrucciones, registros, localidades de memoria, etc.

NIBBLE. Es una secuencia de 4 "bits" operada como una unidad.

PALABRA. Es un grupo básico de "bits" es cual puede ser manipulado (leido, almacenado, sumado, etc.) por el microcontrolador es un solo paso. Hay dos tipos de palabras utilizadas: Palabras de Datos y Palabras de Instrucciones. Las palabras de datos contienen información que debe de ser manipulada. Las palabras de Instrucciones causan que el dispositivo ejecute una operación en particular.

PROGRAMA. Es una colección de instrucciones ordenadas propiamente para la realización de alguna tarea en particular.

PROM (Programmable Read Only Memory). Memoria que viene vacía de fábrica para que se grabe solo una vez convirtiendose así en una ROM.

RAM (Random Access Memory). Es una memoria para lectura y escritura y como su nombre lo indica su acceso a una localidad de memoria es inmediato. Es una memoria volátil lo cual significa que al ser interrumpida la energía se pierde la información que se encontraba almacenada en ella. Hay dos variaciones de esta memoria: Dinámica y Estática. Una memoria dinámica es aquella la cual necesita un ciclo de "refresco" para la información constante, y una memoria estática es la que solo con Vcc mantiene su información sin necesidad del ciclo antes mencionado.

REGISTRO. Circuito de rápido acceso utilizado para almacenar "bits" o palabras. Los registros juegan un papel importante en las operaciones realizadas

por el dispositivo principal. En la mayoría de las aplicaciones la eficiencia de los programas es relativa al número de registros.

REGISTRO DE DATOS. Cualquier registro el cual contiene un dato.

RELOJ. Dispositivo el cual envia pulsos de tiempo para la sincronización de las actividades del/los circuitos.

REQUISICION DE INTERRUPCION. Es una señal la cual suspende temporalmente la secuencia de una rutina y transfiere el control a una rutina especial y puede darse el caso de que se regrese a la rutina que fue interrumpida. La habilidad de tener interrupciones es muy útil para aplicaciones de comunicación en las cuales se permita al dispositivo principal servir varios canales.

ROM (Read Only Memory). Memoria de solo lectura que viene grabada de fábrica y no es volátil.

RUTINA. Usualmente se refiere a sub-programas.

RUTINA DE SERVICIO DE INTERRUPCION: Rutina (o programa) para almacenar lejos del "Stack" el estado presente del programa para responder a una requisición de una interrupción, realizando el "trabajo real" requerido por la interrupción, reestablece el estado que había sido almacenado y regresa a la operación que había sido interrumpida.

SALTO. Designación para un incremento normal de un paso del Contador de Programa, forzando con esto que un nuevo valor de dirección sea cargado en el mismo y de este modo la nueva instrucción puede ser buscada en una localidad arbitraria (puede ser para el salto hacia adelante o hacia atrás).

"SOFTWARE". Programas de computadoras. Utilizado comúnmente para denotar programas de propósito general provistos por algún fabricante, como lo son ensambladores, editores, compiladores, etc.

"STACK". Durante el procesamiento de información es muy útil contar con cierta área de memoria RAM donde se puedan almacenar temporalmente datos en forma rápida. Para agilizar ésta función es muy importante no tener la necesidad de buscar la dirección disponible en esa área cada vez que se desea almacenar

un dato ó rescatar el último dato almacenado. Esta área de memoria recibe el nombre de "Stack".

SUBRUTINA. Es un sub-programa (o grupo de instrucciones) la cual puede ser accesada por más de un lugar desde el programa principal.

TIEMPO DE INSTRUCCION. Es el tiempo requerido para realizar un ciclo de instrucción completo.

BIBLIOGRAFIA

BIBLIOGRAFIA

- **MALVINO, Albert Paul.**
  Principios de Electrónica.
  Mc. Graw Hill.
  México, 1984.

- **BOYLESTAD, Robert; NASHELSKY, Louis.**
  Electrónica, Teoría de Circuitos.
  Prentice Hall.
  México, 1989.

- **INTEL.**
  "8-Bit Embedded Controller Handbook".
  Intel.
  Santa Clara California, U.S.A., 1989.

- **MOTOROLA.**
  "Memory Data".
  Motorola.
  U.S.A., Mayo 1984.

- **NATIONAL SEMICONDUCTOR.**
  "Interface Databook".
  National Semiconductor.
  U.S.A., 1988.

- **TEXAS INSTRUMENTS INCORPORATED.**
  "The TTL Data Book for Design Engineers".
  Texas Instruments Incorporated.
  U.S.A., 1981.

- **THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS.**
  "Spectrum".
  I.E.E.E..
  U.S.A., Noviembre 1990.

- **M. en C. GARCIA, Narcia Octavio F..**
  Microprocesadores Z-80.
  México, 1984.